XML C Technology for HP OpenVMS Installation Guide and Release Notes

November 2007

Version 3.0, based on
Apache Xerces C Version 2.7.0 and
Apache Xalan C Version 1.10

Contents

Before Installing XML C Technology

- Hardware Prerequisites
- Software Prerequisites

Installing XML C Technology

- Downloading the Kit
- Expanding the Kit
- Installing the Kit

After Installing XML C Technology

Creating Xerces C and Xalan C Applications using XML-C Version 3.0 on OpenVMS Alpha and Integrity servers

Creating Xerces C Applications

- Compiling and Linking Xerces C Applications

Creating Xalan C Applications

- Compiling and Linking Xalan C Applications

Building the XML C Sources

Release Notes

Support

Before Installing XML C Technology

Hardware Prerequisites

XML C Technology Version 3.0 for OpenVMS is available on the OpenVMS Alpha and Integrity server platforms.

The *XML C Technology for OpenVMS* self-extracting file requires approximately 116,408 blocks (57 MB) of disk space for OpenVMS Alpha and 390,000 blocks (191 MB) of disk space for OpenVMS Integrity servers. Expanding the file requires an additional 173,103 (85 MB) blocks for OpenVMS Alpha and 590,000 (287 MB) blocks for OpenVMS Integrity servers.

To install the product, a minimum of 295,018 blocks of disk space is required for OpenVMS Alpha and 1,124,191 blocks for OpenVMS Integrity servers. Documentation and Sources require an additional of 96,881 blocks of disk space for OpenVMS Alpha and 177,454 blocks of disk space for OpenVMS Integrity servers.

Software Prerequisites

For OpenVMS Alpha:

HP OpenVMS Alpha Version 7.3-2 or higher
HP C++ Version 7.1 or higher for OpenVMS Alpha

For OpenVMS Integrity servers:

HP OpenVMS Integrity servers Version 8.2 or higher
HP C++ Version 7.1 or higher for OpenVMS Integrity servers

Installing on an ODS-5 enabled disk is required. Because of long file names and directory depth issues, the installed code base and the accompanying documentation cannot be guaranteed to function properly in a non-ODS5 environment.

Installing XML C Technology

Removing the Previous Version

To remove the previous installation, perform the following steps:

- `Run $ @XML-C$ROOT:[XML-C-2_0]UNINSTALL_XML-C-2_0`

`Or`

- The PCSI product installation will provide an option to delete the previous JAR based installation. It checks for XML-C$ROOT:[XML-C-2_0]UNINSTALL_XML-C-2_0.COM and prompts for removal.

Note: The *XML C Technology Version 3.0 for OpenVMS* kit provides you an option to retain the previous version (XML C V2.0-1) of the kit. If you want to use the previous version (i.e., XML C V2.0-1) of the kit, the XML-C$ROOT, XERCES-C$ROOT and XALAN-C$ROOT logicals must be defined to point to that kit.

Downloading the Kit

The XML C Technology Version 3.0 for OpenVMS can be obtained in the following ways:

- Software Products Library (SPL) for OpenVMS Alpha (from Q2 2008)
- I64 Delta CDs for OpenVMS Integrity servers
- OpenVMS website for both OpenVMS Alpha and Integrity. To download the kit, please submit the registration form.

Expanding the Kit

To expand the *XML C Technology Version 3.0 for OpenVMS* file, perform the following steps:

1. Set default to a directory on the ODS-5 disk where the installation files can be extracted. You can download and extract the files to an ODS-2 disk. However, you must install the kit on an ODS-5 disk.

   For example:

   ```
   $ SET DEFAULT DISK:[DOWNLOADS]
   ```

2. Enter the following command to extract relevant files from the XML C kit:

   ```
   $ RUN DISK:[DOWNLOADS]XMLC-V0300-AXP.EXE (for OpenVMS Alpha)

   $ RUN DISK:[DOWNLOADS]XMLC-V0300-I64.EXE (for OpenVMS Integrity servers)
   ```

   Following is an example of expanding the self-extractable kit:

   On OpenVMS Alpha:

   ```
   $ RUN XMLC-V0300-AXP.EXE
   UnZipSFX 5.41 of 16 April 2000, by Info-ZIP (Zip-Bugs@lists.wku.edu)
   inflating: hp-axpvms-xml_c-v0300--1.pcsi$compressed
   inflating: hp-axpvms-xml_c-v0300--1.pcsi$compressed_esw
   ```

   On OpenVMS Integrity servers:

   ```
   $ RUN XMLC-V0300-I64.EXE
   UnZipSFX 5.42 of 14 January 2001, by Info-ZIP (Zip-Bugs@lists.wku.edu)
   inflating: hp-i64vms-xml_c-v0300--1.pcsi$compressed
   inflating: hp-i64vms-xml_c-v0300--1.pcsi$compressed_esw
   ```

Installing the Kit

Note: This kit creates logicals in the system name table. Therefore, the SYSNAM privilege is required.

To install the *XML C Technology for OpenVMS* kit, enter the following command:

Installation on an OpenVMS Alpha V7.3-2 and V8.2 systems:

```
$ PRODUCT INSTALL XML_C /DEST=DISK:[DOWNLOADS]

The following product has been selected:
HP AXPVMS XML_C V3.0 Layered Product

Do you want to continue? [YES]

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product and
for any products that may be installed to satisfy software dependency
requirements.
```

```
HP AXPVMS XML_C V3.0: XML Technology for OpenVMS is based on Apache Xerces
C Version 2.7.0 and Apache Xalan C Version 1.10

© Copyright 2007 Hewlett-Packard Development Company, L.P.

Hewlett-Packard Company

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
HP AXPVMS XML_C V3.0 DISK:[DOWNLOADS.]

Portion done: 0%...10%...20%...30%...40%...50%...60%...70%...80%...90%

The supported version of CXX compiler is V7.1 or higher

XML-C$ROOT, XERCES-C$ROOT and XALAN-C$ROOT have been defined.
The following lines must be added to SYS$MANAGER:SYLOGICALS.COM so that it
will be defined each time the system is rebooted.

$ define/system/nolog/trans=concealed XML-C$ROOT DISK:[DOWNLOADS.XML.]
$ define/system/nolog/trans=concealed XERCES-C$ROOT
DISK:[DOWNLOADS.XML.Xerces-C-3_0.]
$ define/system/nolog/trans=concealed XALAN-C$ROOT
DISK:[DOWNLOADS.XML.Xalan-C-3_0.]

Verification of the installation can be performed using the XML-C Test
Procedure. To run the XML-C Test Procedure, enter the following command:

$ @XML-C$ROOT:[XML-C-3_0]XML-C-3_0-TP
...100%

The following product has been installed:
HP AXPVMS XML_C V3.0 Layered Product

$
```

Installation on an OpenVMS Alpha V8.3 system:

```
$ PRODUCT INSTALL XML_C /DEST=DISK:[DOWNLOADS]
Performing product kit validation ...
%PCSI-I-VALPASSED, validation of DISK:[DOWNLOADS]HP-AXPV
MS-XML_C-V0300--1.PCSI$COMPRESSED;1 succeeded

The following product has been selected:
HP AXPVMS XML_C V3.0 Layered Product

Do you want to continue? [YES]

Configuration phase starting ...

You will be asked to choose options, if any, for each selected
product and for any products that may be installed to satisfy
software dependency requirements.
```

```
HP AXPVMS XML_C V3.0: XML Technology for OpenVMS is based on Apache
Xerces C Version 2.7.0 and Apache Xalan C Version 1.10

© Copyright 2007 Hewlett-Packard Development Company, L.P.

Hewlett-Packard Company

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
HP AXPVMS XML_C V3.0 DISK:[DOWNLOADS.]

Portion done:
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%

The supported version of CXX compiler is V7.1 or higher

XML-C$ROOT, XERCES-C$ROOT and XALAN-C$ROOT have been defined.
The following lines must be added to SYS$MANAGER:SYLOGICALS.COM so
that it will be defined each time the system is rebooted.

$ define/system/nolog/trans=concealed XML-C$ROOT
DISK:[DOWNLOADS.XML.]
$ define/system/nolog/trans=concealed XERCES-C$ROOT
DISK:[DOWNLOADS.XML.Xerces-C-3_0.]

$ define/system/nolog/trans=concealed XALAN-C$ROOT
DISK:[DOWNLOADS.XML.Xalan-C-3_0.]

Verification of the installation can be performed using the XML-C
Test Procedure.
To run the XML-C Test Procedure, enter the following command:

$ @XML-C$ROOT:[XML-C-3_0]XML-C-3_0-TP
...100%

The following product has been installed:

HP AXPVMS XML_C V3.0 Layered Product

$
```

Installation on an OpenVMS Integrity servers Versions 8.2 and 8.2-1 system:

```
$ PRODUCT INSTALL XML_C /DEST=DISK:[DOWNLOADS]

The following product has been selected:
    HP I64VMS XML_C V3.0                    Layered Product

Do you want to continue? [YES]

Configuration phase starting ...
```

```
You will be asked to choose options, if any, for each selected product and
for any products that may be installed to satisfy software dependency
requirements.

HP I64VMS XML_C V3.0: XML Technology for OpenVMS is based on Apache Xerces
C Version 2.7.0 and Apache Xalan C Version 1.10

    © Copyright 2007 Hewlett-Packard Development Company, L.P.

    Hewlett-Packard Company

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    HP I64VMS XML_C V3.0                  DISK:[DOWNLOADS.]

Portion done: 0%...10%...20%...40%...50%...60%...70%...80%...90%

The supported version of CXX compiler is V7.1 or higher

XML-C$ROOT, XERCES-C$ROOT and XALAN-C$ROOT  have been defined.
The following lines must be added to SYS$MANAGER:SYLOGICALS.COM so
that it will be defined each time the system is rebooted.

$ define/system/nolog/trans=concealed XML-C$ROOT DISK:[DOWNLOADS.XML.]
$ define/system/nolog/trans=concealed XERCES-C$ROOT
DISK:[DOWNLOADS.XML.Xerces-C-3_0.]
$ define/system/nolog/trans=concealed XALAN-C$ROOT
DISK:[DOWNLOADS.XML.Xalan-C-3_0.]

Verification of the installation can be performed using the XML-C
Test Procedure. To run the XML-C Test Procedure, enter the following
command:

    $ @XML-C$ROOT:[XML-C-3_0]XML-C-3_0-TP
...100%

The following product has been installed:
    HP I64VMS XML_C V3.0                  Layered Product
$
```

Installation on an OpenVMS Integrity servers V8.3 system:

```
$ PRODUCT INSTALL XML_C /DEST=DISK:[DOWNLOADS]
Performing product kit validation ...
%PCSI-I-VALPASSED, validation of DISK:[DOWNLOADS]HP-I64VMS-XML_C-V0300--
1.PCSI$COMPRESSED;1 succeeded

The following product has been selected:
    HP I64VMS XML_C V3.0                  Layered Product

Do you want to continue? [YES]
```

```
Configuration phase starting ...

You will be asked to choose options, if any, for each selected product and
for any products that may be installed to satisfy software dependency
requirements.

HP I64VMS XML_C V3.0: XML Technology for OpenVMS is based on Apache Xerces
C Version 2.7.0 and Apache Xalan C Version 1.10

    © Copyright 2007 Hewlett-Packard Development Company, L.P.

    Hewlett-Packard Company

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
          HP I64VMS XML_C V3.0                  DISK:[DOWNLOADS.]

Portion done: 0%...10%...20%...40%...50%...60%...70%...80%...90%

The supported version of CXX compiler is V7.1 or higher

XML-C$ROOT, XERCES-C$ROOT and XALAN-C$ROOT  have been defined.
The following lines must be added to SYS$MANAGER:SYLOGICALS.COM so
that it will be defined each time the system is rebooted.

$ define/system/nolog/trans=concealed XML-C$ROOT DISK:[DOWNLOADS.XML.]
$ define/system/nolog/trans=concealed XERCES-C$ROOT
DISK:[DOWNLOADS.XML.Xerces-C-3_0.]
$ define/system/nolog/trans=concealed XALAN-C$ROOT
DISK:[DOWNLOADS.XML.Xalan-C-3_0.]

Verification of the installation can be performed using the XML-C
Test Procedure. To run the XML-C Test Procedure, enter the following
command:

        $ @XML-C$ROOT:[XML-C-3_0]XML-C-3_0-TP
...100%

The following product has been installed:
    HP I64VMS XML_C V3.0                  Layered Product
$
```

After Installing XML C Technology

After the installation is complete, perform the following tasks.

Add the following lines to SYS$MANAGER:SYLOGICALS.COM so that XML C Technology is defined each time the system is rebooted.

```
$ define/system/nolog/trans=concealed XML-C$ROOT DISK:[DOWNLOADS.XML.]
$ define/system/nolog/trans=concealed XERCES-C$ROOT
DISK:[DOWNLOADS.XML.Xerces-C-3_0.]
```

```
$ define/system/nolog/trans=concealed XALAN-C$ROOT DISK:[DOWNLOADS.XML.Xalan-
C-3_0.]
```

Optionally run the XML C Test Procedure. To run the test procedure, enter the following command:

```
$ @XML-C$ROOT:[XML-C-3_0]XML-C-3_0-TP
```

Interpreting the results of the XML C Test Procedure

The XML Test Procedure compares the output from the tests with a set of benchmarks. Because of the nature of the tests, some differences between the results and the benchmarks are to be expected. The following describes the common differences which are expected to occur.

Many of the tests display the amount of time it takes to parse the document. For example, SAXCount will output:

```
personal.xml: 736 ms (37 elems, 12 attrs, 134 spaces, 134 chars)
```

The values are the parse time, and the count of elements, attributes, ignorable whitespaces, and characters appearing in the document. The parse time will vary based on a number of factors and is not likely to match the time in the benchmark, however, the element counts should match.

Note: The results produced by this and other similar programs should never be accepted as true performance measurements.

Following is a session log containing a run of the XML C Test Procedure followed by a section showing the differences between the run log and supplied benchmark. Your log should look very similar depending on the platform.

```
$ @XML-C$ROOT:[XML-C-3_0]XML-C-3_0-TP

No test specified.  All applicable tests will be run.
Starting Xerces-C tests...
%DCL-S-SPAWNED, process SYSTEM_11010 spawned
%DCL-S-ATTACHED, terminal now attached to process
SYSTEM_11010
%DCL-S-RETURNED, control returned to process _TNA27:
Starting Xalan-C tests...
%DCL-S-SPAWNED, process SYSTEM_56454 spawned
%DCL-S-ATTACHED, terminal now attached to process
SYSTEM_56454
%DCL-S-RETURNED, control returned to process _TNA27:
Tests complete. Check XML-C$ROOT:[XML-C-3_0]XML-C-TP.LOG
for errors.
$type XML-C$ROOT:[XML-C-3_0]XML-C-TP.LOG
************
File XML-C$ROOT:[Xerces-C-3_0.samples.RESULTS]XML-XERCES-TP.OUT;1
6  21
7  ms (
******
File XML-C$ROOT:[Xerces-C-3_0.samples.BENCHMARKS]XML-XERCES-TP.BMK;1
6   12
7   ms (
```

```
************
File XML-C$ROOT:[Xerces-C-3_0.samples.RESULTS]XML-XERCES-TP.OUT;1
20    9
21    ms(
******
File XML-C$ROOT:[Xerces-C-3_0.samples.BENCHMARKS]XML-XERCES-TP.BMK;1
20    11
21    ms (
************
File XML-C$ROOT:[Xerces-C-3_0.samples.RESULTS]XML-XERCES-TP.OUT;1
34   8
35   ms (
******
File XML-C$ROOT:[Xerces-C-3_0.samples.BENCHMARKS]XML-XERCES-TP.BMK;1
34   7
35   ms (
************
File XML-C$ROOT:[Xerces-C-3_0.samples.RESULTS]XML-XERCES-TP.OUT;1
48   11
49   ms (
******
File XML-C$ROOT:[Xerces-C-3_0.samples.BENCHMARKS]XML-XERCES-TP.BMK;1
48   10
49   ms(
************
************
File XML-C$ROOT:[Xerces-C-3_0.samples.RESULTS]XML-XERCES-TP.OUT;1
93    13
94    ms (
******
File XML-C$ROOT:[Xerces-C-3_0.samples.BENCHMARKS]XML-XERCES-TP.BMK;1
93    14
94    ms (
************
************
File XML-C$ROOT:[Xerces-C-3_0.samples.RESULTS]XML-XERCES-
TP.OUT;1
101   14
102  ms(
******
File XML-C$ROOT:[Xerces-C-3_0.samples.BENCHMARKS]XML-XERCES-TP.BMK;1
101   13
102   ms (
```

```
************
************
File XML-C$ROOT:[Xerces-C-3_0.samples.RESULTS]XML-XERCES-TP.OUT;1
117   13
118   ms (
******
File XML-C$ROOT:[Xerces-C-3_0.samples.BENCHMARKS]XML-XERCES-TP.BMK;1
117   12
118   ms (
************
************
File XML-C$ROOT:[Xerces-C-3_0.samples.RESULTS]XML-XERCES-TP.OUT;1
125 14
126 ms (
******
File XML-C$ROOT:[Xerces-C-3_0.samples.BENCHMARKS]XML-XERCES-TP.BMK;1
125  13
126  ms (
************
Number of difference sections found: 8
Number of difference records found: 8
DIFFERENCES /IGNORE=()/MERGED=1-
     XML-C$ROOT:[Xerces-C-3_0.samples.RESULTS]XML-XERCES-TP.OUT;1-
     XML-C$ROOT:[Xerces-C-3_0.samples.BENCHMARKS]XML-XERCES-TP.BMK;1
Number of difference sections found: 0
Number of difference records found: 0
 DIFFERENCES /IGNORE=()/MERGED=1-
     XML-C$ROOT:[Xalan-C-3_0.c.samples.RESULTS]XML-XALAN-TP.OUT;1-
     XML-C$ROOT:[Xalan-C-3_0.c.samples.BENCHMARKS]XML-XALAN-TP.BMK;1
     $
In order to build and run all the samples provided, refer to
XML-C$ROOT:[XML-C-3_0]BuildAllSamples_Readme.txt

For documentation on XML-C, please access the Apache website at the following
URLs:
   - http://xml.apache.org/xerces-c/
   - http://xml.apache.org/xalan-c/
```

Installing the Sources and Documentation

The sources and documentation are provided in the form of a backup savesets. The "sources" restore operation requires approximately 43,617 blocks of disk space for OpenVMS Alpha and 124,190 blocks of disk space for OpenVMS Integrity servers, and the "documentation" restore operation requires approximately 53,264 blocks of disk space. Execute the following commands to restore the documentation and source files.

```
$ SET DEF XML-C$ROOT:[000000]
$ @XMLC_RESTORE_BACKUPS.COM

*** THIS PROCEDURE WILL LET YOU RESTORE DOCUMENTATION OR SOURCE OR BOTH
ON TO XML-C$ROOT:[000000…] ***

   1. RESTORE DOCUMENTATION ONLY
   2. RESTORE SOURCES ONLY
   3. RESTORE DOCUMENTATION AND SOURCES
   4. EXIT

TYPE 1 OR 2 OR 3 OR 4 : 1

BACKUP RESTORE OPERATION STARTS.....

DOCUMENTATION IS RESTORED.
$ @XMLC_RESTORE_BACKUPS.COM
*** THIS PROCEDURE WILL LET YOU RESTORE DOCUMENTATION OR SOURCE OR BOTH
ON TO XML-C$ROOT:[000000…] ***

   1. RESTORE DOCUMENTATION ONLY
   2. RESTORE SOURCES ONLY
   3. RESTORE DOCUMENTATION AND SOURCES
   4. EXIT

TYPE 1 OR 2 OR 3 OR 4 : 2

BACKUP RESTORE OPERATION STARTS.....

SOURCE IS RESTORED
$ @XMLC_RESTORE_BACKUPS.COM
*** THIS PROCEDURE WILL LET YOU RESTORE DOCUMENTATION OR SOURCE OR BOTH
ON TO XML-C$ROOT:[000000…] ***

   1. RESTORE DOCUMENTATION ONLY
   2. RESTORE SOURCES ONLY
   3. RESTORE DOCUMENTATION AND SOURCES
   4. EXIT

TYPE 1 OR 2 OR 3 OR 4 : 3
BACKUP RESTORE OPERATION STARTS.....
```

```
DOCUMENTATION AND SOURCES ARE RESTORED
$ @XMLC_RESTORE_BACKUPS.COM
*** THIS PROCEDURE WILL LET YOU RESTORE DOCUMENTATION OR SOURCE OR BOTH
ON TO XML-C$ROOT:[000000…] ***

   1. RESTORE DOCUMENTATION ONLY
   2. RESTORE SOURCES ONLY
   3. RESTORE DOCUMENTATION AND SOURCES
   4. EXIT

TYPE 1 OR 2 OR 3 OR 4 : 4

$
```

Removing XML C Technology for OpenVMS

To remove XML C Technology, execute the following command:

On OpenVMS Alpha:

```
$ PRODUCT REMOVE XML_C

  The following output is then displayed:

  The following product has been selected:
  HP AXPVMS XML_C V3.0                    Layered Product

  Do you want to continue? [YES]

  The following product will be removed from destination:

  HP AXPVMS XML_C V3.0                    DISK:[DOWNLOADS.]

 Portion done:
 0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%

 The following product has been removed:
 HP AXPVMS XML_C V3.0                     Layered Product

 $
```

On OpenVMS Integrity servers:

```
$ PRODUCT REMOVE XML_C

  The following output is then displayed:

  The following product has been selected:
  HP I64VMS XML_C V3.0                    Layered Product

  Do you want to continue? [YES]
```

```
The following product will be removed from destination:
HP I64VMS XML_C V3.0                    DISK:[DOWNLOADS.]

Portion done:
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%

The following product has been removed:
HP I64VMS XML_C V3.0                    Layered Product

$
```

Creating Applications using XML-C Version 3.0 on OpenVMS Alpha and Integrity servers

Setting up the Environment

To set up the environment for creating Xerces C and Xalan C applications, perform the following steps:

1. To extract the sources, refer to the Installing the Sources and Documentation section in this document.
2. Define XERCESC as a rooted logical name to point to the top level Xerces directory as follows:

   ```
   $ DEFINE/JOB XERCESC XERCES-C$ROOT:[SRC.XERCESC]
   ```

   This is required for including the XERCES C header files.
3. Define XALANC as a rooted logical name to point to the top level Xalan directory as follows:

   ```
   $ DEFINE/JOB XALANC XALAN-C$ROOT:[C.SRC.XALANC]
   ```

   This is required for including the XALAN C header files.
4. Define the DECC feature logical DECC$FILENAME_UNIX_REPORT as follows:

   ```
   $ DEFINE/JOB DECC$FILENAME_UNIX_REPORT   ENABLE
   ```
5. If you are using shareable images, the logical names for XERCES_SHR and XALAN_SHR must be defined as follows:

   For OpenVMS Alpha:

   ```
   $ DEFINE/JOB XERCES_SHR XERCES-C$ROOT:[BIN]XERCES_SHR.EXE_ALPHA
   $ DEFINE/JOB XML_SHR XALAN-C$ROOT:[C.BIN]XML_SHR.EXE_ALPHA
   ```

   For OpenVMS Integrity servers:

   ```
   $ DEFINE/JOB XERCES_SHR XERCES-C$ROOT:[BIN]XERCES_SHR.EXE_I64
   $ DEFINE/JOB XML_SHR XALAN-C$ROOT:[C.BIN]XML_SHR.EXE_I64
   ```

Creating Xerces C Applications

Independent of the API you want to use, DOM, SAX, or SAX2; your application must initialize the Xerces system before using the API, and terminate it after you are done. This is achieved by the following code:

```
#include <xercesc/util/PlatformUtils.hpp>
// Other include files, declarations, and non-Xerces-C++ initializations.
XERCES_CPP_NAMESPACE_USE

int main(int argc, char* argv[])
{
  try {
    XMLPlatformUtils::Initialize();
  }
  catch (const XMLException& toCatch) {
    // Do your failure processing here
    return 1;
  }

  // Do your actual work with Xerces-C++ here.

  XMLPlatformUtils::Terminate();

  // Other terminations and cleanup.
  return 0;
}
```

`XMLPlatformUtils::Initialize()` and `XMLPlatformUtils::Terminate` must be called at least once in each process. You are allowed to call `XMLPlatformUtils::Initialize()` and `XMLPlatformUtils::Terminate` multiple times, but each call to `XMLPlatformUtils::Initialize()` must be matched with a call to `XMLPlatformUtils::Terminate`.

For details on the DOM API, refer to the DOM Programming Guide:
http://xml.apache.org/xerces-c/program-dom.html

For details on the SAX API, refer to the SAX Programming Guide:
http://xml.apache.org/xerces-c/program-sax.html

For details on the SAX2 API, refer to the SAX2 Programming Guide:
http://xml.apache.org/xerces-c/program-sax2.html

Compiling and Linking Xerces C Applications

- Compile Xerces C applications on OpenVMS using the following CXX compiler qualifiers:
  - /NOMEMBER_ALIGNMENT
  - /ASSUME=NOALIGNED_OBJECTS
  - /DEFINE=__USE_STD_IOSTREAM

- Statically link Xerces C applications with XERCES-C$ROOT:[LIB]XERCES.OLB

- Dynamically link Xerces C applications with the Xerces C shareable image, refer to XML-C$ROOT:[XML-C-3_0] create_shareable_readme.txt

Creating Xalan C Applications

To perform a transformation, use one of the XalanTransformer Transform() methods. The transformation requires an XML source document and an XSL stylesheet. Both of these objects may be represented by instances of XSLTInputSource You can construct an XSLTInputSource with a string (the system ID for a file or URI), an input stream, or a DOM.

If you are using an XSL stylesheet to perform a series of transformations, you can improve performance by calling transform() with a compiled stylesheet, an instance of XalanCompiledStylesheet. If you are transforming an XML source more than once, you should call transform() with a parsed XML source, an instance of XalanParsedSource.

If your XML source document contains a stylesheet Processing Instruction (PI), you do not need to include a stylesheet object when you call transform().

The transformation output is represented by an XSLTResultTarget, which you can set up to refer to an output stream, the system ID for a file or URI, or a Formatter for one of the various styles of DOM output.

For detailed Xalan API documentation, refer to Xalan-C++ API.

Using XalanTransformer and the C++ API, you can perform one or more transformations as described in the following steps.

Note: For a working sample that illustrates these steps, see the XalanTransform sample.

Step1: Include the required header files

1. Always start with xalanc/Include/PlatformDefinitions.hpp, the Xalan-C++ base header file. Also include xercesc/util/PlatformUtils.hpp, xalanc/XalanTransformer/XalanTransformer.hpp, and any other header files your particular application requires.

```
#include <xalanc/Include/PlatformDefinitions.hpp>
#include <xercesc/util/PlatformUtils.hpp>
#include <xalanc/XalanTransformer/XalanTransformer.hpp>
...
```

Step 2: Define Namespaces

As of version 1.5, Xalan-C++ now uses C++ namespaces for those platforms which support them. A number of macros are provided to make using the Xalan-C++ (and Xerces-C++) namespaces easier.

In the following example, the XALAN_USING_XERCES and XALAN_USING_XALAN macros are used to declare that the program is using XMLPlatformUtils and XalanTransformer from the Xerces-C++ and Xalan-C++ namespaces respectively.

```
XALAN_USING_XERCES(XMLPlatformUtils)
XALAN_USING_XALAN(XalanTransformer)
```

These macros can be used immediately after the included header files (for global applicability in a given source file) or within functions and methods for local applicability.

Note: You can use the standard C++ namespace syntax directly, the Xerces-C++ and Xalan-C++ namespaces are linked to the version number. For example, the Xalan namespace is currently

xalanc_1_9. The macros will automatically take care of this when code is re-compiled against a new version of the libraries. Using the namespaces directly will require each namespace related statement be updated by hand.

Step 3: Initialize Xerces and Xalan

Use the static initializes to initialize the Xalan-C++ and Xerces-C++ platforms. You must initialize Xerces-C++ once per process. You may initialize and terminate Xalan-C++ multiple times, but this is not recommended: it is inefficient and is not thread safe.

```
XMLPlatformUtils::Initialize();
XalanTransformer::initialize();
```

Step 4: Create a XalanTransformer

```
XalanTransformer theXalanTransformer;
```

Step 5: Perform each transformation

You can explicitly instantiate XSLTInputSource objects for the XML source document and XSL stylesheet, and an XSLTResultTarget object for the output, and then call XalanTransformer transform() with those objects as parameters. For example:

```
XSLTInputSource xmlIn("foo.xml");
XSLTInputSource xslIn("foo.xsl");
XSLTResultTarget xmlOut("foo-out.xml");
int theResult = theXalanTransformer.transform(xmlIn,xslIn,xmlOut)
```

Alternatively, you can call transform() with the strings (system identifiers), streams, and/or DOMs that the compiler needs to implicitly construct the XSLTInputSource and XSLTResultTarget objects. For example:

```
const char* xmlIn = "foo.xml";
const char* xslIn = "foo.xsl";
const char* xmlOut = "foo-out.xml";
int theResult = theXalanTransformer.transform(xmlIn,xslIn,xmlOut)
```

Note that XSLTInputSource and XSLTResultTarget provide a variety of single-argument constructors that you can use in this manner:

- `XSLTInputSource(const char* systemID);`
- `XSLTInputSource(const XMLCh* systemID);//Unicode chars`
- `XSLTInputSource(istream* stream);`
- `XSLTInputSource(XalanNode* node);`

- `XSLTResultTarget(char* fileName);`
- `XSLTResultTarget(XalanDOMString& fileName);`
- `XSLTResultTarget(ostream* stream);`
- `XSLTResultTarget(ostream& stream);`
- `XSLTResultTarget(Writer* characterStream);`
- `XSLTResultTarget(XalanDocument* document);`
- `XSLTResultTarget(XalanDocumentFragment* documentFragment);`
- `XSLTResultTarget(XalanElement* element);`
- `XSLTResultTarget(FormatterListener& flistener);`

Note: Each transform() method returns an integer code, 0 for success. If an error occurs, you can use the getLastError() method to return a pointer to the error message.

Step 6: Shut down Xalan

When you shut down Xalan, you may also want to shut down Xerces and ICU support (if enabled). Keep the following considerations in mind:

- Once you have shut down Xerces, you can no longer make Xalan or Xerces calls in the current process.
- Ensure that there are no Xalan-C++ or Xerces-C++ objects extant at the point of termination. Any deletion of objects after termination could cause errors.

- Use the static terminators.

```
XalanTransformer::terminate();
XMLPlatformUtils::Terminate();
XalanTransformer::ICUCleanUp();
```

For more information, refer to http://xml.apache.org/xalan-c/usagepatterns.html

Compiling and Linking Xalan C Applications

- Compile Xalan C applications on OpenVMS using the following CXX compiler qualifiers:
    - /NOMEMBER_ALIGNMENT
    - /ASSUME=NOALIGNED_OBJECTS
    - /DEFINE=__USE_STD_IOSTREAM

- Statically link the Xalan C applications with XALAN-C$ROOT:[C.LIB]XALAN.OLB and XERCES-C$ROOT:[LIB]XERCES.OLB

- Dynamically link the Xalan C applications with Xalan C and Xerces C shareable image, refer to XML-C$ROOT:[XML-C-3_0] create_shareable_readme.txt

Building the XML C Sources

Building the XML-C sources is not required unless you plan to make modifications to the sources. Optionally, you can build a shareable image from the object files shipped in the object library.

If you do plan to modify the sources, it is recommended that you perform a full build of the existing code before you make any modifications. A full build will likely require several hours and is best submitted as a batch job to run overnight. To submit the build, modify the main build procedure BUILD_XML-C-3_0.COM found in XML-C$ROOT:[XML-C-3_0] to specify the batch queue to use (this procedure will submit additional procedures). Then submit the main procedure using the command:

```
$ SUBMIT/QUEUE=<your-batch-queue> XML-C$ROOT:[XML-C-3_0]BUILD_XML-C-3_0
```

Further details are provided in the BUILD_XML-C-3_0.COM procedure.

You can also invoke the main build procedure interactively using the following command, in which case all build procedures will be run interactively:

```
$ @XML-C$ROOT:[XML-C-3_0]BUILD_XML-C-3_0
```

However, based on the length of time required, this method is not recommended. Using a batch job also has the added advantage of providing log files of any errors that may occur.

Note: XALAN-C$ROOT and XERCES-C$ROOT must be defined in order for the build procedures to work. These logicals are defined by the XML-C installation procedure; however they are defined only on the node in which XML-C was installed, not on all nodes in a cluster. Be sure they are defined on the node which will be executing the batch jobs. You should either manually define them on the nodes on which you will be executing the batch jobs (assuming they point to a cluster accessible disk), or you should restrict which nodes execute the batch jobs.

Partial Builds

Once the full build has completed, subsequent builds will only recompile the sources that you modify or any sources that failed to produce an object file in the previous build. The determination as to which sources to build is based on modification dates of the sources and objects, so modifying a source will mean that the source is newer than the object and cause that object to be rebuilt. Note that if you then delete the modified source in an attempt to restore the original version, you must also delete the corresponding object to force the object to be rebuilt.

Note that modifying the source will also cause shareable images to be rebuilt, which can also be a lengthy process. It may be possible to modify the build procedure to reduce the length of time it takes to create the shareable image when only minor modifications to the sources are being made. Doing so is beyond the scope of this document. It is recommended that you disable the creation of the shareable image and/or the related options files and link against the object libraries until you are satisfied with the source changes.

Additional Parameters

You can force a full build using the CLEAN parameter. When specified, all objects, libraries, and shareables are deleted, then all sources are compiled and the object libraries and shareable images will be recreated. Samples images will be relinked.

You can compile and link with the debugger using the DEBUG parameter. When specified, any sources that are compiled will be compiled with the /DEBUG/NOOPTIMIZE qualifiers. Any images that are relinked will be relinked /DEBUG.

For example:

```
$ SUBMIT/QUEUE=MY_QUEUE/PARAM=CLEAN XML-C$ROOT:[XML-C-3_0]BUILD_XML-C-3_0
$ SUBMIT/QUEUE=MY_QUEUE/PARAM=(CLEAN,DEBUG) -
_$ XML-C$ROOT:[XML-C-3_0]BUILD_XML-C-3_0
```

Additional Build Options

There are additional build options contained within the XALAN-C$ROOT:[000000]BUILD_XALAN.COM and XERCES-C$ROOT:[000000]BUILD_XERCES.COM procedures which will control the actions of the procedures and the amount of information output. These are described in the procedures themselves.

Shareble Images

For information about creating Xerces and Xalan shareable images, refer to:

```
XML-C$ROOT:[XML-C-3_0]CREATE_SHAREABLE_README.TXT
```

XERCES_SHR contains only Xerces code. XML_SHR contains both Xerces and Xalan code. To link against these shareables, specify /OPTIONS on the LINK command, and specify the Xerces shareable file spec with the /SHARE qualifier when working with Xerces code. When working with Xalan code, specify both the Xerces and XML shareable file specifications with the /SHARE qualifier.

For example:

On OpenVMS Alpha:

```
$ LINK -
.
<objects>,-
.
SYS$INPUT/OPTIONS,-
XERCES-C$ROOT:[BIN]XERCES_SHR.EXE_ALPHA/SHARE
XALAN-C$ROOT:[C.BIN]XML_SHR.EXE_ALPHA/SHARE
```

On OpenVMS Integrity servers:

```
$ LINK -
.
<objects>,-
.
SYS$INPUT/OPTIONS,-
XERCES-C$ROOT:[BIN]XERCES_SHR.EXE_I64/SHARE
XALAN-C$ROOT:[C.BIN]XML_SHR.EXE_I64/SHARE
```

To use the shareable images, you must define XERCES_SHR and XML_SHR.

For example:

On OpenVMS Alpha:

```
$ DEFINE/JOB XERCES_SHR XERCES-C$ROOT:[BIN]XERCES_SHR.EXE_ALPHA
$ DEFINE/JOB XML_SHR XALAN-C$ROOT:[C.BIN]XML_SHR.EXE_ALPHA
```

On OpenVMS Integrity servers:

```
$ DEFINE/JOB XERCES_SHR XERCES-C$ROOT:[BIN]XERCES_SHR.EXE_I64
$ DEFINE/JOB XML_SHR XALAN-C$ROOT:[C.BIN]XML_SHR.EXE_I64
```

Note: Using shareable images typically provides several advantages. For example, your images will be smaller than if you link them against the object libraries. They also provide entry vectors which typically do not change from build to build. This would normally mean that you would not need to relink your images for a subsequent release of XML-C. However, because XML-C is produced from open source code, and because of the large number of entry points in the shareables, HP cannot guarantee that you would not need to relink your images.

Object Libraries

Two object libraries are provided:

```
XERCES-C$ROOT:[LIB]XERCES.OLB
XALAN-C$ROOT:[C.LIB]XALAN.OLB
```

If you choose to link against these object libraries, you should specify XERCES.OLB when working with Xerces code and both XERCES.OLB and XALAN.OLB when working with Xalan code. For example:

```
$ LINK -
     .
     <objects>,-
     .
XERCES-C$ROOT:[LIB]XERCES.OLB/LIB,-
XALAN-C$ROOT:[C.LIB]XALAN.OLB/LIB
```

Sample Images

Sample images for Xerces and Xalan are provided in the XERCES-C$ROOT:[BIN] and XALAN-C$ROOT:[C.BIN] directories. These images have been linked against the object libraries. If desired, they can be linked against the sharable images by changing the LINK_W_SHARE option in the build procedures. These images are executed as part of the XML-C$ROOT:[XML-C-3_0]XML-C-3_0-TP.COM procedure. The sources for these images can be found in XERCES-C$ROOT:[SAMPLES...] and XALAN-C$ROOT:[C.SAMPLES...].

Release Notes

XML C Version 3.0 for OpenVMS Alpha

The following are the changes from Version 2.0-1 to Version 3.0.

- The Xerces public APIs have changed significantly between V2.0-1 (based on Apache Xerces 2.2.0) and V3.0 (based on Apache Xerces 2.7.0).

  There have been a lot of changes to the Public API methods. Existing V2.0-1 Xerces applications MAY NOT be compatible with V3.0. Check the following links for changes to Xerces Public APIs:

  Migration information for Xerces 2.7.0 –
  http://xml.apache.org/xerces-c/migrate.html

  Migration information for changes between 2.2.0 to 2.6.0 –
  http://xml.apache.org/xerces-c/migrate_archive.html

- The Xalan public APIs have changed significantly between V2.0-1 (based on Apache Xalan 1.5) and V3.0 (based on Apache Xalan 1.10).

  There have been a lot of changes to the Public API methods. Existing V2.0-1 Xalan applications MAY NOT be compatible with V3.0. Check the following links for changes to Xalan Public APIs:

  Migration information for Xalan 1.10 -
  http://xml.apache.org/xalan-c/whatsnew.html

  Migration information for changes between Xalan 1.5 and 1.9 -
  http://xml.apache.org/xalan-c/whatsnew.html#history

XML C Version 3.0 for OpenVMS Integrity servers

The following are the changes from Version 2.0-1 to Version 3.0.

- This kit now ships separate Xerces and Xalan object libraries (.OLB) files, and options (.OPT) files for CXX V7.1 compiler and CXX V7.2 compiler on OpenVMS Integrity servers.

  The files for the CXX V7.1 compiler are as follows:

  XERCES-C$ROOT:[LIB]XERCES.OLB
  XALAN-C$ROOT:[C.LIB]XALAN.OLB
  XERCES-C$ROOT:[BIN]XERCES.OPT
  XALAN-C$ROOT:[C.BIN]XML.OPT

  The files for the CXX V7.2 compiler are as follows:

  XERCES-C$ROOT:[LIB]V72_XERCES.OLB
  XALAN-C$ROOT:[C.LIB]V72_XALAN.OLB
  XERCES-C$ROOT:[BIN]V72_XERCES.OPT
  XALAN-C$ROOT:[C.BIN]V72_XML.OPT

- Source build for XML-C Version 3.0 on OpenVMS Integrity servers Version 8.2-1 may fail because certain versions of CRTL have a definition for the routine "realpath" in the header file “stdlib.h”. This routine is not implemented on OpenVMS Integrity servers Version 8.2-1 and is only available on OpenVMS Integrity servers Version 8.3.

  Note: XML-C applications can be built on OpenVMS Integrity servers Version 8.2-1 using the object libraries that are shipped with the kit. This problem is seen only when the Xerces C and Xalan C source code is built on OpenVMS Integrity servers Version 8.2-1.

- The Xerces public APIs have changed significantly between V2.0-1 (based on Apache Xerces 2.2.0) and V3.0 (based on Apache Xerces 2.7.0).

  There have been a lot of changes to the Public API methods. Existing V2.0-1 Xerces applications MAY NOT be compatible with V3.0. Check the following links for changes to Xerces Public APIs:

  Migration information for Xerces 2.7.0 –
  http://xml.apache.org/xerces-c/migrate.html

  Migration information for changes between 2.2.0 to 2.6.0 –
  http://xml.apache.org/xerces-c/migrate_archive.html

- The Xalan public APIs have changed significantly between V2.0-1 (based on Apache Xalan 1.5) and V3.0 (based on Apache Xalan 1.10).

  There have been a lot of changes to the Public API methods. Existing V2.0-1 Xalan applications MAY NOT be compatible with V3.0. Check the following links for changes to Xalan Public APIs:

  Migration information for Xalan 1.10 -
  http://xml.apache.org/xalan-c/whatsnew.html

  Migration information for changes between Xalan 1.5 and 1.9 -
  http://xml.apache.org/xalan-c/whatsnew.html#history

XML C Version 2.0-1

The following are the changes from Version 2.0 to Version 2.0-1.

- The XML C documentation has been removed from the kit. For documentation on XML C, see the following URLs:

  http://xml.apache.org/xerces-c/
  http://xml.apache.org/xalan-c/

- Separate kits are available for Alpha and Integrity server platforms.
- This kit now ships Xerces and Xalan object libraries (.OLB) instead of the shareable images.

  See the file create_shareable_readme.txt if you want to build Xerces and Xalan shareable images.

- Two Xerces samples and one Xalan sample are included in the kit.

  See the file BuildAllSamples_Readme.txt if you want to build all samples.

- Removed requirement for C compiler.
- Removed requirement for the HP Secure Web Server for OpenVMS.

Support

Version 3.0 Customer Release

Please see the XML Technology Support Page for support information.

If you do not have a support contract and are not interested in acquiring one, you can informally exchange information with other users in the OpenVMS newsgroup `comp.os.vms`.

For technical feedback to the XML C Technology for OpenVMS engineering team, please send mail to *OpenVMS.eBusiness@hp.com.*