Transaction Processing Facility

XML User's Guide

Version 4 Release 1

GTPX-1MST-00

Contents

Tables v

About This Guide vii

The Basics 1
Why XML? 1
XML Support on TPF 2
Character Encodings 2
Learn XML 4

The Parser 7
Application Programming 7
InputSource: Where Is the XML Document? 8
Document Type Definitions (DTDs): Defining the Rules 9
XML Schema: Another Way to Define Rules 9
Namespaces: A Two-Part Naming System 10
For More Information. 10
DOM 10
How Do I Use DOM? 10
When Should I Use DOM? 10
For More Information 11
SAX 11
How Do I Use SAX? 11
When Should I Use SAX? 11
For More Information. 12
DOM versus SAX 12
Validation 13
When Should I Use Validation? 13
How Do I Use Validation? 13
Results of Validation 13
For More Information. 13

Examples 15
XML on TPF: A Short Tutorial 15
Samples from the Source: XML4C 15

Supplements 21
Glossary 21
Migration Information. 21
XML4C Parser 3.5.1 (APAR PJ28176) 22
Prerequisite APARs 22
Functional Overview 22
Architecture 23
Operating Environment Requirements and Planning Information 23
Interface Changes. 23
Functional and Operational Changes 25
Performance or Tuning Changes 25
Storage Considerations and Changes 25
System Initialization Program (SIP) and System Generation Changes . . . 25
Loading Process Changes. 25
Online System Load Changes 25
Publication Changes 25
Host System Changes 26

| Application Programming Interface (API) Changes 26
| Database Changes 26
| Feature Changes 26
| Installation Validation. 26
| Migration Scenarios 26
Resources 27
XML4C Version 3.5.1 Documentation. 28

Tables

1. XML Character Encodings Supported on TPF 3
2. Comparing the DOM API with the SAX API 12
3. Summary of Expected Validation Results 13
4. Changes to Link-Edited Modules for XML4C Parser 3.5.1 24
5. Changes to TPF Publications for XML4C Parser 3.5.1 25

About This Guide

This guide is intended for TPF application programmers who already understand some general TPF programming concepts, and who will be programming in C++ language to access XML data. Users of this guide are expected to be C/C++ programmers who have some familiarity with TPF and a markup language such as HTML or BookMaster.

This guide contains the following:

- Get Started With "The Basics" on page 1
  - Learn how XML can be beneficial in the process of programming TPF applications.
  - Read detailed information about XML support on TPF.
  - Find additional resources for learning the XML language.
- Understand How To Use "The Parser" on page 7 on TPF
  - Learn details about using XML data when programming applications on TPF that access XML data.
  - Learn about the DOM and SAX specifications.
  - Understand validation.
- Apply What You Have Learned Through "Examples" on page 15
  - Take an online tutorial that walks you through the development of a sample TPF application that interacts with XML data.
  - Experiment with sample programs that came with the ported XML4C code.
- Clarify and Research More XML Information in "Supplements" on page 21
  - Understand terminology used throughout the guide.
  - Read Migration Information for XML4C parser 3.5.1 (APAR PJ28176).
  - Find out where you can learn more about XML and all of its components.
  - Open the documentation that came with the ported XML4C code.

The Basics

Before you can start programming TPF applications that use the XML parser, you need to understand some XML basics.

This section contains the following:

"Why XML?"
Understanding how you can benefit from using XML on TPF will give you insight into how you design your XML applications.

"XML Support on TPF" on page 2
As an evolving technology, XML is constantly growing and changing, so understanding what is supported on TPF (and what is not) is critical to your application programming.

"Learn XML" on page 4
There are many pieces needed to complete the XML puzzle. Before you begin writing TPF applications that interact with XML data, you will need to understand these pieces of XML.

Why XML?

XML allows you to tag data in a way that is similar to how you tag data when creating an HTML file. XML incorporates many of the successful features of HTML, but was also developed to address some of the limitations of HTML. XML tags are actually user-defined through a schema, which can either be a Document Type Definition (DTD) or a document written in the XML Schema language. In addition, namespaces can help ensure you have unique tags for your XML document. The syntax of XML has more restrictions than HTML, but this results in faster and cheaper browsing. The ability to create your own tagging structure gives you the power to categorize and structure data for both ease of retrieval and ease of display. XML is already being used for publishing, as well as for data storage and retrieval, data interchange between heterogeneous platforms, data transformations, and data displays. As it evolves and becomes more powerful, XML may allow for single-source data retrieval ***and*** data display.

XML4C parser 3.5.1 (APAR PJ28176) is a port of XML Parser for C++ (XML4C) version 3.5.1. With this APAR installed, applications on your TPF 4.1 system can interact with tagged XML data.

The benefits of using XML vary but, overall, marked-up data and the ability to read and interpret that data provide the following benefits

- With XML, TPF applications can more easily read information from a variety of platforms. The data is platform-independent, so now the sharing of data between you and your customers can be simplified.
- Companies that work in the business-to-business (B2B) environment are developing DTDs for their industry. The ability to parse XML documents gives TPF an opportunity to be exploited in the B2B environment.
- XML data can be read even if you do not have a detailed picture of how that data is structured. Your clients will no longer need to go through complex processes to update how to interpret data that you send to them because the DTD gives the ability to understand the information.

- Changing the content and structure of data is easier with XML. The data is tagged so you can add and remove elements without impacting existing elements. You will be able to change the data without having to change the application.

However, despite all the benefits of using XML, there are some things to be aware of. First of all, working with marked up data can be additional work when writing applications because it physically requires more pieces to work together. (Go to "Learn XML" on page 4 for more information.) Given the benefits of using XML, this additional work up front can reduce the amount of work needed to make a change in the future. Second, although it is a recommendation developed by the World Wide Web Consortium (W3C), XML is still a developing technology.

There are many resources available for learning about XML, some of which are included in "Resources" on page 27.

XML Support on TPF

XML4C parser 3.5.1 (APAR PJ28176) is a port of XML Parser for C++ (XML4C) Version 3.5.1 to the TPF 4.1 system. The parser is XML Version 1.0 compliant and allows TPF 4.1 applications written in C++ language to do the following:

- Parse XML documents using the Document Object Model (DOM) Level 1.0 or 2.0. You can also parse XML documents using the experimental IDOM API, but this is not formally supported by the XML4C parser and, therefore, not formally supported on TPF.
- Parse XML documents using the Simple API for XML (SAX) Version 1.0 or 2.0 specification.
- Parse XML documents with or without validation against a specified Document Type Definition (DTD).
- Parse XML documents with or without validation against a document written in the XML Schema language.**Note:** XML Schema support is experimental and only includes a subset of the W3C Schema language.

In addition, the parser fully implements the ability to use namespaces in support of unique tagging structures.

IBM contributed the XML4C parser to the Apache XML Project (http://xml.apache.org) as open source in November 1999. XML4C Version 3.5.1 is based on Xerces-C Version 1.5.0 and is fully compliant with the Unicode 3.0 specification. While the Apache Xerces-C parser can be updated by the open source community, the XML4C parser is maintained only by IBM and may differ from Xerces-C.

- For more information about Unicode specifications, go to http://www.unicode.org.
- For more information about XML and the DOM specification, go to http://www.w3.org/.
- For more information about the SAX specification, go to http://www.megginson.com.

Character Encodings

XML documents, DTDs, and XML Schema documents require that you declare which version of XML you are using as well as what encoding you are using. This declaration is done in the first line and is similar to the following: `<?xml version="1.0" encoding="ISO-8859-1"?>`

Note: In general, parsers often have the ability to auto-detect certain encodings. When using this version of the XML4C parser, you do not need to specify the encoding when your documents are written in either UTF-8, UTF-16 Little Endian, or UTF-16 Big Endian.

The following table shows which character encodings are supported on TPF. The first column indicates the encoding and the second column lists common names associated with that encoding. The third column shows acceptable values for the `encoding=` portion of the XML declaration. The fourth column indicates if the encoding is supported on TPF, and the last column indicates if the encoding is supported in XML4C version 3.5.1. Note that some encodings supported in XML4C version 3.5.1 are ***not*** supported on TPF and some encodings supported on TPF are not supported in XML4C version 3.5.1.

Table 1. XML Character Encodings Supported on TPF

Encoding	Common Name	Declaration (`encoding=`)	Supported on TPF 4.1	Supported in XML4C

ASCII		`US-ASCII`
		`USASCII`
		`ASCII`
		`US_ASCII`
			X	X

IBM037 [1]	EBCDIC US	`EBCDIC-CP-US` `IBM037`	X	X
IBM500 [1]		`IBM-500`	X	
IBM1047 [1] [2]		`IBM-1047`	X	
IBM1140 [1]	EBCDIC with Euro symbol	`IBM1140`	X	X

ISO-8859-1	ISO Latin 1	`ISO8859-1`
		`ISO-8859-1`
		`ISO_8859-1`
		`IBM-819`
		`IBM819`
		`LATIN1`
		`LATIN-1`
		`LATIN_1`
			X	X

UTF-8	8-bit Unicode	`UTF-8` `UTF8`	X	X

UTF-16 Little Endian		`UTF-16 (LE)`
		`UTF-16LE`
		`UTF-16`
		`UCS2`
		`IBM1200`
		`IBM-1200`
			X	X

UTF-16 Big Endian		`UTF-16 (BE)`
		`UTF-16BE`
		`UTF-16`
		`UCS2`
		`IBM1200`
		`IBM-1200`
			X	X

UCS4 Little Endian		`UCS-4 (LE)`
		`UCS-4LE`
		`UCS4`
		`UCS-4`
		`UCS_4`
			X	X

Table 1. XML Character Encodings Supported on TPF (continued)

Encoding	Common Name	Declaration (`encoding=`)	Supported on TPF 4.1	Supported in XML4C

UCS4 Big Endian		`UCS-4 (BE)`
		`UCS-4BE`
		`UCS4`
		`UCS-4`
		`UCS_4`
			X	X

Windows-1252		`WINDOWS-1252`	X	X
Big5	Chinese, Big5			X
euc-kr	Korean, Extended UNIX code			X
gb2312	Chinese, PRC			X
ISO-8859-2	ISO Latin 2			X
ISO-8859-3	ISO Latin 3			X
ISO-8859-4	ISO Latin 4			X
ISO-8859-5	ISO Latin Cyrillic			X
ISO-8859-6	ISO Latin Arabic			X
ISO-8859-7	ISO Latin Greek			X
ISO-8859-8	ISO Latin Hebrew			X
ISO-8859-9	ISO Latin 5			X
koi8-r	Cyrillic			X
Shift_JIS	Japanese, Shift JIS			X

Notes:

1. This encoding is an EBCDIC code page.
2. IBM1047 is the code page used by the C language compiler for TPF.

How satisfied are you with this encoding support? If you would like support for additional encodings that are not currently supported on TPF, contact your TPF service representative to open a requirement or enhancement request.

Learn XML

Learning the basics about XML is important in the successful use of XML on TPF. Because XML is an evolving technology, any XML education that could be provided here would most likely be out of date within a few months. In addition, there are many resources available for learning the detailed syntax of XML documents, DTDs, and the XML Schema language. To find these resources, do a search on any Internet search engine or browse the technology section of your local book store. You will find many books, Web sites, and tools that will assist you in writing XML documents. We have included a few book titles and Web pages in “Resources” on page 27 that we found useful while porting the XML4C parser to TPF. The list is by no means exhaustive, but provides you with a few places to start.

The following list identifies the core pieces of XML that you should understand before writing applications that use XML4C parser 3.5.1 on TPF:

- XML is Unicode 3.0 compliant.
- A schema is used to define the tags and structure of an XML document. Two types of schema are (1) XML Schema ; and (2) DTD.

  Note: XML Schema support in XML4C parser 3.5.1 is limited to a subset of the W3C XML Schema language and is considered experimental at this time. For more information about XML Schema support in this parser, go to "XML4C Version 3.5.1 Documentation" on page 28.
- An XML namespace is simply a two-part naming system used for qualifying element and attribute names used in an XML document.
- XML documents must be well-formed and may be valid based on an associated DTD. Depending on the encoding used, you may be able to open your XML document in Microsoft Internet Explorer Version 5.0 or later to see if it is well-formed.
- XML documents contain elements and attributes.
- The parser specifications supported on TPF are DOM (versions 1.0 and 2.0) and SAX (versions 1.0 and 2.0). (The experimental IDOM API is also available, but is not supported for production work.)
- On TPF, the schema (DTD or XML Schema) must reside either in the XML document or in the TPF file system. An XML document can reside in the file system, reside in memory, or be passed through standard input (stdin). See the following TPF books for more information about the TPF file system:
  - *TPF Application Programming*
  - *TPF C/C++ Language Support User's Guide*
  - *TPF Concepts and Structures*
  - *TPF Operations*

"XML on TPF: A Short Tutorial" on page 15 explores many of these topics and provides you with a sample application that interacts with XML data.

The Parser

After gaining an understanding of the XML pieces identified in “Learn XML” on page 4, you need to know specific information about how these pieces work together. This section provides you with specific information about how the XML4C parser works.

This section contains the following:

“Application Programming”
Learn the big picture of how the parser works on TPF and read key points to remember when writing an application that uses the parser.

“DOM” on page 10
Learn how and when to use the DOM specification.

“SAX” on page 11
Learn how and when to use the SAX specification.

“DOM versus SAX” on page 12
Before writing an application that uses the parser, you need to decide which specification to use. We help you with this decision by comparing DOM and SAX.

“Validation” on page 13
Learn about validation: what it is, how to use it, and when not to use it.

Application Programming

The XML4C parser lets you interact with XML data. You can choose to use either the DOMpy or the SAX API with or without validation against a DTD or XML Schema document.

An application on TPF is coded in C++ language such that it uses one of the two APIs to interact with the parser. The chosen API determines how the parser accesses XML data. If the parser is asked to validate the data, it simultaneously interacts with the XML data and the associated DTD or XML schema document; otherwise, the parser interacts only with the XML document.

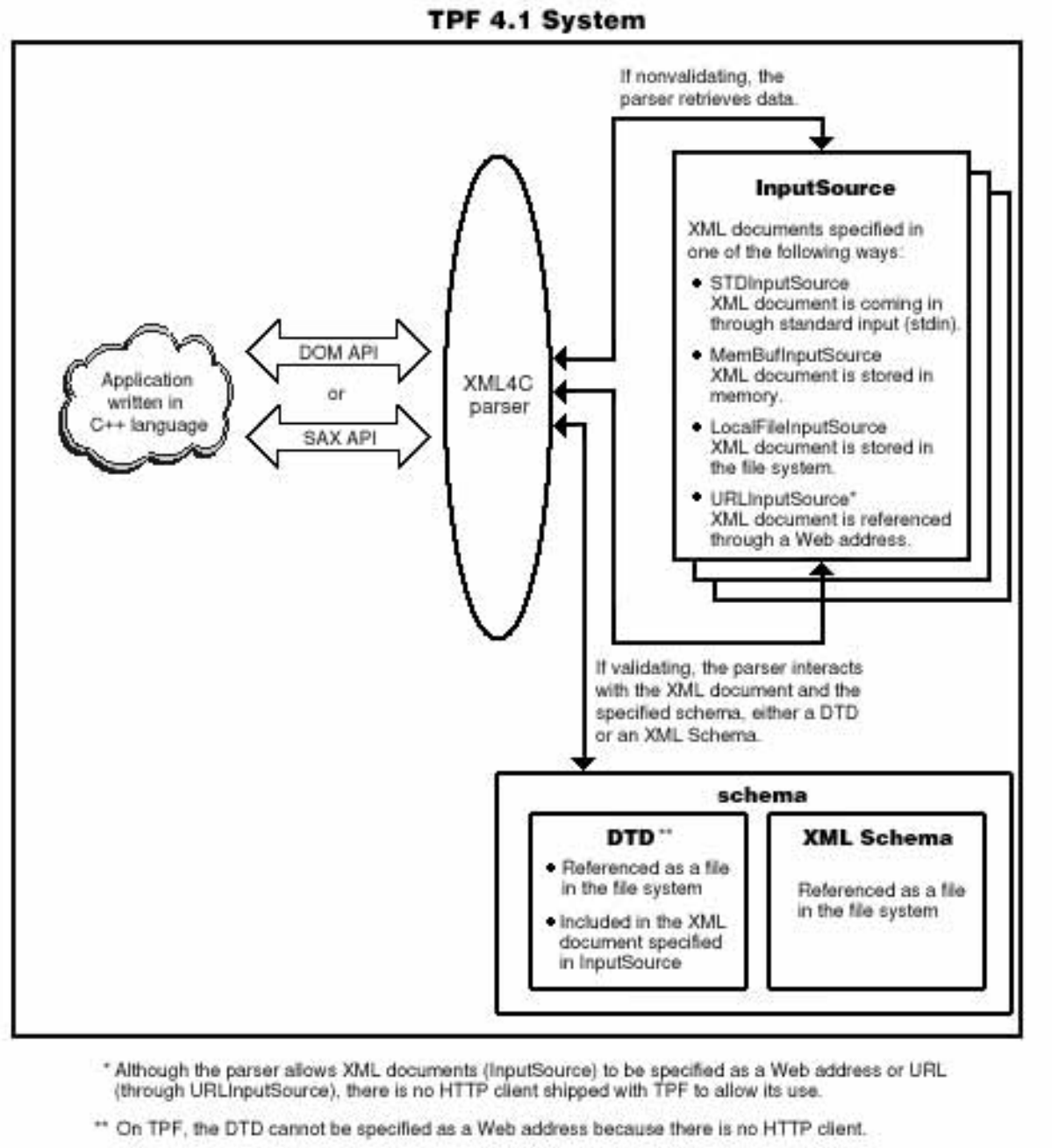

Figure 1. The XML Parser on TPF

Go to “DOM” on page 10 or “SAX” on page 11 for more information about each API; go to “Validation” on page 13 for more information about the process of validating your XML document.

InputSource: Where Is the XML Document?

Whether you use the DOM or SAX API, your applications will need to tell the parser where to find the XML document. The location of the document is specified through the **InputSource** class and can be one of the following:

LocalFileInputSource
The XML document is in the TPF file system.

MemBufInputSource
The XML document is stored in memory.

STDInputSource
The XML document is coming into TPF through standard input (`stdin`).

URLInputSource
The XML document is located at the specified Web address (URL).

Note: Although the XML4C parser allows an XML document to be specified as a uniform resource locator (URL), there is no HTTP client shipped with TPF to allow its use.

For more information about InputSource, go to "XML4C Version 3.5.1 Documentation" on page 28.

Document Type Definitions (DTDs): Defining the Rules

A DTD is one type of schema that is used for defining the tagging rules and structure for an XML document and is written using a strict (and specific) syntax. If you specify DOM or SAX as validating and your XML document uses a DTD to define the rules, the parser compares the XML document with the DTD to ensure that it conforms to the specified rules. The DTD can be in one of the following locations:

- In the XML document An XML document can have a DTD in the same file.
- In a separate file An XML document can reference a DTD that resides in a separate file. For example, the following is taken from an XML document that references an external DTD called `pnr.dtd`: `<!DOCTYPE PassengerNameRecord SYSTEM ″pnr.dtd″ >`

Note: On other platforms, DTDs can also be referenced through a URL. However, TPF does not have an HTTP client to support the use of this reference.

XML Schema: Another Way to Define Rules

XML Schema language can be used in place of a DTD to define the tagging structure and rules for an XML document. It is different from using a DTD because an XML Schema generally provides a more complete and precise definition document and is actually written in the XML language. An XML Schema is housed in a separate document and is referenced at the start of the XML document. For example, the following is taken from an XML document that references an XML Schema document called `pnr.xsd`:

```
<PNRroot xmlns:xsi=″http://www.w3.org/2001/XMLSchema-instance″
xsi:noNamespaceSchemaLocation=″pnr.xsd″>
```

According to the Schema information in the "XML4C Version 3.5.1 Documentation" on page 28, ″The schema must be specified by the xsi:schemaLocation or xsi:noNamespaceSchemaLocation attribute on the root element of the document. The xsi prefix must be bound to the Schema document instance namespace as specified by the Recommendation.″

Note: On other platforms, an XML Schema can also be referenced through a URL. However, TPF does not have an HTTP client to support the use of this reference.

Namespaces: A Two-Part Naming System

Simply put, an XML namespace is a collection of names and allows for two-part naming. Namespaces are often confused as a separate entity and something that exists or is coded. The W3C recommendation for namespaces simply uses a Universal Resource Identifier (URI) as a way of ensuring uniquely tagged elements. A namespace has an associated URI, of which each is unique by definition. By using this identifier with the words chosen for tag names, you can be assured that your tags are completely unique and can be used exactly the way you want them to be used. For example, an XML Schema based on a namespace will identify that namespace: `<xsd:schema xmlns:xsd=″http://www.w3.org/2001/XMLSchema″>`

Each time a tag is used, associate that tag with that namespace as follows: `<xsd:complexType name=″USAddress″>`

You can use more than one namespace in a document. Using multiple namespaces can assist in using the same tag name in two different ways. For example, a tag name of `<address>` might mean a home address when associated with one namespace and a work address when associated with another namespace.

XML Schema documents do not require an associated namespace. If you do not have a namespace to associate with your XML Schema document, use the **noNamespaceSchemaLocation** attribute. For example, `xsi:noNamespaceSchemaLocation=″pnr.xsd″`

Although a Web address (URL) is a type of URI, the use of URIs for namespaces does not require an HTTP client. The parser does not attempt to access the URI and, in fact, the URI does not even need to contain any data or content because namespaces ***only*** create a two-part naming convention.

For More Information

For more information, do the following:

- Go to “Resources” on page 27 for a list of book titles and Web pages about writing XML documents, DTDs, XML Schema documents, and namespaces.
- See *TPF Application Programming* for more information about writing applications on TPF.

DOM

The DOM specification is an object-based interface developed by the World Wide Web Consortium (W3C) that builds an XML document as a tree structure in memory. An application accesses the XML data through the tree in memory, which is a replication of how the data is actually structured. The DOM also allows you to dynamically traverse and update the XML document.

How Do I Use DOM?

When writing applications using the DOM API, you will use a set of C/C++ APIs to interact with the XML data. The documentation for these functions was included in the XML4C package that was ported to TPF. To view the API documentation, go to “XML4C Version 3.5.1 Documentation” on page 28.

When Should I Use DOM?

The DOM API is ideal when you want to manage XML data or access a complex data structure repeatedly. The DOM API:

- Builds the data as a tree structure in memory
- Parses an entire XML document at one time
- Allows applications to make dynamic updates to the tree structure in memory. (As a result, you could use a second application to create a new XML document based on the updated tree structure that is held in memory.)

Using the DOM API preserves the structure of the document (and the relationship between elements) and does the parsing up front so that you do not have to do the parsing process over again each time you access a piece of data. If you choose to validate your document, you can be assured that the syntax of the data is valid while you are working with it. However, the DOM API requires the memory needed to store the data, which can be expensive in terms of machine cycles. In addition, the DOM API is, by nature, a two-step process:

1. It parses the entire XML document.
2. Applications interact with the XML data held in memory using the C/C++ APIs.

As a result, you cannot begin working with the data until the DOM API has completely parsed the entire document.

XML4C parser 3.5.1 supports both DOM level 1.0 and DOM level 2.0. Level 2.0 builds on the APIs in level 1.0, which allow for data manipulation by having additional APIs that allow for items such as namespaces and cascading style sheets (CSSs). IDOM is an experimental API included in this version of the XML4C parser that is a prototyped redesign of the DOM API. For more information about the DOM API at each of these levels, go to “XML4C Version 3.5.1 Documentation” on page 28.

For More Information

For more information, do the following:

- Go to the W3C Web page at http://www.w3.org/ for more information about the DOM specification.
- Go to “Resources” on page 27 for a list of books and Web sites that provide more information about the DOM specification.

SAX

The Simple API for XML (SAX) specification is an event-based interface developed by members of the XML-DEV mailing list. It uses the parser to access XML data as a series of events in a straight line, which means that the parser finds information in the XML document without retaining state or context information.

How Do I Use SAX?

When writing applications using the SAX specification, you will use a set of C/C++ APIs to interact with the XML data. The documentation for these functions was included in the XML4C package that was ported to TPF. To view the API documentation, go to “XML4C Version 3.5.1 Documentation” on page 28.

When Should I Use SAX?

The SAX API can provide faster and less costly processing of XML data when you do not need to access ***all*** of the data in an XML document. The SAX API does the following:

- Accesses data through a series of events, eliminating the need to build a tree structure in memory. (As a result, speed of data retrieval is faster than with the DOM API because it is viewed as a flat document, or data stream.)
- Gives more control to the application by only parsing and returning the information it is asked for; the application builds the object model each time it parses the information.
- Allows you to access a small number of elements at one time rather than an entire document.

The SAX API is best for applications that need to access a specific piece of data and do not need to understand its relationship to surrounding elements. SAX is also ideal for information that is both generated by and readable by a machine. However, SAX cannot traverse the data, which makes it more expensive when you want to access data repeatedly from an XML document.

According to the Megginson Technologies Web site, SAX2 is a new version of Simple API for XML (SAX) that adds support for namespaces. among other things. For more information about the SAX API at each of these levels, go to “XML4C Version 3.5.1 Documentation” on page 28.

For More Information

For more information, do the following:

- Go to the Megginson Technologies Web site at http://www.megginson.com for more information about the SAX specification.
- Go to “Resources” on page 27 for a list of books and Web sites that provide more information about the SAX specification.

DOM versus SAX

The DOM and SAX APIs can each parse documents efficiently given appropriate conditions. The following table summarizes and compares the characteristics of the DOM API with those of the SAX API:

Table 2. Comparing the DOM API with the SAX API

	DOM	**SAX**
Type of Interface	Object-based	Event-based
Object Model	Created automatically	Must be created by application
Element Sequencing [1]	Preserved	Ignored in favor of single events
Use of TPF Memory	Higher	Lower
Speed of Initial Data Retrieval	Slower	Faster
Stored Information	Better for complex structures	Better for simple structures
Validation	Optional	Optional
Ability to Update XML Document	Yes (in memory)	No

Notes:

1. Element sequencing refers to the ability of the API to remember the order of the elements. DOM can traverse the tree structure in memory; SAX locates a specific element and ignores the surrounding elements.

For more detailed information about the two APIs, go to “DOM” on page 10 or “SAX” on page 11.

Validation

A *valid* document is one that follows the XML syntax and also conforms to the rules of an associated DTD or XML Schema. (A *well-formed* document is one that follows the XML syntax.) Both the DOM and SAX specifications enforce the rules for XML syntax whether you are using validation or not.

Validation is the process of comparing an XML document with a specified DTD or XML Schema. It ensures that the document uses only those tags that have been defined in the DTD or XML Schema as well as ensuring that it conforms to the element rules specified in the DTD or XML Schema. Both the DOM and SAX specifications have the ability to validate an XML document, but this is not a requirement for either.

When Should I Use Validation?

Validation of an XML document is expensive in terms of machine cycles. If the document is received from a reliable source and the format of the document has been predetermined, validation may not be necessary. However, using validation ensures that only elements defined in the DTD or XML Schema are used and, therefore, the structure of the XML document remains consistent.

If you do not want to validate the document each time you access data, you can, as an example, code an application so that it may reject tags that it does not recognize and takes an appropriate error path. If you do this, you may want to use validation during testing and initial implementation of a new version of an application or temporarily until the source of a document has been accredited.

How Do I Use Validation?

Validation is controlled by an API in the SAXParser and DOMParser classes. For more information about the classes for both DOM and SAX, go to “XML4C Version 3.5.1 Documentation” on page 28.

Results of Validation

The following table summarizes the expected results of validation:

Table 3. Summary of Expected Validation Results

	Validate Against a DTD	**Do Not Validate**
Document is Valid	Once validation is completed, parsing continues.	Validation is ignored and processing continues.
Document is Not Valid	Validation will result in an error response that will help you determine the error. Parsing is discontinued.	Validation is ignored and processing continues.

For More Information

For more information, do the following:

- Go to the World Wide Web Consortium (W3C) Web site at http://www.w3.org/ for more information about XML syntax.

- Go to “Resources” on page 27 for a list of books and Web sites that contain more information about the XML syntax, DTDs, XML Schema, and validation.

Examples

This section provides you with an XML tutorial and examples that use the parser.

"XML on TPF: A Short Tutorial"
This tutorial will help you to understand the basics of writing applications and drivers that use the parser. You can choose to just view the tutorial or you can participate to gain hands-on experience with XML on TPF.

"Samples from the Source: XML4C"
Get directions for using the sample programs that came with XML4C parser 3.5.1, which was ported to TPF.

XML on TPF: A Short Tutorial

To help you learn about using TPF applications to access XML data, this tutorial takes you through the process step-by-step. We give you a sample Document Type Definition (DTD), a sample XML Schema document, some sample XML data, and two sample applications that interact with the XML data. Overall, this tutorial:

- Demonstrates how TPF applications written in C++ language interact with XML data through the DOM and SAX APIs
- Identifies the core skills that are needed before writing applications that access XML data using the parser
- Gives you an opportunity to gain hands-on experience with XML on TPF.

This tutorial is only accessible when you are connected to the Internet. If you are, click the following link to launch the tutorial:

http://www.ibm.com/software/ts/tpf/pubs/xml4ic/xmltut/xtut.htm

Note: This tutorial has been tested using Netscape version 4.72 and Microsoft Internet Explorer version 5.5. Problems may occur when using earlier versions of these browsers to view the XML documents and DTDs included with this tutorial.

Samples from the Source: XML4C

XML Parser for C++ (XML4C) version 3.5.1 contains sample programs to help you understand how the APIs work as well as the differences between DOM and SAX.

As noted in the Migration Information (MAKE THIS A LINK), the actual XML4C code that was ported to the TPF 4.1 system has been included ***for your information only***. If the XML4C `tar` file has been extracted and placed onto your OS/390 UNIX System Services (OS390 UNIX) system, you may choose to install the samples that come with XML4C on TPF.

The following instructions guide you through the process of installing and running the XML4C samples on TPF.

On your OS/390 UNIX system and do the following:

1. To define the value for the variable `XERCESCROOT`, enter: **export XERCESCROOT=***"/dirname"* where *dirname* is the full path to the directory in which the XML4C `tar` file was extracted. The directory specified for *dirname* will

contain a subdirectory called samples. The XERCESCROOT variable is used in the configure and make processes; do not change the name XERCESCROOT.

2. To configure the samples for your TPF 4.1 system, do the following:
   a. Enter **chmod -R 777 $XERCESCROOT** to set the correct permission values for the files in the samples directory.
   b. Enter **cd $XERCESCROOT/samples** to change your directory to the samples directory.
   c. Enter **export USE_TPF_MAKEFILES=true** to create the environment variable.
   d. Enter **configure —cache-file=/dev/null local-ibm-tpf** to begin the configure process.
3. To compile the samples, do the following:
   a. Enter the following export commands: (After you enter these export commands, you cannot go back to the configure step without first exiting your OS390 UNIX session.)

      Note: You may need to change these export commands to match your development environment.

      - **export _C89_CCMODE=1**
      - **export _CXX_INCDIRS="$XERCESCROOT/include /u/tpf41/currentmaint/include"**

        (Enter the entire command as a single line.)

        Note: _CXX_INCDIRS must contain the directory where the TPF system header files are as well as the directory that contains the extracted XML4C header files.
      - **export _C89_INCDIRS="$_CXX_INCDIRS"**
      - **export _CXX_INCLIBS=""**
      - **export _C89_INCLIBS=""**
      - **export _CXX_CSYSLIB=""**
      - **export _C89_CSYSLIB=""**
      - **export _CXX_CXXSUFFIX="cpp"**
      - **export _CXX_OPTIONS="-W 0,langlvl(extended) -W0,NOSTART -D_POSIX_SOURCE -Uerrno -DTPF"**

        (Enter the entire command as a single line.)
   b. Enter **cd $XERCESCROOT/samples** to ensure you are in the samples directory.
   c. Enter **make** to begin the compiling process. The following warning may be issued when compiling the sample programs using the OP2 (-2) compiler option; it is expected and can be ignored:

```
WARNING CBC5109: Infinite loop detected in function xxx::fatalError(const SAXParseException&). Program may not stop.
```

4. To link the samples, do the following:
   - For each sample identified in the following list, link the appropriate .ofiles with CSTRTD(40) and CGETOP40.pds. (The .o files are generated in the $XERCESCROOT/bin/obj directory.)
   - CreateDOMDocument: CreateDOMDocument.o
   - DOMCount: DOMCount/DOMCount.o
   - DOMPrint: DOMPrint/DOMPrint.o DOMPrint/DOMTreeErrorReporter.o

- EnumVal: EnumVal.o
- MemParse: MemParse/MemParse.o MemParse/MemParseHandlers.o
- PParse: PParse.o PParseHandlers.o
- Redirect: Redirect.o RedirectHandlers.o
- SAXCount: SAXCount/SAXCount.o SAXCount/SAXCountHandlers.o
- SAXPrint: SAXPrint/SAXPrint.o SAXPrint/SAXPrintHandlers.o
- StdInParse: StdInParse.o StdInParseHandlers.o
- IDOMCount: IDOMCount/IDOMCount.o
- IDOMprint: IDOMPrint/IDOMPrint.o IDOMPrint/IDOMTreeErrorReporter.o
- SAX2Count: SAX2Count/SAX2Count.o SAX2Count/SAX2CountHandlers.o
  Note: The following warnings are expected when linking this sample and can be ignored:

  ```
  WARNING EDC4016: Duplicate objects are detected:
  {NameIdPoolEnumerator}XMLEnumerator::virtual-fn-table-ptr
  {NameIdPoolEnumerator}XMLEnumerator::virtual-fn-table-ptr
  {RefHash2KeysTableOfEnumerator}XMLEnumerator::virtual-fn-table-ptr
  {RefHash3KeysIdPoolEnumerator}XMLEnumerator::virtual-fn-table-ptr
  {NameIdPoolEnumerator}XMLEnumerator::virtual-fn-table-ptr
  ```
- SAX2Print: SAX2Print/SAX2Print.o SAX2Print/SAX2PrintHandlers.o

5. Create a loadset for the samples you linked in the previous step.

On a TPF 4.1 test system, complete the remaining steps:

6. To set up a test system, do the following:
   a. Ensure you do not have the hash (#) and double quotation (″) key strokes mapped to have special meanings. For example, if you use eNetwork Personal Communications as your emulator, you can enter **q term** at the CP prompt to determine if these two symbols are mapped to special meanings or not.
   b. To associate the names of the sample drivers with each respective program name, use the ZFILE echo and ZFILE chmod commands. An example of what to enter for each sample is included in the following list. The `CXM` names are only examples; use the same names that you used in step 4 when you linked each sample. **Note:** The following ZFILE entries are case-sensitive.
      - **ZFILE echo ″#!CXMA″ > /bin/CreateDOMDocument ZFILE chmod 777 /bin/CreateDOMDocument**
      - **ZFILE echo ″#!CXMB″ > /bin/DOMCount ZFILE chmod 777 /bin/DOMCount**
      - **ZFILE echo ″#!CXMC″ > /bin/DOMPrint ZFILE chmod 777 /bin/DOMCount**
      - **ZFILE echo ″#!CXMD″ > /bin/EnumVal ZFILE chmod 777 /bin/EnumVal**
      - **ZFILE echo ″#!CXME″ > /bin/MemParse ZFILE chmod 777 /bin/MemParse**
      - **ZFILE echo ″#!CXMF″ > /bin/PParse ZFILE chmod 777 /bin/PParse**
      - **ZFILE echo ″#!CXMG″ > /bin/Redirect ZFILE chmod 777 /bin/Redirect**
      - **ZFILE echo ″#!CXMH″ > /bin/SAXCount ZFILE chmod 777 /bin/SAXCount**
      - **ZFILE echo ″#!CXMI″ > /bin/SAXPrint ZFILE chmod 777 /bin/SAXPrint**
      - **ZFILE echo ″#!CXMJ″ > /bin/StdInParse ZFILE chmod 777 /bin/StdInParse**

- **ZFILE echo ″#!CXMN″ > /bin/IDOMCount ZFILE chmod 777 /bin/IDOMCount**
- **ZFILE echo ″#!CXMO″ > /bin/IDOMPrint ZFILE chmod 777 /bin/IDOMPrint**
- **ZFILE echo ″#!CXMP″ > /bin/SAX2Count ZFILE chmod 777 /bin/SAX2Count**
- **ZFILE echo ″#!CXMQ″ > /bin/SAX2Print ZFILE chmod 777 /bin/SAX2Print**

c. Load and activate the new loadsets.

d. TFTP or FTP the following documents from $XERCESCROOT/samples/data to your test system using binary mode:
   - personal.xml
   - personal.dtd
   - redirect.dtd

7. To run the samples, enter each of the following commands. (These commands are case-sensitive.) Each entry is accompanied by the output. **Note:** Some characters may be displayed differently because of console restrictions.
   - **ZFILE CreateDOMDocument**

     ```
     The tree just created contains: 4 elements.
     ```

   - **ZFILE DOMCount personal.xml**

     ```
     personal.xml: xx ms (37 elems).
     ```

   - **ZFILE DOMPrint -x=IBM-1047 personal.xml**

     ```
     <?xml version='1.0' encoding='IBM-1047' ?>
     <!-- @version: -->
     <personnel>
         <person id = "Big.Boss">
         <name><family>Boss</family> <given>Big</given></name>
         <email>chief@foo.com</email>
         <link subordinates = "one.worker two.worker three.worker four.worker five.worker"/>
         </person>
     ...
     ```

   - **ZFILE EnumVal personal.xml**

     ```
     ELEMENTS:
     ---------------------------
         Name: personnel
         Content Model: (person)+
         Name: person
         Content Model: (name,email*,url*,link?)
         Attributes:
             Name:id, Type: ID
     ...
     ```

   - **ZFILE MemParse**

     ```
     Finished parsing the memory buffer containing the following XML statements:
     <?xml version='1.0' encoding='ibm-1047'?>
     <!DOCTYPE company [
     <!ELEMENT company     (product,category,developedAt)>
     <!ELEMENT product     (#PCDATA)>
     <!ELEMENT category    (#PCDATA)>
     <!ATTLIST category idea CDATA #IMPLIED>
     <!ELEMENT developedAt (#PCDATA)>
     ]>
     ...
     ```

   - **ZFILE PParse personal.xml**

     ```
     Got the required 16 elements
     ```

   - **ZFILE Redirect personal.xml**

     ```
     personal.xml: xx ms (37 elems, 12 attrs, 0 spaces, 268 chars)
     ```

- **ZFILE SAXCount personal.xml**

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

- **ZFILE SAXPrint -x=IBM-1047 personal.xml**

```
<?xml version="1.0" encoding="IBM-1047"?>
<personnel>
<person id="Big.Boss">
            <name><family>Boss</family> <given>Big</given></name>
            <email>chief@foo.com</email>
            <link subordinates="one.worker two.worker three.worker four.worker five.worker"></link>
        </person>
```

- **ZFILE StdInParse < personal.xml**

  stdin: xx ms (37 elems, 12 attrs, 134 spaces, 134 chars)

- **ZFILE SAX2Count personal.xml**

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

- **ZFILE SAX2Print -x=IBM-1047 personal.xml**

```
<?xml version="1.0" encoding="IBM-1047"?>
<personnel>
<person id="Big.Boss">
            <name><family>Boss</family> <given>Big</given></name>
            <email>chief@foo.com</email>
            <link subordinates="one.worker two.worker three.worker four.worker five.worker"></link>
        </person>
```

- **ZFILE IDOMCount personal.xml**

  personal.xml: xx ms (37 elems).

- **ZFILE IDOMPrint -x=IBM-1047 personal.xml**

```
<?xml version='1.0' encoding='IBM-1047' ?>
<!-- @version: -->
<personnel>
    <person id = "Big.Boss">
    <name><family>Boss</family> <given>Big</given></name>
    <email>chief@foo.com</email>
    <link subordinates = "one.worker two.worker three.worker four.worker five.worker"/>
    </person>
...
```

Supplements

This section provides you with supporting information for the contents of this guide and contains the following:

"Glossary"
Use the glossary to look up terms you are unfamiliar with.

"Migration Information"
Learn how to install APAR PJ28176. This chapter is also included in TPF Migration Guide: Program Update Tapes.

"Resources" on page 27
Learn more about XML, DTDs, Unicode, DOM, and SAX. A list of book titles and Web pages is provided to help you learn about the different XML pieces.

"XML4C Version 3.5.1 Documentation" on page 28
Open XML4C Version 3.5.1 documentation for more information about XML4C, including the API documentation for both DOM and SAX.

Glossary

See *TPF Library Guide* for glossary definitions of important terms and abbreviations that are used in this guide.

Migration Information

XML4C Parser 3.5.1 (APAR PJ28176)

The following section discusses the migration considerations for XML4C parser 3.5.1.

Prerequisite APARs

XML4C Parser 3.5.1 (APAR PJ28176) obsoletes the version of the parser that was ported for XML parser (APAR PJ27634) on PUT 14. As a result, only the pieces of APAR PJ27634 that are TPF-specific are considered prerequisites for APAR PJ28176:

- The updates to the CICONT library member (object file) and to the CISO link-edited module support changes to the `iconv` function.
- The SPPGML and IBMPAL macro updates support the addition of the CXML link-edited module. (The parser code contained in the CXML link-edited module, however, is not required because it is replaced by XML4C Parser 3.5.1.)

Functional Overview

XML4C parser 3.5.1 allows your applications to read (parse) and write Extensible Markup Language (XML) data on the TPF 4.1 system. XML is a markup language that combines the power of Standard Generalized Markup Language (SGML) and the simplicity of Hypertext Markup Language (HTML). XML allows you to mark up data based on what information the data contains rather than on how it is to be rendered. Data marked up in XML is easy to share across various platforms and across various companies. For more information about the XML specification, go to http://www.w3.org/.

XML4C parser 3.5.1 is the XML Parser for C++ (XML4C) Version 3.5.1 ported to the TPF 4.1 system. The parser is XML Version 1.0 compliant and allows TPF 4.1 applications written in C++ language to do the following:

- Parse XML documents using the Document Object Model (DOM) Level 1.0 or 2.0 specification. You can also parse XML documents using the experimental IDOM API, but this is not formally supported by the XML4C parser and, therefore, not formally supported on the TPF 4.1 system.
- Parse XML documents using the Simple API for XML (SAX) Version 1.0 or 2.0 specification.
- Parse XML documents with or without validation against a specified Document Type Definition (DTD).
- Parse XML documents with or without validation against a document written in the XML Schema language.

  Note: XML Schema support is experimental and only includes a subset of the W3C Schema language.

In addition, the parser fully implements the ability to use namespaces in support of unique tagging structures.

Applications on the TPF 4.1 system interact with XML documents that are in the file system, coming in through standard input (`stdin`) or residing in memory. This interaction is made possible through application programming interfaces (APIs) specified by either the DOM or SAX specifications and can be either nonvalidating or validating against a schema (DTD or XML Schema). The API definitions are contained in a set of header files that application programmers will need to have in their `#include` (or search) path. See General Use C/C++ Language Header Files on page 23 for more information about these header files.

TPF information for XML4C parser 3.5.1 is exclusively online. The browser-readable HTML files are available on *IBM TPF Product Information Center*, which is the CD-ROM included with TPF 4.1 PUT 16, or on the TPF Web site (http://www.ibm.com/tpf/pubs/tpfpubs.htm). See Publication Changes on page 25 for more information about XML4C parser 3.5.1 written information.

Architecture

IBM contributed the XML4C parser to the Apache XML Project (http://xml.apache.org) as open source in November 1999. XML Parser for C++ (XML4C) Version 3.5.1 is based on Xerces-C Version 1.5.0 and is fully compliant with the Unicode 3.0 specification. While the Apache Xerces-C parser can be updated by the open source community, the XML4C parser is maintained only by IBM and may differ slightly from the Xerces-C parser.

- For more information about the Unicode specification, go to the Unicode Consortium's Web page (http://www.unicode.org).
- For more information about XML and the DOM specification, go to http://www.w3.org/.
- For more information about the SAX specification, go to http://www.saxproject.org.

Operating Environment Requirements and Planning Information

There are none.

Interface Changes

The following section summarizes interface changes.

C/C++ Language: The following section summarizes C/C++ language changes. This information is presented in alphabetic order by the type of C/C++ language information. See the *TPF C/C++ Language Support User's Guide* and *TPF Application Programming* for more information about the C/C++ language.

Build Scripts: There are no changes.

Dynamic Load Module (DLM) Stubs: There are no changes.

General Use C/C++ Language Header Files: *General use* means these header files are available for your use.

The ported XML4C parser code provides a set of header files (or `#include` files) that are needed for applications that use the XML4C parser to access XML data. The header files are included with XML4C parser 3.5.1 and must be copied into a directory accessible by application programmers. (Application programmers are instructed to concatenate the directory that contains these header files in their `#include` path.)

When copying the XML4C header files to your TPF 4.1 system, ensure that you first remove any header files you copied with the XML parser (APAR PJ27634) on PUT 14 and that you retain the directory structure in which the XML4C parser 3.5.1 (APAR PJ28176) header files reside. The following shows the directory structure for the XML4C parser 3.5.1 files (note that there are many header files in each directory):

```
include
    |- dom
    |- framework
    |- internal
    |- parsers
    |- sax
    |- sax2
    |- util
        |- regx
        |- Compilers
        |- MsgLoaders
            |- InMemory
        |- Platforms
            |- TPF
        |- Transcoders
            |- IconvTPF
    |- validators
        |- common
        |- datatype
        |- schema
        |- DTD
```

See Publication Changes on page 25 for more information about XML4C parser 3.5.1 information. (For specific information about the XML4C header files, open *XML on TPF: An Online User's Guide* and click **XML4C Version 3.5.1 Documentation**.)

Note: The XML4C header files cannot reside in a partitioned data set (PDS) because the first eight characters of all header files that reside in a PDS must be unique, and the XML4C header file names do not follow this rule. To ensure that your application programs work, put the header files into a hierarchical file system (HFS); these header files cannot be renamed and each must reside in its native directory in the HFS.

Implementation-Specific C/C++ Language Header Files (IBM Use Only): There are no changes.

Library Interface Scripts: There are no changes.

Link-Edited Modules: Table 4 summarizes changes to the link-edited modules shipped by IBM, which should go into a data set with attributes DCB=(RECFM=U,LRECL=80,BLKSIZE=1200). This information is presented in alphabetic order by the name of the link-edited module.

Table 4. Changes to Link-Edited Modules for XML4C Parser 3.5.1

Link-Edited Module	New, Changed, or No Longer Supported?	Description of Change
CXML	Changed	Updated for XML4C Parser 3.5.1.

Members: There are no changes.

Object Code Only (OCO) Stubs: There are no changes.

Configuration Constant (CONKC) Tags: There are no changes.

Control Program Interface (CINFC) Tags: There are no changes.

Copy Members: There are no changes.

Fixed File Records: There are no changes.

Macros: There are no changes.

Segments: There are no changes.

System Equates: There are no changes.

User Exits: There are no changes.

Functional and Operational Changes

There are no changes.

Performance or Tuning Changes

There are no changes.

Storage Considerations and Changes

There are no changes.

System Initialization Program (SIP) and System Generation Changes

There are no changes.

Loading Process Changes

There are no changes.

Online System Load Changes

There are no changes.

Publication Changes

Table 5 summarizes changes to the publications in the TPF library. This information is presented in alphabetic order by the publication title. See the *TPF Library Guide* for more information about the TPF library.

Table 5. Changes to TPF Publications for XML4C Parser 3.5.1

Publication Title	Softcopy File Name	Description of Change
TPF Migration Guide: Program Update Tapes	GTPMG206	Updated with migration considerations for XML4C parser 3.5.1.
TPF C/C++ Language Support User's Guide	GTPCLU0G	Updated the list of classes that are supported by XML Parser for C++ (XML4C) Version 3.5.1 but not documented in the *TPF C/C++ Language Support User's Guide*.
XML User's Guide	Not Applicable	Updated for the delivery of TPF information on XML4C parser 3.5.1. This guide was delivered previously as browser-readable HTML files only, called *XML on TPF: An Online User's Guide*. The contents of this guide have been changed slightly so that you can view XML information in PDF as well as in HTML format. From the IBM TPF Product Information Center, do one of the following: • Click **Tasks –> XML4C Parser 3.5.1** to view task-oriented information such as a short tutorial that walks you through a real example of XML on TPF, step-by-step instructions for using sample programs included with the ported source code, and migration information. • Click **Concepts –> XML4C Parser 3.5.1** to view conceptual information about XML on TPF, including a list of resources and the documentation included with the ported source code. • Click **Full Library** and select **XML User's Guide** to view the information in either HTML or PDF.

Host System Changes
There are no changes.

Application Programming Interface (API) Changes
XML4C parser 3.5.1 supports the DOM and SAX specifications, which are each comprised of several APIs. These APIs are supported on the TPF 4.1 system but are not documented in the library. Some of the APIs included with this version of the parser may be different than those included with the original port of the XML parser (APAR PJ27634) on PUT 14. The information for the APIs was included with the ported code for XML4C parser 3.5.1 and is available in *XML on TPF: An Online User's Guide*. See Publication Changes on page 25 for more information.

Database Changes
There are no changes.

Feature Changes
There are no changes.

Installation Validation
There are no changes.

Migration Scenarios
Use the following procedure to install XML4C parser 3.5.1 on your TPF 4.1 system:

1. Unpack program update tape (PUT) 16, which contains APAR PJ28176 for XML4C parser 3.5.1. See *TPF Memo to Licensees* for more information about unpacking the tape.
2. Copy the XML4C C/C++ language header files into an HFS so that they are available to application programmers. The directory structure must be maintained and any header files copied for XML parser (APAR PJ27634) on PUT 14 must be removed. See General Use C/C++ Language Header Files on page 23 for more information about the XML4C header files.
3. Load the CXML link-edited module listed in Table 4 on page 24.
4. IPL your TPF 4.1 system.

Additional Information
- Listing files are available on a CD-ROM. See your IBM service representative for more information about these CD-ROMs.
- Source code information:
  1. The `xml4c3_5_1.tpf.ascii.tar.Z` `tar` file is provided and contains the XML Parser for C++ (XML4C) Version 3.5.1 source code. The files created after extracting this `tar` file contain source code, listings, samples, written information, and makefiles. (IBM does not, however, warrant that these makefiles will run in your development environment because these files may use tools that are not required for your TPF 4.1 system.) The `tar` file is provided ***for your information only*** and this source code is not supported on the TPF 4.1 system.
  2. Extracting the contents of the `tar` file will create a `samples` directory. The samples in this directory may be useful to programmers who are writing applications that interact with XML data. Directions for using these samples (included in *XML on TPF: An Online User's Guide*) assume that the programmers have access to the extracted `samples` directory.

Resources

The following list of resources may be helpful to you while learning XML. This is not a complete list and you are encouraged to find other resources on the Web or through other sources.

Book Titles Part of the power of XML is that it can be used for both publishing to an end user and for data definitions (schema) for the interchange of data between computer programs. Many of the books in the list below favor one of the two points of view. *XML for the World Wide Web: Visual QuickStart Guide* presents a publishing point of view. *Beginning XML* bridges the gap between both, while *Professional XML* focuses on the data perspective.

- Benoit, Marcahl. *XML by Example*. Indianapolis, IN: QUE, 2000.
- Castro, Elizabeth. *XML for the World Wide Web: Visual QuickStart Guide*. Berkeley, CA: Peachpit Press, 2001.
- Graham, Tony. *Unicode: A Primer*. United States of America: M& T Books, 2000.
- Harold, Elliotte Rusty. *XML Bible*. United States of America: IDG Books Worldwide, Inc, 1999.
- Hunter, David, Curt Cagle, Dave Gibbons, Nikola Ozu, Jon Pinnock, and Paul Spencer. *Beginning XML*. Birmingham, UK: Wrox Press Ltd, 2000.
- Martin, Didier, Mark Birbeck, Michael Kay, Brian Loesgen, Jon Pinnock, Steven Livingstone, Peter Stark, Kevin Williams, RIchard Anderson, Stephen Mohr, David Baliles, Bruce Peat, and Nikola Ozu. *Professional XML*. Birmingham, UK: Wrox Press Ltd, 2000.
- North, Simon and Paul Hermans. *Teach Yourself XML in 21 Days*. United States of America: Sams, March 1999.
- Ray, Eric T. *Learning XML*. United States of America: O'Reilly & Associates, 2001. A sample chapter from this book can be found on the O'Reilly Web site at http://www.oreilly.com/catalog/learnxml/chapter/ch02.html

IBM Web Sites

- Alphaworks: http://alphaworks.ibm.com/
- developerWorks' XML Zone: http://www.ibm.com/developerworks/xml/
- IBM Red Books: http://www.redbooks.ibm.com/
- XML Toolkit of z/OS and OS/390 Home Page: http://www.s390.ibm.com/xml/

Non-IBM Web Sites IBM accepts no responsibility for the content or use of non-IBM Web sites.

- Apache XML Project: http://xml.apache.org
- developerlife.com: http://developerlife.com
- OASIS: http://www.oasis-open.org/
- SAX: http://www.megginson.com and http://www.saxproject.org
- Unicode: http://www.unicode.org
- W3C (for the XML specification, the DOM specification, namespaces, and XML Schema): http://www.w3.org/
- The XML Cover Pages: http://xml.coverpages.org/
- XML.com: http://www.xml.com
  - Annotated XML specification: http://www.xml.com/axml/testaxml.htm.
  - Myths about Namespaces: (http://www.xml.com/pub/a/2000/03/08/namespaces/index.html)

.

- XML.ORG: http://www.xml.org

XML4C Version 3.5.1 Documentation

When the TPF development lab ported XML Parser for C++ (XML4C) version 3.5.1, online documentation was included. The XML4C online documentation provides you with additional information about the XML4C parser, including the API documentation used for both the (DOM) and (SAX) specifications. (Some of the APIs included with this version of the parser may be different than those included with the original port of the XML parser (APAR PJ27634) on PUT 14.)

IBM has not committed to including any future release of this product for TPF. Discussion of future support (such as (XML Schema)) in the following documents should not be interpreted to imply that such support will be provided on TPF.

This documentation is only available when you are connected to the Internet. If you are connect, the following link opens a new browser window. To return to this page, simply close the new browser window by using the X in the upper-right corner.

To open the XML4C Version 3.5.1 Documentation, click the following:

http://www.ibm.com/software/ts/tpf/pubs/xml4ic/xml4c351/html/index.html

Note: This documentation was not developed by the TPF development lab and the lab is, therefore, not responsible for the information and links contained in the documentation.