# Mapping File Format

## Contents

# Overview

The mapping file contains instructions written in XML for the Data Import Service. These instructions define how to handle certain files and the fields contained in them. In general, these instructions are intended for structured data, such as a document with fixed-width or delimited fields, but they're also flexible enough to be used for unstructured data.

Basically, these mappings define the following about the document:

### *Structure*

- Document name
- Encoding or character set
- Line separator character
- The number of lines to skip before parsing data

### *Records*

- Record name
- Record rank, highest parsed first
- The lines contained in the record
- The line structure, if any
- The fields contained in each line
- Any lines contained in the field

### *Filters*

- Filters a record by comparing against certain criteria

### *Processors*

- Alters the incoming data in some way

### *Transforms*

- Allows you to convert to and from dates, numbers, currency, choices, flags, etc.

## XML Basics

The mapping file is written in XML. XML is an abbreviation for “eXtensible Markup Language”. There are several fundamental elements of an XML document that you should know about:

### *Header / XML Declaration*

- Optional in some XML documents, but mandatory in a mapping file.
- Specifies the encoding used in this file, you should ensure this is correct!
- Typically, looks like:

```
<?xml version="1.0" encoding="UTF-8"?>
```

### *Tags*

- A *tag* starts with < and ends with >
- There are three types of *tags*:

*Start-tags*:
```
<section>
```

*End-tags*:
```
</section>
```

*Empty-element Tags*:
```
<line-break />
```

### *Elements*

- An *element* either begins with a *start-tag* and ends with a matching *end-tag*, or consists of only an *empty-element tag*. If the *element* has a *start-tag* and an *end-tag*, then whatever is between those tags is the element’s *content*. The *content* may contain *elements* (which are then *child elements*) or text.

```
<section>
    This is an XML
    <bold>element</bold>
    it can contain text or more elements.
    <line-break />
</section>
```

### *Attributes*

- An *attribute* is a name/value pair that is linked to an *element*. It must be within a *start-tag* or a *empty-element tag*.

```
<section id="First Section">
    <section id="Subsection"></section>
    <img src="Wysk.jpg" alt='Wysk Logo' />
</section>
```

### *Predefined Entities*

- There are certain characters that cannot be used in XML normally. Fortunately, there are five *predefined entities* that you can use in place of these characters.

| Character | Escape Code |
|---|---|
| < | < |
| > | > |
| & | &amp |
| ' | ' |
| " | " |

## Example 1 — Comma Separated, Multiple Record Mappings

```
<?xml version="1.0" encoding="UTF-8"?>
<file-format id="Example 1"
             encoding="UTF-8"
             lineSeparator="\r\n"
             skipLines="1">
    <record id="Company" rank="100">
        <filter>
            <reference>Record Type</reference>
            <matches>
                <constant>C</constant>
            </matches>
        </filter>
        <line delimiter=","
              escape="\"
              quote="&amp;quot;">
            <mapping id="Type"/>
            <mapping id="Company Name"/>
            <mapping id="Wysk Number">
                <process>
                    <uppercase/>
                </process>
            </mapping>
        </line>
    </record>
    <record id="Person" rank="90">
        <filter>
            <reference>Record Type</reference>
            <matches>
                <constant>P</constant>
            </matches>
        </filter>
        <line delimiter=","
              escape="\"
              quote="&amp;quot;">
            <mapping id="Type"/>
            <mapping id="Person Name"/>
            <mapping id="Wysk Number">
                <process>
                    <uppercase/>
                </process>
            </mapping>
        </line>
    </record>
</file-format>
```

## Document

```
1    TYPE,NAME,WYSK NO,DATE OF BIRTH
2    "P","Khechar Boorla","3KFYF4M"
3    "P","Jennifer Gohlke","33CFUQZ","1/20/1989"
4    "P","Shahin \"Sean\" Jamea","X3EP99Z"
5    "P","Natalie Hudson","H73L8EZ"
6    "C","Adept Developer, LLC","X3GYFBZ"
7    "P","Steve Parks","33ecdkz","1/14/1978"
8    "C","Bain Capital, LLCD"
9    "P","Mitt \"Mittens\" Romney",,"3/12/1947"
10   "P","Fiona Apple"
11   "M","Björk Guðmundsdóttir"
12   "P","Edward
13   Earl of Wessex","66EQXPZ"
```

## Records

### *Person*

| Index | Record ID | Type | Person Name | Wysk Number |
|---|---|---|---|---|
| **2** | Person | P | Khechar Boorla | 3KFYF4M |
| **3** | Person | P | Jennifer Gohlke | 33CFUQZ |
| **4** | Person | P | Shahin "Sean" Jamea | X3EP99Z |
| **5** | Person | P | Natalie Hudson | H73L8EZ |
| **7** | Person | P | Steve Parks | 33ECDKZ |
| **9** | Person | P | Mitt "Mittens" Romney | |
| **10** | Person | P | Fiona Apple | |
| **12, 13** | Person | P | Edward<br>Earl of Wessex | 66EQXPZ |

### *Company*

| Index | Record ID | Type | Company Name | Wysk Number |
|---|---|---|---|---|
| **6** | Company | C | Adept Developer, LLC | 3KFYF4M |
| **8** | Company | C | Bain Capital, LLCD | 33CFUQZ |

### *Unmapped*

| Index | Status |
|---|---|
| **1** | Skipped, specified by skipLines attribute. |
| **11** | Skipped, line did not get filtered into any records. |

## Example 2 — Fixed-Width, Multiple Record Mappings

```
<?xml version="1.0" encoding="UTF-8"?>
<file-format id="Example 2"
             encoding="UTF-8"
             lineSeparator="\r\n">
    <record id="Company" rank="100">
        <filter>
            <reference>Record Type</reference>
            <matches>
                <constant>C</constant>
            </matches>
        </filter>
        <line length="32">
            <mapping id="Type">
                <span length="1"/>
            </mapping>
            <mapping id="Company Name">
                <span length="24"/>
            </mapping>
            <mapping id="Wysk Number">
                <span length="7"/>
                <process>
                    <uppercase/>
                </process>
            </mapping>
            <process scope="global" order="after">
                <trim/>
            </process>
        </line>
    </record>
    <record id="Person" rank="90">
        <filter>
            <reference>Record Type</reference>
            <matches>
                <constant>P</constant>
            </matches>
        </filter>
        <line minLength="40">
            <mapping id="Type">
                <span length="1"/>
            </mapping>
            <mapping id="Person Name">
                <span length="24"/>
            </mapping>
            <mapping id="Wysk Number">
                <span length="7"/>
                <process>
                    <uppercase/>
                </process>
            </mapping>
            <mapping id="DOB">
                <span length="8"/>
                <process>
                    <trim/>
                    <transform>
                        <datetime pattern="yyyyMMdd"/>
                        <datetime pattern="MMM d, yyyy"/>
                    </transform>
                </process>
            </mapping>
```

```
60              <process scope="global" order="after">
61                  <trim/>
62              </process>
63          </line>
64      </record>
65  </file-format>
```

## Document

```
1    PKhechar·Boorla·········3KFYF4M········
2    PJennifer·Gohlke········33CFUQZ19890120
3    PShahin "Sean" Jamea····X3EP99Z········
4    PNatalie Hudson·········H73L8EZ········
5    CAdept Developer, LLC···X3GYFBZ
6    PSteve Parks···········33ecdkz19780114
7    CBain Capital, LLCD·····------
8    PMitt "Mittens" Romney·····19470312
9    PFiona Apple···························
10   MBjörk Guðmundsdóttir··················
11   PEdward
12   Earl of Wessex···66EQXPZ········
```

- Represents a space

## Records

### *Person*

| Index | Record ID | Type | Person Name | Wysk Number | DOB |
|---|---|---|---|---|---|
| **1** | Person | P | Khechar Boorla | 3KFYF4M | |
| **2** | Person | P | Jennifer Gohlke | 33CFUQZ | Jan 20, 1989 |
| **3** | Person | P | Shahin "Sean" Jamea | X3EP99Z | |
| **4** | Person | P | Natalie Hudson | H73L8EZ | |
| **6** | Person | P | Steve Parks | 33ECDKZ | Jan 14, 1989 |
| **8** | Person | P | Mitt "Mittens" Romney | | Mar 12, 1947 |
| **9** | Person | P | Fiona Apple | | |

### *Company*

| Index | Record ID | Type | Company Name | Wysk Number |
|---|---|---|---|---|
| **5** | Company | C | Adept Developer, LLC | 3KFYF4M |
| **7** | Company | C | Bain Capital, LLCD | 33CFUQZ |

### *Unmapped*

| Index | Status |
|---|---|
| **10** | Skipped, line did not get filtered into any records. |
| **11** | Skipped, line does not meet length requirement of any records. |
| **12** | Skipped, line does not meet length requirement of any records. |

## Example 3 — Fixed-Width, Single Record Mapping

```
<?xml version="1.0" encoding="UTF-8"?>
<file-format id="Example 3"
             encoding="UTF-8"
             lineSeparator="\r\n">
    <record id="Entity">
        <line minLength="36">
            <mapping id="Type Code">
                <span length="1"/>
            </mapping>
            <mapping id="Name">
                <span length="24"/>
            </mapping>
            <mapping id="Wysk Number">
                <span length="7"/>
                <process>
                    <uppercase/>
                </process>
            </mapping>
            <mapping id="DOB">
                <span length="8"/>
                <process>
                    <trim/>
                    <transform>
                        <datetime pattern="yyyyMMdd"/>
                        <datetime pattern="MMM dd, yyyy"/>
                    </transform>
                </process>
            </mapping>
            <mapping id="Type">
                <process>
                    <set>
                        <reference>Type Code</reference>
                    </set>
                    <trim/>
                    <replace pattern="[^PCM]"
                             replacement=""
                             scope="all"/>
                    <replace pattern="^[P]$"
                             replacement="Person"
                             scope="all"/>
                    <replace pattern="^[C]$"
                             replacement="Company"
                             scope="all"/>
                    <replace pattern="^[M]$"
                             replacement="Musician"
                             scope="all"/>
                </process>
            </mapping>
            <process scope="global" order="after">
                <trim/>
            </process>
        </line>
    </record>
</file-format>
```

## Document

```
1     PKhechar·Boorla·········3KFYF4M········
2     PJennifer·Gohlke········33CFUQZ19890120
3     PShahin "Sean" Jamea····X3EP99Z········
4     PNatalie Hudson·········H73L8EZ········
5     CAdept Developer, LLC···X3GYFBZ
6     PSteve Parks···········33ecdkz19780114
7     CBain Capital, LLCD····················
8     PMitt "Mittens" Romney·········19470312
9     PFiona Apple···························
10    MBjörk Guðmundsdóttir···
11    PEdward
12    Earl of Wessex···66EQXPZ········
```

· Represents a space

## Records

### *Entity*

| Index | Record ID | Type Code | Type | Name | Wysk Number | DOB |
|---|---|---|---|---|---|---|
| **1** | Entity | P | Person | Khechar Boorla | 3KFYF4M | |
| **2** | Entity | P | Person | Jennifer Gohlke | 33CFUQZ | Jan 20, 1989 |
| **3** | Entity | P | Person | Shahin "Sean" Jamea | X3EP99Z | |
| **4** | Entity | P | Person | Natalie Hudson | H73L8EZ | |
| **5** | Entity | C | Company | Adept Developer, LLC | X3GYFBZ | |
| **6** | Entity | P | Person | Steve Parks | 33ECDKZ | Jan 14, 1989 |
| **7** | Entity | C | Company | Bain Capital, LLCD | | |
| **8** | Entity | P | Person | Mitt "Mittens" Romney | | Mar 12, 1947 |
| **9** | Entity | P | Person | Fiona Apple | | |
| **10** | Entity | M | Musician | Björk Guðmundsdóttir | | |
| **12** | Entity | E | | arl of Wessex 66EQXPZ | | |

### *Unmapped*

| Index | Status |
|---|---|
| **11** | Skipped, line does not meet length requirement of any records. |

## Example 4 — Date, Time, and Timezone Transforms

```
<?xml version="1.0" encoding="UTF-8"?>
<file-format id="Example 4">
    <record id="Time Period">
        <line delimiter="|">
            <mapping id="Start Date">
                <process>
                    <trim/>
                    <transform>
                        <datetime pattern="yyyy-MM-dd HH:mm:ssZZ"/>
                        <datetime pattern="MMM dd, yyyy h:mm:ss a, ZZZ"
                                  timezone="CST6CDT"/>
                    </transform>
                </process>
            </mapping>
            <mapping id="End Date">
                <process>
                    <trim/>
                    <transform>
                        <datetime pattern="yyyy-MM-dd HH:mm:ssZZ"/>
                        <datetime pattern="MMM dd, yyyy h:mm:ss a, ZZZ"
                                  timezone="CST6CDT"/>
                    </transform>
                </process>
            </mapping>
        </line>
    </record>
</file-format>
```

## Document

```
1    1845-12-29 04:00:00-08:00|1845-12-31 00:00:00-08:00
2    1789-11-21 15:05:26-07:00|1789-11-21 23:04:01-07:00
3    2012-10-27 03:26:00-06:00|2012-10-27 04:26:00-06:00
4    2045-02-02 02:47:01-08:00|2045-02-02 02:47:02-08:00
5    1789-11-21 15:05:00-07:00|1789-12-28 15:05:00-07:00
6    1959-01-03 13:00:00-00:00|1959-01-03 13:00:00+07:00
7    2012-11-15 03:13:43-05:00|2013-01-01 00:00:00-05:00
```

## Records

### *Time Period*

| Index | Record ID | Start Date | End Date |
|---|---|---|---|
| **1** | Time Period | Dec 29, 1845 6:00:00 AM, CST6CDT | Dec 31, 1845 2:00:00 AM, CST6CDT |
| **2** | Time Period | Nov 21, 1789 4:05:26 PM, CST6CDT | Nov 22, 1789 12:04:01 AM, CST6CDT |
| **3** | Time Period | Oct 27, 2012 4:26:00 AM, CST6CDT | Oct 27, 2012 5:26:00 AM, CST6CDT |
| **4** | Time Period | Feb 02, 2045 4:47:01 AM, CST6CDT | Feb 02, 2045 4:47:02 AM, CST6CDT |
| **5** | Time Period | Nov 21, 1789 4:05:00 PM, CST6CDT | Dec 28, 1789 4:05:00 PM, CST6CDT |
| **6** | Time Period | Jan 03, 1959 7:00:00 AM, CST6CDT | Jan 03, 1959 12:00:00 AM, CST6CDT |
| **7** | Time Period | Nov 15, 2012 2:13:43 AM, CST6CDT | Dec 31, 2012 11:00:00 PM, CST6CDT |

### *Unmapped*

| Index | Status |
|---|---|

## Example 5 — Nested Line and Mapping Elements

```
1    <?xml version="1.0" encoding="UTF-8"?>
2    <file-format id="Example 5">
3        <record id="Contractor" rank="90">
4            <line delimiter="," escape="\" quote="&quot;">
5                <mapping id="First Name"/>
6                <mapping id="Last Name"/>
7                <mapping id="Street Address">
8                    <process>
9                        <address type="short"/>
10                   </process>
11               </mapping>
12               <mapping id="City, State, Postal Code">
13                   <line delimiter=",">
14                       <mapping id="City"/>
15                       <mapping>
16                           <line delimiter=" ">
17                               <mapping id="State"/>
18                               <mapping>
19                                   <line>
20                                       <mapping id="Postal Code">
21                                           <span length="5"/>
22                                       </mapping>
23                                       <mapping id="Postal Code Extension">
24                                           <span length="4"/>
25                                       </mapping>
26                                   </line>
27                                   <process>
28                                       <replace pattern="[^0-9]" replacement=""
scope="all"/>
29                                   </process>
30                               </mapping>
31                               <process>
32                                   <uppercase/>
33                               </process>
34                           </line>
35                       </mapping>
36                       <process>
37                           <trim/>
38                           <strip pattern=","/>
39                           <capitalize/>
40                       </process>
41                       <process scope="global" order="after">
42                           <trim/>
43                           <compress/>
44                       </process>
45                   </line>
46               </mapping>
47               <mapping id="Phone Number">
48                   <process>
49                       <replace pattern="[^0-9]" replacement="" scope="all"/>
50                       <transform>
51                           <extract pattern="^1?([0-9][0-9][0-9])([0-9][0-9][0-
9])([0-9][0-9][0-9][0-9])$"
52                                    output="+1 (\1) \2-\3"/>
53                       </transform>
54                   </process>
55               </mapping>
56               <mapping id="Have W-2">
57                   <process>
```

```
                        <transform>
                            <boolean format="any"/>
                            <boolean format="truefalse"/>
                        </transform>
                    </process>
                </mapping>
                <mapping id="SSN">
                    <process>
                        <replace pattern="[^0-9]" replacement="" scope="all"/>
                        <transform>
                            <extract pattern="^([0-9][0-9][0-9])([0-9][0-9])([0-9][0-
9][0-9][0-9])$"
                                     output="\1-\2-\3"/>
                        </transform>
                    </process>
                </mapping>
            </line>
        </record>
    </file-format>
```

## Document

```
1       Adam,Creel,2901 Donegal Ln,"Pearland, TX 77581-5008",899-921-2637,Y,481-48-2066
2       Eric,Armstrong,,,(844) 322-0809,Yes,501-17-1482
3       Laura,Bowles,350 LIMESTONE CIR,"Irvington, AL",8444586101,TRUE,751015965
4       Avis, Chatfield,7600 WASHINGTON AVE,"Brittow, VA 20136-2053",844.213.4374,1,
5       Ben,Dixon,4000 Smiling Wood Ln,"Houston, TX 77086",+1 (855) 065-6931,N,574-39-0162
6       Chris,Estrada,PO BOX 1052,"Daphne, AL 36526-7635",+ 1 855-543-3426,No,516-27-8662
7       Anushka,Sen,109 Lakeside Terrace,"PANAMA CITY, FL 32404-7545",,false
8       Doug,Wyatt,PO BOX 510,"Miami, FL 32301",(811)-973-2585,0,408-05-2354
```

## Records

### *Contractor*

| Index | Street Address | City, State, Postal Code | City | State | Postal Code | Postal Code Extension |
|---|---|---|---|---|---|---|
| **1** | 2901 Donegal Ln | Pearland, TX 77581-5008 | Pearland | TX | 77581 | 5008 |
| **2** | | | | | | |
| **3** | 350 Limestone Cir | Irvington, AL | Irvington | AL | | |
| **4** | 7600 Washington Ave | Brittow, VA 20136-2053 | Brittow | VA | 20136 | 2053 |
| **5** | 4000 Smiling Wood Ln | Houston, TX 77086 | Houston | TX | 77086 | |
| **6** | Po Box 1052 | Daphne, AL 36526-7635 | Daphne | AL | 36526 | 7635 |
| **7** | 109 Lakeside Ter | PANAMA CITY, FL 32404-7545 | Panama City | FL | 32404 | 7545 |
| **8** | Po Box 510 | Miami, FL 32301 | Miami | FL | 32301 | |

### *Contractor, Continued*

| Index | First Name | Last Name | Phone Number | SSN | Have W-2 |
|---|---|---|---|---|---|
| **1** | Adam | Creel | +1 (899) 921-2637 | 481-48-2066 | true |
| **2** | Eric | Armstrong | +1 (844) 322-0809 | 501-17-1482 | true |
| **3** | Laura | Bowles | +1 (844) 458-6101 | 751-01-5965 | true |
| **4** | Avis | Chatfield | +1 (844) 213-4374 | | true |
| **5** | Ben | Dixon | +1 (855) 065-6931 | 574-39-0162 | false |
| **6** | Chris | Estrada | +1 (855) 543-3426 | 516-27-8662 | false |
| **7** | Anushka | Sen | | | false |
| **8** | Doug | Wyatt | +1 (811) 973-2585 | 408-05-2354 | false |

### *Unmapped*

| Index | Status |
|---|---|
| | |

## Recipes

*Processor: Generic field value cleanup, intended to go in first line element.*

```
1    <process scope="global" order="after">
2        <trim/>
3        <replace pattern="\n" replacement=", " scope="all"/>
4        <compress/>
5        <strip pattern=","/>
6    </process>
```

*Processor: Format integer, removing leading zeros.*

```
1    <process>
2        <replace pattern="[^0-9]" replacement="" scope="all"/>
3        <transform>
4            <integer/>
5        </transform>
6    </process>
```

*Processor: Format generic numeric type.*

```
1    <process>
2        <replace pattern="[^0-9]" replacement="" scope="all"/>
3    </process>
```

*Processor: Remove "N/A" fields.*

```
1    <process>
2        <replace pattern="^N/A$" replacement="" scope="all"/>
3    </process>
```

*Processor: Remove "NULL" fields.*

```
1    <process>
2        <replace pattern="^NULL$" replacement="" scope="all"/>
3    </process>
```

*Transform: Excel "Accounting Format" with two decimal places.*

```
1    <process>
2        <transform>
3            <number pattern="#,##0.00;(#,##0.00)"/>
4        </transform>
5    </process>
```

*Mapping: 3-letter country code normalize*

```
1    <mapping id="Country Code">
2        <span length="3"/>
3        <process>
4            <trim/>
5            <replace pattern="USA" replacement="US" scope="all"/>
6            <set if="empty">
7                <constant>US</constant>
8            </set>
9            <uppercase/>
10       </process>
11   </mapping>
```

### *Mapping: 2-letter state normalize*

```
<mapping id="State">
    <span length="2"/>
    <process>
        <trim/>
        <uppercase/>
    </process>
</mapping>
```

### *Mapping: Separate full postal code into postal code and extension*

```
<mapping id="Full Postal Code">
    <span length="9"/>
    <process>
        <replace pattern="[^0-9]" replacement="" scope="all"/>
    </process>
    <line>
        <mapping id="Postal Code">
            <span length="5"/>
        </mapping>
        <mapping id="Postal Code Extension">
            <span length="4"/>
        </mapping>
    </line>
</mapping>
```

### *Mapping: US Phone Number, with optional country code*

```
<mapping id="Phone Number">
   <process>
       <replace pattern="[^0-9]" replacement="" scope="all"/>
       <transform>
           <extract pattern="^1?([0-9][0-9][0-9])([0-9][0-9][0-9])([0-9][0-9][0-9][0-9])$"
                    output="+1 (\1) \2-\3"/>
       </transform>
   </process>
</mapping>
```

### *Mapping: Social Security Number*

```
<mapping id="SSN">
    <process>
        <replace pattern="[^0-9]" replacement="" scope="all"/>
        <transform>
            <extract pattern="^([0-9][0-9][0-9])([0-9][0-9])([0-9][0-9][0-9][0-9])$"
                     output="\1-\2-\3"/>
        </transform>
    </process>
</mapping>
```

## Data Import Service

The Wysk DIS is easy to interface with through a mapping file.

1) On each <record> element that you want to import add the spname attribute.
   Format: `[DatabaseName].[dbo].[StoredProcedureName]`
2) Under each of those <record> elements, on each <mapping> element that you want to import, add the spvar attribute.
   Format: `@VariableName`

All other records and mappings that don't specify these additional attributes will be ignored by the DIS.

# Element Documentation

## <file-format>

The <file-format> element is the *root element* of the document. Everything else will go in here.

| *Attribute* | *Description* |
|---|---|
| id | A short, descriptive name of the document. |
| encoding | A valid character set name, common values include:<br>`ISO-8859-1` – Default<br>`ASCII`<br>`CP1251`<br>`UTF-8`<br>`UTF-16BE`<br>`UTF-16LE` |

| *Valid Child Elements* | *Multiple* |
|---|---|
| <record> | Yes |

### *Valid Content*

*None*

## <record>

The <record> element contains a filter for the record and the lines inside the record.

| *Attribute* | *Description* |
|---|---|
| id | A short, descriptive name of the record. |
| rank | The highest ranked record is the first that will be tried when trying to parse a line. If the record does not accept the line, the next highest ranked record is tried. |
| spname | The name of a stored procedure to use when importing this record. Must be of the form:<br>`[DatabaseName].[dbo].[StoredProcedureName]` |

| *Valid Child Elements* | *Multiple* |
|---|---|
| <line> | Yes |
| <filter> | No |

### *Valid Content*

*None*

## <filter>

The <filter> element allows you to specify a condition that must be true for the parent record to be parsed.

### *Attribute*

*None*

| *Valid Child Elements* | *Multiple* |
|---|---|
| *Expression Values* | |
| <reference> | No |
| <constant> | No |
| *Expression Patterns* | |
| <regex> | No |
| *Expression Operations* | |
| <matches> | No |
| <not> | No |
| <allof> | Yes |
| <oneof> | Yes |

### *Valid Content*

*None*

## <allof>

The <allof> element allows you to combine if statements inside a filter so that only if all of them are true will the filter pass.

### *Attribute*

*None*

| *Valid Child Elements* | *Multiple* |
|---|---|
| <if> | Yes |

## <oneof>

The <oneof> element allows you to combine if statements inside a filter so that if only one of them are true the filter pass.

### *Attribute*

*None*

| *Valid Child Elements* | *Multiple* |
|---|---|
| <if> | Yes |

## <if>

The <if> element allows you to specify a condition that must be true for the parent record to be parsed.

### *Attribute*

*None*

| *Valid Child Elements* | *Multiple* |
|---|---|
| *Expression Values* | |
| <reference> | No |
| <constant> | No |
| *Expression Patterns* | |
| <regex> | No |
| *Expression Operations* | |
| <matches> | No |

## <line>

The <line> element contains the processors, transforms, and mappings that make up the bulk of your mapping document. A record may have several lines to indicate that it spans multiple lines in the document.

| *Attribute* | *Description* |
|---|---|
| length | If specified, the record must be exactly this long to be parsed. |
| minLength | If specified, the record must be at least this long to be parsed. |
| maxLength | If specified, the record must be at most this long to be parsed. |
| delimiter | Specifies that the line's mappings will be delimited by one or more characters. |
| quote | If delimiter is specified, this is the character used to specify a field that may contain delimiters inside it. |
| escape | If delimiter and quote are both specified, this is the character used to escape quotes that are inside a quoted field. |

| *Valid Child Elements* | *Multiple* |
|---|---|
| <mapping> | Yes |
| <process> | Yes |

### *Valid Content*

*None*

## <mapping>

The <mapping> element tells the parser where to find a value of a field in an arbitrary line of text. The mapping element can also be thought of as a "field" in the "record". If no index is set manually, the index is automatically set where 0 is the first mapping, 1 is the next mapping, and so on.

| *Attribute* | *Description* |
|---|---|
| id | A short, descriptive name for this value. |
| index | If the parent line is delimited, manually set the index of this value where 0 is the first value on the line. |
| spvar | The name of a variable in the stored procedure that was specified in the <record> element. It must take the form of:<br>`@VariableName` |

| *Valid Child Elements* | *Multiple* |
|---|---|
| <process> | Yes |
|  | No |
| <line> | No |

### *Valid Content*

*None*

## 

The  element sets a logical span in an arbitrary line of text. This span is used by the parser to retrieve the value of a field. An invalid span element will prevent the record from being parsed.

Valid attribute combinations are:

- Length
- start, length
- start, end

| *Attribute* | *Description* |
|---|---|
| length | The length of this span. If start is not specified, it is relative to the last mapping elements span end position or 0. |
| start | The start index of this span, where 0 is the beginning of the text. This must be a positive integer that is less than or equal to end index. |
| end | The end index of this span, where 0 is the beginning of the text. This must be a positive integer that is greater than or equal to start index. |

### *Valid Child Elements*

*None*

### *Valid Content*

*None*

## <process>

The <process> element instructs the parser to modify the resulting value(s) in some way. If a processor is set to `before`, and there are child mappings/nested line elements, the mapping closest to the processor will be parsed and processed first, followed by the next closest and so on. If a processor is set to `after`, and there are child mappings/nested line elements, the mapping closest to the processor will be parsed *but not processed*, followed by the next closest until there are no more mappings to parse, then the furthest from the processor is processed, followed by the next furthest until the parser returns to the parent element. This can drastically affect how nested lines are handled if you use `global` processors.

| *Attribute* | *Description* |
|---|---|
| scope | Specifies if this processor will be run on nested lines or not.<br>`local` – Default; Run this on local line/mapping only.<br>`global` – Run this on local line/mapping and also child lines/mappings. |
| order | Specifies the order this processor should be run relative to any children.<br>`before` – Default; Run this before parsing globally.<br>`after` – Run this after parsing globally. |

| *Valid Child Elements* | *Multiple* |
|---|---|
| <transform> | Yes |
| *Processor Actions* | |
| <address> | Yes |
| <append> | Yes |
| <prepend> | Yes |
| <insert> | Yes |
| <left> | Yes |
| <right> | Yes |
| <substring> | Yes |
| <replace> | Yes |
| <capitalize> | Yes |
| <lowercase> | Yes |
| <uppercase> | Yes |
| <compress> | Yes |
| <trim> | Yes |
| <strip> | Yes |
| <pad> | Yes |
| <set> | Yes |

### *Valid Content*

*None*

## <transform>

The <transform> element takes a string value and puts it through a series of transformations. The last transformation in the element is responsible for transforming the value back into a string.

### *Attribute*

*None*

| *Valid Child Elements* | *Multiple* |
|---|---|
| <transform> | Yes |
| *Transformations* | |
| <datetime> | Yes |
| <boolean> | Yes |
| <number> | Yes |
| <integer> | Yes |
| <decimal> | Yes |
| <split> | Yes |
| <join> | Yes |
| <extract> | Yes |
| *Processor Actions* | |
| <address> | Yes |
| <append> | Yes |
| <prepend> | Yes |
| <insert> | Yes |
| <left> | Yes |
| <right> | Yes |
| <substring> | Yes |
| <replace> | Yes |
| <capitalize> | Yes |
| <lowercase> | Yes |
| <uppercase> | Yes |
| <compress> | Yes |
| <trim> | Yes |
| <strip> | Yes |
| <pad> | Yes |
| <set> | Yes |

### *Valid Content*

*None*

## Expression Values

### *Valid Content*

*None*

## <reference>

The <reference> element specifies that the referenced ID be parsed, and the field content should be used in place of this tag.

### *Valid Child Elements*

*None*

### *Valid Content*

Must contain the ID of a <mapping> element.

## <constant>

The <constant> element specifies a constant piece of text.

### *Valid Child Elements*

*None*

### *Valid Content*

Any text, line breaks are ignored.

# Expression Patterns

## <regex>

The <regex> element specifies a regular expression pattern that will be used to match against another value.

### *Valid Child Elements*

*None*

### *Valid Content*

Must contain a valid regular expression pattern.

# Expression Operations

## <matches>

The <matches> element is an expression that allows you to match a reference or constant to either a regular expression, reference, or constant. Matching a regular expression to a regular expression will always return false.

| *Valid Child Elements* | *Multiple* |
|---|---|
| *Expression Values* | |
| <reference> | No |
| <constant> | No |
| *Expression Patterns* | |
| <regex> | No |

### *Valid Content*

*None*

## <not>

The <not> element is negates any expression found within it.

| *Valid Child Elements* | *Multiple* |
|---|---|
| *Expression Operations* | |
| <matches> | No |
| <not> | No |

### *Valid Content*

*None*

# Processor Actions

## <address>

The <address> action attempts to clean up addresses into a consistent format.

| *Attribute* | *Description* |
|---|---|
| type | Specifies how we should return the normalized addresses.<br>`full` – Default; Expand abbreviations.<br>`short` – Abbreviate where possible. |

### *Valid Child Elements*

*None*

## <append>

The <append> action appends text to the field value. If the text attribute is specified, no child elements are allowed.

| *Attribute* | *Description* |
|---|---|
| text | The text to append. |

| *Valid Child Elements* | *Multiple* |
|---|---|
| *Expression Values* | |
| <reference> | No |
| <constant> | No |

## <prepend>

The <prepend> action prepends text to the field value. If the text attribute is specified, no child elements are allowed.

| *Attribute* | *Description* |
|---|---|
| text | The text to append. |

| *Valid Child Elements* | *Multiple* |
|---|---|
| *Expression Values* | |
| <reference> | No |
| <constant> | No |

## <insert>

The <insert> action inserts text at the specified index. If the text attribute is specified, no child elements are allowed.

| *Attribute* | *Description* |
|---|---|
| text | The text to append. |
| index | The index of the character to insert before inside the field value, where 0 is the start of the string. |

| *Valid Child Elements* | *Multiple* |
|---|---|
| *Expression Values* | |
| <reference> | No |
| <constant> | No |

## <left>

The <left> action gets an arbitrary number of characters starting from the left of the field value.

| *Attribute* | *Description* |
|---|---|
| length | The amount of characters to retrieve. |

### *Valid Child Elements*

*None*

## <right>

The <right> action gets an arbitrary number of characters starting from the right of the field value.

| *Attribute* | *Description* |
|---|---|
| length | The amount of characters to retrieve. |

### *Valid Child Elements*

*None*

## <state>

The <state> action attempts to clean up state names or identifiers into a consistent format.

| *Attribute* | *Description* |
|---|---|
| type | Specifies how we should return the normalized state name(s).<br>`full` – Default; Expand abbreviations.<br>`short` – Abbreviate where possible. |

### *Valid Child Elements*

*None*

## <substring>

The <substring> action gets a span of characters in an arbitrary line of text.

Valid attribute combinations are:

- length
- start, length
- start, end

| Attribute | Description |
|---|---|
| length | The length of this span. If start is not specified, it is relative to the last mapping elements span end position or 0. |
| start | The start index of this span, where 0 is the beginning of the text. This must be a positive integer that is less than or equal to end index. |
| end | The end index of this span, where 0 is the beginning of the text. This must be a positive integer that is greater than or equal to start index. |

### *Valid Child Elements*

*None*

## <replace>

The <replace> action finds the specified pattern and replaces it with the specified text.

| *Attribute* | *Description* |
|---|---|
| pattern | A regular expression pattern. |
| replacement | Text that will replace anything that matches the pattern in the field value. |
| scope | `all` – Default; Replace every instance.<br>`first` – Only replace the first instance. |

### *Valid Child Elements*

*None*

## <capitalize>

The <capitalize> action capitalizes the string as a proper noun.
“WHAT YOU SHOULD KNOW” turns into “What You Should Know”.

### *Attribute Description*

*None*

### *Valid Child Elements*

*None*

## <lowercase>

The <lowercase> action changes the field value to all lower case characters.
“What You Should Know” turns into “what you should know”.

### *Attribute Description*

*None*

### *Valid Child Elements*

*None*

## <uppercase>

The <uppercase> action changes the field value to all upper case characters.
“What You Should Know” turns into “WHAT YOU SHOULD KNOW”.

### *Attribute Description*

*None*

### *Valid Child Elements*

*None*

## <compress>

The <compress> action merges consecutive whitespace into a single space.
“What You Should Know” turns into “What You Should Know”.

### *Attribute Description*

*None*

### *Valid Child Elements*

*None*

## <trim>

The <trim> action removes whitespace from the beginning and end of the field value.
“ What You Should Know ” turns into “What You Should Know”.

### *Attribute* *Description*

*None*

### *Valid Child Elements*

*None*

## <strip>

The <strip> action removes the specified character(s) from the beginning and end of the field value.
If `pattern` is a comma, then “,,,,,,What You Should Know,,,,,,,” turns into “What You Should Know”.

### *Attribute* *Description*

| | |
|---|---|
| pattern | The text to strip. |

### *Valid Child Elements*

*None*

## <pad>

The <pad> action pads the field value with text to the specified length.

### *Attribute* *Description*

| | |
|---|---|
| type | `left` – Default; Appends to the left. |
| | `right` – Append to the right. |
| with | The text to pad with. |
| length | The length to pad to. |

### *Valid Child Elements*

*None*

## <set>

The <set> action sets the field value explicitly. If the text attribute is specified, no child elements are allowed.

### *Attribute* *Description*

| | |
|---|---|
| text | The text to set the field value to. |
| if | `always` – Default; Always sets the field value. |
| | `empty` – Sets only if the field value is currently empty. (Empty is a blank string or null.) |

### *Valid Child Elements*

*Expression Values*

| | |
|---|---|
| <reference> | No |
| <constant> | No |

# Transformations

## <datetime>

The <datetime> transform converts dates and times to and from different formats. It also supports converting from one timezone into another, or using an alternative chronology. Any changes created by the timezone or chronology attributes will automatically account for timezone differences, daylight savings time, and leap years.

| *Attribute* | *Description* |
|---|---|
| pattern | A valid datetime pattern used to convert the date, time, and timezone information.<br>*See Appendix A.* |
| timezone | If converting a string to a datetime, and no timezone was specified by the pattern, assume this one.<br>If converting a string to a datetime, but timezone was specified by the pattern, change the date and time to reflect the appropriate timezone.<br>If converting a datetime to a string, change the date and time to reflect the appropriate timezone.<br>*See Appendix B.* |
| chronology | `ISO` – Default; Use the standard modern calendar.<br>`ISO8601` – Alias for `ISO`<br>`GregorianJulian` – Use the Gregorian Julian calendar.<br>`Gregorian` – Use the Gregorian calendar.<br>`Julian` – Use the strict Julian calendar.<br>`Coptic` – Use the Coptic calendar.<br>`Buddhist` – Use the Buddhist calendar.<br>`Ethiopic` – Use the Ethiopic calendar.<br>*See Appendix C.* |

## <boolean>

The <boolean> transform converts between boolean (yes/no) values. Case-insensitive.
If the format is `any`, this transform cannot be the last transform in a <transform> element.

| *Attribute* | *Description* |
|---|---|
| format | `any` – Default; Looks at all available formats to determine what the value is.<br>`truefalse` – Only TRUE and FALSE are allowed.<br>`yn` – Only Y and N are allowed.<br>`yesno` – Only YES and NO are allowed.<br>`bit` – Only `1` and `0` are allowed. |

## <number>

The <number> transform converts and formats arbitrary numbers.

| *Attribute* | *Description* |
|---|---|
| pattern | A valid number pattern for conversion and formatting.<br>*See Appendix D.* |

## <integer>

The <integer> transform converts and outputs arbitrary integers.

| Attribute | Description |
|---|---|
| radix | The base, or radix, of the number to convert to/from. |

## <decimal>

The <decimal> transform converts and outputs arbitrary decimals.

| Attribute | Description |
|---|---|
| *None* | |

## <extract>

The <extract> finds a regular expression pattern within the field value and allows you to output a string that uses the capturing groups found by the regular expression.
The strings `$1`, `$2`, … `$9` are replaced by capturing groups 0 through 9.
The strings `\1`, `\2`, … `\9` are replaced by capturing groups 0 through 9.

| Attribute | Description |
|---|---|
| pattern | A valid regular expression pattern. |
| output | The replacement string. |

## <split>

The <split> transform splits strings and outputs several strings. Only certain transforms can operate on this array of strings, but all processors are supported.

| Attribute | Description |
|---|---|
| separator | The character(s) to split the string on. |

## <join>

The <join> transform joins together an array of strings into one single string.

| Attribute | Description |
|---|---|
| separator | The character(s) to put between each string. |

## Appendix A — Datetime Patterns

| Symbol | Meaning | Presentation | Examples |
|---|---|---|---|
| G | era | text | AD |
| C | century of era (>=0) | number | 20 |
| Y | year of era (>=0) | year | 1996 |
| x | weekyear | year | 1996 |
| w | week of weekyear | number | 27 |
| e | day of week | number | 2 |
| E | day of week | text | Tuesday; Tue |
| y | year | year | 1996 |
| D | day of year | number | 189 |
| M | month of year | month | July; Jul; 07 |
| d | day of month | number | 10 |
| a | halfday of day | text | PM |
| K | hour of halfday (0~11) | number | 0 |
| h | clockhour of halfday (1~12) | number | 12 |
| H | hour of day (0~23) | number | 0 |
| k | clockhour of day (1~24) | number | 24 |
| m | minute of hour | number | 30 |
| s | second of minute | number | 55 |
| S | fraction of second | number | 978 |
| z | time zone | text | Pacific Standard Time; PST |
| Z | time zone offset/id | zone | -0800; -08:00; America/Los_Angeles |
| ' | escape for text | delimiter | |
| '' | single quote | literal | ' |

## Appendix B — Common Timezone IDs

| Standard Offset | Canonical ID | Aliases |
|---|---|---|
| **-11:00** | Pacific/Pago_Pago | Pacific/Samoa, US/Samoa |
| **-10:00** | America/Adak | America/Atka, US/Aleutian |
| **-10:00** | Etc/GMT+10 | |
| **-10:00** | HST | |
| **-10:00** | Pacific/Honolulu | US/Hawaii |
| **-09:00** | America/Anchorage | US/Alaska |
| **-09:00** | Etc/GMT+9 | |
| **-08:00** | Etc/GMT+8 | |
| **-08:00** | PST8PDT | |
| **-07:00** | Etc/GMT+7 | |
| **-07:00** | MST | |
| **-07:00** | MST7MDT | |
| **-06:00** | CST6CDT | |
| **-06:00** | Etc/GMT+6 | |
| **-05:00** | EST | |
| **-05:00** | EST5EDT | |
| **-05:00** | Etc/GMT+5 | |

## Appendix C — Available Chronologies

| ID | Description |
|---|---|
| **ISO** | Based on the ISO8601 standard and suitable for use after about 1600 |
| **GregorianJulian** | Historically accurate calendar with Julian followed by Gregorian |
| **Gregorian** | The Gregorian calendar system used for all time (proleptic) |
| **Julian** | The Julian calendar system used for all time (proleptic) |
| **Buddhist** | The Buddhist calendar system which is an offset in years from GJ |
| **Coptic** | The Coptic calendar system which defines 30 day months |
| **Ethiopic** | The Ethiopic calendar system which defines 30 day months |

## Appendix D — Number Patterns

The number patterns support different kinds of numbers, including integers (123), fixed-point numbers (123.4), scientific notation (1.23E4), percentages (12%), and currency amounts ($123). All of these can be localized. A negative subpattern is optional, in case you want to format negative numbers differently.

Number patterns have the follow syntax:

| Symbol | Location | Localized? | Meaning |
|---|---|---|---|
| `0` | Number | Yes | Digit |
| `#` | Number | Yes | Digit, zero shows as absent |
| `.` | Number | Yes | Decimal separator or monetary decimal separator |
| `-` | Number | Yes | Minus sign |
| `,` | Number | Yes | Grouping separator |
| `E` | Number | Yes | Separates mantissa and exponent in scientific notation. *Need not be quoted in prefix or suffix.* |
| `;` | Subpattern boundary | Yes | Separates positive and negative subpatterns |
| `%` | Prefix or suffix | Yes | Multiply by 100 and show as percentage |
| `\u2030` | Prefix or suffix | Yes | Multiply by 1000 and show as per mille |
| `¤(\u00A4)` | Prefix or suffix | No | Currency sign, replaced by currency symbol. If doubled, replaced by international currency symbol. If present in a pattern, the monetary decimal separator is used instead of the decimal separator. |
| `'` | Prefix or suffix | No | Used to quote special characters in a prefix or suffix, for example, `"'#'#"` formats 123 to `"#123"`. To create a single quote itself, use two in a row: `"# o''clock"`. |

# Appendix E — Valid SQL Type Formats

### *Date and Time*

| Standard | Datetime Pattern |
|---|---|
| **Default** | `MMM dd yyyy hh:mmaa` |
| **Default + milliseconds** | `MMM dd yyyy hh:mm:ss:SSSaa` |
| **U.S.** | `MM/dd/yyyy` |
| **ODBC canonical** | `yyyy-MM-dd HH:mm:ss` |
| **ODBC canonical + milliseconds** | `yyyy-MM-dd HH:mm:ss.SSS` |
| **ISO8601** | `yyyy-MM-dd'T'HH:mm:ss.SSS` |
| **ISO8601 + timezone** | `yyyy-MM-dd'T'HH:mm:ss.SSSZZ` |

### *Date*

| Standard | Datetime Pattern |
|---|---|
| **Default** | `MM/dd/yyyy` |
| **ISO8601** | `yyyy-MM-dd` |
| **ANSI** | `yy.MM.dd` |

### *Time*

| Standard | Datetime Pattern |
|---|---|
| **Default (24h)** | `HH:mm:ss` |
| **Default (24h) + milliseconds** | `HH:mm:ss:SSS` |
| **Default (12h)** | `hh:mmaa` |
| **Default (12h) + milliseconds** | `hh:mm:ss:SSSaa` |

### *Money*

| Standard | Number Pattern |
|---|---|
| **Default (USD)** | `$#,##0.00` |