Outbound (CX Contact) CX Contact Help

Advanced Input - Preprocessing

12/9/2023

Contents

- 

- Administrator

Learn how to use various advanced input techniques to modify data when creating data mapping schemas.

Related documentation:

-
-

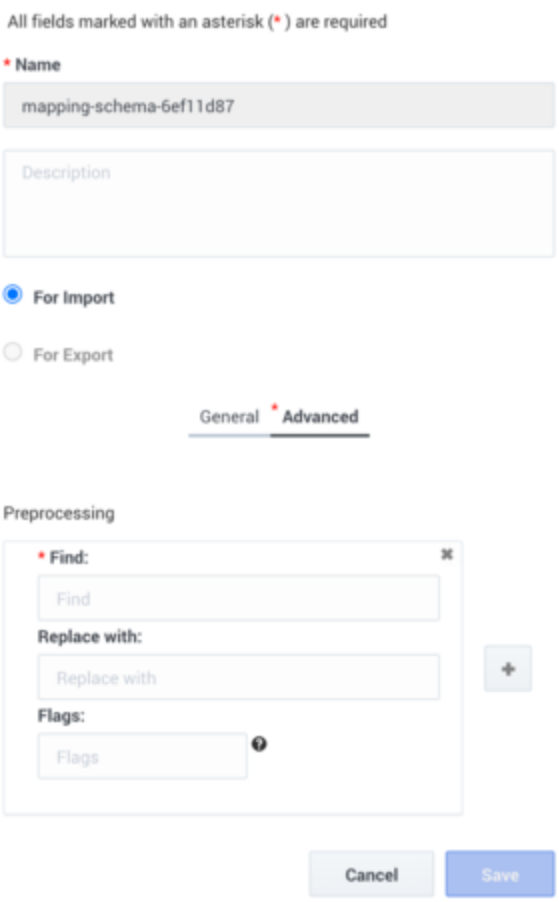
Periodically, to achieve your goals when creating data mapping schemas for input files, data must be modified when it is loaded. The technique used to alter data during import is called **preprocessing**.

This section explains the tips and techniques used to achieve some of the more common preprocessing requests.

When using an input specification file, the pre-process lines in the input spec must appear before everything else that is grabbed.

Also, you can use several preprocessing statements in one input spec. The statements must be added in the order you want them to run. Preprocessing can be added to Data Mapping Schema using the Advance tab (see the image below) when creating or editing a Data Mapping Schema.

> ### Important
> One preprocessing statement can create a result that may be unintentionally manipulated by a second preprocessing statement. For this reason, you must check the results when using preprocessing.



Replace Abbreviations With Full Words For Proper TTS

The following statement should be used when searching for occurrences of case insensitive data, and it can significantly reduce the number of required preprocessing statements.

For example, the statement below will find and replace instances of dr, Dr, DR, and dR.

Find	Replace
^([^,]*,[^,]*,[^,]*,["]*[^",]*)((?i) dr,)(.*)$	$1 Drive,$3

 # Pull out Dr from address and replace with Drive

Find	Replace
^([^,]*,[^,]*,[^,]*,[^,]*,[^,]*,["]*[^",]*) (Dr)(.*)$	$1 Drive$3

Find	Replace
^([^,]*,[^,]*,[^,]*,[^,]*,[^,]*,["]*[^",]*) (dr)(.*)$	$1 Drive$3

 # Pull out St from address and replace with Street

Find	Replace
^([^,]*,[^,]*,[^,]*,[^,]*,[^,]*,["]*[^",]*) (St)(.*)$	$1 Street$3
^([^,]*,[^,]*,[^,]*,[^,]*,[^,]*,["]*[^",]*) (st)(.*)$	$1 Street$3
^(.*) DR,(.*)$	$1 Drive,$2
^(.*) ST,(.*)$	$1 Street,$2
^(.*) HWY,(.*)$	$1 Highway,$2
^(.*) LN,(.*)$	$1 Lane,$2
^(.*) PL,(.*)$	$1 Place,$2
^(.*) CT,(.*)$	$1 Court,$2
^(.*) WY,(.*)$	$1 Way,$2
^(.*) CIR,(.*)$	$1 Circle,$2
^(.*) NE,(.*)$	$1 Northeast,$2
^(.*) NW,(.*)$	$1 Northwest,$2
^(.*) SE,(.*)$	$1 Southeast,$2
^(.*) SW,(.*)$	$1 Southwest,$2
^(ST)(.*)$	Saint$2
^(St)(.*)$	Saint$2

TTS evaluates certain abbreviations in an unintended manner. For example, Dr is spoken as doctor and St is spoken as saint if the TTS engine cannot put abbreviations in the proper context.

This could be a problem is you performed TTS on an address, since TTS would result in 65 Burlington Doctor, instead of 65 Burlington Drive.

The following preprocess statements are examples of how it is possible to convert some of these abbreviations (in this case, the 6th comma-separated field), into what they should be.

To perform this type of abbreviation replacement on a fixed width file, you should perform the following so that you do not encounter a situation in which data will be replaced with variable length words.

Find	Replace
^(.*) DR (.*)$	$1 $2Drive
^(.*) ST (.*)$	$1 $2Street
^(.*) HWY (.*)$	$1 $2Highway
^(.*) LN (.*)$	$1 $2Lane

Find	Replace
^(.*) PL (.*)$	$1 $2Place
^(.*) CT (.*)$	$1 $2Court
^(.*) WY (.*)$	$1 $2Way
^(.*) CIR (.*)$	$1 $2Circle
^(.*) NE (.*)$	$1 $2Northeast
^(.*) NW (.*)$	$1 $2Northwest
^(.*) SE (.*)$	$1 $2Southeast
^(.*) SW (.*)$	$1 $2Southwest

In the following example, specific abbreviations with specific amounts of space around them are replaced with the a specific amount of space with the addition of words at the end of the record. For example, if you insert an address in a field, you will pull the old street address, skip a bunch of characters that would normally bring you directly to the end of the record, and finally grab the amount of characters that you added at the end. For instance, note that the word replacements in the above example are all 9 characters long.

 # Remove ND only if it is preceded with a number. As a result, SECOND does not become SECO

Find	Replace
^(.*)([0-9]+)ND (.*)$	$1$2 $3
^(.*)([0-9]+)Nd (.*)$	$1$2 $3
^(.*)([0-9]+)nd (.*)$	$1$2 $3

 # Remove TH only if it is preceded with a number. As a result, NORTH does not become NOR

In the following table, TH or ND are removed from an address so that 19th street is replaced with 19 street.

Find	Replace
^(.*)([0-9]+)TH (.*)$	$1$2 $3
^(.*)([0-9])Th (.*)$	$1$2 $3
^(.*)([0-9]+)th (.*)$	$1$2 $3

Replace Text With Something Else

Use pre process to format any part of the data file before the rest of the input spec looks at the data file.

In the first example, an attempt is made to verify if either one of the telephone fields only includes

zeros. If it does only include zeros, the zeros will be replaced by the number 9.

 # Replace phone number

Find	Replace
^(.{103})0000000000(.{40})0000000000(.*)$	$19999999999$2 $3

In the second example, an attempt is made to verify if one of the calling fields only includes zero. Both the first and second examples should be used together to avoid records being rejected.

 # Blank the phone fields

Find	Replace
^(.{103})0000000000(.*)$	$1 $2
^(.{153})0000000000(.*)$	$1 $2

In the third example, the delimiter is changed. It is possible to use this example in various ways to show a field with its own special delimiter, appear as its own field.

 # Change all ~ to,

Find	Replace
^([^~]*)~([^~]*)~([^~]*)~([^~]*)$	$1,$2,$3,$4

In the fourth example, a static value is replaced with a different value.

 # Find a value and change it

Find	Replace
^([^|]*|[^|]*|[^|]*|)(6)(.*)$	$18882493432$3

Add Fields After an Entire Record

 # Add a campaign name to the end of a record

In the following example, the campaign name always appears in the data. If for example, the campaign name is imported as Other1, you can create a field in an output spec to display a Campaign Name as well as a Sub Campaign name.

Find	Replace
^(.*)$	$1,Campaign Name

 # Add a campaign name to the end of a record

In the following example, ther campaign name always appears in the data. If for example, the

campaign name is imported as Other1, you can create a field in an output spec to display a Campaign Name as well as a Sub Campaign name.

Find	Replace
^(.*)$	$1,Campaign Name

 # Add fields when you see specific numbers in a specific position

In the following example, 8001231234 is added to the end of each record in the original file. 8001231234 is then treated as part of the original file and is grabbed into one of the Other fields for use as a DC number, and so on. The position for this additional field is one more than the number of field positions in the original data file.

Find	Replace
^([^,]*,)(12345)(.*)$	$1$2$3,STUFF

The following is another example in which the client supplies 2 phone numbers in the 9th and 10th fields of each row. These 2 fields are the last 2 fields in each record. All records should always have a phone number in the 9th field. Some records have a number in the 10th field. If a valid number exists in the 10th field, then that should be the only contact number. If the number in the 10th field is missing or mangled, then the number in the 9th field will be used instead. In addition, notice that a comma is added to the end of each data line so that a series of comparisons can be performed to see which number should be grabbed.

Find	Replace
^(.*)$	$1,
^([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^	$1,$2,$3,$4,$5,$6,$7,$8,$9,,$9
^([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^	$1,$2,$3,$4,$5,$6,$7,$8,$9,$(10),$9
^([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^	$1,$2,$3,$4,$5,$6,$7,$8,$9,$(10),$9
^([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^	$1,$2,$3,$4,$5,$6,$7,$8,$9,$(10),$9
^([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^	$1,$2,$3,$4,$5,$6,$7,$8,$9,$(10),$9
^([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^	$1,$2,$3,$4,$5,$6,$7,$8,$9,$(10),$9
^([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^	$1,$2,$3,$4,$5,$6,$7,$8,$9,$(10),$9
^([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^	$1,$2,$3,$4,$5,$6,$7,$8,$9,$(10),$9
^([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^	$1,$2,$3,$4,$5,$6,$7,$8,$9,$(10),$9
^([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^	$1,$2,$3,$4,$5,$6,$7,$8,$9,$(10),$9
^([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^	$1,$2,$3,$4,$5,$6,$7,$8,$9,$(10),$(10)
^([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^,]*),([^	$1,$2,$3,$4,$5,$6,$7,$8,,$(10),$(10)

Past Custom Example - Force Variable-Length Field to Trim or Pad

In the following examples, the client has a score value in the 13th data field. The score value can be up to 4 digits long (for example, 1, 12, 123, 1234). If the score is less than 3 digits long, you must

pad zeroes at the beginning to turn it into 3 digits. If the score is 3 digits long, do not make any changes. If the score is 4 digits long, remove the first digit to turn it into a 3 digit number. This solution incorporates the preprocessing field order. Begin by reducing 4 digit scores into 3 digits. Add 3 zeros in front of all the scores. Finally, reduce scores of 6, 5, or 4 digits into 3 digits.

 # Turn a 4 character score into 3 by removing the first character

Find	Replace
^([^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],)(\d)(\d{3})(["].*)$	$1$2$3

 # Add 3 zeros in front anytime digits are found

Find	Replace
^([^,]*,[^,]*,[^,]*,[^,]*,[^,]*,[^,]*,[^,]*,[^,]*,[^,]*,[^,]*,[^,]*,["]*)(\d)(["]*.*)$	$1000$2$3

 # Turn 6 character numbers into 3

Find	Replace
^([^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],)(\d{3})(\d{3})(["].*)$	$1$2$3

 # Turn 5 character numbers into 3

Find	Replace
^([^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],)(\d{2})(\d{3})(["].*)$	$1$2$3

 # Turn 4 character numbers into 3

Find	Replace
^([^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],[^,],)(\d)(\d{3})(["]*)$	$1$2$3

Grab Required Data Even If Records Are Not The Same Length

In the following example, the client supplied data where the first 2 fields were the Client ID followed by the name. The 3rd field represents the home phone number. If the home phone number is the only data available, the row ends with no placeholder commas to indicate that there were additional fields without a value. Also, when a cell number does not exist, but a home phone number and a work phone number do exist, the data will not indicate that a cell number does not exist.

 # Fun

Find	Replace
^["]*([^",]*,[^",]*)["]*.*$	$1, , ,

For example:
12345,JAY DEE,7818972639
12346,FRANK DEE,7818971234,7818972354
12348,SUSIE DEE,7818975142,7818971092,7818976523

Using preprocessing, the solution adds extra fields to all records. In the preprocess statement above, there are 10 spaces between each comma. The length of a record row is not significant as long as the rightmost field the system is trying to grab can be grabbed (For example, in this case the cell phone number in the 5th field).

Therefore, by adding empty fields to all the rows, all possible data situations are satisfied. If there is only 1 phone number, extra fields are added to allow the Work and Cell grabs to work as expected. If there are only have 2 phone numbers, the extra fields are added to allow the Cell grab to work as expected. When all 3 numbers are available there is no significance to appending blank fields at the end of the row since they are already referenced.

Change Inconsistent Dates To Consistent Dates

The following is an example of when a client sends dates with a single digit month, single digit day, or both (refer to the fourth field of example below). In this example, that data must be changed so that is can be used with the makeDate function.
Travis Nelson,7818972639,53214,1/22/1979,$325.45 ,M,50000001
David Durski,7818972640,53211,12/3/1975,$148.45 ,M,T0050000002
Monika Mitchell,7818972641,85954,3/4/1980,$123.45 ,M,50000003

When importing into an Other field, the date format for the date format should be MMddyyyy. In this example, the preprocess removes the slashes from the date. Three preprocess lines are required to handle the possible month or day single digit scenarios. The result is that the dates are all the same and do not include slashes. The plus character (+) is added to the preprocess statement to enable slashes within the statement.

Find	Replace
^([^,]*,[^,]*,[^,]*,)(\d)/(\d{2})	(.*)$+$10$2$3$4+
^([^,]*,[^,]*,[^,]*,)(\d{2})/(\d)	(.*)$+$1$20$3$4+
^([^,]*,[^,]*,[^,]*,)(\d)/(\d)	(.*)$+$10$20$3$4+

Replace More Than One Instance Of Something

In the following example, the client wanted to remove the asterisks from their data and replace it with spaces. Each record could contain one or two stars depending on how many names were in the record. Example data:
Joe A*Schmoe Lucy K*Schmoe 6031231234 12345

Before performing the replacement, you must remember that preprocess matches and replaces the first instance it finds in a record. If the record contains another instance in the same record, the same preprocess statement will not replace the second instance. To overcome this, repeat the preprocess statement as many times as possible for the data you are trying to replace. The following was performed for the above example. The first statement replaces the first instance and the second statement finds the second instance.

Find	Replace
^(.*)*(.*)$	$1 $2
^(.*)*(.*)$	$1 $2

Secondary Name in Fixed Position Data

In the following example, there is no delimiter, commas in the name field, and the second name fields are predominately empty. The second name field is 40 bytes long and since the system needs to skip 62 bytes, a pipe must be seeded at byte 103 to use as a reference point later on.

 # Seed a pipe at byte 103 in order to grab the secondary name (other2)

Find	Replace
^(.{102})(.*)$	$1|$2

Remove Commas From Business Names

Some data includes commas in business names. Typically, the technique is used to remove the commas, but occasionally the data must be preprocessed. The following is an example of removing the commas using preprocessing. In this example, the field in question is the 3rd field. Anytime commas are embedded in a field, double quotes are entered around the field. In this example, the 3 lines will remove instances of 3 commas, 2 commas, or 1 comma respectively in that field. Since ampersands have been entered in the business names and it is important to keep them, the ampersands are included as part of the character classes for the match (word characters, white space characters, and ampersands).

Find	Replace
^([^,]*,[^,]*,)"([ws&]+),([ws&]+),([ws&]+),([ws&]+)"(.*)$	$1$2$3$4$5$6
^([^,]*,[^,]*,)"([ws&]+),([ws&]+),([ws&]+)"(.*)$	$1$2$3$4$5
^([^,]*,[^,]*,)"([ws&]+),([ws&]+)"(.*)$	$1$2$3$4

Remove Leading Zeros in a Currency Field

To remove leading zeros in a fixed position field you will need to replace them with spaces. The first

statement replaces the first zero with a space, the second statement detects the space and then a zero and replaces the zero with a space, and so on.

For example:
device other1 clientid
781897252000020134 1
781897252151781258 2
781897252200000134 3

The following solution turns 00000134 tp 1.34, or 51781258 to 517812.58, or 00000004 to 0.04.

Find	Replace
^(\d{10})[0]{1}(.*)$	$1 $2
^(\d{10}\s{1})[0]{1}(.*)$	$1 $2
^(\d{10}\s{2})[0]{1}(.*)$	$1 $2
^(\d{10}\s{3})[0]{1}(.*)$	$1 $2
^(\d{10}\s{4})[0]{1}(.*)$	$1 $2

Add Static Values to Variable Length Records in One Field

In this scenario, the client must add a static value to the end of all records, based on a specific flag. For example purposes, refer to the **Add some fields after an entire record** section above.

When static values are added to variable length records in one field the record appears as follows (for example, Test Camp is added at the end of all records with a colon delimiter ahead of it):
TEST,20408,TEST2:Test Camp

Using a unique delimiter with this static value for all records, enables you to locate specific information within the record.

This technique can only be used if the secondary delimiter is not used anywhere else in the original data file.

In the example below, the second field and the added campaign value are merged into a single field.

Find	Replace
^[^,]*,([^,]*),[^:]*:(.*)$	$1:$2

Use Regular Expression to Map Additional Devices

You can use regular expression (RegEx) in your data mapping schema when it includes importing devices greater than 1 to 10. CX Contact uses this functionality to extend the supported maximum of 10 devices.

For example, because there are only 10 supported device fields in the mapping schema, device11

and device12 might not be imported into CX Contact. However, fields 1 to 10 do not always contain devices—typically, some fields are left empty. By creating a regular expression, you can map device11 and device12 to the first and second empty field, so that they are imported along with all other devices. See the example below and in the tables (where D1 = device1, D2 = device2, and so on).

Use regular expression to include additional devices

In the current format (below), device 11 and 12 can remain mapped to their user-defined fields and labelled appropriately for reporting purposes.

Previous format

firstName	lastName	Device1	Device 2	Device 3	Device 4	Device 5	Device 6	Device 7	Device 8	Device 9	Device 10	Device11	Device12
Joe	Schmoe	+447973772619										+4412155555555	+4412166666666

Current format

firstName	lastName	Device1	Device2	Device3	Device4	Device5	Device6	Device7	Device8	Device9	Device 10	Device11	Device12
Joe	Schmoe	+447973772619	+4412155555555	+4412166666666								+4412155555555	+4412166666666

Therefore, in the following case:

```
firstName,lastName,Device1,Device2,Device3,Device4,Device5,Device6,Device7,Device8,Device9,
Device10,Device11,Device12
Joe,Schmoe,,,,,,,,,,+4412155555555,+4412166666666
```

You would use the following find/replace values to pull Device11 to Device1 (if Device11 exists and Device1 is empty).

Find	Replace
^([^,]*\,[^,]*\,)([^,]*)(\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*)$	(function(m,p1,p2,p3,p4,p5,p6){return p2===''&&p4!==''?p1+p4+p3+p5+p6:m})

After this operation, the result would be:

```
firstName,lastName,Device1,Device2,Device3,Device4,Device5,Device6,Device7,Device8,Device9,
Device10,Device11,Device12
Joe,Schmoe,+4412155555555,,,,,,,,,,+4412166666666
```

To pull Device12 to Device1 (if Device12 exists and Device1 is empty):

Find	Replace
^([^,]*\,[^,]*\,)([^,]*)(\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*)$	(function(m,p1,p2,p3,p4,p5,p6){return p2===''&&p6!==''?p1+p6+p3+p4+p5:m})

After this operation, the result would be:

```
firstName,lastName,Device1,Device2,Device3,Device4,Device5,Device6,Device7,Device8,Device9,
```

```
Device10,Device11,Device12
Joe,Schmoe,+4412166666666,,,,,,,,,+4412155555555
```

To continue:

- Pull Device11 to Device2 (if Device11 exists and Device2 is empty) - see first row below.
- Pull Device12 to Device2 (if Device12 exists and Device2 is empty) - see second row below.
- Pull Device11 to Device3 (if Device11 exists and Device is empty)
- Pull Device12 to Device3 (if Device12 exists and Device3 is empty)
- and so on...

Find	Replace
^([^,]*\,[^,]*\,[^,]*\,)([^,]*)(\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*)$	(function(m,p1,p2,p3,p4,p5,p6){return p2===''&&p4!==''?p1+p4+p3+p5+p6:m})($&
^([^,]*\,[^,]*\,[^,]*\,)([^,]*)(\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*\,[^,]*)$	(function(m,p1,p2,p3,p4,p5,p6){return p2===''&&p6!==''?p1+p6+p3+p4+p5:m})($&