

Configuration Guide
Message Transformation Guidelines - MessageTransformBean
July 2024
English

Message Transformation Guidelines - MessageTransformBean

Content

| | | |
|-------|------------------|----|
| 1 | Introduction | 3 |
| 2 | Conversion Types | 4 |
| 2.1 | SimplePlain2XML | 4 |
| 2.1.1 | Properties | 4 |
| 2.1.2 | Example | 6 |
| 2.2 | StructPlain2XML | 7 |
| 2.2.1 | Properties | 7 |
| 2.2.2 | Example | 9 |
| 2.3 | SimpleXML2Plain | 10 |
| 2.3.1 | Properties | 10 |
| 2.3.2 | Example | 11 |
| 2.4 | StructXML2Plain | 12 |
| 2.4.1 | Properties | 12 |
| 2.4.2 | Example | 14 |

1 Introduction

The MessageTransformBean (MTB) adapter module is often used on SAP Process Integration/Orchestration to aid the transformation of the message during the runtime, allowing data to be converted from Plain format to XML and vice-versa, according to the technical requirements.

The document presents an alternative solution using a groovy script which supports most of the features that existed on the MTB module.

The document also describes in detail the required and available properties/configurations and how to use them, followed by an example for each scenario.

There are four conversion types:

1. **SimplePlain2XML**
2. **StructPlain2XML**
3. **SimpleXML2Plain**
4. **StructXML2Plain**

2 Conversion Types

2.1 SimplePlain2XML

Simple Plain files are files which contains only 1 structure object to be parsed, meaning that, all the lines belong to the same structure. In each line, the fields can be split by a separator (xml.fieldSeparator) or each field can have fixed size (xml.fieldFixedLengths). In the next subsection there is an explanation about required and available properties to use this conversion type.

Sample:

Header1,header2,header3

Field1,field2,field3

Field1,field2,field3

2.1.1 Properties

| Parameter Name | Required | Default value | Allowed values / Samples | Additional remarks |
|-----------------------|------------------|-----------------|---|---|
| xml.conversionType | YES | SimplePlain2XML | | |
| xml.addXMLDeclaration | NO | false | true,false | Adds or not initial XML declaration. |
| xml.documentName | NO | root | Sample output: <root> <row/> </root> | 'root' xml node. |
| xml.documentNamespace | NO | | Sample: http://namespace.com Sample output: <root ns1:http://namespace.com> <row/> </root> | 'root' xml node namespace. |
| xml.documentOffset | NO | 0 | Value automatically set If xml.processFieldNames = fromFile, documentOffset = 1, if fromFileWithBlankLine, documentOffset = 2 | Number of rows to be skipped from the start of the file. |
| xml.structureTitle | NO | row | Sample output: <root> <row> <A/> </row> </root> | Main xml node. |
| xml.fieldFixedLengths | YES ¹ | | Sample: 1,2,3 | ¹ Either fieldFixedLengths or fieldSeparator must be populated |
| xml.fieldSeparator | YES ¹ | | | ¹ Either fieldFixedLengths or fieldSeparator must be populated |

| Parameter Name | Required | Default value | Allowed values / Samples | Additional remarks |
|-----------------------|------------------|---------------|--|--|
| xml.endSeparator | NO | \n | | Separator for each line |
| xml.fieldNames | YES ² | | Sample: A,B,C | ² Required parameter if xml.processFieldNames = fromConfiguration |
| xml.processFieldNames | YES | | <p>fromFile – means that the field name information is in the header line of the file to be converted.</p> <p>fromFileWithBlankLine – corresponds to fromFile. After the header line there also follows a blank line or separator that is skipped.</p> <p>fromConfiguration – means that no header information exists in the file to be converted, but it will be delivered by the present configuration.</p> <p>notAvailable – no field name information is in the configuration or in the file to be converted. In this case, the columns in the XML document are identified using a simple counter tag (<columnX>, X=0,1,2...).</p> | Required parameter |

2.1.2 Example

Input body:

| Date | Mat | PO | GR1 | GR2 | GRP3 | GRP4 | AMT | CRDT |
|--------|---------|-----------|-------------------------------|----------|------|------|-----|------|
| 202310 | FABCDX | 01TEST123 | 45304508010001000000028259990 | 10152023 | | | | |
| 202310 | FABCDF | 01TEST456 | 35301508010001000000145824440 | 10152023 | | | | |
| 202310 | FABCD A | 01TEST789 | 55302508010001000000232641827 | 10152023 | | | | |

Properties:

xml.conversionType=SimplePlain2XML
xml.processFieldNames=fromFile
xml.structureTitle=row
xml.fieldFixedLengths=6,10,10,3,3,4,4,15,8
xml.documentName=root
xml.documentNamespace=http://xi.com/test

Output:

```
<ns1:root xmlns:ns1="http://xi.com/test">
  <row>
    <Date>202310</Date>
    <Mat>FABCDX</Mat>
    <PO>01TEST123</PO>
    <GR1>453</GR1>
    <GR2>045</GR2>
    <GRP3>0801</GRP3>
    <GRP4>0001</GRP4>
    <AMT>000000028259990</AMT>
    <CRDT>10152023</CRDT>
  </row>
  <row>
    <Date>202310</Date>
    <Mat>FABCDF</Mat>
    <PO>01TEST456</PO>
    <GR1>353</GR1>
    <GR2>015</GR2>
    <GRP3>0801</GRP3>
    <GRP4>0001</GRP4>
    <AMT>000000145824440</AMT>
    <CRDT>10152023</CRDT>
  </row>
  <row>
    <Date>202310</Date>
    <Mat>FABCD A</Mat>
    <PO>01TEST789</PO>
    <GR1>553</GR1>
    <GR2>025</GR2>
    <GRP3>0801</GRP3>
    <GRP4>0001</GRP4>
    <AMT>000000232641827</AMT>
    <CRDT>10152023</CRDT>
  </row>
</ns1:root>
```

2.2 StructPlain2XML

Structured Plain files are files which can contain multiple structured objects to be parsed, meaning that, each line can belong to different structures. The expected structures must be configured in the properties (xml.recordsetStructure). For each line, depending on the structure, the fields can be split by a separator (xml.<StructureName>.fieldSeparator) or each field can have fixed size (xml.<StructureName>.fieldFixedLengths). In the same file, it is possible that some structures have a separator and others a fixed size. In the next subsection there is an explanation about required and available properties to use this conversion type.

Sample:

HeaderA1,headerA2,headerA3

HeaderB1,headerB2,headerB3

FieldA1,fieldA2,fieldA3

FieldB1,fieldB2,fieldB3

2.2.1 Properties

<StructureName> is per the structure names defined in xml.recordsetStructure. For each structure name listed, configure the corresponding xml. <StructureName>.* parameters.

| Parameter Name | Required | Default value | Allowed values/Samples | Additional remarks |
|------------------------|----------|-----------------|--|--|
| xml.conversionType | YES | StructPlain2XML | | |
| xml.addXMLDeclaration | NO | false | true,false | Adds or not initial XML declaration. |
| xml.documentName | NO | root | Sample output: <root> < Recordset /> </root> | 'root' xml node. |
| xml.documentNamespace | NO | | Sample: http://namespace.com Sample output: <root ns1: http://namespace.com > < Recordset /> </root> | 'root' xml node namespace. |
| xml.documentOffset | NO | 0 | Value automatically set if xml.processFieldNames fromFile, documentOffset = 1, if fromFileWithBlankLine, documentOffset = 2 | Number of rows to be skipped from the start of the file. |
| xml.recordsetName | NO | Recordset | Sample output: <root> < Recordset > <A/> <C/> </ Recordset > </root> | |
| xml.recordsetNamespace | NO | | Sample: http://rs.com Sample output: <root> | |

| Parameter Name | Required | Default value | Allowed values/Samples | Additional remarks |
|---------------------------------------|------------------|---------------|---|---|
| | | | < Recordset ns2: http://rs.com > <A/> </ Recordset > </root> | |
| xml.recordsetStructure | YES | | Sample: AA,1,BB,*,CC,1 Sample output: <root> < Recordset > <AA/> <BB/> <BB/> <CC/> </ Recordset > </root> | Values split by ‘,’. Name1,Occur1, Name2,Occur2, Name3,Occur3 Occur allowed values are ‘1’ or ‘*’ |
| xml.recordsetsPerMessage | NO | 0 | | The final payload contains all recordsets, but with a value != 0, the sub-messages are stored into properties, named: output_xml_node_X |
| xml.ignoreRecordSetName | NO | false | true, false | If true, the RecordSet node will not be displayed in the output. |
| xml.endSeparator | | \n | | Separator for each line |
| xml.keyFieldName | YES ² | | Name of the field which identifies the Recordset | ² Required parameter if there is a structure with * in xml.recordsetStructure |
| xml.processFieldNames | YES | | fromFile – means that the field name information is located in the header line of the file to be converted. fromFileWithBlankLine – corresponds to fromFile. After the header line there also follows a blank line or separator that is skipped. fromConfiguration – means that no header information exists in the file to be converted, but it will be delivered by the present configuration. notAvailable – means that no field name information is assumed to be in the configuration or in the file to be converted. In this case, the columns in the XML document are identified using a simple counter tag (<columnX> , X=0,1,2...). | Required parameter |
| xml.<StructureName>.fieldFixedLengths | YES ¹ | | Int values split by ‘,’. Sample: 1,2,3 | ¹ Either fieldFixedLengths or fieldSeparator must be populated |

| Parameter Name | Required | Default value | Allowed values/Samples | Additional remarks |
|---|------------------|---------------|---|--|
| xml.<StructureName>.field Separator | YES ¹ | | Sample: , | ¹ Either fieldFixedLengths or fieldSeparator must be populated |
| xml.<StructureName>.missingLastFields | NO | ignore | ignore: missing fields are not added to the output; add: missing fields are added to the output with empty value; error: an error is thrown indicating missing fields | In case the input contains less values than the fieldnames configured, different approaches can be used. |
| xml.<StructureName>.keyFieldValue | YES ² | | Value of the key field which identifies the Recordset. | ² Required parameter if there is a structure with * in xml.recordsetStructure |
| xml.<StructureName>.keyFieldInStructure | No | add | ignore, add | Add or not the Keyfield into the final payload |
| xml.<StructureName>.fieldNames | YES ³ | | Values splitted by ‘,’. Ex: A1, A2, A3 | ³ Required parameter if xml.processFieldNames = fromConfiguration |

2.2.2 Example

Input:

AA1234567890
BBABCABC
BBXYZXYZ
CC12345

Properties:

xml.conversionType=StructPlain2XML
xml.processFieldNames=fromConfiguration
xml.documentName=root
xml.documentNamespaces=http://xi.com/test
xml.recordsetName=MyRecordset
xml.recordsetStructure=NameA,1,NameB,*,NameC,1
xml.keyFieldName=MyKey
xml.keyFieldType=CaseSensitiveString
xml.NameA.fieldNames=MyKey,field-nameA
xml.NameA.fieldFixedLengths=2,10
xml.NameA.keyFieldValue=AA
xml.NameA.keyFieldInStructure=add
xml.NameB.fieldNames=MyKey,field-nameB1,field-nameB2
xml.NameB.fieldFixedLengths=2,3,3
xml.NameB.keyFieldValue=BB
xml.NameB.keyFieldInStructure=ignore
xml.NameC.fieldNames=MyKey,field-nameC
xml.NameC.fieldFixedLengths=2,5

xml.NameC.keyFieldValue=CC

xml.NameC.keyFieldInStructure=ignore

Output:

```
<ns1:root xmlns:ns1="http://xi.com/test">
  <MyRecordset>
    <NameA>
      <MyKey>AA</MyKey>
      <field-nameA>1234567890</field-nameA>
    </NameA>
    <NameB>
      <field-nameB1>ABC</field-nameB1>
      <field-nameB2>ABC</field-nameB2>
    </NameB>
    <NameB>
      <field-nameB1>XYZ</field-nameB1>
      <field-nameB2>XYZ</field-nameB2>
    </NameB>
    <NameC>
      <field-nameC>12345</field-nameC>
    </NameC>
  </MyRecordset>
</ns1:root>
```

2.3 SimpleXML2Plain

Like the Simple Plain files, Simple XML files are files which contains only 1 structure object to be parsed, meaning that, all nodes belong to the same structure. For the output, the fields can be split by a separator (xml.fieldSeparator) or each field can have fixed size (xml.fieldFixedLengths). In the next subsection there is an explanation about required and available properties to use this conversion type.

Sample:

```
<root>
  <row>
    <Header1>Field1</Header1>
    <Header2>Field2</Header2>
    <Header3>Field3</Header3>
  </row>
  <row>
    <Header1>Field1</Header1>
    <Header2>Field2</Header2>
    <Header3>Field3</Header3>
  </row>
</root>
```

2.3.1 Properties

| Parameter Name | Required | Default value | Allowed values/Samples | Additional remarks |
|---------------------------------|------------------|-----------------|---|---|
| xml.conversionType | YES | SimpleXML2Plain | | Required parameter |
| xml.fieldFixedLengths | YES ¹ | | Sample: 1,2,3 | ¹ Either fieldFixedLengths or fieldSeparator must be populated |
| xml.fieldSeparator | YES ¹ | | , | ¹ Either fieldFixedLengths or fieldSeparator must be populated |
| xml.endSeparator | NO | \n | | Separator for each line |
| xml.addHeaderLine | NO | 0 | 0 - header line is not added. 1 - header line added using the field names from the payload. 2 - same as '1' with a blank line 3 - header line added using the values from 'xml.headerLine'. 4 - same as '3' with a blank line. | |
| xml.headerLine | YES ² | | Values split by ',' | ² Required parameter if xml.addHeaderLine = 3 or 4 |
| xml.fixedLengthTooShortHandling | NO | ERROR | IGNORE – value is added. CUT – value is added with the expected size; ERROR – Exception is thrown, stopping the process | |

2.3.2 Example

Input:

```
<root>
  <row>
    <Date>202310</Date>
    <Mat>FABCDX</Mat>
    <PO>01TEST123</PO>
    <GR1>453</GR1>
    <GR2>045</GR2>
    <GRP3>0801</GRP3>
    <GRP4>0001</GRP4>
    <AMT>000000028259990</AMT>
    <CRDT>10152023</CRDT>
  </row>
  <row>
    <Date>202310</Date>
    <Mat>FABCDF</Mat>
    <PO>01TEST456</PO>
    <GR1>353</GR1>
    <GR2>015</GR2>
    <GRP3>0801</GRP3>
    <GRP4>0001</GRP4>
    <AMT>000000145824440</AMT>
    <CRDT>10152023</CRDT>
  </row>
  <row>
    <Date>202310</Date>
    <Mat>FABCD A</Mat>
    <PO>01TEST789</PO>
    <GR1>553</GR1>
    <GR2>025</GR2>
    <GRP3>0801</GRP3>
    <GRP4>0001</GRP4>
    <AMT>000000232641827</AMT>
    <CRDT>10152023</CRDT>
  </row>
</root>
```

Properties:

```
xml.conversionType=SimpleXML2Plain
xml.fieldSeparator=,
xml.headerLine=Date,Mat,PO,GR1,GR2,GRP3,GRP4,AMT,CRDT
xml.addHeaderLine=3
xml.fixedLengthTooShortHandling=CUT
```

Output

```
Date,Mat,PO,GR1,GR2,GRP3,GRP4,AMT,CRDT
202310,FABCDX,01TEST123,453,045,0801,0001,000000028259
990,10152023
202310,FABCDF,01TEST456,353,015,0801,0001,000000145824
440,10152023
202310,FABCD A,01TEST789,553,025,0801,0001,000000232641
827,10152023
```

2.4 StructXML2Plain

Like the Structured Plain files, Structured XML files can contain multiple structured objects to be parsed, meaning that, each node can belong to different structures. The expected structures must be configured in the properties (xml.recordsetStructure). For the output, each line, depending on the structure, the fields can be split by a separator (xml.<StructureName>.fieldSeparator) or each field can have fixed size (xml.<StructureName>.fieldFixedLengths). In the same file, it is possible that some structures have a separator and others a fixed size. In the next subsection there is an explanation about required and available properties to use this conversion type.

Sample:

```
<root>
  <RecordSet>
    <AA>
      <HeaderA1>FieldA1</HeaderA1>
      <HeaderA2>FieldA2</HeaderA2>
      <HeaderA3>FieldA3</HeaderA3>
    </AA>
    <BB>
      <HeaderB1>FieldB1</HeaderB1>
      <HeaderB2>FieldB2</HeaderB2>
      <HeaderB3>FieldB3</HeaderB3>
    </BB>
  </RecordSet >
</root>
```

2.4.1 Properties

<StructureName> is per the structure names defined in xml.recordsetStructure. For each structure name listed, configure the corresponding xml.<StructureName>.* parameters.

| Parameter Name | Required | Default value | Allowed values/Samples | Additional remarks |
|------------------------|----------|-----------------|---|--|
| xml.conversionType | YES | StructXML2Plain | | Required parameter |
| xml.endSeparator | NO | \n | | Separator for each line |
| xml.addHeaderLine | YES | 0 | 0 - header line is not added. 1 - header line added using the field names from the payload. 2 - same as '1' with a blank line 3 - header line added using the values from 'xml.headerLine'. 4 - same as '3' with a blank line. | |
| xml.recordsetStructure | YES | | Sample: AA,1,BB,*,CC,1 Sample input: <root> <Recordset > <AA/> <BB/> <BB/> | Values splitted by ','. Name1,Occur1, Name2,Occur2, Name3,Occur3 Occur allowed values are '1' or '*' |

| | | | | |
|---|------------------|-------|---|---|
| | | | <CC/> </ Recordset > </root> | |
| xml.<StructureName>.fieldFixedLengths | YES ¹ | | Int values splitted by ‘,’. Sample: 1,2,3 | ¹ Either fieldFixedLengths or fieldSeparator must be populated |
| xml.<StructureName>.fieldSeparator | YES ¹ | | Sample: , | ¹ Either fieldFixedLengths or fieldSeparator must be populated |
| xml.<StructureName>. fixedLengthTooShortHandling | NO | ERROR | IGNORE – value is added; CUT – value is added with the expected size; ERROR – Exception is thrown, stopping the process. | Only valid when using fieldFixedLengths |

2.4.2 Example

Input:

```
<ns1:root xmlns:ns1="http://xi.com/test">
  <MyRecordset>
    <NameA>
      <MyKey>AA</MyKey>
      <field-nameA>1234567890</field-nameA>
    </NameA>
    <NameB>
      <field-nameB1>ABC</field-nameB1>
      <field-nameB2>ABC</field-nameB2>
    </NameB>
    <NameB>
      <field-nameB1>XYZ</field-nameB1>
      <field-nameB2>XYZ</field-nameB2>
    </NameB>
    <NameC>
      <field-nameC>12345</field-nameC>
    </NameC>
  </MyRecordset>
</ns1:root>
```

Properties:

```
xml.conversionType=StructXML2Plain
xml.processFieldNames=fromConfiguration
xml.addHeaderLine=1
xml.recordsetName=MyRecordset
xml.recordsetStructure=NameA,1,NameB,*,NameC,1
xml.NameA.fieldNames=field-nameA,MyKey
xml.NameA.fieldSeparator=,
xml.NameB.fieldNames=MyKey,field-nameB1,field-nameB2
xml.NameB.fieldSeparator=,
xml.NameC.fieldNames=MyKey,field-nameC
xml.NameC.fieldSeparator=,
```

Output:

```
MyKey,field-nameA
field-nameB1,field-nameB2
field-nameC
AA,1234567890
ABC,ABC
XYZ,XYZ
12345
```