



Product:	XMLParser
Product version:	V0.9
Document ID:	PM-XMLParser
Doc revision:	A
Written/Aprr.:	KD / RE
Date:	6. Nov. 2008

Industrial Control Design AS



XMLParser V0.9

Programmer Manual

The content of this document is confidential information not to be published without the consent of Industrial Control Design AS.

Industrial Control Design AS, www.icd.no, support@icd.no, forum.icd.no

Contents

1. INTRODUCTION.....	3
1.1. About.....	3
1.2. Terms and Definitions.....	3
<hr/>	
2. INSTALLATION.....	4
2.1. Preparing for linking with the XMLParser library.....	4
<hr/>	
3. FUNCTIONAL DESCRIPTION.....	5
3.1. Parse.....	5
3.2. Search.....	5
3.3. Modify.....	5
3.4. Write.....	5
3.5. Known design limitations.....	6
3.6. Known bugs.....	6
<hr/>	
4. APPENDIX.....	7
4.1. Example XML file.....	7
4.2. Example XMLParser usage.....	7

1. Introduction

1.1. About

This document describes how the XMLParser library component works, and how to set it up and use it with the CDP system. The XMLParser library component has the following features:

- Reads XML from file or buffer.
- Generates a tree of elements with bidirectional access.
 - Can do GetNext() / GetPrevious(), GetParent() / GetChild() etc.
- Search using XPath-like queries.

1.2. Terms and Definitions

DOM

Document Object Model

CDP

Control Design Platform.

Component

Object with strict interface specification.

SAX

Simple API for XML

XML

eXtensible Markup Language

2. Installation

The XMLParser can be delivered both as source code and as a separate library which is linked into the application. If delivered as separate library, the XMLParserLib Setup will by default install all files in sub-folders of “CDP Developer”.

Prerequisites:

- A valid CDP license
- Familiar with CDP
- Familiar with XML

2.1. Preparing for linking with the XMLParser library

Copy the XMLParserLib folder either into your project, to a location for standard libraries, or copy the XMLParserLib_<config>.lib and header-files to a location where your linker can find it.

Include the XMLParserLib.h header file in your application's Libraries.h file:

```
#include <XMLParserLib.h>
```

If necessary, update the include paths by adding the path to where to find XMLParserLib header-files (and source code if available):

- CDP_Application -> Properties
 - C/C++ -> General -> Additional Include Directories

Link the XMLParser library into your application by putting XMLParser_<config>.lib in a folder included in your linker path. For the CDP_Application project:

- CDP_Application->Properties
 - Linker -> Input -> Additional dependencies

3. Functional description

The XMLParser is a light-weight tree-based parser (DOM), and not an event-based parser (SAX). It is designed and well-suited for data-centric XML, as opposed to document-centric, due to its primary function as a configuration XML reader. The parser reads the entire buffer into memory, making bidirectional access possible.

3.1. Parse

The XMLParser class is used to start parsing an XML document, either from a file:

```
XMLParser xmlParser;
xmlParser.ReadXMLFile("myfile.xml");
```

Or alternatively from a buffer:

```
char* pXml = "<Component><Subcomponent></Subcomponent></Component>";
xmlParser.Set(pXml, true); // copy buffer
```

3.2. Search

The parser allows you to easily find specific elements:

```
XMLElement* pElem = xmlParser.FindFirstElement("Subcomponent");
```

You can also find an element from Xpath-like standardized search strings, using **GetElement**. Example search strings:

- "Model/Signal"
- "Model/Signals/Signal[@Name='Joystick']"
- "Model/Signals/Signal[@FromState='Null' & @ToState='Running']"
- "Component/IOConfig/Module[@Name='RRAIO16_21']/Inputs/Channel[@Name='RRAIO16_21_7']/Scaling/Point[@HardwareValue='7500']"

3.3. Modify

XMLParser library allows modification of the XML tree. In **XMLParser** class, refer to **ParseNewXML** and related methods. In **XMLPrimitive**, refer to **Insert** methods.

3.4. Write

XMLParser library allows writing the XML tree back to a buffer or file. See methods **WriteXMLFile** and **WriteXMLBuffer** in **XMLParser** class.

3.5. Known design limitations

- No encoding support.
- No XML entity support.
- Only the first child item has a valid Parent pointer. All Next-items of the child has Parent == NULL. So, if GetPrevious() returns NULL, GetParent() will return the parent, or NULL if this is the ROOT item.
- Only the first text node is available for element value content that includes other elements.
- The objects (XMLPrimitives, buffers etc) allocated by this parser are only valid as long as the parser object is valid.

3.6. Known bugs

ICD Bug #	Description

4. Appendix

4.1. Example XML file

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Component Name="SomeComponent" Model="SomeModel">
  <Activate>1 </Activate>
  <InitialState>Null</InitialState>
  <NetworkInterface LocalName="ETH0"></NetworkInterface>
  <Description><![CDATA[A simple & short component description.]]></Description>
  <Signals>
  </Signals>
  <Alarms>
  </Alarms>
</Component>
```

4.2. Example XMLParser usage

```
void SomeClass::SomeFunction(...)
{
  XMLParser xmlParser;
  if (xmlParser.ReadXMLFile("myfile.xml"))
  {
    XMLElement* pElement = xmlParser.FindFirstElement("ElementName");
    if (pElement!=NULL)
    {
      // do the stuff...
    }
    // done.
  }
}
```