



**AGETOR®**

*AXT Text*  
*User guide*

## Table of contents

1	Introduction.....	3
2	Installation.....	3
3	AXT configuration.....	3
3.1	Example .....	4
3.2	Errorhandler .....	4
4	AXT Text configuration .....	4
4.1	Handling master-detail input.....	7


## 1 Introduction

This document is a description on how to convert data from a delimiter-separated format to XML. AXT Text can convert lines with either quoted or unquoted data elements.

If you need to transform fixed-positioned text to XML, you should consult the EDI2XML filter guide.

## 2 Installation

This section gives you step-by-step information on how to install AXT TEXT.

 The newest version of AXT TEXT is always available at the Bording Data download center at <http://www.agetor.dk>.

Before you begin the installation of the AXT TEXT, make sure you have the following prerequisites installed:

- Java Development Kit (JDK) version 1.3 or newer.
- AGETOR Development Kit (ADK) version 2.0.0 or newer.
- AXT Basic version 2.0.0 or newer.

Download the newest version of the AXT TEXT and copy the package into your “AGETOR\_HOME/install/packages” directory. If you are upgrading an existing installation the existing configuration will be retained.

- Open a command window with “AGETOR\_HOME/bin/prompt.bat” and run “installer.bat”.
- After the AGETOR® installtool has started open a Internet browser and point to “http://localhost:8020”.
- If prompted for login and password, please type in your login and password.
- Under “Product(s) ready to install” click on “AXT TEXT 2.0.1” and answer the questions.

Please note, that new properties might have been added and you should always consult the release.txt for any changes you might need to incorporate.

## 3 AXT configuration

The AXT Text filter uses the parameters described in the table below.

Name	Mandatory	Default	Description
configuration	No	Value of property “axt.transformation.text.configuration”	The filename of the AXT Text configuration
confname	Yes	N/A	The name of the conversion configuration
buffer	No	false	Decides if the XML output is written

			continuously or buffered until the end of the transformation
xmlencoding	No	ISO-8859-1	The encoding of the output
textencoding	No	The default encoding of the system	The encoding of the text input
indent	No	false	Decides if the output is indented for readability

### 3.1 Example

```
<filter class="dk.bording.axt.tc.txtxml.Text2XML">
  <param name="confname" value="orders" />
</filter>
```

### 3.2 Errorhandler

AXT Text includes an AXT errorhandler that will print the error message as plain text instead of XML.

```
<errorhandler class="dk.bording.axt.tc.err.TextErrorHandler">
  <param name="encoding" value="ISO8859_1" />
</errorhandler>
```

To see the error message correctly, you may need to specify the encoding of the message for this errorhandler. The encoding must be specified using the naming conventions of for the java.io and java.lang APIs, which are described here:

<http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html>

## 4 AXT Text configuration

Definition of the conversion processes is located in the file specified by the property

```
axt.transformation.text.configuration
```

The configuration file can contain many parser configurations/conversion definitions. The only limitation is that the attribute value “name” in the <parser> tag is unambiguous.

The following example shows a configuration of a parser from delimiter input to XML compliant output.

```
<configuration>
  <parser name='delimited_orders' delimiter=',' escape='\' stringdelimiter=''>
    <root name='order'>
      <record name='orderlines'>
        <field name='customer' type='string' />
        <field name='ref' type='string' />
        <field name='partNo' />
        <field name='quantity' oldchar='.' newchar=', '/>
        <field name='unit' type='string' />
        <field name='deliveryDate' oldchar='.' newchar='- '/>
      </record>
    </root>
  </parser>
</configuration>
```

```
</record>
</root>
</parser>
</configuration>
```

If parsing is done using the above definition and the input looks like the following:

```
240667", "rekv, int", 240667, 30.5, "KG", 22.04.2003
"240667", "4", JSF 1010, 32, "KG", 22.04.2003
"240667", "dept1, empl2", JSF 1020, 19.5, "KG", 22.04.2003
```

then the output will look like this:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<order>
  <orderlines>
    <customer>240667</customer>
    <ref>rekv</ref>
    <partNo>240667</partNo>
    <quantity>30,5</quantity>
    <unit>KG</unit>
    <deliveryDate>22-04-2003</deliveryDate>
  </orderlines>
  <orderlines>
    <customer>240667</customer>
    <ref>4</ref>
    <partNo>JSF 1010</partNo>
    <quantity>32</quantity>
    <unit>KG</unit>
    <deliveryDate>22-04-2003</deliveryDate>
  </orderlines>
  <orderlines>
    <customer>240667</customer>
    <ref>4</ref>
    <partNo>JSF 1020</partNo>
    <quantity>19,5</quantity>
    <unit>KG</unit>
    <deliveryDate>22-04-2003</deliveryDate>
  </orderlines>
</order>
```

The following table shows, which attributes are valid for each tag and a description of each attribute:

<b>Tag</b>	<b>Attribute</b>	<b>Description</b>
Parser	name	Unambiguous name that identifies every parser definition.
	delimiter	The character used as separator in the parsed input.
	escape	The character used as escape character in the input.
	stringdelimiter	The character used to enclose strings in input. If this attribute is not set, the whole input is examined for escape and separator signs.
Ignore	start	With this attribute it is possible to specify that lines must be ignored if they start with the given string.
Root	name	The name of the root tag of the output.
Record	name	The name of the tag in output.
	lineno	This attribute is not mandatory. When stated it has to be zero or above. If the attribute is set, the record definition is used on the exact line number in the input. If the LineNo attribute is not set, the record definition is used as the default definition. NB. If there are lines in input, which are to be ignored, they are still counted as one line in input.
Field	type	It is only necessary to specify the type if it is a string field. Other values than "string" are ignored.
	oldchar	Characters in the input can be replaced. oldchar is the character to be replaced.
	newchar	When replacing characters newchar is the value used the output

## 4.1 Handling master-detail input

If the input file contains data that is structured as master-detail information (such as an order with an order head and order lines), the line number feature of the record can be used to reflect this in the output. The following example shows this using a slightly modified version of the order example above.

The input with order head (the first line) and order lines (the following lines):

```
Acme Inc.,498732,2005.02.07
M-234344,22,kg
FK-2234,100,stk
S-21233,15,t
```

The AXT Text configuration:

```
<parser name='orders_with_head' delimiter=',' escape='\ ' stringdelimiter=''>
  <root name ='order'>
    <record name='orderhead' lineno="1">
      <field name='customer' type='string' />
      <field name='ref' type='string' />
      <field name='deliveryDate' oldchar='.' newchar='-'/>
    </record>
    <record name='orderline'>
      <field name='partNo' />
      <field name='quantity' oldchar='.' newchar=', '/>
      <field name='unit' type='string' />
    </record>
  </root>
</parser>
```

The output will be:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<order>
  <orderhead>
    <customer>Acme Inc.</customer>
    <ref>498732</ref>
    <deliveryDate>2005-02-07</deliveryDate>
  </orderhead>
  <orderline>
    <partNo>M-234344</partNo>
    <quantity>22</quantity>
    <unit>kg</unit>
  </orderline>
  <orderline>
    <partNo>FK-2234</partNo>
    <quantity>100</quantity>
    <unit>stk</unit>
  </orderline>
  <orderline>
    <partNo>S-21233</partNo>
    <quantity>15</quantity>
    <unit>t</unit>
  </orderline>
</order>
```