EMu Documentation

Appending data using the EMu Import Facility

Document Version 1.0

EMu Version 4.0

www.kesoftware.com

Contents

SECTION 1

Overview

KE EMu 3.2.01 added the ability to import records specified in either XML (eXtensible Markup Language) or CSV (Comma Separated Values). The new Import Facility enabled records to be created, including the creation of any records referenced. Hence it is possible to import a new Catalogue record and have a Party record created for the *Creator* field if the Party record does not already exist. The Import Facility also provided a mechanism for updating records. Using the update feature it is possible to replace the contents of an existing field with new data. Unfortunately the complete contents of the field are replaced with the data imported.

In some instances it may be useful to update a list of values by:

- appending values after a given position or after the last value.
- prepend values before a given position or before the first value.
- replace values at a given position.

EMu 4.0.02 extends the Import Facility, allowing tables and nested tables of values to be updated rather than replaced. It is now possible to append, prepend and replace values at given positions with imported data.

A new *group* mechanism allows separate columns to be tied together when appending data, ensuring all values are placed on the same row in each column. For example, when updating a Catalogue record you may want the *Creator* and *Date Created* values to appear on the same row, as each piece of information is related to the other.

In the following section we will look at the extensions added to the Import Facility to support the appending of data for both XML and CSV based data sources.

SECTION 2

Appending data

The EMu Import Facility allows XML or CSV to be used for defining records for importing. The XML format used is the same as that produced by the EMu Reporting Facility, while CSV is a de facto standard for data interchange. Consider the XML below:

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatSource_tab'>
            <tuple>
                <atom>Map Number 2314</atom>
            </tuple>
        </table>
    </tuple>
</table>
```

or the equivalent CSV:

irn	LatSource_tab(1)
5000432	Map Number 2314

When imported, all values in the *LatSource_tab* column will be replaced with the value `Map Number 2314`. To provide support for updating rows in a table a new row attribute has been added to the `<tuple>` tag for XML, allowing a row position to be specified. The row attribute provides a mechanism for indicating what type of update should be applied and which row is affected. The format of the attribute is:

`<tuple row='`*value*`'>`

For a CSV data source the row number appears between brackets. CSV allows two formats to be used to specify the row attribute:

`(row='`*value*`')`

or the shorter form:

`(`*value*`)`

While double quotes may be used to enclose the row *value*, single quotes are recommended for CSV files as the CSV format uses double quotes to enclose data values. When saving CSV files in Excel, incorrect output will be produced if double quotes are used when specifying the row *value*.

The *value* of the row attribute may be:

nnn	where **nnn** is a row number. The number indicates the row position in the list of values to be modified. The first row is numbered 1. In essence this setting replaces the current value at that position.
+	indicates the row position is after the last value in the table. Any data will be appended to the list of values.
-	indicates the row position is before the first value in the table. Any data added will be put at the start of the list with existing values moved down.
=	indicates the row position is row one. Any data added will replace the existing value at this position.
nnn+	appends any data after row number *nnn*.
nnn-	prepends any data before row number *nnn*.
nnn=	replaces any data at row number *nnn*.

In all cases, if the new row number does not exist, it is created. For example, if the row setting is:

```
<tuple row='12+'>
```

or for CSV:

```
(row='12+')
```
or
```
(12+)
```

and there are not twelve values in the table, the table would be padded out to twelve values (with empty rows) and the new value appended, creating a thirteenth row. If a row attribute is not defined, the default behaviour is to append to the end of the table.

In order to provide backwards compatibility with the existing Import Facility and to allow all values in a table to be replaced, the first <tuple> in a table for XML, or the first row specifier for CSV, define whether the values in the table are being updated or whether all values are being replaced. If the row attribute is not specified in XML, or just a row number is specified, then the contents of the column will be cleared and the imported values added. If a row attribute is provided and it contains an update operator (that is +, - or = with or without a leading row number), the contents of the table are updated, with the existing values retained.

The following section contains examples on how to replace or update the contents of a table using the new row attribute. The examples provide both XML and CSV solutions.

Updating tables

The examples below show how the row attribute can be used to overwrite or update the contents of table based columns.

Example one - Replacing the contents of a table

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatSource_tab'>
            <tuple>
                <atom>new source 1</atom>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatSource_tab(1)
5000432	new source 1

As the row attribute value in the *LatSource_tab* column does not contain an update modifier (that is, +, - or =), the contents of the column are cleared before adding in the new data. The tables below show before and after cases for the above import:

Before

1	source 1
2	source 2
3	source 3
4	source 4

After

1	new source 1

Example two - Replacing the contents of a table and skipping rows

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatSource_tab'>
            <tuple row="1">
                <atom>new source 1</atom>
            </tuple>
            <tuple row="3">
                <atom>new source 3</atom>
            </tuple>
            <tuple row="5">
                <atom>new source 5</atom>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatSource_tab(1)	LatSource_tab(3)	LatSource_tab(5)
5000432	new source 1	new source 3	new source 5

Once again as the first row attribute does not contain an update modifier, the contents of *LatSource_tab* will be cleared before the imported values are added. The use of the row attribute allows specific row positions to be set. The example below shows before and after cases of the above import:

Before

1	source 1
2	source 2
3	source 3
4	source 4

After

1	new source 1
2	
3	new source 3
4	
5	new source 5

Example three - Appending a value to the end of a table

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatSource_tab'>
            <tuple row="+">
                <atom>append source 1</atom>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatSource_tab(+)
5000432	append source 1

As the row attribute value contains an update modifier, +, the existing contents of the *LatSource_tab* field are retained and added to:

Before

1	source 1
2	source 2
3	source 3
4	source 4

After

1	source 1
2	source 2
3	source 3
4	source 4
5	append source 1

Example four - Appending multiple values to the end of a table

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatSource_tab'>
            <tuple row="+">
                <atom>append source 1</atom>
            </tuple>
            <tuple row="+">
                <atom>append source 2</atom>
            </tuple>
            <tuple row="+">
                <atom>append source 3</atom>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatSource_tab(+)	LatSource_tab(+)	LatSource_tab(+)
5000432	append source 1	append source 2	append source 3

Since the first row attribute contains an update modifier (the + modifier), the imported data updates the existing table values. Notice the use of the append modifier for each row to be appended:

Before

1	source 1
2	source 2
3	source 3
4	source 4

After

1	source 1
2	source 2
3	source 3
4	source 4
5	append source 1
6	append source 2
7	append source 3

Example five - Prepending a value to the front of a table

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatSource_tab'>
            <tuple row="-">
                <atom>prepend source 1</atom>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatSource_tab(-)
5000432	prepend source 1

Inserting a value into the first position in a table column is similar to appending, except that the prepend row operator (-) is used. When the value is inserted, existing values are moved down the table:

Before

1	source 1
2	source 2
3	source 3
4	source 4

After

1	prepend source 1
2	source 1
3	source 2
4	source 3
5	source 4

Example six - Replacing values in a table

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatSource_tab'>
            <tuple row="2=">
                <atom>replace source 1</atom>
            </tuple>
            <tuple row="3=">
                <atom>replace source 2</atom>
            </tuple>
            <tuple row="6=">
                <atom>replace source 3</atom>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatSource_tab(2=)	LatSource_tab(3=)	LatSource_tab(6=)
5000432	replace source 1	replace source 2	replace source 3

The = row operator is used to replace the contents of a row in a table. If the row does not exist, it is created and the appropriate value set:

Before

1	source 1
2	source 2
3	source 3
4	source 4

After

1	source 1
2	replace source 1
3	replace source 2
4	source 4
5	
6	replace source 3

Example seven - Appending a reference to the end of a table

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatDeterminedByRef_tab'>
            <tuple row="+">
                <atom name="NamFirst">firstname</atom>
                <atom name="NamLast">lastname</atom>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatDeterminedByRef_tab(+).NamFirst	LatDeterminedByRef_tab(+).NamLast
5000432	firstname	lastname

All the update modifiers available for tables of values may be used for tables of references. For CSV data sources, columns with the same name and row attribute are handled as one update. For example, the CSV row attributes above both use the append operator (+). This does not result in two new rows, rather the same row is used to contain the reference:

Before

1	reference 1
2	reference 2
3	reference 3
4	reference 4

After

1	reference 1
2	reference 2
3	reference 3
4	reference 4
5	append reference 1

Updating nested tables

The mechanism for updating nested tables, that is tables within tables, is very similar to updating a single table. The only difference is that nested tables have an outer row number and an inner row number. Each outer row contains a table of inner row values:

Outer Row	Inner Row	Data
1	1	opinion 1, comment 1
	2	opinion 1, comment 2
	3	opinion 1, comment 3
2	1	opinion 2, comment 1
3	1	opinion 3, comment 1
	2	opinion 3, comment 2

As you can see from the table above, outer row one contains a table with three values, while outer row two has one value and outer row three, two values. The XML representation of the above table would be:

```
<table table='ecollectionevents'>
    <tuple>
        <table name='LatComment_nesttab'>
            <tuple>
                <table>
                    <tuple>
                        <atom>opinion 1, comment 1</atom>
                    </tuple>
                    <tuple>
                        <atom>opinion 1, comment 2</atom>
                    </tuple>
                    <tuple>
                        <atom>opinion 1, comment 3</atom>
                    </tuple>
                </table>
            </tuple>
            <tuple>
                <table>
                    <tuple>
                        <atom>opinion 2, comment 1</atom>
                    </tuple>
                </table>
            </tuple>
            <tuple>
                <table>
                    <tuple>
                        <atom>opinion 3, comment 1</atom>
                    </tuple>
                    <tuple>
                        <atom>opinion 3, comment 2</atom>
                    </tuple>
                </table>
            </tuple>
        </table>
    </tuple>
</table>
```

The nested table *LatComment_nesttab* starts with the `<table name='LatComment_nesttab'>` tag. Each outer row is enclosed in a `<tuple>` tag. The outer rows are coloured green. Within each outer row is a table of inner rows. The inner rows are enclosed in red `<tuple>` tags. When updating nested tables both the outer and inner row `<tuple>` tags may have row attributes. Hence it is possible to append/prepend/replace outer and/or inner rows.

The equivalent CSV representation is shown below. The table has been turned on its side for ease of viewing. The column names should appear in the first row rather than the first column:

irn	5000432
LatComment_nesttab(1:1)	opinion 1, comment 1
LatComment_nesttab(1:2)	opinion 1, comment 2
LatComment_nesttab(1:3)	opinion 1, comment 3
LatComment_nesttab(2:1)	opinion 2, comment 1
LatComment_nesttab(3:1)	opinion 3, comment 1
LatComment_nesttab(3:2)	opinion 3, comment 2

The outer and inner rows for nested tables in CSV are recorded after the column name enclosed in brackets separated by a colon. The outer row is shown in green, while the inner row is red. The example above uses the short form for the row attributes. The long form would look like:

```
LatComment_nesttab(row='1':row='1')
```

As with XML the outer and/or inner rows may contain update modifiers. The following examples detail some common uses.

Example one - Replacing the contents of a nested table

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatComment_nesttab'>
            <tuple>
                <table>
                    <tuple>
                        <atom>new opinion 1, new comment 1</atom>
                    </tuple>
                </table>
            </tuple>
            <tuple>
                <table>
                    <tuple>
                        <atom>new opinion 2, new comment 1</atom>
                    </tuple>
                    <tuple>
                        <atom>new opinion 2, new comment 2</atom>
                    </tuple>
                </table>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatComment_ nesttab(1:1)	LatComment_ nesttab(2:1)	LatComment_ nesttab(2:2)
5000432	new opinion 1, new comment 1	new opinion 2, new comment 1	new opinion 2, new comment 2

As the first outer row and first inner row do not contain row attributes the current contents of the *LatComments_nesttab* column will be cleared:

Before

1	1	opinion 1, comment 1
	2	opinion 1, comment 2
	3	opinion 1, comment 3
2	1	opinion 2, comment 1
3	1	opinion 3, comment 1
	2	opinion 3, comment 2

After

1	1	new opinion 1, new comment 1
2	1	new opinion 2, new comment 1
	2	new opinion 2, new comment 2

Example two - Appending an outer row to a nested table

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatComment_nesttab'>
            <tuple row="+">
                <table>
                    <tuple>
                        <atom>append opinion 1, comment 1</atom>
                    </tuple>
                    <tuple>
                        <atom>append opinion 1, comment 2</atom>
                    </tuple>
                </table>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatComment_nesttab(+:1)	LatComment_nesttab(+:2)
5000432	append opinion 1, comment 1	append opinion 1, comment 2

The first outer row has a row attribute with a value of +, indicating an outer row is to be appended to the existing data. The inner table contains the values to be added:

Before

1	1	opinion 1, comment 1
	2	opinion 1, comment 2
	3	opinion 1, comment 3
2	1	opinion 2, comment 1
3	1	opinion 3, comment 1
	2	opinion 3, comment 2

After

1	1	opinion 1, comment 1
	2	opinion 1, comment 2
	3	opinion 1, comment 3
2	1	opinion 2, comment 1
3	1	opinion 3, comment 1
	2	opinion 3, comment 2
4	1	append opinion 1, comment 1
	2	append opinion 1, comment 2

Example three - Appending values to an inner table in a nested table

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatComment_nesttab'>
            <tuple row="2">
                <table>
                    <tuple row="+">
                        <atom>opinion 2, append comment 1</atom>
                    </tuple>
                    <tuple>
                        <atom>opinion 2, append comment 2</atom>
                    </tuple>
                </table>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatComment_nesttab(2:+)	LatComment_nesttab(2:+)
5000432	opinion 2, append comment 1	opinion 2, append comment 2

The outer row contains a fixed row number while the first inner row contains an update modifier, the append (+) modifier. The combination indicates that values are to be appended to the end of the second outer row:

Before

1	1	opinion 1, comment 1
	2	opinion 1, comment 2
	3	opinion 1, comment 3
2	1	opinion 2, comment 1
3	1	opinion 3, comment 1
	2	opinion 3, comment 2

After

1	1	opinion 1, comment 1
	2	opinion 1, comment 2
	3	opinion 1, comment 3
2	1	opinion 2, comment 1
	2	opinion 2, append comment 1
	3	opinion 2, append comment 2
3	1	opinion 3, comment 1
	2	opinion 3, comment 2

Example four - Prepending an outer row to a nested table

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatComment_nesttab'>
            <tuple row="2-">
                <table>
                    <tuple>
                        <atom>prepend opinion 2, comment 1</atom>
                    </tuple>
                    <tuple>
                        <atom>prepend opinion 2, comment 2</atom>
                    </tuple>
                </table>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatComment_nesttab(2-:1)	LatComment_nesttab(2-:2)
5000432	prepend opinion 2, comment 1	prepend opinion 2, comment 2

The outer row attribute value of `2-` will prepend an outer row before row two and move the existing second row and subsequent rows down. The inner table contains the values to be set:

Before

1	1	opinion 1, comment 1
	2	opinion 1, comment 2
	3	opinion 1, comment 3
2	1	opinion 2, comment 1
3	1	opinion 3, comment 1
	2	opinion 3, comment 2

After

1	1	opinion 1, comment 1
	2	opinion 1, comment 2
	3	opinion 1, comment 3
2	1	prepend opinion 2, comment 1
	2	prepend opinion 2, comment 2
3	1	opinion 2, comment 1
4	1	opinion 3, comment 1
	2	opinion 3, comment 2

Example five - Replacing an outer row in a nested table

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatComment_nesttab'>
            <tuple row="2=">
                <table>
                    <tuple>
                        <atom>replace opinion 2, comment 1</atom>
                    </tuple>
                    <tuple row="3">
                        <atom>replace opinion 2, comment 3</atom>
                    </tuple>
                </table>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatComment_nesttab(2=:1)	LatComment_nesttab(2=:3)
5000432	replace opinion 2, comment 1	replace opinion 2, comment 3

The `2=` row attribute on the outer row indicates row two is to be replaced. Note the use of the row attribute in the inner table to skip row two (causing a blank value to be created):

Before

1	1	opinion 1, comment 1
	2	opinion 1, comment 2
	3	opinion 1, comment 3
2	1	opinion 2, comment 1
3	1	opinion 3, comment 1
	2	opinion 3, comment 2

After

1	1	opinion 1, comment 1
	2	opinion 1, comment 2
	3	opinion 1, comment 3
2	1	replace opinion 2, comment 1
	2	
	3	replace opinion 2, comment 3
3	1	opinion 3, comment 1
	2	opinion 3, comment 2

Example six - Replacing an inner row in a nested table

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatComment_nesttab'>
            <tuple row="3">
                <table>
                    <tuple row="1=">
                        <atom>opinion 3, replace comment 1</atom>
                    </tuple>
                    <tuple row="+">
                        <atom>opinion 3, append comment 1</atom>
                    </tuple>
                </table>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatComment_nesttab(3:1=)	LatComment_nesttab(3:+)
5000432	opinion 3, replace comment 1	opinion 3, append comment 1

The outer row attribute value indicates row 3 is to be changed, while the first inner row indicates row one is to be replaced. The second inner row attribute will append a value to the end of the inner table:

Before

1	1	opinion 1, comment 1
	2	opinion 1, comment 2
	3	opinion 1, comment 3
2	1	opinion 2, comment 1
3	1	opinion 3, comment 1
	2	opinion 3, comment 2

After

1	1	opinion 1, comment 1
	2	opinion 1, comment 2
	3	opinion 1, comment 3
2	1	opinion 2, comment 1
3	1	opinion 3, replace comment 1
	2	opinion 3, comment 2
	3	opinion 3, append comment 1

Groups

The ability to append data to an existing list allows new values to be added to columns over time. One issue that may arise is the need to append values to more than one column, ensuring the values are all added at the same row position. Consider the following update:

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatSource_tab'>
            <tuple row="+">
                <atom>append source 1</atom>
            </tuple>
        </table>
        <table name='LatLatLongDetermination_tab'>
            <tuple row="+">
                <atom>append determination 1</atom>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatSource_tab(+)	LatLatLongDetermination_tab(+)
5000432	append source 1	append determination 1

with the following result:

Before

1	source 1	determination 1
2	source 2	determination 2
3	source 3	

After

1	source 1	determination 1
2	source 2	determination 2
3	source 3	append determination 1
4	append source 1	

The resulting data is almost certainly not what was envisioned. The two columns, *LatSource_tab* and *LatLatLongDetermination_tab* are related and appear in the one grid. As such there is an implied relationship between values in a row. For example, all values in the first row apply to the first opinion, while values in the second row apply to the second opinion and so on. When appending a new opinion, all the added data values need to appear in the same row regardless of any empty values already present.

As the Import Facility does not know the relationship between columns, a new attribute, the group attribute, has been added allowing columns to be tied together.

The attribute is also used to set the row position to be the first row to which data can be appended across all the columns in the group. For XML data, the group attribute appears in `<tuple>` tags along with the row attribute. For CSV, the group attribute is placed after the row attribute. It only makes sense to set a group where the row attribute is `row="+"`.

Consider the following update where a group is specified:

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatSource_tab'>
            <tuple row="+" group="opinion">
                <atom>append source 1</atom>
            </tuple>
        </table>
        <table name='LatLatLongDetermination_tab'>
            <tuple row="+" group="opinion">
                <atom>append determination 1</atom>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

irn	LatSource_tab(+ group='opinion')	LatLatLongDetermination_tab(+ group='opinion')
5000432	append source 1	append determination 1

Notice how the group attribute is defined on both the *LatSource_tab* and *LatLatLongDetermination_tab* columns. The group value can be any string. All columns with the same group value are examined to determine the row position to use when appending data. More than one group may be used where disjoint (unrelated) groups of related columns exists. Each disjoint set of related columns should have its own group name. The above update will produce:

Before

1	source 1	determination 1
2	source 2	determination 2
3	source 3	

After

1	source 1	determination 1
2	source 2	determination 2
3	source 3	
4	append source 1	append determination 1

which is probably what was required in the first place. Groups may be used with nested tables in the same way they can be with tables. A group may be applied to the outer and/or inner tuples in the nested table. The following update ensures the Source (*LatSource_tab*), Determination (*LatLatLongDetermination_tab*), Latitude

(*LatLatitude_nesttab*) and Longitude (*LatLongitude_nesttab*) values all append to the same row:

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatSource_tab'>
            <tuple row="+" group="opinion">
                <atom>append source 1</atom>
            </tuple>
        </table>
        <table name='LatLatLongDetermination_tab'>
            <tuple row="+" group="opinion">
                <atom>append determination 1</atom>
            </tuple>
        </table>
        <table name='LatLatitude_nesttab'>
            <tuple row="+" group="opinion">
                <table>
                    <tuple>
                        <atom>new opinion 1, latitude 1</atom>
                    </tuple>
                    <tuple>
                        <atom>new opinion 1, latitude 2</atom>
                    </tuple>
                </table>
            </tuple>
        </table>
        <table name='LatLongitude_nesttab'>
            <tuple row="+" group="opinion">
                <table>
                    <tuple>
                        <atom>new opinion 1, longitude 1</atom>
                    </tuple>
                    <tuple>
                        <atom>new opinion 1, longitude 2</atom>
                    </tuple>
                </table>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

The table has been turned on its side for ease of viewing.

irn	5000432
LatSource_tab(+ group='opinion')	append source 1
LatLatLongDetermination_tab(+ group='opinion')	append determination 1
LatLatitude_nesttab(+ group='opinion':1)	new opinion 1, latitude 1
LatLatitude_nesttab(+ group='opinion':2)	new opinion 1, latitude 2
LatLongitude_nesttab(+ group='opinion':1)	new opinion 1, longitude 1
LatLongitude_nesttabb(+ group='opinion':2)	new opinion 1, longitude 2

Notice how the group attribute is applied to the outer row to ensure the new latitude and longitude values are added to the same row position as the source and determination values:

Before

1	1	source 1	determination 1	opinion 1, latitude 1	opinion 1, longitude 1
	2			opinion 1, latitude 2	opinion 1, longitude 2
	3			opinion 1, latitude 3	opinion 1, longitude 3
2	1	source 2		opinion 2, latitude 1	opinion 2, longitude 1

After

1	1	source 1	determination 1	opinion 1, latitude 1	opinion 1, longitude 1
	2			opinion 1, latitude 2	opinion 1, longitude 2
	3			opinion 1, latitude 3	opinion 1, longitude 3
2	1	source 2		opinion 2, latitude 1	opinion 2, longitude 1
3	1	append source 1	append determination 1	new opinion 1, latitude 1	new opinion 1, longitude 1
	2			new opinion 1, latitude 2	new opinion 1, longitude 2

The final point to cover concerning groups is that the row position at which new data is appended is calculated the first time the group is encountered in the data file. Once the row position is determined it is used for all columns in the group for the current import record. This means the group name used to bind columns together should only appear on the first row to be appended, rather than every row. If the group name appears on every row, each value updated will overwrite the previous value. Consider the update:

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatSource_tab'>
            <tuple row="+" group="opinion">
                <atom>append source 1</atom>
            </tuple>
            <tuple row="+" group="opinion">
                <atom>append source 2</atom>
            </tuple>
        </table>
        <table name='LatLatLongDetermination_tab'>
            <tuple row="+" group="opinion">
                <atom>append determination 1</atom>
            </tuple>
            <tuple row="+" group="opinion">
                <atom>append determination 2</atom>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

The table has been turned on its side for ease of viewing.

irn	5000432
LatSource_tab(+ group='opinion')	append source 1
LatSource_tab(+ group='opinion')	append source 2
LatLatLongDetermination_tab(+ group='opinion')	append determination 1
LatLatLongDetermination_tab(+ group='opinion')	append determination 2

This results in:

Before

1	source 1	determination 1
2	source 2	

After

1	source 1	determination 1
2	source 2	
3	append source 2	append determination 2

You may have noticed the value `append source 2` has overwritten the value `append source 1`. The reason is that when group `opinion` was encountered there were two rows in *LatSource_tab*. Hence group `opinion` refers to row three. As the second `<tuple>` uses the same group name as the first, they both refer to the row used by group `opinion` (row three in this case). Hence the second value overwrites the first value. If you want to append multiple values where the columns are related, a separate group name should be given for each row to be appended. That is:

XML

```
<table table='ecollectionevents'>
    <tuple>
        <atom name='irn'>5000432</atom>
        <table name='LatSource_tab'>
            <tuple row="+" group="opinion 1">
                <atom>append source 1</atom>
            </tuple>
            <tuple row="+" group="opinion 2">
                <atom>append source 2</atom>
            </tuple>
        </table>
        <table name='LatLatLongDetermination_tab'>
            <tuple row="+" group="opinion 1">
                <atom>append determination 1</atom>
            </tuple>
            <tuple row="+" group="opinion 2">
                <atom>append determination 2</atom>
            </tuple>
        </table>
    </tuple>
</table>
```

CSV

The table has been turned on its side for ease of viewing.

irn	5000432
LatSource_tab(+ group='opinion 1')	append source 1
LatSource_tab(+ group='opinion 2')	append source 2
LatLatLongDetermination_tab(+ group='opinion 1')	append determination 1
LatLatLongDetermination_tab(+ group='opinion 2')	append determination 2

Which results in:

Before

1	source 1	determination 1
2	source 2	

After

1	source 1	determination 1
2	source 2	
3	append source 1	append determination 1
4	append source 2	append determination 2

Index