Global Profiles

Customer 360 Data Manager

Salesforce, Summer ’20

@salesforcedocs

Last updated: July 9, 2020

CONTENTS

CONTENTS

Global Profile Basics

What data can I include in global profiles?

What sources can I include to build global profiles?
- Salesforce Org Data (Sales, Service, Platform...)
- Salesforce B2C Commerce Data
- Third-Party Systems (CSV)

How do I deal with configs?
- What does each key do?

How do I map every field?

What happens with the data?
- Initial Config Validation
- Input-Record Validation Rules
- Preparation and Validation rules
  - Individual
  - Email Address
  - Phone Number
  - Where is the phone country code?
  - Address
  - Party | External Identifier
- Job Summary
- Job Details Summary

How are records matched?
- Match Keys (Find Candidates)
  - Atomic Keys
  - Composite Keys
- Match Rules (Default)
  - Configure Match Rules
  - Party Identifier
- Fuzzy Name Matching
  - Feature Selection

Global Profile Basics

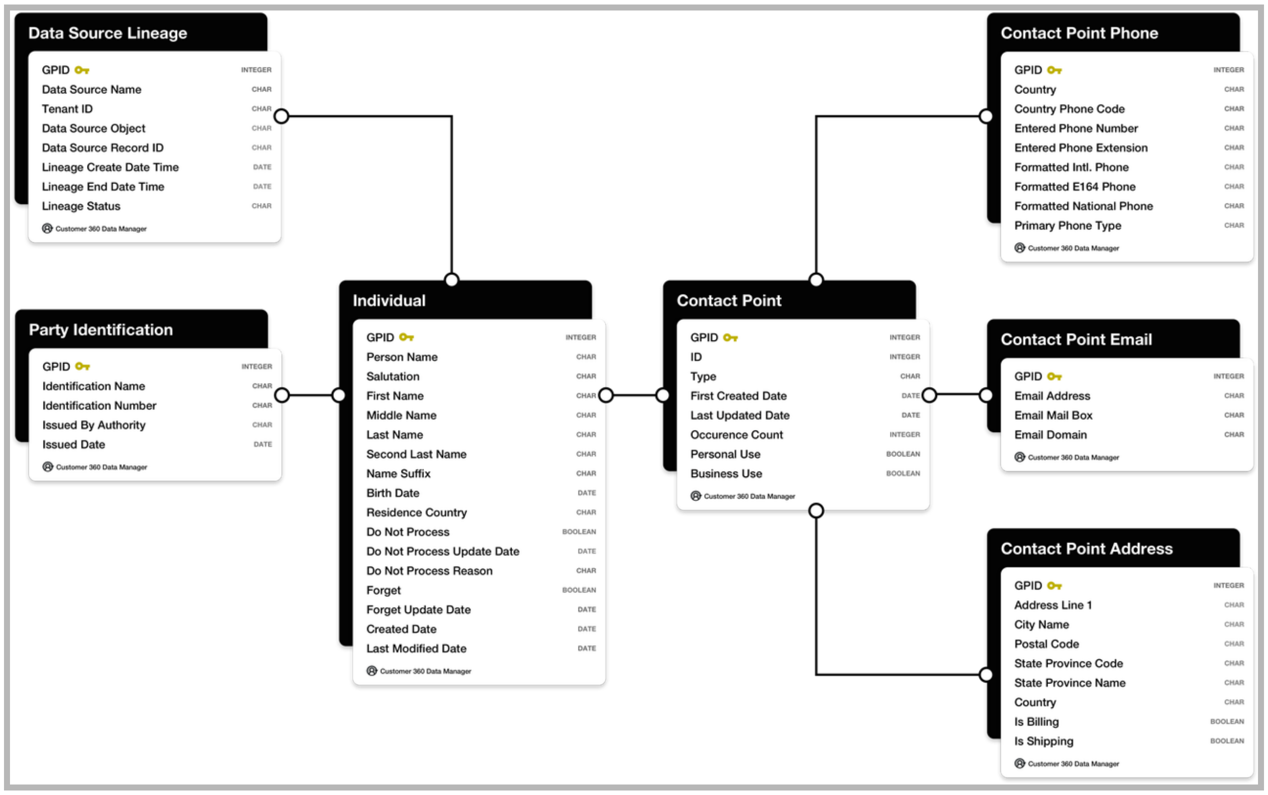
Customer 360 Data Manager creates and maintains a global profile for each customer with a unique, global party identifier.

This global profile enables a comprehensive understanding of an individual's contact points and unique identifiers. Plus, Customer 360 Data Manager addresses many use cases for getting the 360-degree view of the customer, such as a service agent's need to understand the latest order history in real time, or powering marketing and analytics implementations using the single ID.

Learn more in Customer 360 Data Manager Basics.

What data can I include in global profiles?

To build global profiles, you import data sources into Customer 360 Data Manager, such as Commerce Cloud or Service Cloud. The foundation of the global profile is based on a fixed schema.

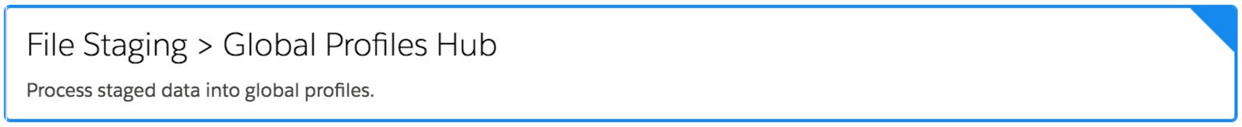


What sources can I include to build global profiles?

Salesforce Org Data (Sales, Service, Platform...)

Export Service Cloud data into a CSV file and import it into Customer 360 Data Manager. To help make the process easier, Customer 360 Data Manager includes a set of SOQL-queries and configuration files that facilitate the data extract for building global profiles. Here's how:

1. Use Workbench to export Service Cloud data in CSV format using:
    a. standard_solq_query-s
    b. standard_config-s
2. Build global profiles using DataJobs.



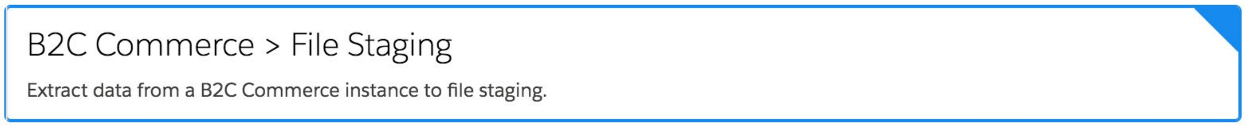
File Staging > Global Profiles Hub

Process staged data into global profiles.

Salesforce B2C Commerce Data

This file format and configuration is reserved for the automated extract from Commerce Cloud. The process produces the export files and deposits them to the staging environment. The files must pair with the standard <Order> or <Customer> file respectively. For all custom files, follow CSV requirements.

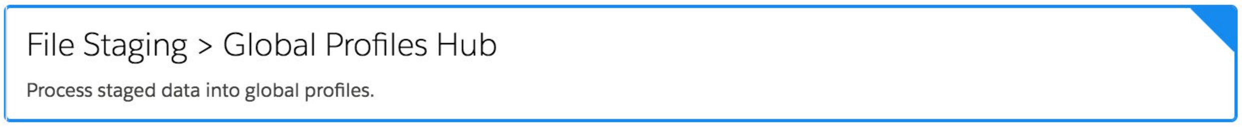
1. Data-Jobs to TXT (JSON).



B2C Commerce > File Staging

Extract data from a B2C Commerce instance to file staging.

2. DataJobs to build a profile.



File Staging > Global Profiles Hub

Process staged data into global profiles.

Third-Party Systems (CSV)

Customer 360 Data Manager can process any CSV-file that contains at least one valid Individual Contact Point and one valid other Contact Point. Ensure that the file is saved with these properties or results can vary.

- UTF-8 encoded

Note for Mac users: If you run file -I <inputFile> you can see the encoding of a particular file.

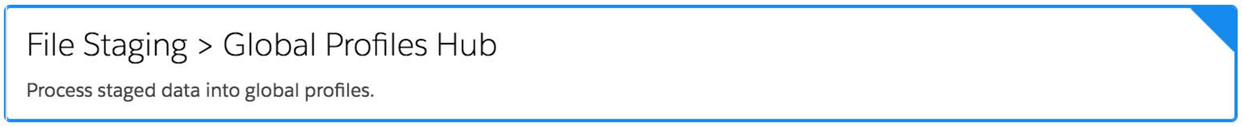
- Comma separated
- Double quotes enclosed

Follow these steps:

1. Export a CSV UTF-8 file from your data source.

Use an ETL tool to do a full data extract from your system.

2. Set up a config file for the data. For more information, review How do I deal with configs?
3. Run data jobs to build a profile.


File Staging > Global Profiles Hub

Process staged data into global profiles.

How do I deal with configs?

Review How do I map every field? for examples on how to map any GA-Schema fields to CRE-data-model for ingest.

What does each key do?

```
"version” : 1
```

Over time the config can evolve, so keeping a version number helps build backward compatibility and reduces the need to manage config files. If Salesforce introduces changes that are not backward compatible, a config validation error and a reason are provided. Latest product documentation will include examples to reflect the latest version syntax with detailed examples.

Note: The version is always an integer number. Use version 1 for the May ‘20 release.

`"phoneCountryCodes": ["US", "CA", "FR"]`

This configuration uses a hint to the phone-validation process. Review where is the phone country code? for more information.

Limits: List can contain no more than 10 codes.

`"auditDataFields"`

dateTimeFormat defaults

- create_ts defaults to now()
- update_ts defaults to create_ts

Important: Defaults for update_ts, e.g. lack of create_ts, will result in all profiles taking the reconciled attributes from the latest submitted files.

Example: dateTimeFormat examples

You can change the date-format in the config file to match your exported shape. Some examples and shortcuts:

- “yyyy-MM-dd'T'HH:mm:ss.SSS'Z” (2018-08-17T16:11:37.000Z) - official format that should be listed as supported. Default for exporting from your Salesforce org for Day0 scenarios via Data Loader
- “dd-M-yyyy hh:mm:ss” (02-1-2018 06:07:59)
- “dd-MM-yyyy hh:mm:ss” (02-01-2018 06:07:59)
- “MM-dd-yyyy” (01-02-2018)
- “dd MMMM yyyy” (02 January 2018)
- MM/dd/yyyy (01/02/2018)
- E, dd MMM yyyy HH:mm:ss z (Tue, 02 Jan 2018 18:07:59 IST)
- If you have ‘commas’ in the field, make sure the whole field is wrapped in double-quotes “Tue, 02 Jan 2018 18:07:59 IST”

```
"columnLimit": 36,
```

This key specifies the expected number of input attributes per row in the CSV file. Records that contain more attributes are discarded from further processing, but are counted for reporting. Review Input-Record Validation Rules for more information.

```
"commonFields": [{
 "locationInFile": "1",
 "entityFieldName": "ID",
 "crePartyContactPointFieldName": "recordExternalID"
}]
```

The ‘commonFields’ section is reserved for the record’s ID from the original source system. A Contact record in Service Cloud has an id:0030M000062rlFuGHB. The field is used to build lineage of the source records that comprise the final global profile. Data files must include a source system ID or you can’t submit the job.

```
"nonCommonFields"
```

This section describes each physical non-Id/time field on the datafile. It includes six mandatory components:

```
{
[a] "locationInFile": "3",
[b] "entityFieldName": "FIRSTNAME",
[c] "crePartyContactPoint": "INDIVIDUAL",
[d] "crePartyContactPointFieldName": "firstName",
[e] "crePartyContactPointSubType": "",
[f] "crePartyContactGroup": "individual1"
},
```

- [a] The position in the file. The first position is 1.
- [b] The label of field in the file.
- [c] Contact-Point that this field belongs to (Name, Phone, Email, Address).
- [d] Contact-Point-FieldName to the Customer 360 Data Managerspecific label of the field.
- [e] Contact-Point-SubType

    - Phone: MOBILE, HOME, BUSINESS, UNKNOWN, OTHER, FAX
    - Address: SHIPPING, BILLING, MAILING, <anything>
    - Individual, Email and Party contact-points do not have subTypes.
    - The field is required in the config file.
- [f] Party-Contact-Group - Specify this group consistently across a single contact point with more than one attribute. Using the 'Billing-Address-1' example, include this value across all attributes that comprise the Billing Address including Street, City, State/Province, Postal Code, and Country. Name and Address are the most prominent contact points where this repetition is required.

How do I map every field?

This config file example covers several challenges when mapping third-party systems.

- Mapping every field
    - All CIM-fields that Customer 360 Data Manager supports are listed as examples in this config file.
- Mapping multiple Address or Phone or Email fields
    - Review fields 15 to 30 for examples of multiple Billing.Addresses.
    - key field: crePartyContactGroup should be different between Contact Points
- Mapping an External Identifier
    - Review fields 31 and 32.

Example: Config File

```
Mapping {
"version": 1,
"phoneCountryCodes": ["US", "CA"],
"auditDataFields": [{
  "locationInFile": "32",
  "entityFieldName": "CREATEDDATE",
  "crePartyContactPoint": "",
  "crePartyContactPointFieldName": "firstCreatedDate",
  "dateTimeFormat": "yyyy-MM-dd'T'HH:mm:ss.SSS'Z'",
  "crePartyContactPointSubType": "",
 },
 {
  "locationInFile": "33",
  "entityFieldName": "LASTMODIFIEDDATE",
  "crePartyContactPoint": "",
  "crePartyContactPointFieldName": "lastUpdatedDate",
  "dateTimeFormat": "yyyy-MM-dd'T'HH:mm:ss.SSS'Z'",
  "crePartyContactPointSubType": "",
 }
],
"columnLimit": 36,
"commonFields": [{
 "locationInFile": "1",
 "entityFieldName": "ID",
 "crePartyContactPoint": "",
 "crePartyContactPointFieldName": "recordExternalID",
 "crePartyContactPointSubType": "",
}],

"nonCommonFields": [{
  "locationInFile": "2",
  "entityFieldName": "SALUTATION",
  "crePartyContactPoint": "INDIVIDUAL",
  "crePartyContactPointFieldName": "salutation",
  "crePartyContactPointSubType": "",
  "crePartyContactGroup": "individual1"
 },
 {
  "locationInFile": "3",
  "entityFieldName": "FIRSTNAME",
  "crePartyContactPoint": "INDIVIDUAL",
  "crePartyContactPointFieldName": "firstName",
  "crePartyContactPointSubType": "",
  "crePartyContactGroup": "individual1"
 },
 {
  "locationInFile": "4",
  "entityFieldName": "MIDDLENAME",
  "crePartyContactPoint": "INDIVIDUAL",
  "crePartyContactPointFieldName": "middleName",
  "crePartyContactPointSubType": "",
  "crePartyContactGroup": "individual1"
```

```
},
{
 "locationInFile": "5",
 "entityFieldName": "LASTNAME",
 "crePartyContactPoint": "INDIVIDUAL",
 "crePartyContactPointFieldName": "lastName",
 "crePartyContactPointSubType": "",
 "crePartyContactGroup": "individual1"
},
{
 "locationInFile": "6",
 "entityFieldName": "SECONDLASTNAME",
 "crePartyContactPoint": "INDIVIDUAL",
 "crePartyContactPointFieldName": "secondLastName",
 "crePartyContactPointSubType": "",
 "crePartyContactGroup": "individual1"
},
{
 "locationInFile": "7",
 "entityFieldName": "NAMESUFFIX",
 "crePartyContactPoint": "INDIVIDUAL",
 "crePartyContactPointFieldName": "nameSuffix",
 "crePartyContactPointSubType": "",
 "crePartyContactGroup": "individual1"
},
{
 "locationInFile": "8",
 "entityFieldName": "BIRTHDATE",
 "crePartyContactPoint": "INDIVIDUAL",
 "crePartyContactPointFieldName": "birthDate",
 "crePartyContactPointSubType": "",
 "dateTimeFormat": "dd-MM-yyyy",
 "crePartyContactGroup": "individual1"
},
{
 "locationInFile": "9",
 "entityFieldName": "RESIDENCECOUNTRYNAME",
 "crePartyContactPoint": "INDIVIDUAL",
 "crePartyContactPointFieldName": "residenceCountryName",
 "crePartyContactPointSubType": "",
 "crePartyContactGroup": "individual1"
},
{
 "locationInFile": "10",
 "entityFieldName": "EMAIL",
 "crePartyContactPoint": "EMAIL",
 "crePartyContactPointFieldName": "emailAddress",
 "crePartyContactPointSubType": "",
 "crePartyContactGroup": "email1"
},
{
 "locationInFile": "11",
  "entityFieldName": "phoneCountry",
  "crePartyContactPoint": "PHONE",
  "crePartyContactPointFieldName": "countryName",
  "crePartyContactPointSubType": "MOBILE",
  "crePartyContactGroup": "phone1"
},
{
 "locationInFile": "12",
 "entityFieldName": "MOBILE1PHONE",
 "crePartyContactPoint": "PHONE",
 "crePartyContactPointFieldName": "telephoneNumber",
 "crePartyContactPointSubType": "MOBILE",
 "crePartyContactGroup": "phone1"
},
{
 "locationInFile": "13",
 "entityFieldName": "MOBILE2PHONE",
 "crePartyContactPoint": "PHONE",
 "crePartyContactPointFieldName": "telephoneNumber",
 "crePartyContactPointSubType": "MOBILE",
 "crePartyContactGroup": "phone2"
},
{
 "locationInFile": "14",
 "entityFieldName": "HOME1PHONE",
 "crePartyContactPoint": "PHONE",
 "crePartyContactPointFieldName": "telephoneNumber",
 "crePartyContactPointSubType": "HOME",
 "crePartyContactGroup": "phone3"
},
{
 "locationInFile": "15",
 "entityFieldName": "BILLING1STREET1",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "address1",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing1"
},
{
 "locationInFile": "16",
 "entityFieldName": "BILLING1STREET2",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "address2",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing1"
},
{
 "locationInFile": "17",
 "entityFieldName": "BILLING1STREET3",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "address3",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing1"
},
{
 "locationInFile": "18",
 "entityFieldName": "BILLING1STREET4",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "address4",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing1"
},
{
 "locationInFile": "19",
 "entityFieldName": "BILLING1CITY",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "city",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing1"
},
{
 "locationInFile": "20",
 "entityFieldName": "BILLING1POSTALCODE",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "postalCode",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing1"
```

```
},
{
 "locationInFile": "21",
 "entityFieldName": "BILLING1STATE",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "stateProvinceCode",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing1"
},
{
 "locationInFile": "22",
 "entityFieldName": "BILLING1COUNTRY",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "countryName",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing1"
},
{
 "locationInFile": "23",
 "entityFieldName": "BILLING2STREET1",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "address1",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing2"
},
{
 "locationInFile": "24",
 "entityFieldName": "BILLING2STREET2",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "address2",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing2"
},
{
 "locationInFile": "25",
 "entityFieldName": "BILLING2STREET3",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "address3",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing2"
},
{
 "locationInFile": "26",
 "entityFieldName": "BILLING2STREET4",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "address4",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing2"
},
{
 "locationInFile": "27",
 "entityFieldName": "BILLING2CITY",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "city",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing2"
},
{
 "locationInFile": "28",
 "entityFieldName": "BILLING2POSTALCODE",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "postalCode",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing2"
},
{
 "locationInFile": "29",
 "entityFieldName": "BILLING2STATE",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "stateProvinceCode",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing2"
},
{
 "locationInFile": "30",
 "entityFieldName": "BILLING2COUNTRY",
 "crePartyContactPoint": "ADDRESS",
 "crePartyContactPointFieldName": "countryName",
 "crePartyContactPointSubType": "BILLING",
 "crePartyContactGroup": "billing2"
},
{
 "locationInFile": "31",
 "entityFieldName": "LinkedIn_URL__c",
 "crePartyContactPoint": "PARTYID",
 "crePartyContactPointFieldName": "identificationNumber",
 "crePartyContactPointPartyIdentificationName": "LinkedIn URL",
 "crePartyContactPointPartyIdentificationAuthority": "LinkedIn",
 "crePartyContactGroup": "party1"
},
{
 "locationInFile": "32",
 "entityFieldName": "LinkedInId.ISSUEDATE",
 "crePartyContactPoint": "PARTYID",
 "crePartyContactPointFieldName": "issuedDate",
 "crePartyContactPointPartyIdentificationName": "LinkedIn URL",
 "crePartyContactPointPartyIdentificationAuthority": "LinkedIn",
 "dateTimeFormat": "yyyy-MM-dd'T'HH:mm:ss.SSS'Z'",
 "crePartyContactGroup": "party1"
}

]
}
```

What happens with the data?

Note: Review Data Preparation Rules for more information.

Initial Config Validation

- You need to map at least one valid Contact Point.
- Create at least one valid Match Rule.

If any of the config validation rules is violated the job stops processing and an error message is posted in the Logs section with details on how to fix the config file or what the detected error is.

Input-Record Validation Rules

`“Total Records Dropped due to Parse Errors non UTF-8 chars detected: 0”`

Records that contain non-UTF-8 characters are discarded. Such records/charsets often cause quality issues when processing data and are dropped from making it to the clean profiles.

`“Total Records Dropped due to Timestamp parsing errors: 0”`

The full input record is dropped if any one of the timestamps inside the row/records (create-ts, update-ts, birthday) doesn’t match the expected config format.

Example: Review examples in How do I deal with configs?

```
“Total Records Dropped exceeding column limit: 0”
```

The full input record is dropped if the input record contains more fields than expected in the mapping config. Review details with key-property in “columnLimit”:36.

```
“Total Records Dropped due to other exceptions during processing: 0”
```

This rare occurrence is associated with unknown errors that can occur during the file-reading process.

```
“Input record structure length is not supported: 0”
```

The input record is discarded if it contains more than 13 contact points. For example, more than four phones + four emails + four addresses + one Name + one party causes an input error.

Preparation and Validation rules

To form a profile, include an individual contact point and at least one other type of contact point.

- If individual is the only valid contact point, the input record is discarded.
- If an individual contact point is invalid, the input record is discarded.

Individual

The individual contact point validation aims to answer the question “Is this a valid person name?” given the first, middle and last name in the mapping config.

- The condition for a valid name is at least one letter character in major alphabets.
- Names that contain only invalid characters are discarded.
- An empty name field is considered invalid. No detailed error is logged for Individual Contact Point.

Email Address

Customer 360 Data Manager removes the leading or trailing white space from email addresses. It tokenizes the mailbox name, before the @ symbol, and domain name. The email is validated for format and any error messages are posted in the validation library.

Input	**Validation**	**Error Message**
A@b@c@domain.com	X	Not a valid email address - Domain contains illegal character

example.com	X	Not a valid email address - Missing final '@domain'
abc is”not\\valid@domain.com	X	Not a valid email address - Local address contains control or whitespace

Phone Number

Customer 360 Data Manager removes the leading or trailing white space, special characters, and non-alphanumeric-digits from phone numbers. It tokenizes country code, phone number, and extension. It also normalizes for international, national, and E. 164 phone number formats.

- To validate phone-contact points, we use **libphonenumber**, which includes multiple benefits and few restrictions. (Review here.)
- To validate phone, the number should begin with + followed by the country code.
- Provide country code with the validation input.

Where is the phone country code?

- The country code is in the Phone field. For example, +1 415 537 1420 or +001 415 537 1420.
- The phone.CountryCode is mapped from the mapping.config.
    - Review "locationInFile": "11"
- The CountryCode list is gathered from the input address.CountryCodes.
- You can provide an input parameter in your config for use in phoneValidation (up to 10 CC-codes).
    - See Key’s section: Where is the phone country code?

Input				**Error Message**	**Normalized Phone**
Phone	phone.CC	address.CC	config.CC	audit_log_details	e.164
+1 (847) 277-9500			["US", "UK", "TR"]		+18472779500
123456789			["US", "UK", "TR"]	Too Short	

+ 312-300-2567			["US", "UK", "TR"]	Not a valid Number	
+ 33 4 74 46 25 56			["US", "UK", "TR"]		+33474462556
+ 41 779816745			["US", "UK", "TR"]		+41779816745
+ 51 (1) 715-5915 Ext.139		["US", "UK", "TR"]	["US", "UK", "TR"]		+5117155915
765-637-1508			["US", "UK", "TR"]	Invalid country calling code	
+765-637-1508			["US", "UK", "TR"]	Too Short	
+1 (765) 637 1508			["US", "UK", "TR"]		+17656371508
17656371508			["US", "UK", "TR"]	Invalid country calling code	
312 427 13 14			["US", "UK", "TR"]	Invalid country calling code	
00312 427 13 14			["US", "UK", "TR"]	Invalid country calling code	
+312 427 13 14			["US", "UK", "TR"]	Not a valid Number	
0312 427 13 14			["US", "UK", "TR"]	Invalid country calling code	

+1312 427 13 14			["US", "UK", "TR"]		+13124271314
+90312 427 13 14			["US", "UK", "TR"]		+903124271314
+90312 427 13 14			["US", "UK", "TR"]		+903124271314
+90312 427 13 14			["US", "UK", "TR"]		+903124271314
(201) 930-0222 * off website			["US", "UK", "TR"]	Invalid country calling code	
+1 (201) 930-0222 * off website			["US", "UK", "TR"]	The string supplied is too long to be a phone number	
(202) 719-5624 #zoom			["US", "UK", "TR"]	Invalid country calling code	
+1.917.753.7636 wrong			["US", "UK", "TR"]	Too Long	

Address

Address tokens are parsed by Salesforce's internal library and normalized to standard formatting. They are arranged based on country-specific rules.

Input Fields					**Normalized**
Address	City	State	Zip	Country	Address
220 Laurier Avenue West Suite 1000	Ottowa	ON	K1P 5Z9	CA	220 Laurier Ave W Ste 1000

Suite 1402 505 Railway Street	Cochrane	AB	T4C 2K9	CA	505 Railway St Ste 1402
Unit 101 2469 Pauline Street	Abbotsford	BC	V2S 3S1	CA	2469 Pauline St Unit 101
Suite 3033 Inco Innovation Centre	St. John's	NL	A1C 5V5	CA	Inco Innovation Centre Ste 3033
251 East Imperial Highway Suite 441	Fullerton	CA	92835	US	251 E Imperial Hwy Ste 441
34098 pacific coast highway suite B	Dana Point	ca	92629	US	34098 Pacific Coast Hwy Ste B
535 Madison Avenue 14 Floor	New York	NY	10022	US	535 Madison Ave Fl 14
8668 West Spring mountain road Suite 100	Las Vegas	NV	89117	US	8668 W Spring Mountain Rd Ste 100
Level 12 - Chifley Tower 2 Chifley Square	Sydney	NSW	2000	AUSTRALIA	Level 12 Chifley Tower 2 Chifley Sq
7925 west sahara ave suite 103	las vegas	NV	89117	US	7925 W Sahara Ave Ste 103
11726 San Vicente Blvd. Suite 280	Los Angeles	CA	90049	us	11726 San Vicente Blvd Ste 280
po Box 560042	Orlando	FL	32856	US	Po Box 560042
15123 southwest Millikan way apt 216	Beaverton	OR	97006	US	15123 SW Millikan Way Apt 21
Schulstraße 4	Düsseldorf	North Rhine-Westphalia	40213	DE	Schulstraße 4
52 Filderstraße	Leinfelden-Echterdingen	Baden-Württemberg	70771	DE	Filderstraße 52

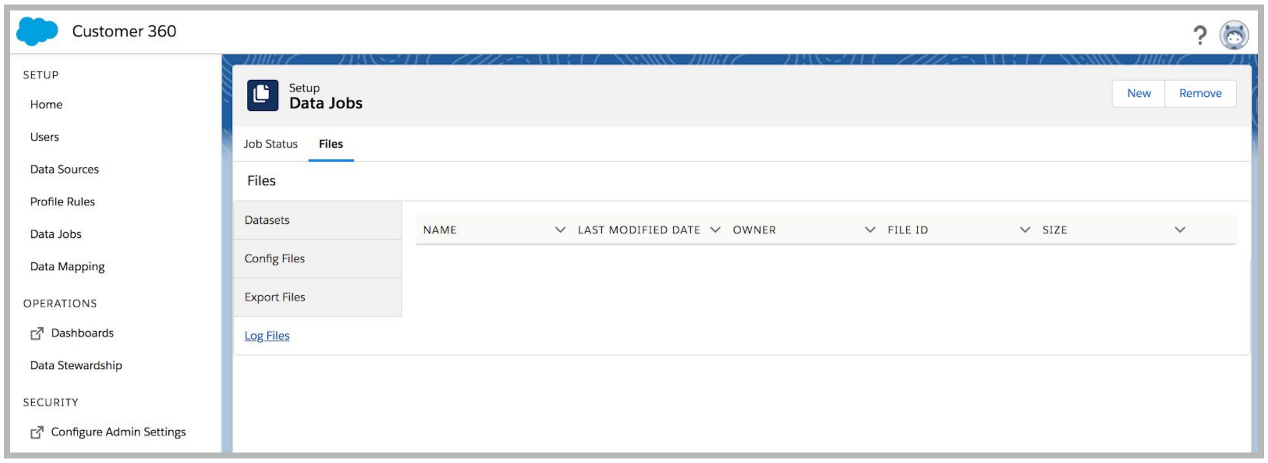
267 boulevard Pereire	PARIS	Île-de-Fran ce	75017	FR	267 Blvd Pereire
Calle de José Abascal 56	Madrid	Madrid	28003	ES	Calle De José Abascal 56
525 Rua Manoel Valim	Franca	SP	14401	BR	Rua Manoel Valim 525

Party | External Identifier

The Party | External Identifier requires both a Number and Identifier. Review Party Identifier to learn how to create a match rule with party identifier.

Job Summary

A job execution summary file is created after every successful job and added to the Log Files section.



Example: This is an example execution summary. The name is the Job ID plus audit_log_summary.log.

```
6888e8b9-0ec3-3e5d-a47b-210c34575851_audit_log_summary.log

Batch Process Completed! TransactionId: 0hjOUv

--- Row Summary ---
Total Records Read: 3
Total Records Ingested: 2
Total Records Dropped: 0
Total Records Dropped due to Parse Errors non UTF-8 chars
detected: 0
Total Records Dropped due to Timestamp parsing errors: 0
Total Records Dropped exceeding column limit: 0
Total Records Dropped due to other exceptions during processing:
0
Input record structure length is not supported: 0

--- Validation Errors ---
Total Fields Truncated: 0
Individual: 0
Email: 0
Phone: 1
Address: 0
PartyId: 0

--- Profile Summary ---
New Master Profiles created: 2
Master Profiles updated: 0
Master Profiles merged: 0
Master Profiles deleted: 0

--- Execution Time ---
Total time taken(min:sec): (1:22)

--- Hub Summary ---
Total Master Profiles created and updated in current job: 2
```

Job Details Summary

Errors associated with Preparation and Validation rules are posted in the Job Details Summary file. You can use this file to carry out targeted data quality and clean-up campaigns in the source systems. You can also develop data stewardship extensions tailored to your needs . The file is posted in Logs within Customer 360 Data Manager Data Jobs.

How are records matched?

Example: Review these matching examples.

firstname	middlename	lastname	email	phone	address1	city	zip	state	country	partyid
Sam	J	Smith	samantha.smith@gmail.com	763 658 5241	699 coughlin	Des Plaines	19632	DE	US	41901
Samantha		Smith	samantha.smith@gmail.com	(163)- 213-5219	461 romer Apartment 2	Oxted	34578	KY	US	
Sam	Johnson	Smith	sam.hobby@yahoo.com	763 658 5241						
Samantha	Johnson	Smith	bills.sam.smith@comcast.com	(600)- 812 1234	461 romer Apartment 2	Oxted	34578	KY	US	
Sam	Johnson	Smith-Willson	samantha.smith@yahoo.com	(800)- 812 1234	50 Freemont	San Francisco	94105	CA	USA	41901

- Records (1) and (2) match because they share the same “Fuzzy.First” and “Exact.Last” and “Exact.Email”
- Records (1) and (3) match because they share the same “Fuzzy.First” and “Exact.Last” and “Exact.Address”(fields)
- Records (1) and (4) match because they share the same “Fuzzy.First” and “Exact.Last” and “Exact.Phone”
- Records (1) and (5) match because they share the same “Fuzzy.First” and “Exact.Last” and “Identification Number”

Match Keys (Find Candidates)

When you add a record to Customer 360 Data Manager, it is compared with all existing records, which can number into the millions. Suitably designed match keys drastically reduce the number of existing records to compare with the new record, from millions to hundreds or less.

For example, if the new record has the name Samantha Smith, a match key based on person name only finds records with Samantha Smith for comparison. However, it does need to accommodate variants of Samantha such as S (initials), Sam, and so on.

With this in mind, match keys are carefully designed to narrow the search significantly, without missing potential matches.

To match person records, Customer 360 Data Manager uses atomic keys and composite keys. Atomic match keys are built with composite ones. However, only the composite keys are intended for final use. Atomic keys are available for reuse or in custom settings.

Atomic Keys

flast - This is composed of the first character of the first name (after normalization) followed by the last name (downcased and white spaces, hyphens, and special characters stripped).

Examples of flast:

First Name	**Last Name**	**flast**	**Explanation**
Bob	Smith	rsmith	Bob → Robert
Jane	O'reilly	joreilly	Strip '
sara	Smith-Willson	ssmithwillson	Strip -

phone - First n digits of phone after international code is stripped. (n is configurable. n = 6 is a common choice.)

address - Key from street address. Assumes the address is parsed into its components. The key is formed by concatenating suitable components, one per rule below.

1. <street number> < street name>.

postal(n) - First n characters of postal code or entire postal code if it's less than n after stripping away white space. (n is configurable. n=5 is a common choice.)

city - Normalized city name.

Composite Keys

Composite keys are based on a combination of one or many of the atomic keys. This record decryption allows for fast retrieval of potentially matching records that share one or many of the following keys:

flast@phone(n) - first initial, last name, and phone number

email - exact email address

flast@address - first initial, last name, and address-line-1

flast@postal(n) - first initial, last name, and postal code

flast@city - first initial, last name, and city

partyId - used to retrieve records for Party Identifier

Match Rules (Default)

Configure Match Rules

1. Fuzzy.FirstName + Exact.LastName + Exact.Email
2. Fuzzy.FirstName + Exact.LastName + Exact.Phone
3. Fuzzy.FirstName + Exact.LastName + Exact.Address
4. Fuzzy.FirstName + Exact.Indentification.Number + Exact.Identification.Name

Note: First name match is fuzzy. Last name, email and phone match is exact. Full address match is exact (on the combination of all components: street, city, postal, state, country). City-state-country match is exact (all components must match exactly)

Party Identifier

You can set up a rule using “Identification Number” and “Identification Name” to map various third-party system information. A common example, provided in How do I map every field? is LinkedIn-id. You can add multiple contact points from many systems when building the global profile in Customer 360 Data Manager.

"crePartyContactPointPartyIdentificationName": "LinkedIn URL",

"crePartyContactPointPartyIdentificationAuthority": "LinkedIn",

Fuzzy Name Matching

One common challenge in matching names is reviewing nicknames and female or male versions of the same name. Often there are fewer than one or two differing letters. With fuzzy name matching in Customer 360 Data Manager, we err on the conservative side and match less often rather than overly matching. We prefer less false positives over higher recall.

Feature Selection

We performed a collaborative and iterative feature selection process and a trained model is included every org. These selected features optimize the desired matching results:

1. metaphone + length
2. character position difference
3. longest common string with prefix, suffix
4. jaro winkler distance
5. normalized name pair

Features that were evaluated during the developer of Customer 360 Data Manager:

1. Phonetic Pattern
    1. Soundex: Smith => S530, Schmidt => S530
    2. Metaphone: Smith => SM0, Schmidt => SXMTT
2. Test Similarity Pattern
    1. Edit distance
    2. Jaro Winkler
3. Longest Common String (LCS) Pattern
    1. LCS Prefix Suffix: eric, erica => +,+a
    2. LCS Prefix Suffix with length: eric, erica => 4, 4a
    3. Missing Space & Hyphens
    4. Missing Components
    5. Spelling Difference
    6. Out of Order Components
    7. NickNames
    8. Truncated Components
    9. Initials

Person.Name to Individual

Name data from different sources comes in many formats, so Customer 360 Data Manager supports names stored as a single field. Although Salesforce stores name data as two separate fields, firstName and lastName, we know that many other systems you work with use only one field. Support for the personName field supports a wider range of data you can bring into Customer 360 Data Manager without completing extra manual tasks first. This feature is available with the May ’20 release.

Mapping Example

Review how do I map every field? to learn more.

Each `crePartyContactPoint` must

- Be `INDIVIDUAL`,
- Have only one `INDIVIDUAL` contact point per input job/file,
- And each INDIVIDUAL contact point must have either
  - PersonName , or
  - FirstName and LastName
- And each INDIVIDUAL contact point must have either
  - `PersonName` , or
  - `FirstName` and `LastName`.

Example

```
{
"locationInFile": 4,
"entityFieldName": "Person Full Name",
"crePartyContactPoint": "INDIVIDUAL",
"crePartyContactPointFieldName": "PersonName",
"crePartyContactPointSubType": "",
"crePartyContactGroup": "individualParty_1"
}
```

Source System A's Data	**Individual Match?**	**Source System B's Data**					**Match Rule**
PersonName		Salutation	FirstName	MiddleName	LastName	Suffix	
Roberto Martinez	**Yes**		Roberto	Mills	Martinez		FirstName + LastName
Peter Neumann	**Yes**		Peter	Jones	Neumann		FirstName + LastName

Ms. Christina Salas	**Yes**	Ms	Christina		Salas		FirstName + LastName
Herr Alexander Malone	**Yes**		Alexander		Malone		FirstName + LastName
Herr Alexander Malone	**Yes**	Herr	Alexander		Malone		FirstName + LastName
Herr Alexander Malone	**No**		Herr Alexander		Malone		FirstName + LastName
Adam Claudia Tabachnikoff	**Yes**		Adam		Tabachinkoff		FirstName + LastName
Adam Claudia Tabachnikoff	**Yes**		Adam	Claudia	Tabachinkoff		FirstName + LastName
Tiffany Embry	**No**			Tiffany	Embry		FirstName + LastName
Tiffany Embry	**No**		Embry		Tiffany		FirstName + LastName

Known Limitations

- PersonName OR FirstName and LastName must be mapped as part of input data

Scrubbing List

Use the scrubbing list to remove contact points with bad field-level source data before they get into your global profiles. By removing contacts that don't meet your data standards, your global profiles are more accurate and you have less manual cleanup later. This feature is in beta, starting with the May '20 release.

How do I remove Contact Points with bad data from my Global Profiles?

Example

```
Mapping {
...
```

```
}

Preparation
{
"version": 1,
"individual": {
"scrubbing": [
{"firstName": "First", "lastName": "Last"},
{"firstName": "Jane", "lastName": "Doe"},
{"firstName": "Nobody", "lastName": "Empty"}
]
},
"email": {
"scrubbing": [
{"emailAddress": "no@no.com"},
{"emailAddress": "no@gmai.com"},
{"emailAddress": "dummy@dummy.dummy"
{"domain": "dummy.dummy"}
]
},
"phone": {
"scrubbing": [
{"countryCode": "1", "telephoneNumber": "6047211212"},
{"e164PhoneNumber": "+16047831212"}
]
}
}
```

Understanding each key

```
"version": 1
```

Version allows the schema for the Preparation section to be changed over time.

○

Example

```
"individual": {
"scrubbing": [
{"firstName": "v", "lastName": "DOE"},
{"firstName": "foo", "lastName": "bar"},
{"firstName": "Nobody", "lastName": "Cocoa"}
]
},
```

A scrubbing list entry specifies one or more fields and associated values. These values are normalized in the same manner that fields in the contact point are normalized. In this example, firstName of "v" would be normalized to "V.".

If all of the normalized values in a scrubbing list entry match the associated normalized values in the contact point, then the contact point is removed from the global profile. Values are case sensitive. You can learn more about which contact points were removed by the scrubbing list in the Audit Log Details, ending in file extension __audit_log_details.csv.

If an input record contains personName instead of individual name fields, the record is decomposed into the individual fields (`salutation, firstName, middleName, lastName`, etc.) before being compared to the values in the scrubbing list.

Example

```
   "email": {
"scrubbing": [
{"emailAddress": "foo@bar.com"},
{"domain": "example.com"}
]
},
```

In this example, the scrubbing list removes contact points where `emailAddress` is "foo@bar.com"or domain is "example.com". Scrubbing list entries are always case sensitive.

Detailed Examples

This section provides examples for all of the fields that can be included in the scrubbing list to remove contact points from global profiles. For a definitive schema specification of the scrubbing list section, see more here.

Individual

Individual Fields with Scrubbing List Support

```
salutation
firstName
middleName
lastName
secondLastName
nameSuffix
```

Example

```
"individual": {
"scrubbing": [
{"firstName": "john", "middleName": "q.", "lastName": "doe"},
{"salutation": "Mr.", "lastName": "Tee"},
{"secondLastName": "IGNOREME"},
{"lastName": "nobody", "nameSuffix": "jr."}
]
}
```

Party ID

Party ID Fields with Scrubbing List Support

```
identifierName
identificationNumber
issuedByAuthority
```

Example

```
"partyId": {
"scrubbing": [
{"identifierName": "membership", "identificationNumber": "00000"},
{"issuedByAuthority": "NO_AUTHORITY"}
]
}
```

Phone

Phone Fields with Scrubbing List Support

```
countryName
areaCode
telephoneNumber
extension
internationalPhoneNumber
e164PhoneNumber
nationalPhoneNumber
phoneType
```

Example

```
"phone": {
"scrubbing": [
{"countryName": "Tuvalu"},
{"e164PhoneNumber": "+16047831234"},
{"telephoneNumber": "6047831111", "extension": "555"},
{"phoneType": "NONE"}
]
}
```

Email

Email Fields with Scrubbing List Support

```
emailAddress
mailbox
domain
```

Example

```
"email": {
"scrubbing": [
{"emailAddress": "foo@bar.com"},
{"domain": "example.com"},
{"mailbox": "postmaster"},
{"mailbox": "webmaster"},
{"mailbox": "na"}
]
}
```

Address

Address Fields with Scrubbing List Support

```
address1
address2
address3
address4
city
postalCode
stateProvinceCode
stateProvinceName
countryName
addressType
```

Example

```
"address": {
"scrubbing": [
{"address1": "1234 Main St.", "city": "Baltimore"},
{"city": "Mainville"},
{"city": "Nowhere"},
{"postalcode": "H0H 0H0"},
{"countryName": "Tuvalu"},
{"addressType": "none"}
]
}
```

Known Limitation

Each implementation of Customer 360 Data Manager can have 5,000 scrubbing list entries. Each line counts as one entry. This includes 6 scrubbing list entries. For example, the scrubbing list address example includes 6 scrubbing list entries.