

2  
TXT2XML  
TXT2XML

TXT2XML

TXT2XML

TXT2XML

TXT2XML

TXT2XML

TXT2XML

TXT2XML

TXT2XML

TXT2XML

TXT2XML

TXT2XML

TXT2XML

TXT2XML

**TXT2XML** : Conversion from text to XML  
and vice-versa using a COBOL copybook as  
reference.

**XML2COB** : Generation of a COBOL  
copybook using an XML file as reference

**Version 1.25<sup>1</sup>**

---

<sup>1</sup>

# Contents

<b>I</b>	<b>T X T 2 X M L</b>	<b>6</b>
<b>1</b>	<b>Introduction</b>	<b>7</b>
<b>2</b>	<b>Principles of conversion</b>	<b>10</b>
2.1	COBOL copybook analysis . . . . .	10
2.1.1	"Cleaning" the copybook . . . . .	10
2.1.2	Extracting the meta-data . . . . .	11
2.1.3	Renumbering the COBOL item levels . . . . .	12
2.2	Text to XML conversion . . . . .	12
2.3	XML to text conversion . . . . .	15
<b>3</b>	<b>Signed and binary data</b>	<b>17</b>
3.1	Warning . . . . .	17
3.2	COBOL compilers . . . . .	17
3.3	Internal representation . . . . .	18
3.4	Low-values, high-values and spaces . . . . .	20
3.5	Functions . . . . .	20
<b>4</b>	<b>Installation</b>	<b>22</b>
4.1	Downloading TXT2XML . . . . .	22
4.2	Under MVS . . . . .	22
4.2.1	Installation . . . . .	22
4.2.2	Tests . . . . .	23
4.2.3	Post-installation . . . . .	23
4.3	Under Windows . . . . .	23
4.3.1	Regina or Reginald Rexx . . . . .	23
4.3.2	Java . . . . .	23
4.3.3	Tests . . . . .	24
4.4	Under Unix/Linux and other platforms . . . . .	24
4.4.1	Regina . . . . .	24
4.4.2	Java . . . . .	24
4.4.3	Tests . . . . .	24
<b>5</b>	<b>Syntax</b>	<b>25</b>
5.1	Mandatory parameters . . . . .	25
5.1.1	TXT txt-file . . . . .	25
5.1.2	COB cob-file . . . . .	26
5.1.3	XML xml-file . . . . .	26

<i>CONTENTS</i>	4
-----------------	---

5.1.4	FORMAT format . . . . .	27
5.2	Optional parameters . . . . .	27
5.2.1	BROWSE . . . . .	27
5.2.2	VERBOSE . . . . .	27
5.2.3	DTD dtd-file . . . . .	27
5.2.4	PREFIX prefix . . . . .	28
<b>6</b>	<b>Examples</b>	<b>29</b>
6.1	MVS . . . . .	29
6.1.1	ISPF . . . . .	29
6.1.2	Batch . . . . .	30
6.2	Windows . . . . .	31
6.3	Unix/Linux . . . . .	32

## **II XML2COB 35**

<b>7</b>	<b>Introduction</b>	<b>36</b>
<b>8</b>	<b>Principles of generation</b>	<b>37</b>
8.1	The meta-data . . . . .	37
8.2	The COBOL copybook generation . . . . .	37
8.3	COBOL level . . . . .	38
8.4	Constraints . . . . .	40
<b>9</b>	<b>Installation</b>	<b>42</b>
<b>10</b>	<b>Syntax</b>	<b>43</b>
10.1	Mandatory parameters . . . . .	43
10.1.1	COB cob-file . . . . .	43
10.1.2	XML xml-file . . . . .	44
10.2	Optional parameters . . . . .	44
10.2.1	PREFIX prefix . . . . .	44
10.2.2	LEVEL01 . . . . .	44
10.2.3	ROUND . . . . .	45
10.2.4	BROWSE . . . . .	45
10.2.5	VERBOSE . . . . .	45
<b>11</b>	<b>Examples</b>	<b>46</b>
11.1	MVS . . . . .	46
11.1.1	ISPF . . . . .	46
11.1.2	Batch . . . . .	47
11.2	Windows . . . . .	48
11.3	Unix/Linux . . . . .	49

## **III The GUI front-end 52**

<b>12</b>	<b>The Graphical User Interface</b>	<b>53</b>
12.1	Introduction . . . . .	53
12.2	Installation . . . . .	54

<i>CONTENTS</i>	5
12.3 Syntax . . . . .	55
12.4 Example . . . . .	55
<b>A License</b>	<b>57</b>
A.1 Preamble . . . . .	57
A.2 GNU GENERAL PUBLIC LICENSE TERMS AND CONDI- TIONS FOR COPYING, DISTRIBUTION AND MODIFICA- TION . . . . .	58
A.3 NO WARRANTY . . . . .	61
A.4 How to Apply These Terms to Your New Programs . . . . .	62
<b>B History &amp; Road map</b>	<b>63</b>
B.1 30/09/02 - Version 0.1 . . . . .	63
B.2 04/10/02 - Version 0.2 . . . . .	63
B.3 06/11/02 - Version 0.3 . . . . .	63
B.4 21/11/02 - Version 1.0 . . . . .	63
B.5 28/07/04 - Version 1.1 RC1 . . . . .	64
B.6 28/08/04 - Version 1.1 . . . . .	64
B.7 20/09/04 - Version 1.15 . . . . .	64
B.8 05/11/04 - Version 1.20 . . . . .	65
B.9 05/03/05 - Version 1.25 . . . . .	65
B.10 Road map . . . . .	65
<b>C Internet resources</b>	<b>66</b>
<b>D Glossary</b>	<b>68</b>

## Part I

**T X T 2 X M L**

# Chapter 1

## Introduction

COBOL, a forty years old computer language, and XML a new revolutionary format, seems to as near as the earth and the moon. However, if you look any further, you will notice some common concepts :

	XML	COBOL
Hierarchy of data	Root and child elements	level 01 and other level items
Indentation for a better visualization	✓	✓
Structured data	Simple and complex elements	Items and grouping items
Occurrence of data	Unique and multiple	OCCURS clause

The need to exchange data in XML between servers is continuously increasing and, in each mainframe around the world, there are plenty of COBOL copybooks and text files. So, was born the idea of TXT2XML : a bridge between the common concepts of COBOL and XML. It's not a GUI<sup>1</sup> tool, it's a Rexx script (for more info on Rexx see appendix C) running in a DOS command interface for the Windows users, a shell for the Unix users, ISPF or JCL for the mainframe users.

TXT2XML is not an XML parser written in Rexx, nor will support the entire XML specification. Some COBOL clauses are also not supported (like REDEFINES, DEPENDING, ... ). TXT2XML works in both directions, i.e. it can convert XML files to text files and vice-versa. The COBOL copybook is always mandatory to perform the conversion, because :

---

<sup>1</sup>However, starting with version 1.25, a GUI front-end has been developed. See chapter 12 for more details.

- During XML to text conversion, XML element names are checked against the COBOL item names. For example, if there is an XML element like `<cobol_item_01>`, there must be an item like *COBOL-ITEM-01* in the COBOL copybook.
- During text to XML conversion, the COBOL copybook item names are used to build the XML elements.
- During XML to text conversion, if a COBOL item is numeric, the XML content is checked to see if it's also numeric. Otherwise conversion will fail.
- ...

TXT2XML was therefore developed primarily to convert data files typically used on a mainframe. TXT2XML works only with well-formed XML documents and valid COBOL copybooks. TXT2XML will never check for COBOL syntax errors or missing XML closing tags. TXT2XML generates 100 % compatible and well-formed XML files, but it's still your responsibility to validate them against a DTD with an XML parser.

TXT2XML was originally written on an IBM mainframe running OS/390 V2R10. But, starting with version 1.15, TXT2XML should also run on the following platforms :

- Linux,
- FreeBSD,
- Solaris,
- AIX,
- HP-UX,
- OS/2,
- eCS,
- DOS,
- Win9x/Me/NT/2k/XP,
- Amiga,
- AROS,
- QNX4.x,
- QNX6.x,
- BeOS,
- MacOS X,
- EPOC32,
- AtheOS,



- OpenVMS,
- SkyOS,
- OpenEdition.

As it's impossible to run all those OS, TXT2XML was tested successfully on :

- Windows 2K,
- Linux Mandrake 10.0,
- MVS (OS/390 V2R10).

It should work without problems on the other platforms<sup>2</sup>. To run TXT2XML on non-mainframe platforms, you must first install Regina Rexx or for windows only, Reginald (see chapter 4). Should you succeed or encounter problems in running TXT2XML on a non-tested platform, please send an e-mail to :

**`sunuraxi@users.sourceforge.net`**

TXT2XML is released under the GPL license (see appendix A).

---

<sup>2</sup>with the exception of binary and packed data.

## Chapter 2

# Principles of conversion

As said early, conversion occurs only if a valid COBOL copybook is present. The first step in the conversion process is the ...

### 2.1 COBOL copybook analysis

The analysis take place in 3 phases :

#### 2.1.1 "Cleaning" the copybook

In this phase, the copybook is read and :

- comments are skipped,
- Multi-line COBOL declaration are concatenated in one line.
- PICTURE clauses are replaced by PIC,
- SIGN clauses are ignored,
- IS clauses are ignored,
- NATIVE clauses are skipped,
- X(4) and 9(6) symbols are replaced respectively by XXXX and 999999,
- Final dot of COBOL declarations is suppressed,
- COBOL declarations are capitalized.<sup>1</sup>

---

<sup>1</sup>According to IBM's COBOL Language Reference (IGYLR204), paragraph 1.1.1.1 COBOL Words with Single-Byte Characters : "Each lowercase letter is considered to be equivalent to its corresponding uppercase letter, except in non numeric literals"

### 2.1.2 Extracting the meta-data

In this phase, the following meta-data are extracted :

Meta-data	Usage
COBOL level	To indent XML element
Name	To create or check XML element tags
Type of data	To check XML data during text to XML conversion
Sign	When converting signed and/or binary data
Sign position	When converting numeric data
Starting position	To create the XML content or a text record
Length	To create the XML content or a text record
Default value	To initialize a text record when converting from XML to text

FILLER clauses are not skipped. They are used to compute the starting positions of all COBOL items. They never appears in the XML file when the text to XML conversion process ends. The copybook is also checked for unsupported COBOL clauses :

- REDEFINES
- DEPENDING ON
- COMPUTATIONAL-1, COMPUTATIONAL-2, COMPUTATIONAL-5, COMP-1, COMP-2, and COMP-5.
- POINTER, PROCEDURE-POINTER and FUNCTION-POINTER
- OBJECT REFERENCE
- NATIONAL
- DISPLAY-1<sup>2</sup>
- P symbol (decimal scaling position)
- INDEX
- level 66, 77, 88,<sup>3</sup>
- Non-numeric COBOL levels or COBOL levels greater than 49,
- DBCS picture symbols.

and for partially supported COBOL clauses :

- OCCURS for elementary and group item with only one dimension,
- VALUES except for QUOTES and NULL clauses.

---

<sup>2</sup>DBCS items are not supported

<sup>3</sup>Those levels generate only a warning message and conversion continues.

### 2.1.3 Renumbering the COBOL item levels

As COBOL item levels are used to indent tags during text to XML conversion, they are internally renumbered to have a smoother indentation. For example, the following COBOL copybooks :

Figure 2.1: A sample COBOL copybook

```
01 COBOL-ITEM-01.
    05 FILLER PIC IS X.
    05 COBOL-ITEM-02 PIC IS X(5).
```

and

Figure 2.2: Same sample COBOL copybook with different levels

```
10 COBOL-ITEM-01.
    20 FILLER PIC IS X.
    20 COBOL-ITEM-02 PIC IS X(5).
```

will always give the same indentation, like in the following XML file :

Figure 2.3: Resulting XML file after conversion

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<DATA>
  <COBOL_ITEM_01>
    <COBOL_ITEM_02>12345</COBOL_ITEM_02>
  </COBOL_ITEM_01>
</DATA>
```

The original COBOL copybook is never modified by TXT2XML and if the conversion ends without errors, a report of this analysis is printed.

## 2.2 Text to XML conversion

This is the easiest conversion. An XML header (`<?xml .... ?>`) is first created. Then, one text record is read, item data is extracted from the current text record and the XML elements and their content are written one by one. The conversion process continues until the end of the text file. The XML elements can be classified in 3 categories :

- **Root** : this element must be unique in the whole XML file. That's why, DATA, a COBOL reserved word, was chosen as root element name. It has no equivalent in the COBOL copybook.

- **First child** : it appears each time a text record is read. It corresponds to the level of the first COBOL item in the copybook. As this first child element will include all the other XML elements, this COBOL item needs to be a grouping item, with the lowest level and ideally with a unique occurrence.
- **Other child elements**: The other XML elements.

All of them are built using the following rules :

- The name of the XML element is the capitalized COBOL item name with dashes (" - ") translated to underscore (" \_ "). For example : *COBOL-ITEM-01* will become *<cobol\_item\_01>*,
- Content of the XML element is extracted from the current text record (using the starting position and the length computed in section 2.1). Characters are escaped if necessary<sup>4</sup>,
- FILLER clauses are skipped.
- Opened XML tags are closed :
  - immediately after the content of an XML element,
  - at the end of each COBOL grouping item,
  - at the end of the text record,
  - at the end of the text file.

If requested, TXT2XML can create a DTD (Document Type Definition), based on the COBOL copybook, which describes the XML structure. This DTD can be internal :

---

<sup>4</sup>& < > “ and ’ need to be converted to &amp; &lt; &gt; &quot; &apos; respectively.

Figure 2.4: Internal DTD

```

<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!DOCTYPE DATA [

    <!ELEMENT NUMERIC-TYPES (EXTERNAL-DECIMAL, BINARIES,
        INTERNAL_DECIMAL)>
    <!ELEMENT EXTERNAL-DECIMAL (EXT-DEC-UNSIGNED,

        EXT-DEC-SIGNED-NEGATIVE, EXT-DEC-SIGNED-POSITIVE,
        EXT-DEC-SIGN-LEADING-NEG, EXT-DEC-SIGN-LEADING-POS,
        EXT-DEC-SIGN-TRAILING-NEG, EXT-DEC-SIGN-TRAILING-POS,
        EXT-DEC-SIGN-SEP-LEAD-NEG, EXT-DEC-SIGN-SEP-LEAD-POS,
        EXT_DEC_SIGN_SEP_TRAIL_NEG, EXT_DEC_SIGN_SEP_TRAIL_POS)>
    <!ELEMENT EXT_DEC_UNSIGNED (#PCDATA)>
    <!ELEMENT EXT_DEC_SIGNED_NEGATIVE (#PCDATA)>
    <!ELEMENT EXT_DEC_SIGNED_POSITIVE (#PCDATA)>
    <!ELEMENT EXT_DEC_SIGN_LEADING_NEG (#PCDATA)>
    <!ELEMENT EXT_DEC_SIGN_LEADING_POS (#PCDATA)>
    <!ELEMENT EXT_DEC_SIGN_TRAILING_NEG (#PCDATA)>
    <!ELEMENT EXT_DEC_SIGN_TRAILING_POS (#PCDATA)>
    <!ELEMENT EXT_DEC_SIGN_SEP_LEAD_NEG (#PCDATA)>
    <!ELEMENT EXT_DEC_SIGN_SEP_LEAD_POS (#PCDATA)>
    <!ELEMENT EXT_DEC_SIGN_SEP_TRAIL_NEG (#PCDATA)>
    <!ELEMENT EXT_DEC_SIGN_SEP_TRAIL_POS (#PCDATA)>
    <!ELEMENT BINARIES (BINARY-UNSIGNED, BINARY-SIGNED-POSITIVE,
        BINARY_SIGNED_NEGATIVE)>
    <!ELEMENT BINARY_UNSIGNED (#PCDATA)>
    <!ELEMENT BINARY_SIGNED_POSITIVE (#PCDATA)>
    <!ELEMENT BINARY_SIGNED_NEGATIVE (#PCDATA)>
    <!ELEMENT INTERNAL-DECIMAL (INT-DEC-UNSIGNED, INT-DEC-SIGNED-POS,
        INT_DEC_SIGNED_NEG)>
    <!ELEMENT INT_DEC_UNSIGNED (#PCDATA)>
    <!ELEMENT INT_DEC_SIGNED_POS (#PCDATA)>
    <!ELEMENT INT_DEC_SIGNED_NEG (#PCDATA)>

]>
<DATA>
    <NUMERIC_TYPES>

    ...

```

or external :

Figure 2.5: XML file with an external DTD reference

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE DATA SYSTEM "../sample/dtd/binary">
<DATA>
  <NUMERIC_TYPES>
  ...
```

## 2.3 XML to text conversion

This conversion is a little bit more difficult and even in some cases impossible. Remember, TXT2XML implements only a subset of XML and COBOL. For example, this XML file :

Figure 2.6: Nonconvertible XML file

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<weather>
  <city name="'Los Angeles'">
    <sky>Blue</sky>
    Hot and sunny
  </city>
</weather>
```

In this file, the XML element **city** is a complex one. It includes the simple element **sky** but also a content (**Hot and sunny**). The corresponding COBOL copybook would be something like :

Figure 2.7: Corresponding COBOL copybook

```
01 city pic x(20).
   03 name pic x(20).
   03 sky pic x(8).
```

This is impossible in COBOL : a grouping item can't have a picture clause.<sup>5</sup>

So, not all the XML files can be converted. But, with a valid COBOL copybook, the XML file is read one line at the time and data is extracted using the following rules :

---

<sup>5</sup>COBOL for MVS & VM V1R2.2 Language Reference (IGYLR204) paragraph 5.3.11 PICTURE Clause : "The PICTURE clause can be specified only at the elementary level."

- Mixed XML content is **not supported**,
- All the XML tags and contents may expand on more than one line,
- XML headers starting with `<?xml` and ending with `?>` are skipped,
- XML entities starting with `<!ENTITY` and ending with `>` are **not supported**. If such a declaration is found, the conversion process is aborted,
- XML internal and external document type declaration starting with `<!DOCTYPE` and ending with either `>` or `/>` are skipped,
- XML comments starting with `<!--` and ending with `-->` are skipped,
- The first child element is then searched. Each time an XML first child closing tag is found, a text record is written and a new text record is initialized with default values from the COBOL copybook,
- Attributes of the first child element are supported. But, attributes of the root element are ignored.
- Each XML element, even without a content, **must** have an equivalent in the COBOL copybook. To find it, the XML element name is capitalized and underscores are translated to dashes. For example : `<cobol_item_01>` will become `COBOL-ITEM-01` and is searched in the COBOL copybook,
- Each XML attribute **may** have an equivalent in the COBOL copybook,
- XML attributes like `xml:space` are ignored,
- Each time a content is found, it's length and data type are checked against the characteristics of the corresponding COBOL item. Numeric items are right justified and left zero filled. If necessary characters are escaped.<sup>6</sup>
- If some XML elements or attributes are missing, conversion continues with the new element found. Missing COBOL items are left to their default value (see section 2.1).
- Empty XML elements, even with attributes, are supported.

---

<sup>6</sup>&amp; &lt; &gt; &quot; &apos; are converted to & < > “ and ’ respectively or content is extracted from CDATA section starting with `<![CDATA[` and ending with `]]>`



## Chapter 3

# Signed and binary data

### 3.1 Warning

Signed and binary data are hardly bound with the operating system and the hardware running TXT2XML. On a PC, binary (and not numeric and packed-decimal) data is stored in little endian mode. It means that the low-order byte of the number is stored in memory at the lowest address, and the high-order byte at the highest address. On a IBM MVS mainframe, binary data are stored in the opposite mode : big endian. For example :

Hexadecimal data	Little endian (PC)	Big endian (MVS)
x'01 02 03 04'	04 03 02 01	01 02 03 04

**Conclusion** : although TXT2XML runs on multiple platforms, you should always run TXT2XML on the platform where you will use the binary and/or signed converted data.

So, forget the idea to run TXT2XML on one platform (i.e a PC running Windows) and to transfer the converted file on another platform (i.e. an IBM mainframe). File transfer means code page (character set) translation which binary, packed-decimal and even numeric data don't support. Don't try also to convert a pure EBCDIC file on a non-mainframe platform : TXT2XML will never find the XML element delimiters '<', '/' and '>' or the COBOL equivalent of an XML element. In all cases, conversion will stop with an error message.

### 3.2 COBOL compilers

Even if COBOL is standardized, internal representation of signed and binary data may differ from a compiler to another, from a platform to another one, etc. That's why, a small COBOL program was written to create binary, packed-decimal and numeric reference data. On IBM mainframe running MVS, compiling it is easy, using a one of the 'Enterprise COBOL for Z/OS' product family. But on Windows and Linux/Unix, there are few compilers, with a active project status, available. One of them is : Tiny-Cobol that you can download from :

<http://tiny-cobol.sourceforge.net/>

Tiny-Cobol runs on :

- BeOS,
- FreeBSD,
- Linux,
- Windows (with MinGW or Cygwin).

### 3.3 Internal representation

The results of execution of this program on the 3 platforms is showed below :

Figure 3.1: Binary, packed-decimal and numeric internal representation

			Internal representation		
Numeric type	COBOL clauses	Value	MVS	Linux/Unix	Windows
External decimal					
	PIC 9999 DISPLAY	1234	F1 F2 F3 F4	31 32 33 34	31 32 33 34
	PIC S9999 DISPLAY	+1234	F1 F2 F3 C4	31 32 33 44	31 32 33 44
		-1234	F1 F2 F3 D4	31 32 33 D4	31 32 33 D4
	PIC S9999 DISPLAY SIGN LEADING	+1234	C1 F2 F3 F4	41 32 33 34	41 32 33 34
		-1234	D1 F2 F3 F4	D1 32 33 34	D1 32 33 34
	PIC S9999 DISPLAY SIGN TRAILING	+1234	F1 F2 F3 C4	31 32 33 44	31 32 33 44
		-1234	F1 F2 F3 D4	31 32 33 D4	31 32 33 D4
	PIC S9999 DISPLAY SIGN LEADING SEPARATE	+1234	4E F1 F2 F3 F4	4E 31 32 33 34	4E 31 32 33 34
		-1234	60 F1 F2 F3 F4	60 31 32 33 34	60 31 32 33 34
	PIC S9999 DISPLAY SIGN TRAILING SEPARATE	+1234	F1 F2 F3 F4 4E	31 32 33 34 4E	31 32 33 34 4E
		-1234	F1 F2 F3 F4 60	31 32 33 34 60	31 32 33 34 60
Binary					
	PIC 9999 COMP-4	1234	04 D2	D2 04	D2 04
	PIC S9999 COMP-4	+1234	04 D2	D2 04	D2 04
		-1234	FB 2E	2E FB	2E FB
Packed-decimal					
	PIC 9999 COMP-3	1234	01 23 4F	01 23 4F	01 23 4F
	PIC S9999 COMP-3	+1234	01 23 4C	01 23 4C	01 23 4C
		-1234	01 23 4D	01 23 4C	01 23 4C

### 3.4 Low-values, high-values and spaces

By default, space-only contents are ignored in XML. To preserve them, they are included in a CDATA tag. High-values (x'FF') and low-values (x'00') are escaped : i.e. converted in a tag like `&#0;` or `&#255;` respectively.

### 3.5 Functions

To convert text data to and from binary, packed-decimal and numeric data, six functions have been written :

#### **txt2bin**

It converts text data to binary. The parameters of the function are

- **text** is the text to convert.
- **length** is the length in bytes of binary data. May only be 2, 4 or 8.
- **signed** is a flag to tell that the binary item is signed. May only be "Y" or "N".

If text data is signed and negative, then text data is converted to binary and two's complement is performed. Returned data is justified to the right and padded with x'00' or x'FF' according to the sign. Finally, binary data is reversed for little endian systems.

#### **txt2pack**

It converts text data to packed-decimal. The parameters of the function are :

- **text** is the text to convert.
- **length** is the length in bytes of packed-decimal data.
- **signed** is a flag to tell that the packed-decimal item is signed. May only be "Y" or "N".

If necessary, a leading zero is added to the packed-decimal data so that the number of half-bytes is always odd. Returned data is justified to the right and padded with x'00'.

#### **txt2num**

It converts text data to numeric. The parameters of the function are :

- **text** is the text to convert.
- **length** is the length in bytes of numeric data.
- **signed** is a flag to tell that the numeric item is signed. May only be "Y" or "N".

- **sign position** is a flag to tell where the sign is located. Allowed values are : " " (empty quotes - for unsigned data), "L" (for LEADING sign), "T" (for TRAILING sign), "LS" (for LEADING SEPARATE sign) and "TS" (for TRAILING SEPARATE sign).

Non-numeric data, like \$ or '.', is converted in hexadecimal. Returned data is justified to the right and padded with zeroes.

### **bin2txt**

It converts binary data to text. The parameters of the function are

- **bin** is the binary to convert.
- **length** is the length in bytes of binary data. May only be 2, 4 or 8.
- **signed** is a flag to tell that the binary item is signed. May only be "Y" or "N".

Firstly, binary data is reversed for little endian systems. If it is signed and negative, then two's complement is performed before converting it to text. Leading zeroes are removed and eventually the sign is added before returning the text data.

### **pack2txt**

It converts packed-decimal data to text. The parameter of the function is :

- **pack** is the packed-decimal data to convert.

Sign is extracted from the last half-byte of the packed-decimal. Leading zeroes are removed and eventually the sign is added before returning the text data.

### **num2txt**

It converts numeric data to text. The parameters of the function are :

- **num** is the numeric data to convert.
- **signed** is a flag to tell that the binary item is signed. May only be "Y" or "N".
- **sign position** is a flag to tell where the sign is located. Allowed values are : " " (empty quotes - for unsigned data), "L" (for LEADING sign), "T" (for TRAILING sign), "LS" (for LEADING SEPARATE sign) and "TS" (for TRAILING SEPARATE sign).

Non-numeric data, like \$ or '.', is converted from hexadecimal to characters. Leading zeroes are removed and eventually the sign is added before returning the text data.

# Chapter 4

## Installation

### 4.1 Downloading TXT2XML

Download the latest TXT2XML version from :

**`http://sourceforge.net/projects/txt2xml-rexx/`**

and corresponding to your platform :

File	Platform
txt2xml.v*r*.mvs.zip	MVS
txt2xml.v*r*.win.zip	Windows
txt2xml.v*r*.unix.tar.gz	Linux/Unix

where \* are the version and release numbers. Create a new directory and decompress the archive file using your favorite decompression tool. For Unix environment, type in a console :

```
tar xvzf txt2xml.v*r*.unix.tar.gz
```

For MVS platform, the zip archive contains 2 files : a 'readme.txt' and an XMIT file that you will upload to the mainframe (see below).

### 4.2 Under MVS

#### 4.2.1 Installation

The installation is straight forward :

1. Before uploading TXT2XML on the mainframe, you need to allocate a new dataset with the following characteristics : *recfm=fb* and *lrecl=80*,
2. Unzip the TXT2XML archive and upload **in binary mode** the TXT2XML.XMIT file to this dataset on your MVS,
3. In TSO, issue the following command :

```
RECEIVE INDATASET(your.TXT2XML.XMIT.dataset)
```

4. Hit enter when it prompts you for restoring parameters. This will create a new dataset,
5. Follow the instructions in the README member of this new dataset. Typically, you will have to exec a Rexx called "receive" that will create the TXT2XML EXEC, CNTL and PANEL datasets.

### 4.2.2 Tests

To run the IVP job, edit the TXT2XML member in the TXT2XML.CNTL dataset and change in the entire member :

- USERID to your TSO userid,
- HLQ to the high level qualifier of TXT2XML.

and submit the job. The JCL step names ending with KO should end with a RC = 12 and the JCL step names ending with OK should end with a RC = 0 or 4.

### 4.2.3 Post-installation

If you want, you can copy :

- the TXT2XML.PANEL(TXT2XML) to your ISPF panel dataset.
- the TXT2XML.EXEC(TXT2XML) to your ISPF EXEC or REXX dataset.

## 4.3 Under Windows

### 4.3.1 Regina or Reginald Rexx

This step is mandatory. You must first download and install the latest Windows version of Regina Rexx from :

<http://sourceforge.net/projects/regina-rexx/>

You can alternatively download and install Reginald Rexx from :

<http://www.borg.com/~jglatt/rexx/reginald/reginald.htm>

Don't forget to add Rexx directory to the *PATH* environment variable (to access it : Start => Parameters => Configuration Panel => System => Advanced => Environment Variables).

### 4.3.2 Java

This step is optional. Starting with version 1.25, a JAVA GUI front-end for non-mainframe platforms has been written. If you want to use it, you must install Java. See section 12.2 for more details.

### 4.3.3 Tests

In the TXT2XML directory, execute in a DOS command interface window the TXT2XML.BAT file. The first five tests should end with an error, the others not.

## 4.4 Under Unix/Linux and other platforms

### 4.4.1 Regina

You must first download and install the latest version of Regina Rexx corresponding to your OS from :

**<http://sourceforge.net/projects/regina-rexx/>**

The Regina binary should be installed in `/usr/bin/` . If it is not the case, modify in the rexx scripts (`txt2xml.rexx` & `xml2cob.rexx`), the magic number `#!/usr/bin/rexx` according to your needs.

### 4.4.2 Java

This step is optional. Starting with version 1.25, a JAVA GUI front-end for non-mainframe platforms has been written. If you want to use it, you must install Java. See section 12.2 for more details.

### 4.4.3 Tests

In the TXT2XML directory, execute in a shell the `./txt2xml.sh` script. The first five tests should end with an error, the others not.



# Chapter 5

## Syntax

The TXT2XML syntax is :

Figure 5.1: TXT2XML syntax

```
txt2xml
    TXT txt-file
    COB cob-file
    XML xml-file
    FORMAT format
    PREFIX prefix
    DTD dtd-file
    VERBOSE
    BROWSE
```

All parameters can be typed in any case (upper, lower or mixed) and only in this order. But, on some case sensitive operating systems like Linux/Unix, file names must be typed with the correct case and the TXT2XML command must be typed in lower case. The file names may include space characters. TXT2XML parameters must be only one line, except if your operating systems allows continuation characters.

### 5.1 Mandatory parameters

#### 5.1.1 TXT txt-file

It's the name of the input or output (depending on the direction of conversion) text file. There is no default value for this parameter.

#### Under MVS

'txt-file' can be one of the following :

- a sequential dataset,

- a member of a partitioned dataset,
- a DDname. In this case, the DDname must be preceded by 'DD:'.

Before running TXT2XML, txt-file must exist, except when doing an XML to text conversion under ISPF.

#### For all other platforms

txt-file has to be a correct file name. The file must exist before running TXT2XML for a text to XML conversion.

### 5.1.2 COB cob-file

It's the name of the COBOL copybook file. There is no default value for this parameter.

#### Under MVS

'cob-file' can be one of the following :

- a sequential dataset,
- a member of a partitioned dataset,
- a DDname. In this case, the DDname must be preceded by 'DD:'.

Before running TXT2XML, cob-file must **always** exist.

#### For all other platforms

cob-file has to be a correct file name. The file must **always** exist before running TXT2XML. Since TXT2XML does not handle some COBOL clauses such as *REDEFINES*, etc, it is best to 'clean up' the copybook by removing all COBOL clauses not handled by the TXT2XML routines.

### 5.1.3 XML xml-file

It's the name of the input or output (depending on the direction of conversion) XML file. There is no default value for this parameter.

#### Under MVS

'xml-file' can be one of the following :

- a sequential dataset,
- a member of a partitioned dataset,
- a DDname. In this case, the DDname must be preceded by 'DD:'.

Before running TXT2XML, xml-file must exist, except when doing a text to XML conversion under ISPF.

**For all other platforms**

xml-file has to be a correct file name. The file must exist before running TXT2XML for an XML to text conversion.

**5.1.4 FORMAT format**

The value for this parameter may only be (all platforms) :

- X for text to XML conversion,
- T to XML to text conversion.

There is no default value for this parameter.

**5.2 Optional parameters****5.2.1 BROWSE**

The default value for this parameter is no.

**Under MVS**

Only under ISPF and if present, will browse the conversion results report.

**For all other platforms**

On all other platforms, this parameter is ignored.

**5.2.2 VERBOSE**

If present, displays all the messages. The default value is no, all messages are not displayed.

**5.2.3 DTD dtd-file**

This parameter is only valid during text to XML conversion. It's the name of an optional output DTD file or the value 'INTERNAL' for a DTD included in the XML file. There is no default value for this parameter. if omitted, no DTD will be created.

**Under MVS**

'dtd-file' can be one of the following :

- a sequential dataset,
- a member of a partitioned dataset,
- a DDname. In this case, the DDname must be preceded by 'DD:'.

Before running TXT2XML under ISPF, dtd-file must not exist.

**For all other platforms**

dtd-file has to be a correct file name.

**5.2.4 PREFIX prefix**

If present :

- During text to XML conversion, if prefix value is 'COBOL-' then the COBOL item COBOL-ITEM-01 will be converted to <ITEM-01> .
- During XML to text conversion, if prefix value is 'COBOL-' then the XML element <ITEM\_01> will be checked against the COBOL item COBOL-ITEM-01.

There is no default value for this parameter.

**Under MVS batch (JCL)**

The dash ('-') used in COBOL item name, like COBOL-ITEM-01, is taken for a continuation character in JCL. If you use prefix within a batch job, replace all dashes ('-') in the COBOL item name by underscores ('\_').

## Chapter 6

# Examples

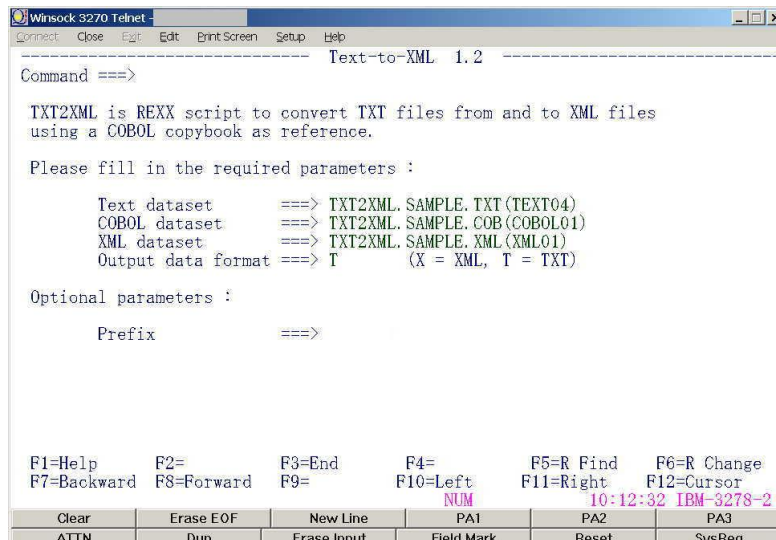
You will find here some examples of TXT2XML. You will find more examples in the test script files (see section 4.2.2, section 4.4.3, or section 4.3.3) or in the IVP job.

### 6.1 MVS

#### 6.1.1 ISPF

Just type 'exec' in front of the TXT2XML member of the EXEC dataset and the ISPF panel will appear :

Figure 6.1: ISPF panel



```
----- Text-to-XML 1.2 -----
Command ==>

TXT2XML is REXX script to convert TXT files from and to XML files
using a COBOL copybook as reference.

Please fill in the required parameters :

Text dataset      ==> TXT2XML.SAMPLE.TXT(TEXT04)
COBOL dataset     ==> TXT2XML.SAMPLE.COB(COB0101)
XML dataset       ==> TXT2XML.SAMPLE.XML(XML01)
Output data format ==> T      (X = XML, T = TXT)

Optional parameters :

Prefix           ==>

F1=Help      F2=      F3=End      F4=      F5=R Find      F6=R Change
F7=Backward  F8=Forward F9=      F10=Left F11=Right F12=Cursor
NUM          10:12:32 IBM-3278-2

Clear Erase EOF New Line PA1 PA2 PA3
ATTN Dup Erase Input Field Mark Reset SvsRed
```

Fill the required parameters, hit enter and you will see :

Figure 6.2: Browsing result dataset

```

Winsock 3270 Telnet
Connect Close Edit Print Screen Setup Help
Menu Utilities Compilers Help

BROWSE      .TXT2XML.REPORT                      Line 00000000 Col 001 076
***** Top of Data *****
Text from & to XML Conversion Utility. Version: 1.2

==>Warning : in the COBOL copybook, level 66, 77, 88 items are ignored.
==>Warning : in the COBOL copybook, level 66, 77, 88 items are ignored.
==>Warning : in the COBOL copybook, level 66, 77, 88 items are ignored.

Txt File:      TXT2XML.SAMPLE.TXT (TEXT04)
Cobol File:    TXT2XML.SAMPLE.COB (COBOL01)
XML File:      TXT2XML.SAMPLE.XML (XML01)
Output data format: TXT
Text records:  10 processed in 0.88 seconds

=====
                                COBOL copybook analysis
=====
! Level ! Name                                     ! Type ! Start ! Length ! Def
-----
Command ==>
Scroll ==> CSR
NUM 10:05:43 IBM-3278-2
Clear Erase EOF New Line PA1 PA2 PA3
ATTN Dup Erase Input Field Mark Reset SvsReq

```

### 6.1.2 Batch

If you submit the following JCL :

Figure 6.3: Job to be submitted

```

Winsock 3270 Telnet
Connect Close Edit Print Screen Setup Help
File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT      .TXT2XML.CNTL(TEMP) - 01.01              Columns 00001 00072
***** Top of Data *****
000001 //USERIDR JOB USERIDR,ABC,CLASS=V,MSGCLASS=Z,REGION=OM,NOTIFY=USERID
000002 /*
000003 //TXT2XML EXEC PGM=IKJEFT1B,DYNAMNBR=50
000004 //SYSEXEC DD DISP=SHR,DSN=HLQ.EXEC
000005 //SYSPRINT DD SYSOUT=*
000006 //SYSTSPRT DD SYSOUT=*
000007 //TXT DD DSN=HLQ.SAMPLE.TXT(MISSING),DISP=SHR
000008 //COB DD DSN=HLQ.SAMPLE.COB(COBOL01),DISP=SHR
000009 //XML DD DSN=HLQ.SAMPLE.XML(MISSING),DISP=SHR
000010 //SYSTSIN DD *
000011 TXT2XML +
000012 TXT DD:TXT COB DD:COB XML DD:XML FORMAT TXT PREFIX XML_
000013 /*
000014 //PRINT EXEC PGM=IEBGENER
000015 //SYSUT1 DD DSN=HLQ.SAMPLE.TXT(MISSING),DISP=SHR
000016 //SYSUT2 DD SYSOUT=*
000017 //SYSPRINT DD SYSOUT=*
000018 //SYSIN DD DUMMY
Command ==>
Scroll ==> CSR
NUM 10:40:20 IBM-3278-2
Clear Erase EOF New Line PA1 PA2 PA3
ATTN Dup Erase Input Field Mark Reset SvsReq

```

You will get :

Figure 6.4: Resulting job log

```

Winsock 3270 Telnet -
Connect Close Edit Print Screen Setup Help

Display Filter View Print Options Help

SDSF OUTPUT DISPLAY R JOB24165 DSID 4 LINE 36 COLS 02- 81
COMMAND INPUT ==>
IEF374I STEP/PRINT /STOP 2004257.1031 CPU OMIN 00.03SEC SRB OMIN 00.00S
IEF375I JOB/ R/START 2004257.1031
IEF376I JOB/ R/STOP 2004257.1031 CPU OMIN 00.34SEC SRB OMIN 00.00S
READY
TXT2XML TXT DD:TXT COB DD:COB XML DD:XML FORMAT TXT PREFIX XML_
TXT2XML: Text from & to XML Conversion Utility. Version: 1.2
TXT2XML:
TXT2XML: ==>Warning : in the COBOL copybook, level 66, 77, 88 items are ignored.
TXT2XML: ==>Warning : in the COBOL copybook, level 66, 77, 88 items are ignored.
TXT2XML: ==>Warning : in the COBOL copybook, level 66, 77, 88 items are ignored.
TXT2XML: ==>Warning : in text record number 1, COBOL item COBOL-ITEM-02 startin
TXT2XML: is filled with default values because the corresponding XML equivalent
TXT2XML: ==>Warning : in text record number 1, COBOL item COBOL-ITEM-04 startin
TXT2XML: is filled with default values because the corresponding XML equivalent
TXT2XML: ==>Warning : in text record number 1, COBOL item COBOL-ITEM-06 startin
TXT2XML: is filled with default values because the corresponding XML equivalent
TXT2XML: ==>Warning : in text record number 1, COBOL item COBOL-ITEM-04 startin
TXT2XML: is filled with default values because the corresponding XML equivalent
TXT2XML: ==>Warning : in text record number 1, COBOL item COBOL-ITEM-05 startin
TXT2XML: ==>Warning : in text record number 1, COBOL item COBOL-ITEM-05 startin
NUM 10:32:24 IBM-3278-2

```

Clear	Erase EOF	New Line	PA1	PA2	PA3
ATTN	Dup	Erase Input	Field Mark	Reset	SvsReq

## 6.2 Windows

In a DOS command interface window and in the TXT2XML directory, just type :

Figure 6.5: Windows example

```

D:\REXX\txt2xml>txt2xml.rexx txt .\sample\txt\database cob
.\sample\cob\database xml .\sample\xml\database format txt

```

and you will get :

Figure 6.6: Resulting message log

```

TXT2XML: Text from & to XML Conversion Utility. Version: 1.2
TXT2XML:
TXT2XML: ==>Warning : in text record number 1 , COBOL item ADMINISTRATOR starting at 115
TXT2XML: is filled with default values because the corresponding XML equivalent was not found
TXT2XML: ==>Warning : in text record number 1 , COBOL item EMAILALIAS starting at 135
TXT2XML: is filled with default values because the corresponding XML equivalent was not found
TXT2XML: ==>Warning : in text record number 1 , COBOL item EXTENSION starting at 145
TXT2XML: is filled with default values because the corresponding XML equivalent was not found
TXT2XML: ==>Warning : in text record number 1 , COBOL item ADMINISTRATOR starting at 149
TXT2XML: is filled with default values because the corresponding XML equivalent was not found
TXT2XML: ==>Warning : in text record number 1 , COBOL item EMAILALIAS starting at 169
TXT2XML: is filled with default values because the corresponding XML equivalent was not found
TXT2XML: ==>Warning : in text record number 1 , COBOL item EXTENSION starting at 179
TXT2XML: is filled with default values because the corresponding XML equivalent was not found
TXT2XML: ==>Warning : in text record number 2 , COBOL item ADMINISTRATOR starting at 149
TXT2XML: is filled with default values because the corresponding XML equivalent was not found
TXT2XML: ==>Warning : in text record number 2 , COBOL item EMAILALIAS starting at 169
TXT2XML: is filled with default values because the corresponding XML equivalent was not found
TXT2XML: ==>Warning : in text record number 2 , COBOL item EXTENSION starting at 179
TXT2XML: is filled with default values because the corresponding XML equivalent was not found
TXT2XML: ==>Warning : in text record number 3 , COBOL item EXTENSION starting at 179
TXT2XML: is filled with default values because the corresponding XML equivalent was not found
TXT2XML:
TXT2XML: Txt File: .\sample\txt\database
TXT2XML: Cobol File: .\sample\cob\database
TXT2XML: XML File: .\sample\xml\database
TXT2XML: Output data format: TXT
TXT2XML: Text records: 3 processed in .29000 seconds
...

```

### 6.3 Unix/Linux

In a shell, just type in lowercase in the TXT2XML directory :

Figure 6.7: Unix/Linux example.

```

./txt2xml.rexx txt ./sample/txt/text01 cob
./sample/cob/cob0101 xml ./sample/xml/xml01 format xml

```

The XML file should look like :



Figure 6.8: Resulting XML file

```

<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!--
Created by TXT2XML on 12 Sep 2004 at 21:46:39
More info on : http://sourceforge.net/projects/txt2xml-rexx/
-->
<DATA>

  <COBOL_ITEM_01>
    <COBOL_ITEM_02>12<COBOL_ITEM_02>45</COBOL_ITEM_02>
    <COBOL_ITEM_02>67<COBOL_ITEM_02>90</COBOL_ITEM_02>
    <COBOL_ITEM_03>
      <COBOL_ITEM_04>1<COBOL_ITEM_04>3</COBOL_ITEM_04>
      <COBOL_ITEM_05>4</COBOL_ITEM_05>
      <COBOL_ITEM_06>5</COBOL_ITEM_06>
    </COBOL_ITEM_03>
    <COBOL_ITEM_03>
      <COBOL_ITEM_04>678</COBOL_ITEM_04>
      <COBOL_ITEM_05>9</COBOL_ITEM_05>
      <COBOL_ITEM_06>0</COBOL_ITEM_06>
    </COBOL_ITEM_03>
    <COBOL_ITEM_03>
      <COBOL_ITEM_04>123</COBOL_ITEM_04>
      <COBOL_ITEM_05>4</COBOL_ITEM_05>
      <COBOL_ITEM_06>5</COBOL_ITEM_06>
    </COBOL_ITEM_03>
    <COBOL_ITEM_03>
      <COBOL_ITEM_04>678</COBOL_ITEM_04>
      <COBOL_ITEM_05>9</COBOL_ITEM_05>
      <COBOL_ITEM_06>0</COBOL_ITEM_06>
    </COBOL_ITEM_03>
    <COBOL_ITEM_08>1234</COBOL_ITEM_08>
    <COBOL_ITEM_09>567890</COBOL_ITEM_09>
    <COBOL_ITEM_12>
      <COBOL_ITEM_13>12345</COBOL_ITEM_13>
      <COBOL_ITEM_13>67890</COBOL_ITEM_13>
      <COBOL_ITEM_13>12345</COBOL_ITEM_13>
      <COBOL_ITEM_13>67890</COBOL_ITEM_13>
      <COBOL_ITEM_14>123456789012345</COBOL_ITEM_14>
      <COBOL_ITEM_15>6789</COBOL_ITEM_15>
    </COBOL_ITEM_12>
  </COBOL_ITEM_01>
  ...
</DATA>

```

with this message log :

Figure 6.9: Log of the conversion

```

TXT2XML: Text from & to XML Conversion Utility. Version: 1.2
TXT2XML:
TXT2XML: ==>Warning : in the COBOL copybook, level 66, 77, 88 items are ignored.
TXT2XML: ==>Warning : in the COBOL copybook, level 66, 77, 88 items are ignored.
TXT2XML: ==>Warning : in the COBOL copybook, level 66, 77, 88 items are ignored.
TXT2XML:
TXT2XML: Txt File:                ./sample/txt/text01
TXT2XML: Cobol File:              ./sample/cob/cobol01
TXT2XML: XML File:                ./sample/xml/xml01
TXT2XML: Output data format: XML
TXT2XML: Text records:           10 processed in .03286 seconds
TXT2XML:
TXT2XML: =====
TXT2XML:
TXT2XML:                                COBOL copybook analysis
TXT2XML:
TXT2XML: +-----+-----+-----+-----+-----+-----+-----+
TXT2XML: | Level | Name                               | Type | Start | Length | Default value |
TXT2XML: +-----+-----+-----+-----+-----+-----+-----+
TXT2XML: | 1     | COBOL-ITEM-01                     | GROUP | 1     | 0     |                |
TXT2XML: | 2     | FILLER                           | CHAR  | 1     | 1     |                |
TXT2XML: | 2     | COBOL-ITEM-02                     | CHAR  | 2     | 5     | AB"CD          |
TXT2XML: | 2     | COBOL-ITEM-02                     | CHAR  | 7     | 5     | AB"CD          |
TXT2XML: | 2     | COBOL-ITEM-03                     | GROUP | 12    | 0     |                |
TXT2XML: | 3     | COBOL-ITEM-04                     | CHAR  | 12    | 3     |                |
TXT2XML: | 3     | COBOL-ITEM-05                     | CHAR  | 15    | 1     |                |
TXT2XML: | 3     | COBOL-ITEM-06                     | NUM   | 16    | 1     |                |
TXT2XML: | 2     | COBOL-ITEM-03                     | GROUP | 17    | 0     |                |
TXT2XML: | 3     | COBOL-ITEM-04                     | CHAR  | 17    | 3     |                |
TXT2XML: | 3     | COBOL-ITEM-05                     | CHAR  | 20    | 1     |                |
TXT2XML: | 3     | COBOL-ITEM-06                     | NUM   | 21    | 1     |                |
TXT2XML: | 2     | COBOL-ITEM-03                     | GROUP | 22    | 0     |                |
TXT2XML: | 3     | COBOL-ITEM-04                     | CHAR  | 22    | 3     |                |
TXT2XML: | 3     | COBOL-ITEM-05                     | CHAR  | 25    | 1     |                |
TXT2XML: | 3     | COBOL-ITEM-06                     | NUM   | 26    | 1     |                |
TXT2XML: | 2     | COBOL-ITEM-03                     | GROUP | 27    | 0     |                |
TXT2XML: | 3     | COBOL-ITEM-04                     | CHAR  | 27    | 3     |                |
TXT2XML: | 3     | COBOL-ITEM-05                     | CHAR  | 30    | 1     |                |
TXT2XML: | 3     | COBOL-ITEM-06                     | NUM   | 31    | 1     |                |
TXT2XML: | 2     | COBOL-ITEM-08                     | NUM   | 32    | 4     |                |
TXT2XML: | 2     | COBOL-ITEM-09                     | CHAR  | 36    | 6     |                |
TXT2XML: | 2     | COBOL-ITEM-12                     | GROUP | 42    | 0     |                |
TXT2XML: | 3     | COBOL-ITEM-13                     | NUM   | 42    | 5     |                |
TXT2XML: | 3     | COBOL-ITEM-13                     | NUM   | 47    | 5     |                |
TXT2XML: | 3     | COBOL-ITEM-13                     | NUM   | 52    | 5     |                |
TXT2XML: | 3     | COBOL-ITEM-13                     | NUM   | 57    | 5     |                |
TXT2XML: | 3     | COBOL-ITEM-14                     | CHAR  | 62    | 15    |                |
TXT2XML: | 3     | COBOL-ITEM-15                     | NUM   | 77    | 4     | 0000          |
TXT2XML: +-----+-----+-----+-----+-----+-----+-----+
TXT2XML:
TXT2XML: Total computed COBOL length : 80

```

## Part II

# X M L 2 C O B

## Chapter 7

# Introduction

XML2COB is a complementary tool of TXT2XML. It generates a COBOL copybook, using an XML file as template. Of course, you can use the generated COBOL copybook to convert, with TXT2XML, XML files to text files and vice-versa. However, like TXT2XML, it is not an XML parser written in Rexx, nor will support the entire XML specification.

XML2COB works only with well-formed XML documents. For example, it will never check for missing XML closing tags. XML2COB generates valid COBOL copybooks, but it's still your responsibility to check COBOL clauses like :

- OCCURS
- PICTURE (symbol and length of the item).
- SIGN

As TXT2XML, XML2COB was tested successfully on :

- Windows 2K,
- Linux Mandrake 10.1,
- MVS (OS/390 V2R10).

It should work without problems on other platforms. To run XML2COB on non-mainframe platforms, you must first install Regina Rexx or for windows only, Reginald (see chapter 4). Optionally, for the GUI front-end, you may also install Java (see 12.2 for more details). Should you succeed or encounter problems in running XML2COB on a non-tested platform, please send an e-mail to :

**`sunuraxi@users.sourceforge.net`**

XML2COB is released under the GPL license (see appendix A).

## Chapter 8

# Principles of generation

Except for some constraints (see section 8.4), there is no special requirements about the XML file used for the generation. However, there is a gold rule : the bigger the XML file is, more accurate will be the generated COBOL copybook. The reason is very simple : a bigger file means more tags, more contents analyzed, and more accuracy in ...

### 8.1 The meta-data

The following meta-data are extracted from the XML elements, attributes and content :

Meta-data	Usage
Element name	To create the COBOL item names
Type of content	To create the PICTURE clause
Sign position	To create SIGN clause
Length	To create the PICTURE clause
Occurrence	To create the OCCURS clause

### 8.2 The COBOL copybook generation

In an XML file, the elements can be classified in 3 categories :

- **Root** : the first XML element. It's ignored in the generation process and is not part of the generated COBOL copybook.
- **First child** : the second XML element. Each time it appears, the meta-data analysis of the XML elements is (re-)started. It corresponds to the first COBOL item in the copybook. As this first child element includes all the other XML elements, its corresponding COBOL item is a grouping item, with the lowest level and a unique occurrence.
- **Other child elements**: the other XML elements. They are analyzed to extract the meta-data needed to build the rest of the COBOL copybook.

The COBOL copybook generation process take place in the following steps :

- The XML file is read record per record until the end of the file.
- The first child is then searched.
- Each XML element or attributes generates a COBOL item applying the following rules :
  - The name of the XML element or attribute is capitalized and dashes (" - ") are translated to underscore (" \_ "). For example : `<cobol_item_01>` becomes `COBOL-ITEM-01`. COBOL item names are checked against a COBOL reserved word list<sup>1</sup>. A warning message is issued if the generated item name appears in this list,
  - Meta-data of the content<sup>2</sup> of the XML element and attributes is extracted : length, type and sign.
  - Length of item is rounded if requested (see sub-section 10.2.3). A COBOL item is then created.
- Each time the XML first child is found, the analysis is restarted and the meta-data is updated if necessary. For example, if the content length of an XML element is greater than the current length of its corresponding COBOL item, the length is updated. The contrary is, of course, not true.
- When the end of the XML file is reached :
  - The item list is then checked against elementary or grouping items occurring more than one time. If this is the case, only the first occurrence of the item is kept, the others are deleted and an occurrence counter is created
  - The starting position of each COBOL item is computed. Although, it's not necessary for the COBOL copybook, it is used for final reporting.
  - The COBOL copybook is written, indenting the items according to their COBOL level.

### 8.3 COBOL level

COBOL levels are not part of the XML file. They are computed using the following rules :

- At each XML first child, level starts at 01 or 02 according to the *level01* parameter (see sub-section 10.2.2)
- The level of COBOL item increases :
  - at each XML element
  - at each XML content

---

<sup>1</sup>according to IBM's "Enterprise COBOL for z/OS V3R3 Language Reference" (IGY3LR20)

<sup>2</sup>Content is escaped if necessary. Characters like &lt; &gt; " and ' are converted to &amp; &lt; &gt; &quot; &apos; respectively.

- at each XML attribute
- The level of COBOL item decreases :
  - at each XML empty element
  - after each XML content
  - after each XML closing tag,
  - after each XML attribute.

For example, with the following XML file,

Figure 8.1: an XML file with attributes and content

```
<?xml version="1.0"?>
<DatabaseInventory>
  <DatabaseName>

    <DatabaseDomain>iDevelopment.info</DatabaseDomain>
    <Administrator EmailAlias="jhunter" Extension="6007">Jeffrey Hunter
  </Administrator>
</DatabaseName>
</DatabaseInventory>
```

XML2COB will generate the following COBOL copybook :

Figure 8.2: Generated COBOL copybook

```
02 DATABASENAME.
  04 DATABASEDOMAIN PICTURE X(17).
  04 ADMINISTRATOR.
    06 EMAILALIAS PICTURE X(7).
    06 EXTENSION PICTURE 9(4).
    06 ADMINISTRATOR PICTURE X(14).
```

As you can see, the COBOL item *ADMINISTRATOR* appears 2 times :

1. The first time as a grouping item, because the XML element *Administrator* has attributes
2. The second time as an elementary item at a lower level than the grouping item, because the XML element *Administrator* also has a content.

## 8.4 Constraints

In some cases, generating a COBOL copybook from an XML file is impossible. For example, this XML file :

Figure 8.3: Nonconvertible XML file

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<weather>
  <city name='Los Angeles'>
    <sky>Blue</sky>
    Hot and sunny
  </city>
</weather>
```

In this file, the XML element **city** is a complex one. It includes the simple element **sky** but also a content (**Hot and sunny**). The corresponding COBOL copybook would be something like :

Figure 8.4: Corresponding COBOL copybook

```
01 city pic x(20).
   03 name pic x(20).
   03 sky pic x(8).
```

This is impossible in COBOL : a grouping item can't have a picture clause.<sup>3</sup>

So, not all the XML files can be used and some constraints apply when generating a COBOL copybook :

- Mixed XML content is **not supported**,
- XML entities starting with `<!ENTITY` and ending with `>` are **not supported**. If such a declaration is found, the generation process is aborted,
- Attributes of XML element are supported. But, attributes of the root element are ignored.
- XML attributes like `xml:space` are ignored,
- All the XML tags and contents may expand on more than one line,
- XML headers starting with `<?xml` and ending with `?>` are skipped,
- XML internal and external document type declaration starting with `<!DOCTYPE` and ending with either `>` or `/>` are skipped,

<sup>3</sup>COBOL for MVS & VM V1R2.2 Language Reference (IGYLR204) paragraph 5.3.11 PICTURE Clause : "The PICTURE clause can be specified only at the elementary level."



- XML comments starting with `<!--` and ending with `-->` are skipped,
- Empty XML elements, even with attributes, are supported.

## Chapter 9

# Installation

See chapter 4

# Chapter 10

## Syntax

The XML2COB syntax is :

Figure 10.1: XML2C0B syntax

```
xml2cob
    COB cob-file
    XML xml-file
    PREFIX prefix
    LEVEL01
    ROUND
    VERBOSE
    BROWSE
```

All parameters can be typed in any case (upper, lower or mixed) and only in this order. But, on some case sensitive operating systems like Linux/Unix, file names must be typed with the correct case and the XML2C0B command must be typed in lower case. The file names may include space characters. XML2C0B parameters must be only one line, except if your operating systems allows continuation characters.

### 10.1 Mandatory parameters

#### 10.1.1 COB cob-file

It's the name of the COBOL copybook file. There is no default value for this parameter.

##### Under MVS

'cob-file' can be one of the following :

- a sequential dataset,
- a member of a partitioned dataset,

- a DDname. In this case, the DDname must be preceded by 'DD:'.

Before running XML2COB, 'cob-file' must **not** exist.

#### **For all other platforms**

'cob-file' has to be a correct file name.

### **10.1.2 XML xml-file**

It's the name of the input XML file. There is no default value for this parameter.

#### **Under MVS**

'xml-file' can be one of the following :

- a sequential dataset,
- a member of a partitioned dataset,
- a DDname. In this case, the DDname must be preceded by 'DD:'.

Before running XML2COB, 'xml-file' must exist.

#### **For all other platforms**

'xml-file' has to be a correct file name. The file must exist before running XML2COB.

## **10.2 Optional parameters**

### **10.2.1 PREFIX prefix**

If prefix value is set to 'COBOL-' then the XML element <cobol\_item\_01> will generate a COBOL item called ITEM-01. There is no default value for this parameter.

#### **Under MVS batch (JCL)**

The dash ('-') used in *prefix*, like COBOL-, is taken for a continuation character in JCL. If you use prefix within a batch job, replace all dashes ('-') by underscores ('\_').

### **10.2.2 LEVEL01**

If present, the COBOL copybook will start at level 01. The default value is no level 01, it starts at level 02.

### 10.2.3 ROUND

If present, all COBOL item length will be rounded to the next 5 or 0. For example :

COBOL item length	
Before round	After round
8	10
48	50
113	125
205	250
517	600

The default value is no round, item length is not rounded.

### 10.2.4 BROWSE

The default value for this parameter is no.

#### Under MVS

Only under ISPF and if present, will browse the generation results report.

#### For all other platforms

On all other platforms, this parameter is ignored.

### 10.2.5 VERBOSE

If present, displays all the messages. The default value is no, all messages are not displayed.

# Chapter 11

## Examples

You will find here some examples of XML2COB. You will find more examples in the test script files or in the IVP job (see section 4.2.2, see 4.4.3 or section 4.3.3).

### 11.1 MVS

#### 11.1.1 ISPF

Just type 'exec' in front of the XML2COB member of the EXEC dataset and the ISPF panel will appear :

Figure 11.1: ISPF panel

Clear	Erase EOF	New Line	PA1	PA2	PA3
ATTN	Dup	Erase Input	Field Mark	Reset	SysReq

Fill the required parameters, hit enter and you will see :

Figure 11.2: Browsing result dataset

```

Winsock 3270 Telnet
Connect Close Edit PrintScreen Setup Help
Menu Utilities Compilers Help

BROWSE XML2COB REPORT Line 00000000 Col 001 080
***** Top of Data *****
XML to COBOL Copybook Conversion Utility. Version: 1.0

XML File:      TXT2XML.SAMPLE.XML (XML01)
Cobol File:    TXT2XML.SAMPLE.COB.XML2COB (XML01)
Prefix:        COBOL-
Round:         Y
Level01:       N

XML first child: 11 processed in 3.3712 seconds

=====
COBOL copybook created :

+-----+-----+-----+-----+-----+-----+
| Level | Name      | Type | Occurs | Sign | Start | Length |
+-----+-----+-----+-----+-----+-----+
| 02    | ITEM-01   |      | 1      | N    | 1      | 0       |
+-----+-----+-----+-----+-----+-----+
Command ==>
Scroll ==> CSR

Clear Erase EOF New Line PA1 PA2 PA3
ATTN Dup Erase Input Field Mark Reset SysReq

12:39:40 IBM-3278-2

```

### 11.1.2 Batch

If you submit the following JCL :

Figure 11.3: Job to be submitted

```

Winsock 3270 Telnet
Connect Close Edit PrintScreen Setup Help
File Edit Edit.Settings Menu Utilities Compilers Test Help

EDIT TXT2XML.CNTL (XML2COB) - 01.03 Columns 00001 00072
***** Top of Data *****
000001 //USERIDR JOB USERIDR, ABC, CLASS=V, MSGCLASS=Z, REGION=0M, NOTIFY=USERID
000002 //*JOBPARM LINES=1000
000003 //*
000004 //TEST010K EXEC PGM=IKJEFT1B, DYNAMNBR=50, COND=(16, LT)
000005 //SYSEXEC DD DISP=SHR, DSN=HLQ.EXEC
000006 //SYSPRINT DD SYSOUT=*
000007 //SYSTSPRT DD SYSOUT=*
000008 //COB DD DSN=HLQ.SAMPLE.COB.XML2COB (COB0L01),
000009 // XML DD DISP=SHR
000010 //XML DD DSN=HLQ.SAMPLE.XML (XML01),
000011 // XML DD DISP=SHR
000012 //SYSTSIN DD *
000013 XML2COB +
000014 COB DD:COB +
000015 XML DD:XML +
000016 PREFIX COBOL_ VERBOSE
000017 //*
000018 //PRINT01 EXEC PGM=IEBGENER, COND=(16, LT)
000019 //SYSUT1 DD DSN=HLQ.SAMPLE.COB.XML2COB (COB0L01),
Command ==>
Scroll ==> CSR

Clear Erase EOF New Line PA1 PA2 PA3
ATTN Dup Erase Input Field Mark Reset SysReq

12:51:30 IBM-3278-2

```

You will get :

Figure 11.4: Resulting job log

```

SDSF OUTPUT DISPLAY
JOB18807 DSID 4 LINE 22 COLUMNS 02- 81
COMMAND INPUT ==>
IEF375I JOB/ /START 2005049.1258
IEF376I JOB/ /STOP 2005049.1258 CPU OMIN 01.96SEC SRB OMIN 00.00S
READY
XML2COB COB DD:COB XML DD:XML PREFIX COBOL_ VERBOSE
XML2COB: XML to COBOL Copybook Conversion Utility. Version: 1.0
XML2COB:
XML2COB: XML DD: XML
XML2COB: Cobol DD: COB
XML2COB: Prefix: COBOL-
XML2COB: Round: N
XML2COB: Level01: N
XML2COB: XML first child: 11 processed in 2.1733 seconds
XML2COB:
XML2COB: =====
XML2COB:
XML2COB: COBOL copybook created :
XML2COB:

```

Clear	Erase EOF	New Line	PA1	PA2	PA3
ATTN	Dup	Erase Input	Field Mark	Reset	SysReq

## 11.2 Windows

In a DOS command interface window and in the XML2COB directory, just type :

Figure 11.5: Windows example

```

D:\REXX\txt2xml>xml2cob.rexx cob .\sample\cob\xml2cob\database xml
.\sample\xml\database level01 round verbose

```

and you will get :



Figure 11.6: Resulting message log

```

XML2COB: XML to COBOL Copybook Conversion Utility.  Version: 1.0
XML2COB:
XML2COB:
XML2COB: XML File:                .\sample\xml\database
XML2COB: Cobol File:              .\sample\cob\xml2cob\database
XML2COB: Prefix:
XML2COB: Round:                   Y
XML2COB: Level01:                 Y
XML2COB:
XML2COB: XML first child:         4 processed in .26100 seconds
XML2COB:
XML2COB: =====
XML2COB:
XML2COB:                                COBOL copybook created :
XML2COB:
XML2COB: +-----+-----+-----+-----+-----+-----+-----+
XML2COB: | Level | Name                               | Type | Occurs | Sign | Start | Length |
XML2COB: +-----+-----+-----+-----+-----+-----+-----+
XML2COB: | 01    | DATABASENAME                       |      | 1       | N    | 1     | 0       |
XML2COB: | 03    | GLOBALDATABASENAME                 | CHAR | 1       | N    | 1     | 210     |
XML2COB: | 03    | ORACLESID                          | CHAR | 1       | N    | 211   | 15      |
XML2COB: | 03    | DATABASEDOMAIN                     | CHAR | 1       | N    | 226   | 110     |
XML2COB: | 03    | ADMINISTRATOR                      |      | 3       | N    | 336   | 0       |
XML2COB: | 05    | EMAILALIAS                         | CHAR | 1       | N    | 336   | 15      |
XML2COB: | 05    | EXTENSION                          | NUM  | 1       | N    | 351   | 5       |
XML2COB: | 05    | ADMINISTRATOR                      | CHAR | 1       | N    | 356   | 15      |
XML2COB: | 03    | DATABASEATTRIBUTES                 |      | 1       | N    | 441   | 0       |
XML2COB: | 05    | GENRE                             |      | 1       | N    | 441   | 10      |
XML2COB: | 05    | VERSION                           |      | 1       | N    | 451   | 5       |
XML2COB: | 03    | COMMENTS                           |      | 1       | N    | 456   | 185     |
XML2COB: +-----+-----+-----+-----+-----+-----+-----+
XML2COB:
XML2COB: Total COBOL record length : 640
XML2COB:

```

### 11.3 Unix/Linux

In a shell, just type in lower case in the XML2COB directory :

Figure 11.7: Unix/Linux example.

```
./xml2cob.rexx cob ./sample/cob/xml2cob/book xml ./sample/xml/book verbose
```

The XML file should look like :

Figure 11.8: Resulting COBOL copybook

```
*-----
*
* Created by XML2COB on 20 Feb 2005 at 21:17:12
*
* More info on : http://sourceforge.net/projects/txt2xml-rexx/
*
* Total COBOL record length : 255
*
*-----
02 BOOK.
   04 BOOK-ID                PICTURE X(5) .
   04 BOOK-AUTHOR            PICTURE X(20) .
   04 BOOK-TITLE             PICTURE X(38) .
   04 GENRE                  PICTURE X(15) .
   04 PRICE                  PICTURE 9(5) .
   04 PUBLISH-DATE           PICTURE X(10) .
   04 DESCRIPTION            PICTURE X(162) .
```

with this message log :

Figure 11.9: Log of the generation

```

XML2COB: XML to COBOL Copybook Conversion Utility.  Version: 1.0
XML2COB:
XML2COB:
XML2COB: XML File:                ./sample/xml/book
XML2COB: Cobol File:              ./sample/cob/xml2cob/book
XML2COB: Prefix:
XML2COB: Round:                   N
XML2COB: Level01:                 N
XML2COB:
XML2COB: XML first child:        13 processed in .11121 seconds
XML2COB:
XML2COB: =====
XML2COB:
XML2COB:                                COBOL copybook created :
XML2COB:
XML2COB: +-----+-----+-----+-----+-----+-----+-----+
XML2COB: | Level | Name                               | Type | Occurs | Sign | Start | Length |
XML2COB: +-----+-----+-----+-----+-----+-----+-----+
XML2COB: | 02    | BOOK                               |      | 1      | N    | 1     | 0      |
XML2COB: | 04    | BOOK-ID                           | CHAR | 1      | N    | 1     | 5      |
XML2COB: | 04    | BOOK-AUTHOR                       | CHAR | 1      | N    | 6     | 20     |
XML2COB: | 04    | BOOK-TITLE                        | CHAR | 1      | N    | 26    | 38     |
XML2COB: | 04    | GENRE                             | CHAR | 1      | N    | 64    | 15     |
XML2COB: | 04    | PRICE                             | NUM  | 1      | N    | 79    | 5      |
XML2COB: | 04    | PUBLISH-DATE                      | CHAR | 1      | N    | 84    | 10     |
XML2COB: | 04    | DESCRIPTION                       | CHAR | 1      | N    | 94    | 162    |
XML2COB: +-----+-----+-----+-----+-----+-----+-----+
XML2COB:
XML2COB: Total COBOL record length : 255
XML2COB:

```

# Part III

## The GUI front-end

## Chapter 12

# The Graphical User Interface

### 12.1 Introduction

Although TXT2XML and XML2COB are not GUI tools (see chapter 1), it's perfectly understandable that some people don't like the command line. That's why a GUI front-end was written for non-mainframe platforms. There is a lot of programming techniques for GUI applications available on the Internet and choosing one of them was not simple. Some requirements were established : the GUI front-end had to :

- be portable,
- be easy to write,
- use known standards and software.

Thinlet <http://thinlet.sourceforge.net/> was chosen. It's a GUI toolkit, a single Java class that parses the hierarchy and properties of the GUI, handles user interaction, and calls business logic. It separates the graphic presentation (described in an XML file) and the application methods (written as Java code). Thinlet uses XUL (XML User Interface Language) which is a markup language for describing user interfaces. With XUL you can create rich, sophisticated cross-platform applications easily. Here is an example of an XUL file :

Figure 12.1: XUL sample file.

```

<?xml version="1.0"?>
<panel columns="1">

    <menubar>
    <menu text="File" mnemonic="0">
        <menuitem text="Exit" action="exit"/>
    </menu>
    </menubar>
    <panel columns="3" gap="10" top="10" left="10"
    bottom="10" right="10">

        <label text="Text file :" mnemonic="0" for="TextFile"/>
        <textfield name="TextFile" columns="50"
        tooltip="Enter here the text file name"/>
        <button text="Select..."
        action="openTextFileDialog(thinlet,TextFile)"
        tooltip="Click here to select the text file"/>
        <checkbox name="format" text="XML" group="format"
        tooltip="Click here to convert from text to XML"
        selected="true" colspan="2"
        action="SetFormat(this.text)"/>
        <label/>
        <checkbox text="text" group="format"
        tooltip="Click here to convert from XML to text"
        colspan="2" action="SetFormat(this.text)"/>
        <separator colspan="3" />
        <label text="Prefix :" mnemonic="0" for="Prefix"/>
        <textfield name="Prefix" columns="25"
        tooltip="Enter here the prefix"/>
        <label/>
        <button text="Convert"
        action="CallTXT2XML(TextFile.text,COBOLFile.text,
        XMLFile.text,DTDFile.text,Prefix.text)"
        tooltip="Click here to start the conversion" mnemonic="2"/>
    </panel>
</panel>

```

Of course, if you are a shell or a 'DOS command prompt' fanatic, you can still continue to use TXT2XML and XML2COB Rexx scripts on a command line.

## 12.2 Installation

To use the GUI front-end, you need to download and install the Java Software Developer Kit (Java SDK) :

<http://java.sun.com>

The GUI front-end was tested successfully on Windows and Linux Mandrake 10.1 with J2EE 1.4. Make sure that your path includes the directory where you have installed the Java SDK.

## 12.3 Syntax

To launch the GUI front-end, you can :

- Type on a command line :

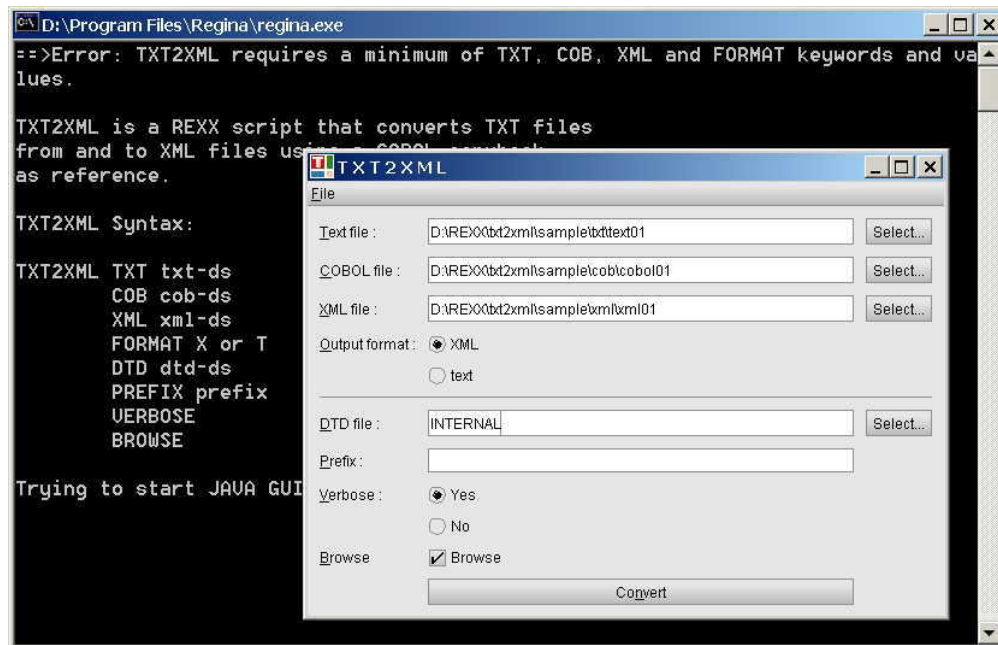
```
java -jar txt2xml.jar
or
java -jar xml2cob.jar
according to which REXX script you want to execute.
```

- or simply execute txt2xml.rexx or xml2cob.rexx without parameters.

## 12.4 Example

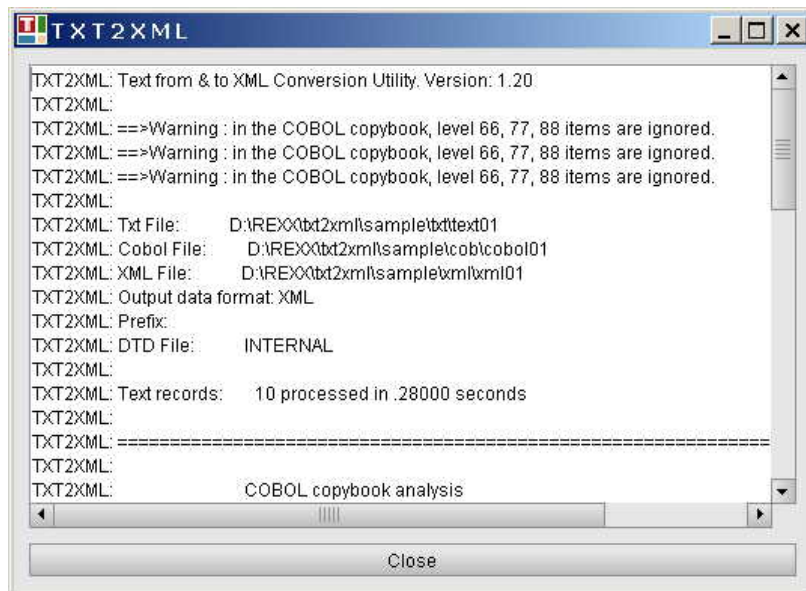
Here is an example when starting TXT2XML without parameters :

Figure 12.2: The GUI front-end



and the result of the conversion :

Figure 12.3: Browsing the results with the GUI front-end





# Appendix A

## License

GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### A.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation’s software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author’s protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the

software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

## A.2 GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for non-commercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code

from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License

incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

### A.3 NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

## A.4 How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>  
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

## Appendix B

# History & Road map

### **B.1 30/09/02 - Version 0.1**

- start,
- indent XML according to the item level.

### **B.2 04/10/02 - Version 0.2**

- handle multi-line COBOL item declaration,
- ignore line numbers in columns 1-6 & 73-80,
- ignore level 66, 77, 88 items,
- stop if level is greater than 50,
- stop if level is not numeric,
- stop if some COBOL reserved words are found,
- replace 9(4) by 9999 and X(3) by XXX, etc.

### **B.3 06/11/02 - Version 0.3**

- handle OCCURS clause for group and elementary items.

### **B.4 21/11/02 - Version 1.0**

- read line by line instead of reading all lines,
- write line by line instead of writing all lines,
- COBOL item names are capitalized.

## B.5 28/07/04 - Version 1.1 RC1

- bug corrected : VALUES COBOL clauses are ignored,
- make conversion in both directions from XML to TXT and from TXT to XML,
- COBOL levels are renumbered from 1 by 1 so that indentation of XML is independent of absolute COBOL levels,
- change input file parameter name to txt and output file parameter name to XML,
- added an "x000 Records processed" message,
- added an error message if the file transfer of TXT2XML has changed vertical bars (concatenation and OR operator) to ! ,
- added a report of COBOL items: level, name, type, start and length,
- during conversion from XML to TXT, check that XML numeric values are really numeric,
- error force termination of the program with return code set to 12.

## B.6 28/08/04 - Version 1.1

- attributes of XML elements are now supported,
- check that all COBOL items are filled during XML to TXT conversion. A warning is issued if it's not the case,
- during XML to text conversion, initialize the output record with the default values specified in the VALUES clause of the COBOL copybook,
- support of CDATA,
- support of escaped chars like &lt; for "<",
- support of element content on more than one line,
- support of comments, declaration, ... on more than one line,
- ignore XML attributes : xml:space, xml:language, ...

## B.7 20/09/04 - Version 1.15

- TXT2XML runs on almost any platform with Regina Rexx
- XML Entity declaration are not supported,
- new parameter PREFIX that suppress constant part of an item name. I.e. if prefix is set to COBOL- then the COBOL item COBOL-ITEM-01 will be translated to <ITEM-01> during text to XML conversion.
- bug corrected : during text to XML conversion, missing tags are now correctly handled.



## B.8 05/11/04 - Version 1.20

- Support of COMP-3, COMP-4, BINARY and PACKED-DECIMAL clauses.
- Support of SIGN clause even for group items.
- bug corrected under MVS : allocation of a dataset with BLOCK as units works correctly now.
- New parameter VERBOSE to replace NOCONFIRM parameter which is deprecated.
- Generation of internal and external DTD based on the COBOL copybook during Text to XML conversion.

## B.9 05/03/05 - Version 1.25

- New JAVA GUI front-end for non-mainframe platforms
- Bug corrected : COBOL picture clauses like 'PIC 99V999' are now correctly handled.
- New web site : <http://txt2xml-rexx.sourceforge.net/>
- New rexx script to generate a COBOL copybook from an XML file : XML2COB

## B.10 Road map

- Generation of XSD files based on COBOL copybooks and vice-versa.
- Integration of XML2COB into TXT2XML.

# Appendix C

## Internet resources

### XML

XML.org	<a href="http://www.xml.org/">http://www.xml.org/</a>
Extensible Markup Language (XML)	<a href="http://www.w3.org/XML/">http://www.w3.org/XML/</a>
The XML FAQ	<a href="http://www.ucc.ie/xml/">http://www.ucc.ie/xml/</a>
XML.com: XML From the Inside Out – XML development, XML resources, XML specifications	<a href="http://www.xml.com">http://www.xml.com</a>
ZVON.org	<a href="http://www.zvon.org/">http://www.zvon.org/</a>

### REXX

The REXX Language Association	<a href="http://www.rexxla.org/">http://www.rexxla.org/</a>
IBM REXX home page	<a href="http://www-306.ibm.com/software/awdtools/obj-rexx/">http://www-306.ibm.com/software/awdtools/obj-rexx/</a>
Reginald REXX Developer's Page for Windows	<a href="http://www.borg.com/~jglatt/rexx/reginald/win32/rxdevw32.htm">http://www.borg.com/~jglatt/rexx/reginald/win32/rxdevw32.htm</a>
Regina - Cross-platform REXX Interpreter	<a href="http://regina-rexx.sourceforge.net/">http://regina-rexx.sourceforge.net/</a>
REXX Anywhere!	<a href="http://www.planetmvs.com/rexxanywhere/">http://www.planetmvs.com/rexxanywhere/</a>
Style Guide for REXX	<a href="http://www.neilhancock.co.uk/Computers/REXXStyleGuide.html">http://www.neilhancock.co.uk/Computers/REXXStyleGuide.html</a>
FTE Text Editor	<a href="http://fte.sourceforge.net/">http://fte.sourceforge.net/</a>

### COBOL

IBM COBOL for OS/390 & VM V2R1 (IGYSH208)	<a href="http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/Shelves/IGYSH208">http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/Shelves/IGYSH208</a>
IBM COBOL - Family Overview - IBM Software	<a href="http://www-306.ibm.com/software/awdtools/cobol/">http://www-306.ibm.com/software/awdtools/cobol/</a>
Kasten COBOL Page: Coding Style and Techniques	<a href="http://home.swbell.net/mck9/cobol/cobol.html">http://home.swbell.net/mck9/cobol/cobol.html</a>
The TinyCOBOL project home page	<a href="http://tiny-cobol.sourceforge.net/">http://tiny-cobol.sourceforge.net/</a>

**Thinlet, Java & XUL**

Java home	<a href="http://java.sun.com">http://java.sun.com</a>
Thinlet home	<a href="http://thinlet.sourceforge.net/">http://thinlet.sourceforge.net/</a>
Thing	<a href="http://thing.sourceforge.net/">http://thing.sourceforge.net/</a>
XUL planet	<a href="http://www.xulplanet.com/">http://www.xulplanet.com/</a>

**Other**

TCP/IP and Mail Tools by Lionel B. Dyck	<a href="http://www.lbdsoftware.com/tcpip.html">http://www.lbdsoftware.com/tcpip.html</a>
Typo generator <sup>1</sup>	<a href="http://mtd.nussnet.at/typo/index2.php">http://mtd.nussnet.at/typo/index2.php</a>
LyX - The Document Processor <sup>2</sup>	<a href="http://www.lyx.org/">http://www.lyx.org/</a>
SDS - Dictionary of the Mainframe World <sup>3</sup>	<a href="http://www.sdsusa.com/dictionary/">http://www.sdsusa.com/dictionary/</a>

---

<sup>1</sup>

The cover page was made with Typo generator.

<sup>2</sup>

This document was written with Lyx.

<sup>3</sup>

Most of glossary definitions come from this site.

## Appendix D

# Glossary

<b>Batch</b>	An accumulation of data brought together for processing or transmission, usually unattended. Less formally, the processing of such data, as opposed to on-line processing where an user is present to respond interactively, one record at a time.
<b>Dataset</b>	An unit of data storage and retrieval consisting of one or more data records. Outside of the IBM mainframe environment, people call them files. Dataset can be partitioned (i.e. containing multiple files) or sequential (i.e. containing only data).
<b>DBCS</b>	stands for Double Byte Character Set. A mainframe way of coding non-latin character sets (i.e. chinese, japanese, etc).
<b>DTD</b>	Document Type Definition. This is a specific markup language, written using SGML.
<b>ISPF/PDF</b>	ISPF/Program Development Facility. ISPF facility providing access to application development services for end-users and programmers. Incorporates C and REXX programming support, and some support for programmable workstations.
<b>GUI</b>	stands for Graphical User Interface. A GUI application must be installed on each PC and makes extensive use of the mouse. By opposition, TUI is Text User Interface. A TUI application needs a connection program called an emulator and use only the keyboard to input data. BUI stands for Browser User Interface. A BUI program is an intranet or extranet application.
<b>HLQ</b>	High Level Qualifier. The left part of a dataset name.
<b>IVP</b>	Installation Verification Procedure. A set of tests to check if a product was correctly installed.
<b>JCL</b>	Job Control Language. The language used to describe the steps of a batch job (files to be used, programs to be run, etc). A generic term, hijacked by IBM to refer to the batch control languages for its System/360 operating systems. Then, as now, z/OS and VSE/ESA JCL is a clumsy and cumbersome system that is hard to learn, full

of inconsistencies, and avoided by anyone with an iota of common sense and access to an alternative. The only excuse for it comes from one of the original OS/360 developers: they ran out of time to build a JCL interpreter, so just used the Assembler macro processor, and built a language (JCL) around it.

**Mainframe** Mainframes used to be defined by their size, and they can still fill a room, cost millions, and support thousands of users. But now a mainframe can also run on a laptop and support two users. So today's mainframes are best defined by their operating systems: Unix and Linux, and IBM's z/OS, OS/390, MVS, VM, and VSE. Mainframes combine four important features:

- Reliable single-thread performance, which is essential for reasonable operations against a database.
- Maximum I/O connectivity, which means mainframes excel at providing for huge disk farms.
- Maximum I/O bandwidth, so connections between drives and processors have few choke-points.
- Reliability—mainframes often allow for "graceful degradation" and service while the system is running.

**Member** A file of a partitioned dataset.

**MVS** Multiple Virtual Storage. In z/OS's long history, MVS has the honor of being its name for the longest period: about two decades. Admittedly, it had many suffixes during those years: MVS/SP, MVS/370, MVS/XA and MVS/ESA. Many users believed that MVS stood for Man Versus System.

**OS/390** The replacement for MVS/ESA announced at the end of 1995. It was an attempt to repackage MVS in a way that allowed IBM to offer attractive pricing at the lower levels, encourage developers to write applications in a shorter period of time, and generally improve the image of an operating system that is still largely identified with big iron and huge IT budgets. It also reduced IBM's testing costs dramatically because there was no longer a need to test every combination of supported releases of what were now components instead of separate system software products. This approach was first tested with DFSMS three and a half years earlier. In the longer term, OS/390 shielded MVS behind a layer of middleware that disguised many of the proprietary functions of MVS and provided users with common services across all the major IBM platforms. Replaced by z/OS on October 3, 2000, along with the introduction of eserver, including a complete line of mainframes called zSeries 900. Version 2 Release 10 was the last release of OS/390 and first became available September 29, 2000.

**Panel** IBM speak for screen layout.

- REXX** Restructured EXtended eXecutor language. A command procedure programming language which was initially available on z/VM only, replacing EXEC and EXEC2, but later became an SAA standard. Although SAA is a distant memory, REXX availability in z/OS TSO has seen it replace CLIST as the tool of choice, especially given the fact that TSO, and therefore REXX, can be run in batch. REXX is an effective programming language in its own right with powerful string processing facilities and is used to drive certain program products, notably GDDM. REXX is also available in VSE/ESA, AIX1, OS/2, Linux and Windows. Although normally interpretive, a REXX compiler and library is available for z/OS and z/VM. REXX for CICS is available for z/OS and VSE/ESA; it consists of REXX Development System and REXX Development System. Object REXX is available for Windows, OS/2, AIX, and Linux for Intel and zSeries 900.
- Shell** Generic term with a lot of different meanings. Early Internet service often involved an interface, called a Shell, on a host computer, rather than a direct connection to the Internet. And there was the IBM DOS Shell, a menu driven interface to basic PC-DOS functionality. Shell is also widely used in the expert systems and Unix communities, to mean software providing a skeleton which can be customized to produce a specific application.
- TSO** Time Sharing Option. These days, everyone just says TSO when they mean TSO/E. Back in the 1980s, TSO was included with MVS/XA and you had to pay extra for TSO/E. Well worth the money given that TSO left you stranded below the 16MB line.
- TSO/E** Time Sharing Option/Extensions. An element of z/OS that provides an on-line interactive environment for programmers and users. Best known for the ISPF/PDF environment that runs on TSO/E. Can also be used to test batch programs.
- Two's**
- complement** is a method of signifying negative numbers in binary. It is also an operation which may be applied to positive binary values in order to perform subtraction using the method of complements, effectively allowing subtraction of one binary number from another using only the addition operation.
- XUL** The XML User Interface Language (XUL) is a markup language for describing user interfaces. With XUL you can create rich, sophisticated cross-platform web applications easily.

# Index

- Attribute, 16
- attribute, 16, 37–41, 64
- big endian, 17
- binary, 17, 20–22, 24, 65, 70
- capitalize, 10, 13, 16, 38, 63
- case, 25, 32, 43, 49
- CDATA, 20, 64
- child, 13, 37
- comment, 10, 16, 41, 64
- COMP, COMPUTATIONAL, 11, 65
- content, 8, 12, 13, 15, 16, 20, 37–40, 64
- continuation character, 25, 28, 43, 44
- dash, 13, 16, 28, 38, 44
- DBCS, 11, 68
- default value, 16, 25–28, 43–45, 64
- DEPENDING, 7, 11
- DISPLAY-1, 11
- DTD, 8, 13, 27, 65, 68
- element, 8, 12, 13, 15–17, 28, 37–41, 44, 64
- elementary item, 11, 38, 39, 63
- ENTITY, 16, 40, 64
- equivalent, 12, 16, 17
- escape, 13, 16, 20, 64
- file transfer, 17, 64
- FILLER, 11, 13
- First child, 13, 16, 37, 38
- group item, 11, 13, 15, 37–40, 63, 65
- GUI, 7, 23, 24, 36, 53–55, 65, 68
- high-value, 20
- INDEX, 11
- ISPF, 7, 23, 26, 27, 29, 45, 46, 68
- java, 23, 24, 36, 53–55, 65
- JCL, 7, 23, 28, 30, 44, 47, 68
- level, 11–13, 37–39, 44, 63, 64
- level 66, 77, 88, 11, 63
- license, 9, 36
- little endian, 17, 20, 21
- low-value, 20
- magic number, 24
- meta-data, 11, 37, 38
- Mixed XML content, 16, 40
- MVS, 9, 17, 22, 25–29, 36, 43–46, 65, 69
- NATIONAL, 11
- NATIVE, 10
- NOCONFIRM, 65
- numeric, 8, 11, 16, 17, 19–21, 63, 64
- OBJECT REFERENCE, 11
- OCCURS, occurrence, 11, 13, 36–38, 63
- OS/390, 8, 9, 36, 69
- packed-decimal, 17, 20, 21, 65
- parser, 7, 8, 36
- PICTURE, 10, 11, 15, 36, 40, 65
- platform, 8, 9, 17, 18, 22–24, 26–28, 36, 44, 45, 53, 64, 65
- POINTER, 11
- REDEFINES, 7, 11, 26
- Regina, 9, 23, 24, 36, 64
- root, 12, 16, 37, 40
- sign, 10, 17, 20, 21, 36, 38, 65
- tag, 8, 12, 13, 16, 20, 36, 37, 39, 40, 64
- Thinlet, 53
- Tiny-Cobol, 17, 18
- two's complement, 20, 21, 70
- underscore, 13, 16, 28, 38, 44

valid COBOL copybook, 8, 10, 15, 36

VALUE, 11, 64

well-formed XML document, 8, 36

XUL, 53