



UNOFFICIAL SALESFORCE COMMUNITY CLOUD MIGRATION GUIDE

Table of Contents

1) Community Setup	4
1.1) Community settings checklist	4
1.2) Chatter settings checklist	4
2) Import Of Data	5
2.1) Import of contacts	5
2.2) Import of users	6
2.3) Import of User pics	7
2.4) Import of FeedItem records	8
2.4.1) Feed Items with images	9
2.4.1.1) DataLoader images import	10
2.4.2) Feed Items with @mentions	11
2.4.2.1) REST API	11
2.4.2.2) ConnectAPI	14
2.4.3) Internal Links Consideration	15
2.5) Feed Comments	15
2.5.1 Best Answer	20
2.6) Topics	21
2.6.1) Knowledgeable People	21
2.7) Likes and Votes	22
2.8) Follows	22
2.9) Reputation Points	22
2.10) Chatter Groups	25
3) Examples	26
3.1) Question import	26
3.1.1) The html source of the question	26
3.1.2) Processing question body	27
3.2) Importing question via DataLoader	28
3.2.1) Topic assigning	31
3.3) Comments importing	32

About This Guide

The main objective of this guide is to provide a practical and comprehensive resource for executives, developers and community admins describing migration and import of data process from any platform to Salesforce Community Cloud.

To keep things simple we use the standard Salesforce Data Loader tool [Setup → Data Management → Data Loader]. We assume that you already have some experience with DataLoader.

1) Community Setup

1.1) Community settings checklist

We recommend the following steps for setting up your community:

- Create a community based on one of the standard templates. Currently the most popular one is “Customer Service” Template
- Create community user profile(s) by cloning one of the standard profiles as standard profiles cannot be edited
- Define members (select appropriate profile/permission set as member profile in Community -> Administration -> Members)
- Community -> Administration -> Preferences:
 - Enable setup and display of reputation levels (This is to see how influential members are. [More details](#))
 - Enable knowledgeable people on topics (Knowledgeable people are displayed in key areas, such as topic detail pages. [More details](#))
 - Allow discussion threads (Threaded discussions add the ‘Reply’ feature to feeds. In addition to commenting on a post or poll or answering a question, you can reply to an answer or a comment. Threaded discussions are indented to be up to three levels deep.)
 - Allow members to upvote and downvote (Optional, the default setting is “Like”)

1.2) Chatter settings checklist

In order to have a fully functional Chatter which provides Social network/forum functionality, make sure that the following Chatter settings are ticked:

- Enabled
- Allow users to edit posts and comments
- Allow users to compose rich text posts

The user who is performing import should have “Insert System Field Values for Chatter Feeds” user permission to be able to update at least the CreateByID and CreatedDate fields.

2) Import Of Data

When migrating data to Salesforce the first thing is to load user records because most Salesforce objects (such as questions, comments, posts) require an owner.

2.1) Import of contacts

Salesforce Community users are created from contact records. So for each community user there should be a contact and each contact should have an account (company) record. If your community users don't have any company data, the most obvious way to deal with this is to create a dummy account record.

However, it's not recommended to have more than 10 000 contacts belonging to the same account so you may need to create a number of dummy accounts – for example, 'Dummy Account 1', 'Dummy Account 2'.

This is because there may be a possible 'Account data skew' problem for large numbers of contacts (more than 10 000).

You can find more information on Account and Lookup data skew issues here

<https://developer.salesforce.com/blogs/engineering/2012/04/avoid-account-data-skew-f-or-peak-performance.html>

<https://developer.salesforce.com/blogs/engineering/2013/04/managing-lookup-skew-to-avoid-record-lock-exceptions.html>

For the sake of simplicity in this particular example we will use only one account. Let's assume that we have created an account with the name "Forum Users" and <Account-Id>.

The standard Salesforce profiles are not editable so normally you can clone one of the standard profiles such as "Customer Community User" profile and name it, for example, "Forum User". If your community would be used by developers, you should also tick "Allow Inclusion of Code Snippets from UI" on system permission to enable code formatting in chatter posts.

When importing contacts the bare minimum fields you should have are FirstName, LastName, email plus an externalId custom field with a predefined account id column.

To save time on importing users you can avoid a merging step by adding temporary additional data from your legacy system into Contact object I have added a few temporary custom fields: legacyUsername, legacyIsActive, legacyReputation.

My example of fields map for DataLoader:

#Mapping values (left legacy system - right salesforce fields)

```
c_realname=LastName  
c_name=legacy_Username__c  
c_email=Email  
c_id=legacy_id__c  
c_active=legacy_is_active_user__c  
accountid=<Account-Id>
```

It can of course be extended to have as many fields as needed.

2.2) Import of users

Please note that when importing Portal Users, a welcome email is automatically sent which contains a link prompting the user to set a password (no password reset required) if the community is already active. If the community is not active, all the members get the email when the community is activated. If needed, this email can be cancelled by going into Community Administration → Emails → and unticking “Send welcome email”.

Also note that if you are bringing the users from the external system it's better to create a custom temporary field “legacy_Id” - as the external id for the User object which would contain the id of the record from your old system. This would make import of the data easier.

Here is the list of csv columns as an example for user import with some predefined values:

```
CONTACT_ID  
LASTNAME  
EMAIL  
EMAIL2 = <DUPLICATE-OF-EMAIL>  
TIMEZONESIDKEY = GMT  
LOCALESIDKEY = en_GB  
EMAILENCODINGKEY = ISO-8859-1  
LANGUAGELOCALEKEY = en_US  
PROFILEID = <Profile “ForumUser”>  
LEGACY_USERNAME__C  
LEGACY_USERNAME2__C = <DUPLICATE-OF-USERNAME>  
LEGACY_ID__C  
LEGACY_IS_ACTIVE_USER__C
```

#Mapping values for DataLoader (save this with *.sdl extension)

```
LEGACY_USERNAME__C=CommunityNickname  
LEGACY_IS_ACTIVE_USER__C=IsActive
```

```
LEGACY_USERNAME2=Alias
profileid=ProfileId
LANGUAGELOCALEKEY=LanguageLocaleKey
EMAIL=Email
TIMEZONESIDKEY=TimeZoneSidKey
EMAIL2=Username
ID=ContactId
LOCALESIDKEY=LocaleSidKey
LASTNAME=LastName
EMAILENCODINGKEY=EmailEncodingKey
LEGACY_ID__C=LEGACY_ID__C
#Contact and User should have same external ID value (LEGACY_ID__C)
```

2.3) Import of User pics

Importing user profiles images is not straight forward. Out of the box the Data Loader doesn't support image import, but you can store profile images in custom field and process with Apex batch job.

Let's assume one of the users has the following user picture:
<https://i.imgur.com/xYEWNpw.jpg> somewhere on their external legacy platform.

In order to get that image on our system we need to:

First, whitelist the website "https://i.imgur.com" by adding the record into "Setup -> Remote site setting" to allow Apex to make callouts.

Second, grab the image and store it as a content version:

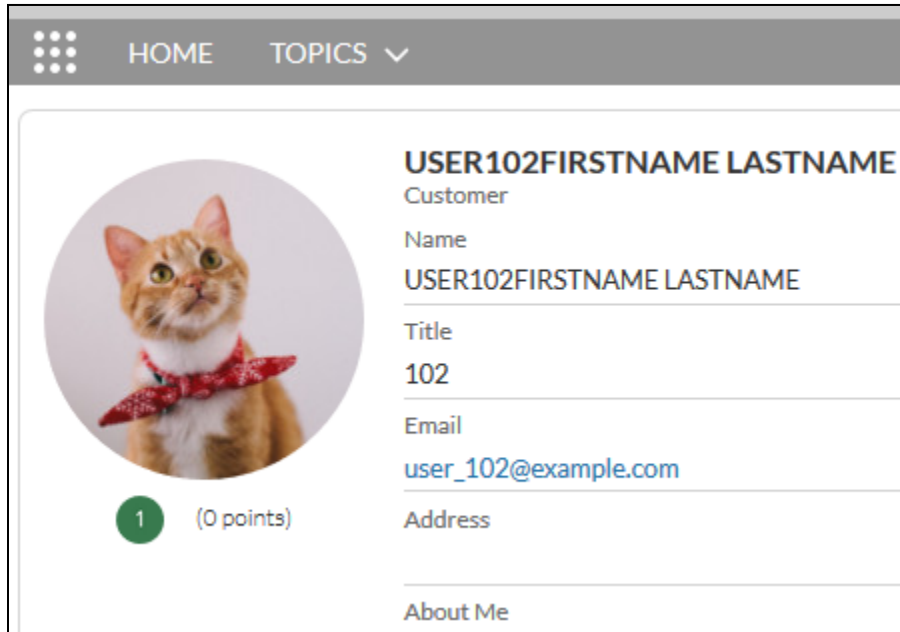
```
String imageURL = 'https://i.imgur.com/xYEWNpw.jpg';
PageReference pageRef = new PageReference(imageURL);
ContentVersion cv = new ContentVersion();
cv.VersionData = pageRef.getContent();
cv.Title = 'Image Name';
cv.PathOnClient = imageURL;
cv.Origin = 'H';//Chatter
insert cv;
```

```
String contentDocId = [SELECT ContentDocumentId FROM ContentVersion
WHERE Id = :cv.Id LIMIT 1].ContentDocumentId;
```


Third, set it as a user photo:

```
ConnectApi.UserProfiles.setPhoto('<communityId>', '<userId>', 'contentDocId',  
null);
```

(Null parameter means latest version of file).



2.4) Import of FeedItem records

FeedItem object is the beating heart of Salesforce chatter. Feed Item stores chatter posts, questions, polls and posted files.

Please be aware that for security reasons FeedItem body supports only limited set of HTML tags:

p, b, code, i, u, s, ul, ol, li, img,

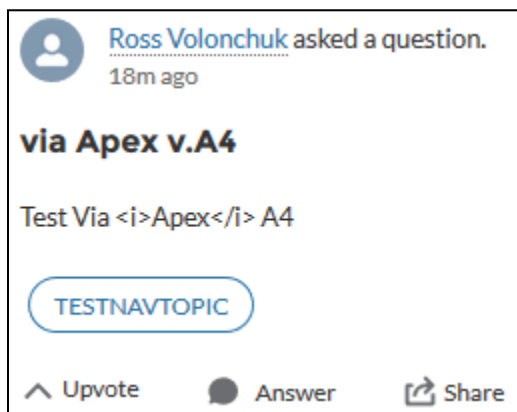
FeedItem records can be created with Apex in the community:

```
FeedItem post = new FeedItem();  
post.ParentId = '<USER-OR-GROUP-OR-RECORD-ID>'; //where to post message  
post.CreatedById = '<OWNER-USER-ID>';  
post.CreatedDate = Datetime.valueOf('YYYY-MM-DD hh:mm:ss');  
post.Title = '<TITLE>';  
post.Body = '<CONTENT>';  
post.NetworkScope = '<COMMUNITY-ID>';  
post.Type = 'QuestionPost';  
post.isRichText = true;  
insert post;
```


To import via Data Loader, the CSV should contain the same fields as in the Apex example above. You can find more information on supported formats for Date/DateTime fields here:

https://help.salesforce.com/articleView?id=000004680&language=en_US&type=1

FeedItem won't automatically recognize HTML content so FeedItem attribute isRichText has to be set isRichText=TRUE



If parentId is set to other user Id, the FeedItem will look like this (where owner is user102 and parentId is my user):



2.4.1) Feed Items with images

Chatter doesn't accept any value for "src" attribute besides the file Id, so it will only work like this ``.

This means that if you are importing posts with images, the image files should be imported first and all image references should be appropriately replaced later.

To be able to import images using Apex from any external website, this website URL must be registered in "Remote Site Settings". Let's assume it's `https://image.domain`.

The image files can have multiple versions. The file is stored in ContentDocument object record that stores the id of latest ContentVersion record of the file. The ContentVersion contains the actual binary file. ContentDocument is created automatically after the ContentVersion record has been created.

The example of importing one image:

```
String USERCUSTOMER = '<FORUM-USER-OWNER-ID>';
String COMMUNITY_ID = '<COMMUNITY-ID>';

String imageURL = 'https://image.domain/some-url-to-image.png';
PageReference pageRef = new PageReference(imageURL);
ContentVersion cv = new ContentVersion();
cv.VersionData = pageRef.getContent();
cv.Title = 'Image Name';
cv.PathOnClient = imageURL ;
cv.Origin = 'H';//Chatter
insert cv;
```

Note: getContent() method is not allowed to use inside the triggers.

To be able use image id during import, the DataLoader will accept only the files owned by the same owner as the FeedItem.

You can't specify the owner on record when creating the record but it is possible to set another owner by using the update operation:

```
cv.OwnerId = '<FEED-ITEM-OWNER-ID>';
update cv;
```

As I mentioned previously, ContentDocument object are created automatically, so it can be retrieved this way.

```
String contentDocId = [SELECT ContentDocumentId FROM ContentVersion
WHERE Id = :cv.Id LIMIT 1].ContentDocumentId;
```

Please note: The image size is limited to script available memory allocation - "heap size" for regular method and 6Mb, 12Mb for future methods. If this is not enough, it is also possible to use [REST](#) up to 500Mb per one request (see ContentVersion).

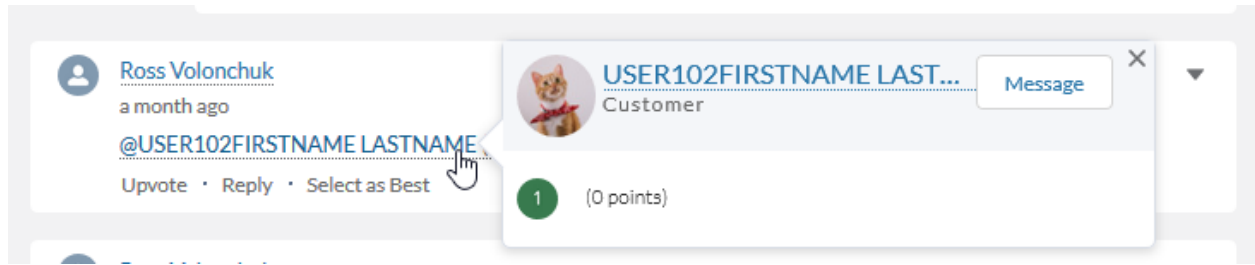
2.4.1.1) DataLoader images import

DataLoader also can be used to upload images if they are available locally. In this case the import object is ContentVersion and the path to the image should be specified in VersionData field:

	A	B	C	D
1	<u>VersionData</u>	<u>Title</u>	<u>PathOnClient</u>	<u>Origin</u>
2	D:\Pic\Lc0cn5n.jpg	testLocalPic	D:\Pic\Lc0cn5n.jpg	H
3				

where Origin "H" means Chatter.

2.4.2) Feed Items with @mentions



Another thing to consider is mentions of other users. It's not possible to import them with Data Loader or any other similar tool but it can be done via rest feedupdate request.

In our particular example the mention on a legacy system looks like this @Username. Salesforce would not automatically link the user to "@username" instance in plain text.

It looks like the only way to properly add the mention is to use Chatter API.

Let's assume that the FeedItem has already been imported:

Question F

Line before

inline before @USER97FIRSTNAME LASTNAME inline after

Line after

2.4.2.1) REST API

This block contains the example of usage REST API directly from Salesforce itself. There is ConnectAPI namespace provided with a bunch of classes to do that but we want to show some parts of the raw content as it comes from CRM.

The first step to retrieve it is via get request:

```
HttpRequest req = new HttpRequest();
HttpResponse resp = new HttpResponse();
http httpMeth = new http();

req.setMethod('GET');
req.setHeader('Authorization', 'Bearer '+ UserInfo.getSessionID());
req.setHeader('Content-Type', 'application/json');
```

```
req.setEndpoint('https://<SF-DOMAIN>.my.salesforce.com/services/data/v44.0/
connect/communities/<COMMUNITY-ID>/chatter/feed-elements/<FEED-ITEM-
ID>');
```

```
resp = httpMeth.send(req);
```

The response will look like this (only body):

```
"body": {
  "isRichText": true,
  "messageSegments": [
    {"htmlTag": "p", "markupType": "Paragraph", "text": "", "type": "MarkupBegin"},
    {"text": "Line before", "type": "Text"},
    {"htmlTag": "p", "markupType": "Paragraph", "text": "\n", "type": "MarkupEnd"},
    {"htmlTag": "p", "markupType": "Paragraph", "text": "", "type": "MarkupBegin"},
    {"text": "inline before @USER97FIRSTNAME LASTNAME inline after", "type":
    "Text"},
    {"htmlTag": "p", "markupType": "Paragraph", "text": "\n", "type": "MarkupEnd"},
    {"htmlTag": "p", "markupType": "Paragraph", "text": "", "type": "MarkupBegin"},
    {"text": "Line after", "type": "Text"},
    {"htmlTag": "p", "markupType": "Paragraph", "text": "\n", "type": "MarkupEnd"}
  ],
  "text": "Line before\ninline before @USER97FIRSTNAME LASTNAME inline
after\nLine after\n"
}
```

As you can see, the body block consists of an array of message segments that contain markup with text blocks separately.

You need to incorporate the mention block with the two available parameters:

```
{ "type": "Mention", "id": "<USER-ID>" }
```

So this:

```
{ "text": "inline before @USER97FIRSTNAME LASTNAME inline after", "type":
"Text" },
```

Becomes:

```
{ "text": "inline before", "type": "Text" },
{ "type": "Mention", "id": "<USER-ID>" },
{ "text": "inline after", "type": "Text" },
```

How to update post with mention:

```
httpRequest req = new HttpRequest();
httpResponse resp = new HttpResponse();
http httpMeth = new http();

req.setMethod('POST');
req.setHeader('Authorization', 'Bearer '+ UserInfo.getSessionID());
req.setHeader('Content-Type', 'application/json');
req.setEndpoint('https://<SF-DOMAIN>.my.salesforce.com/services/data/v44.0/
connect/communities/<COMMUNITY-ID>/chatter/feed-elements/<FEED-ID>?_
HttpMethod=PATCH');
req.setBody('{ "body": { "messageSegments": [
{"markupType": "Paragraph", "type": "MarkupBegin"},
{"text": "Linebefore", "type": "Text"},
{"markupType": "Paragraph", "type": "MarkupEnd"},
{"markupType": "Paragraph", "type": "MarkupBegin"},
{"text": "Inlinebefore", "type": "Text"},
{"type": "Mention", "id": "0051t000001bmmi"},
{"text": "Inlineafter", "type": "Text"},
{"markupType": "Paragraph", "type": "MarkupEnd"},
{"markupType": "Paragraph", "type": "MarkupBegin"},
{"text": "Lineafter", "type": "Text"},
{"markupType": "Paragraph", "type": "MarkupEnd"}
] } }');

resp = httpMeth.send(req);
```

Note: [1] as for v44 {"markupType": "Paragraph", "type": "MarkupBegin"},

This didn't support properties text, htmltag

[2] To make an update, the request method type can be overridden by adding ?_HttpMethod=PATCH to the endpoint.

Question F

Linebefore

Inlinebefore@USER97FIRSTNAME LASTNAME (Customer)Inlineafter

Lineafter

2.4.2.2) ConnectAPI

One more way to do almost the same thing but without processing a lot of json text is ConnectAPI. It has methods to retrieve and update FeedItem but they are in different classes.

ConnectApi.ChatterFeeds.getFeedElement - returns ConnectApi.FeedElement but

ConnectApi.ChatterFeeds.updateFeedElement - requires ConnectApi.FeedElementInput

There is no built-in method to convert FeedItem output (ConnectApi.FeedElement) into FeedItem input (ConnectApi.FeedElementInput), but Salesforce provides an external helper class [ConnectApiHelper](#) where createInputFromBody method can handle conversion for us.

Here is a mock example of how apex script may look:

```
ConnectApi.FeedElement feedObj =  
ConnectApi.ChatterFeeds.getFeedElement('<COMMUNITY-ID>', '<FEED-EL-ID>');
```

```
ConnectApi.FeedItemInput feedItemInputVAR = new  
ConnectApi.FeedItemInput();
```

```
ConnectApi.MessageBodyInput messageBodyInputVAR = new  
ConnectApi.MessageBodyInput();  
messageBodyInputVAR.messageSegments = new  
List<ConnectApi.MessageSegmentInput>();
```

```
ConnectApi.MessageBodyInput messageBodyInputVARFeedItem =  
ConnectApiHelper.createInputFromBody(feedObj.body);
```

```
for (ConnectApi.MessageSegmentInput segment  
: messageBodyInputVARFeedItem.messageSegments) {
```

```
    if (segment instanceof TextSegmentInput) {  
        //process segment.text value  
        //detect if have @  
        //extract value after @ (assuming @username)  
        //detect by value user-id  
        //split original text into text before and after, and insert mention between
```

```
        //ConnectApi.TextSegmentInput segmentA = new ConnectApi.TextSegmentInput();
```

```

//segmentA.text = '<TEXT-BEFORE-MENTION>';
//messageBodyInputVAR.messageSegments.add(segmentA);

//ConnectApi.MentionSegmentInput mention = new
ConnectApi.MentionSegmentInput();
//mention.id = '<USER-ID>';
//messageBodyInputVAR.messageSegments.add(mention);

//ConnectApi.TextSegmentInput segmentB = new ConnectApi.TextSegmentInput();
//segmentB.text = '<TEXT-AFTER-MENTION>';
//messageBodyInputVAR.messageSegments.add(segmentB);
} else {
    messageBodyInputVAR.messageSegments.add(segment);
}

}

```

To handle bigger amounts a batch version [getFeedElementBatch](#) is also available.

2.4.3) Internal Links Consideration

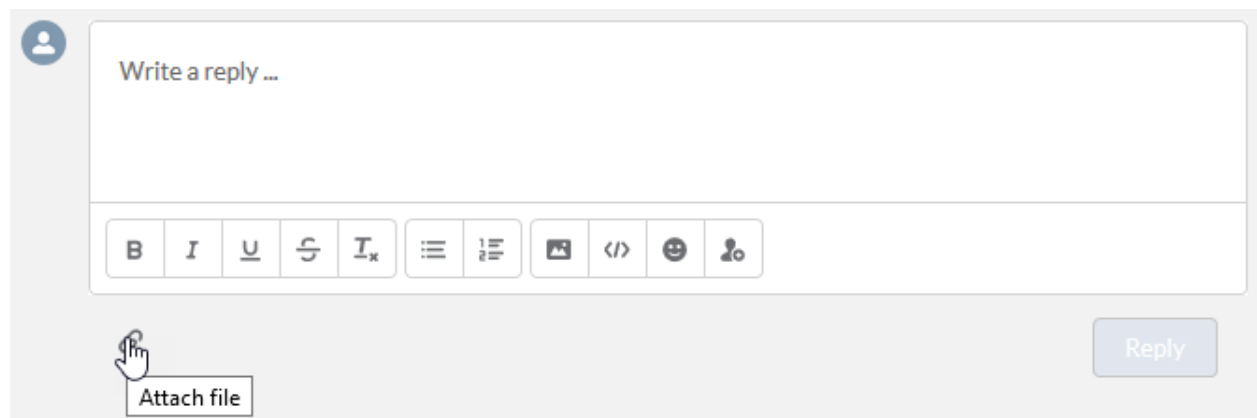
The external system may have internal links that should be relinked accordingly when moving content to community.

2.5) Feed Comments

FeedComments are used to store answers of FeedItems. They have the same content formatting restriction as the FeedItem.


There are two types of comments depending on if it has an attachment or not - TextComment and ContentComment.

If attachment is added by clip-button as shown on the image below then it's ContentComment:

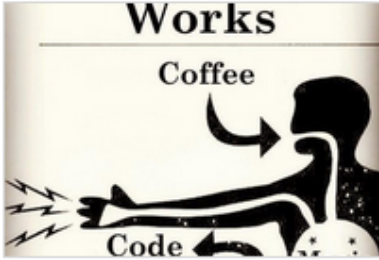


Here is an example of how it differs in appearance (same image used):

A) Attached image

**Ross Volonchuk**
21 minutes ago

hey, look at the attachment



Upvote · Reply · Select as Best

B) Inline image

**Ross Volonchuk**
3 minutes ago

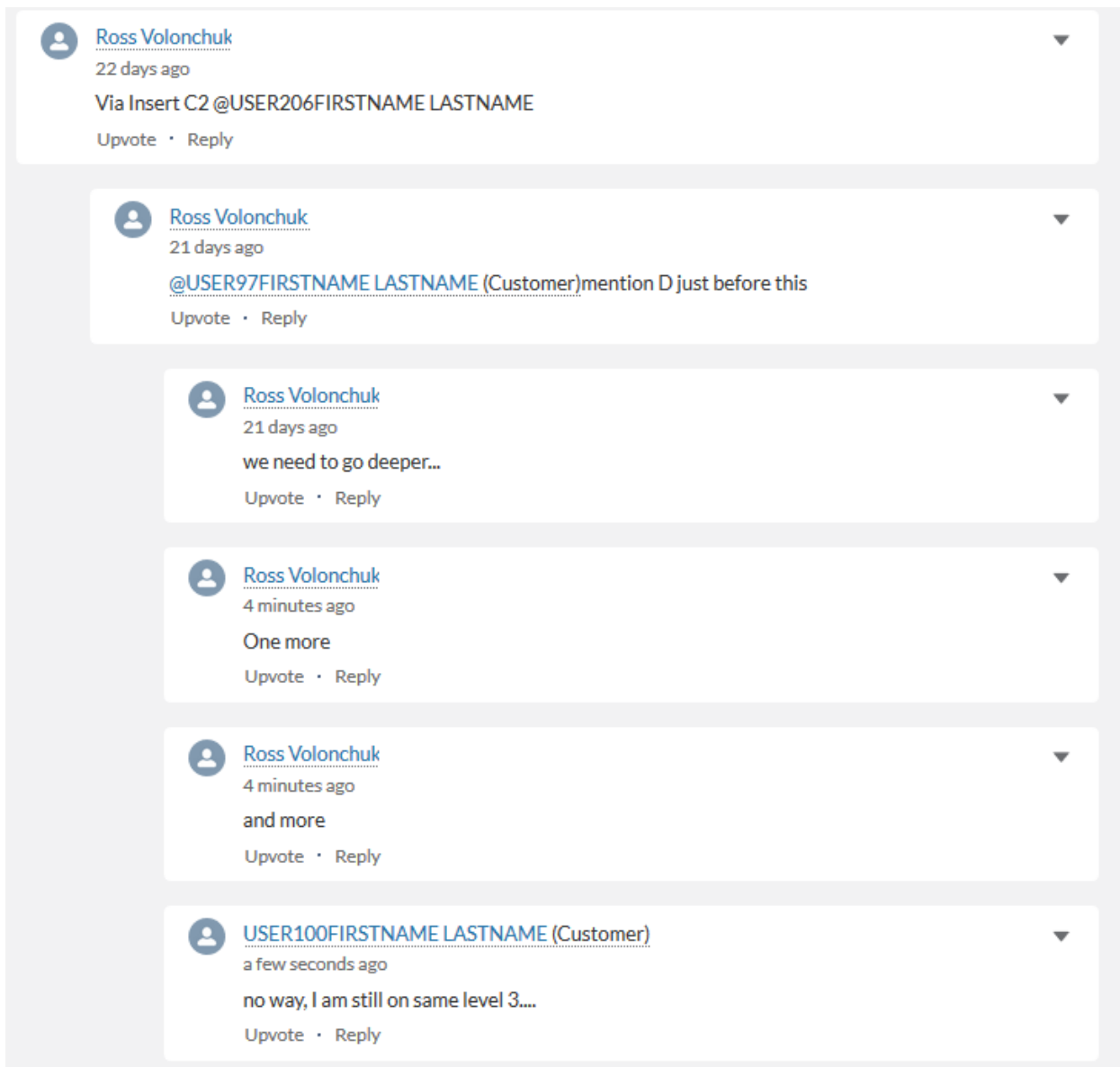
and this same but inline image



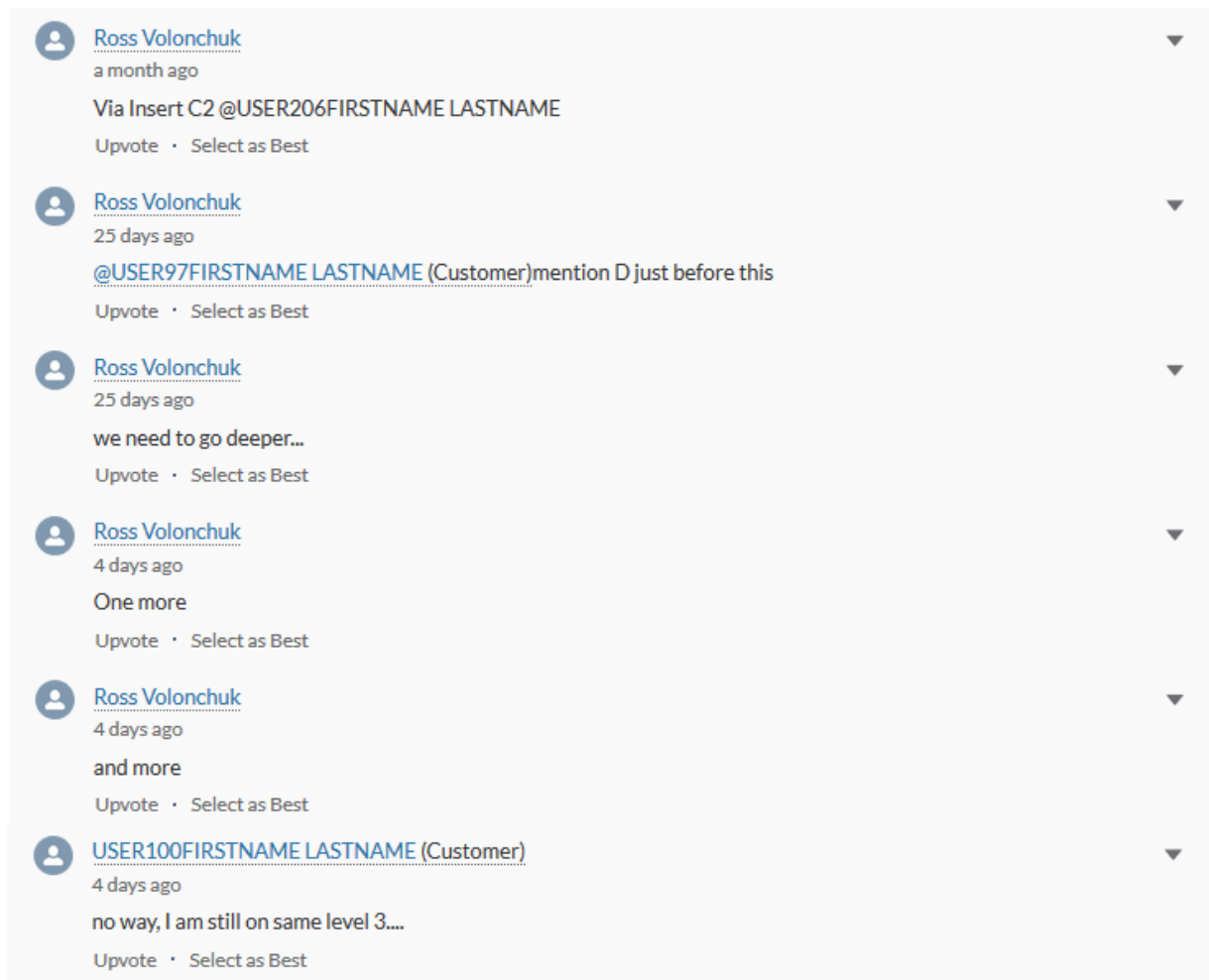
Upvote · Reply · Select as Best

Both images are clickable to open a full-size image view.

As we have mentioned in the setup section, threaded discussions are indented up to three levels. In the example below, every new comment was added by reply to the previous one:



If threaded discussions are not enabled, the same tree as on the picture above will look like this:



You can use Apex to create comments:

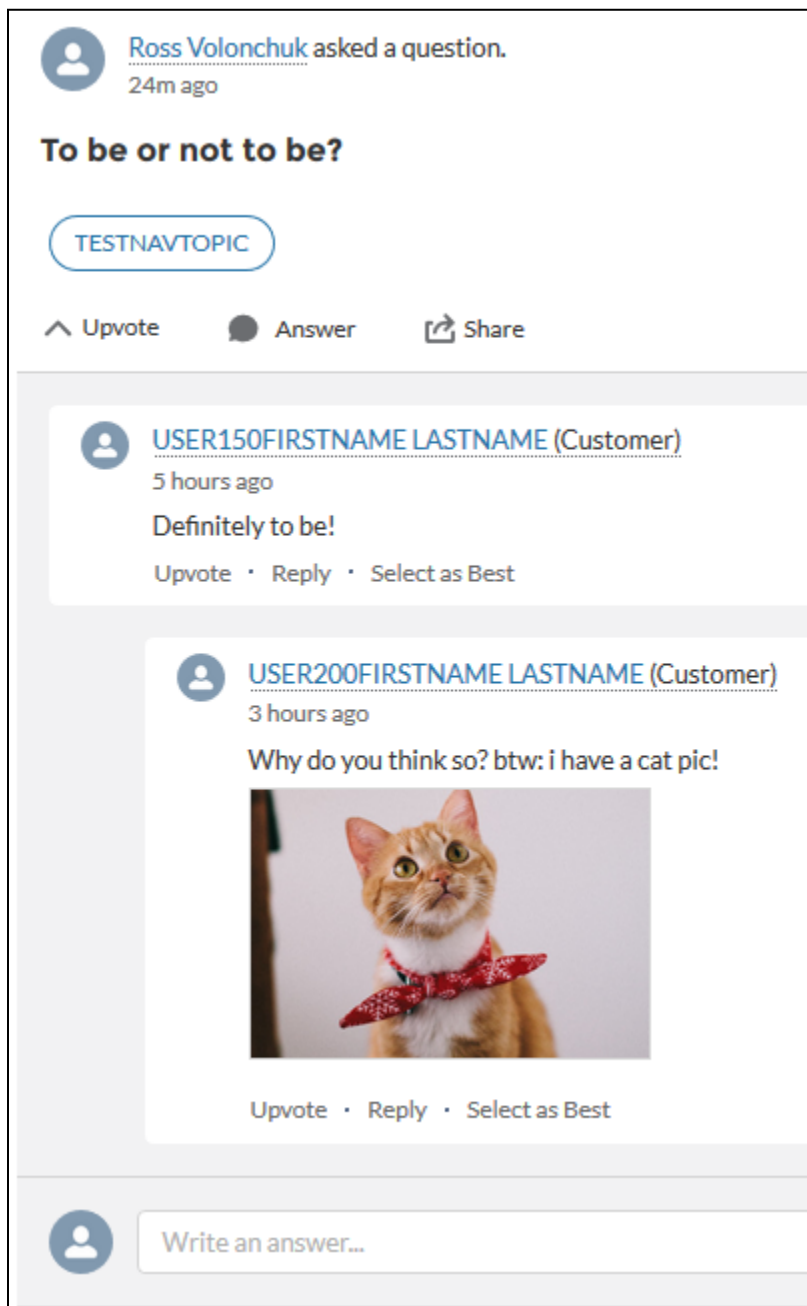
Let's assume that we already have created FeedItem and are going to add two comments:

```
FeedComment commentA = new FeedComment();
commentA.CreatedById = USER_A;
commentA.CreatedDate = Datetime.valueOf('2019-01-08 10:11:55');
commentA.FeedItemId = FEED_ITEM_ID;
commentA.isRichText = true;
commentA.CommentBody = 'Definitely to be!';
insert commentA;
```

```
FeedComment commentB = new FeedComment();
commentB.CommentBody = 'Why do you think so? btw: i have a cat pic!';
commentB.CreatedById = USER_B;
commentB.CreatedDate = Datetime.valueOf('2019-01-08 12:11:55');
commentB.FeedItemId = FEED_ITEM_ID;
```

```
commentB.ThreadParentId = commentA.Id;  
commentB.isRichText = true;  
commentB.RelatedRecordId = FILE_VERSION_ID;  
insert commentB;
```

Second comment commentB has attribute **ThreadParentId** set to first comment commentA id which means the second comment CommentB is a reply to the first CommentA and also CommentB has an attached file which is linked via **RelatedRecordId** attribute.



There is a CSV of a basic example below (there are two users with comments to the same FeedItem):

B	C	D	E
user_id	feedElement	body	c_creation_date
0051t000001bmnZ	0D51t00000C9K5BCAV	Etiam volutpat dui eget malesuada dignissim.	2018-12-20 11:15:00
0051t000001bmoE	0D51t00000C9K5BCAV	Aliquam id erat iaculis, sodales felis vitae, laoreet leo.	2018-12-20 15:00:00

This will give us the following result:

Upvote
Answer
Share
2 answers · Seen by 1

USER150FIRSTNAME LASTNAME (Customer)

10 days ago

Etiam volutpat dui eget malesuada dignissim.

Upvote · Reply · Select as Best

USER200FIRSTNAME LASTNAME (Customer)

10 days ago

Aliquam id erat iaculis, sodales felis vitae, laoreet leo.

Upvote · Reply · Select as Best

2.5.1 Best Answer

Upvote
Answer
Share

USER150FIRSTNAME LASTNAME (Customer)

22 days ago

Etiam volutpat dui eget malesuada dignissim.

Selected as Best · Upvote · Reply · Remove as Best

USER200FIRSTNAME LASTNAME (Customer)

22 days ago

Aliquam id erat iaculis, sodales felis vitae, laoreet leo.

Upvote · Reply · Select as Best

`FeedItem.BestCommentId` store the comment Id that should be marked as “Best”.

2.6) Topics

Topics are the equivalent of hashtags and can be used to organize content on chatter and community. Every FeedItem can have up to 10 topics assigned to it. Find out more about Chatter Allocations [info](#).

Topics can be related to FeedItem records by object TopicAssignment:

```
EntityType=FeedItem  
EntityId=<Feed-Item-Id>  
TopicId=<Topic-Id>
```

Apex example:

```
TopicAssignment ta = new TopicAssignment();  
ta.TopicId = '<TOPIC-ID>';  
ta.EntityId = '<FEED-ITEM-ID>';  
ta.NetworkId = '<COMMUNITY-ID>';  
insert ta;
```

ConnectAPI also has methods related to topics, some of which are:

```
ConnectApi.Topics.createTopic(  
    '<communityId>', '<topic-name>', '<topic-description>');  
ConnectApi.Topics.assignTopicByName(  
    '<communityId>', '<recordId>', '<topicName>');  
ConnectApi.Topics.assignTopic(  
    '<communityId>', '<recordId>', '<topicId>');
```

The Community topics can be managed (i.e. the admin would be able to update/delete/create/merge them from community workspaces section of Salesforce), more details can be found [here](#). The managed topics can be created via API

```
ConnectApi.createManagedTopic(  
    '<communityId>',  
    '<topicId>',  
    '<ConnectApi.ManagedTopicType.TopicType>');
```

where **TopicType** can have the following values: Content, Featured, Navigational.

2.6.1) Knowledgeable People

The Knowledgeable People component displays up to five experts. The experts are displayed based on their activity on the topic and the recognition they've received for the topic. For example, Chatter considers how often people:

- Are mentioned in posts or in comments on posts with the topic
- Receive likes on comments on posts with the topic
- Receive endorsements for the topic

Endorsements are a strong indication of knowledge, so if someone endorses you, then you're automatically included in the list of knowledgeable people.

Note: Knowledgeable people are calculated and updated once a day.

2.7) Likes and Votes

Salesforce chatter allows you to have either votes or likes as a way for community users to give feedback on posts or questions. The Community Preference checkbox tick "Allow members to upvote and downvote" controls this option.

Likes are stored at FeedLike object (note: FeedLike object cannot be queried with SOQL)

FeedEntityId = FeedItem / FeedComment
FeedItemId
CreatedById

Votes are stored at FeedSignal object (they don't support query either)

FeedEntityId = FeedItem / FeedComment
FeedItemId
CreatedById
SignalType = UpDownVote
SignalValue = 0 - downvote / 1 - upvote (int)

2.8) Follows

Typically most social platforms allow some form of "friending" or "following" of other users or topics.

On Salesforce Community Cloud there is a special EntitySubscription object for this purpose.

SubscriberId = <user-Id>
ParentId = <object-To-Follow-Id>

2.9) Reputation Points

Salesforce Community Cloud supports gamification features such as Reputation levels and points which can be updated when importing data from legacy system.

Chatter stores reputation points at NetworkMember object.

It is not possible to create NetworkMember records so that we can only update them.

If we want to update reputation points we need to follow these steps:

First, export all members of our community with filter by NetworkId (select Id, MemberId fields):

Export [Close]

Step 3: Edit your Query
Edit the SOQL query for extraction.

Choose the query fields below.

- ☐ PreferencesDisabled
- ☐ ReputationPoints
- ☐ LastChatterActivityD

Create the where clauses to your query below.

Fields	Operation	Value
NetworkId	equals	00000CbMcGAK

The generated query will appear below. You may edit it before finishing.

```
Select Id, MemberId FROM NetworkMember WHERE NetworkId = '00000CbMcGAK'
```

We should already have csv file with pair sf-user-id and reputation points.

This needs to merge on “sf-user-id” to “member-id” (same key).

After merging, we should get every corresponding reputation points its record-id.

It may look like this:

	A	B	C	D
1	ID_SF_USER	FORUM_ID_C	c_reputation	network_member_id
2	0051t000001bmmqAAA	105	5768	0DO1t000000Ce24GAC
3	0051t000001bmpeAAA	288	826	0DO1t000000Ce3KGAS
4	0051t000001bmpbAAA	285	183	0DO1t000000Ce1mGAC
5	0051t000001bmp1AAA	249	106	0DO1t000000Ce3VGAS
6	0051t000001bmniAAA	160	76	0DO1t000000Ce3NGAS
7	0051t000001bmnoAAA	103	53	0DO1t000000Ce3kGAC
8	0051t000001bmnsAAA	169	51	0DO1t000000Ce2HGAS
9	0051t000001bmnaAAA	117	31	0DO1t000000Ce1zGAC
10	0051t000001bmmbAAA	7	26	0DO1t000000Ce2ZGAS
11	0051t000001bmnoAAA	115	16	0DO1t000000Ce3zGAC
12	0051t000001bmnnAAA	164	16	0DO1t000000Ce32GAC
13	0051t000001bmp7AAA	255	16	0DO1t000000Ce1TGAS
14	0051t000001bmppAAA	284	16	0DO1t000000Ce3SGAS
15	0051t000001bmphAAA	291	16	0DO1t000000Ce14GAC
16	0051t000001bmnaVAAQ	146	6	0DO1t000000Ce1cGAC

The import map is pretty simple:






#Mapping values

c_reputation=ReputationPoints

network_member_id=Id

As a proof that all went well we can open any community user profile or take a look at the LeaderBoard:

LEADERBOARD

1.		USER105FIRSTNAME LAS...	5768 Points
2.		USER261FIRSTNAME LAS...	826 Points
3.		USER249FIRSTNAME LAS...	106 Points
4.		Ross Volonchuk	100 Points
5.		USER103FIRSTNAME LAS...	53 Points

2.10) Chatter Groups

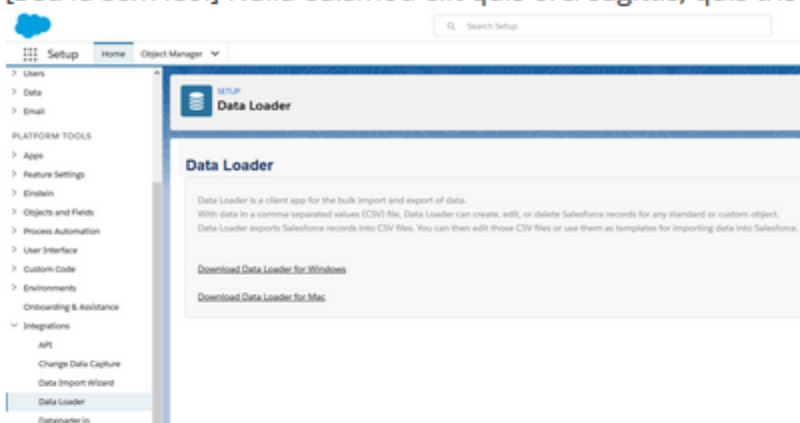
Chatter also provides groups functionality as a way of putting users and their discussions/topics together. The groups can be public, private or hidden. Chatter can be set up so that the members of the group get email notifications for each new message or a daily digest of all messages or a weekly digest of all messages.

3) Examples

3.1) Question Import

Let's assume that we have some "raw" FeedItem that needs to be imported:

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;
[Sed id sem leo.] Nulla euismod elit quis orci sagittis, quis tristique turpis ornare.



Aliquam a venenatis lorem. **Vivamus dictum luctus sollicitudin.** Integer at ante vel felis
porttitor accumsan. @USER150FIRSTNAME LASTNAME

Duis vitae iaculis eros.

- Donec
- laoreet
- blandit

Cras pellentesque faucibus sapien, quis convallis urna mattis sit amet. Nulla et neque
aliquam, faucibus massa auctor, placerat purus.

3.1.1) The html source of the question

Text version of input data:

-----QUESTION-----

@USER100FIRSTNAME LASTNAME

On 2018-12-20 11:00:00

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia
Curae; [Sed id sem leo.] Nulla euismod elit quis orci sagittis, quis tristique turpis
ornare.

```

<a href = "https://i.imgur.com/9jfzvc0.png"></a>
<br><br>
Aliquam a venenatis lorem. <b>Vivamus dictum luctus sollicitudin.</b> Integer at
ante vel felis porttitor accumsan. @USER150FIRSTNAME LASTNAME
<pre>
<code>
Duis vitae iaculis eros.
</code>
</pre>
<ul>
<li>Donec</li>
<li>laoreet</li>
<li>blandit</li>
</ul>
Cras pellentesque faucibus sapien, quis convallis urna mattis sit amet. Nulla et
neque aliquam, faucibus massa auctor, placerat purus.
<br>
-----first comment-----
@USER150FIRSTNAME LASTNAME
On 2018-12-20 11:15:00
---
Etiam volutpat dui eget malesuada dignissim.
-----second comment-----
@USER200FIRSTNAME LASTNAME
On 2018-12-20 15:00:00
---
Aliquam id erat iaculis, sodales felis vitae, laoreet leo.
-----

```

3.1.2) Processing question body

FeedItem body supports only limited tags (p, b, code, i, u, s, ul, ol, li, img), so needs to get rid of the others:

 replace with "<p> </p>"

<pre> remove

<a> remove but leave href attribute value

 src attribute supports only file-id value like this sfdc://069xxxxxxxxxxxxx

To upload image by link, use the code snippet at FeedItem with the Image section above. It is important to set the owner of the file as the same user as the author of the question.

After importing this image we got ID 0691t000000hVT7AAM

Users corresponding SF Ids (found by common "forum_Id" custom field):

@USER100FIRSTNAME LASTNAME = 0051t000001bmml

@USER150FIRSTNAME LASTNAME = 0051t000001bmnZ

@USER200FIRSTNAME LASTNAME = 0051t000001bmoE

User mention is not possible to handle on this step; it will be processed later on the FeedItem update.

After processing the question body, the code that is ready for import may look as follows':

<p> </p>

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; [Sed id sem leo.] Nulla euismod elit quis orci sagittis, quis tristique turpis ornare.

<https://i.imgur.com/9jfzvc0.png>

<p> </p><p> </p>

Aliquam a venenatis lorem. Vivamus dictum luctus sollicitudin. Integer at ante vel felis porttitor accumsan. @USER150FIRSTNAME LASTNAME

<code>

Duis vitae iaculis eros.

</code>

Donec

laoreet

blandit

Cras pellentesque faucibus sapien, quis convallis urna mattis sit amet. Nulla et neque aliquam, faucibus massa auctor, placerat purus.

<p> </p>

3.2) Importing Question using DataLoader

| sf_id2 | sf_id | c_body | c_creation_date | c_title | type | NetworkScope | IsRichText | Status | Visibility |
|-----------------|-----------------|---|---------------------|---------|--------------|-----------------|------------|-----------|------------|
| 0051t000001bmml | 0051t000001bmml | <p> </p>
Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; [Sed id sem leo.] Nulla euismod elit quis orci sagittis, quis tristique turpis ornare.
https://i.imgur.com/9jfzvc0.png

<p> </p><p> </p>
Aliquam a venenatis lorem. Vivamus dictum luctus sollicitudin. Integer at ante vel felis porttitor accumsan. @USER150FIRSTNAME LASTNAME
<code>
Duis vitae iaculis eros.
</code>


Donec
laoreet
blandit

Cras pellentesque faucibus sapien, quis convallis urna mattis sit amet. Nulla et neque aliquam, faucibus massa auctor, placerat purus.
<p> </p> | 2018-12-20 11:00:00 | testG1 | QuestionPost | 0DB1t000000CbMc | TRUE | Published | AllUsers |

Load Inserts

Step 2: Select data objects

Select your Salesforce object and your CSV file.



Select Salesforce object:

☒ Show all Salesforce objects

Event (Event)
Event Relation (EventRelation)
External Data User Authentication (ExternalDataUserAuth)
Feed Attachment (FeedAttachment)
Feed Comment (FeedComment)
Feed Item (FeedItem)
Feed Like (FeedLike)
Feed Signal (FeedSignal)

Choose CSV file:

Browse...

< Back

Next >

Finish


Cancel

Load Inserts

×

Step 3: Mapping

Map your fields (CSV columns) to the Salesforce object.



Choose an Existing Map

Create or Edit a Map

Current Field Mapping:

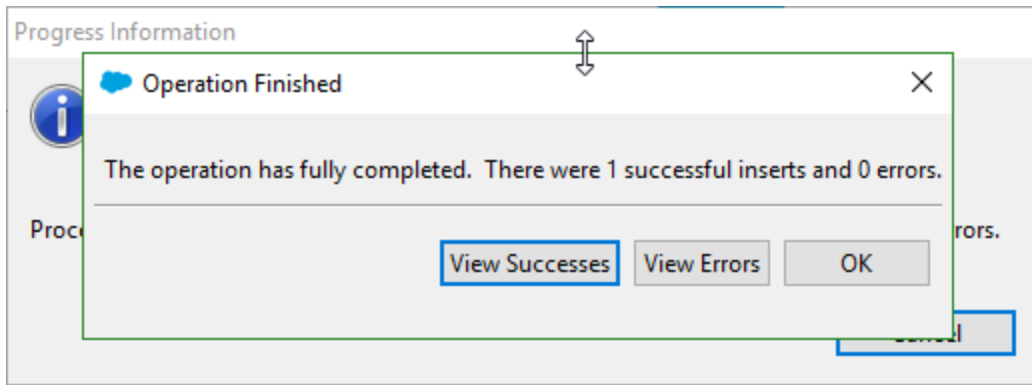
| File Column Header | Name | |
|--------------------|--------------|--|
| Status | Status | |
| Visibility | Visibility | |
| sf_id | ParentId | |
| c_title | Title | |
| c_body | Body | |
| NetworkScope | NetworkScope | |
| sf_id2 | CreatedById | |
| type | Type | |
| IsRichText | IsRichText | |
| c_creation_date | CreatedDate | |

< Back

Next >

Finish

Cancel



After import we got FeedItem Id 0D51t00000C9K5BCAV

3.2.1) Topic assigning

Assign topic:

```
TopicAssignment ta = new TopicAssignment();  
ta.TopicId = 'OTO1t000000TZOXGA4';  
ta.EntityId = '0D51t00000C9K5BCAV';//FeedItemId  
ta.NetworkId = '0DB1t000000CbMc';  
insert ta;
```



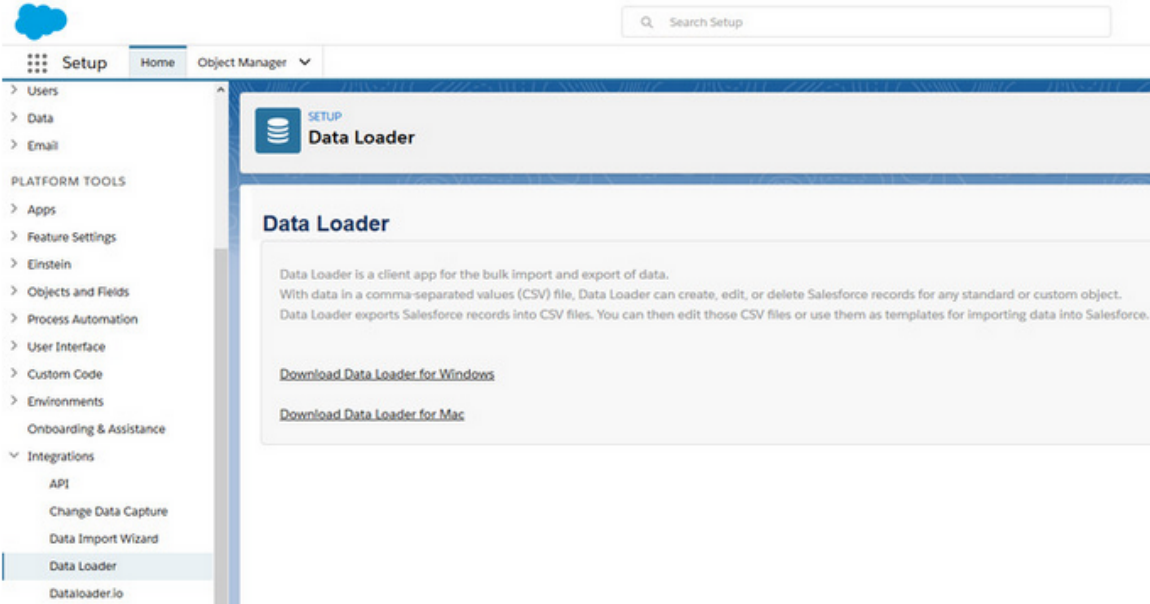
USER100FIRSTNAME LASTNAME (Customer) asked a question.

20 December 2018 at 09:00



testG1

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; [Sed id sem leo.] Nulla euismod elit quis orci sagittis, quis tristique turpis ornare. <https://i.imgur.com/9jfzvc0.png>



Aliquam a venenatis lorem. Vivamus dictum luctus sollicitudin. Integer at ante vel felis porttitor accumsan.

@USER150FIRSTNAME LASTNAME

1 Duis vitae iaculis eros.

- Donec
- laoreet
- blandit

Cras pellentesque faucibus sapien, quis convallis urna mattis sit amet. Nulla et neque aliquam, faucibus massa auctor, placerat purus.

TESTNAVTOPIC

Upvote



Answer



Share

Seen by 1

Next let's handle mention (following the steps as described in the Mention section).

3.3) Comments Importing

Imported FeedItem has id 0D51t00000C9K5BCAV.

| B | C | D | E | F | G |
|-----------------|--------------------|--|---------------------|------------|-----------|
| user_id | feedElement | body | c_creation_date | IsRichText | Status |
| 0051t000001bmnZ | 0D51t00000C9K5BCAV | Etiam volutpat dui eget malesuada dignissim. | 2018-12-20 11:15:00 | TRUE | Published |
| 0051t000001bmoE | 0D51t00000C9K5BCAV | Aliquam id erat iaculis, sodales felis vitae, laoreet leo. | 2018-12-20 15:00:00 | TRUE | Published |

Select Salesforce object:

☒ Show all Salesforce objects

Entity Subscription (EntitySubscription)
Event (Event)
Event Relation (EventRelation)
External Data User Authentication (ExternalDataUserAuth)
Feed Attachment (FeedAttachment)
Feed Comment (FeedComment)
Feed Item (FeedItem)
Feed Like (FeedLike)
Feed Signal (FeedSignal)

Choose CSV file:

Mapping Dialog

Match the Salesforce fields to your columns.

Clear Mapping

Auto-Match Fields to Columns

| Name | Label | Type |
|-----------------------|--------------------------|-----------|
| CommentType | Comment Type | picklist |
| LastEditById | Last Edit By ID | reference |
| LastEditDate | Last Edit Date | datetime |
| RelatedRecordId | Related Record ID | reference |
| Revision | Revision | int |
| ThreadChildrenCount | Thread Children Count | int |
| ThreadLastUpdatedDate | Thread Last Updated Date | datetime |
| ThreadLevel | Thread Level | int |
| ThreadParentId | Feed Comment ID | reference |

Drag the Salesforce fields down to the column mapping. To remove

| File Column Header | Name | |
|--------------------|-------------|--|
| body | CommentBody | |
| c_creation_date | CreatedDate | |
| feedElement | FeedItemId | |
| IsRichText | IsRichText | |
| parentId | | |
| Status | Status | |
| user_id | CreatedById | |
| | | |
| | | |
| | | |
| | | |
| | | |

OK

Save Mapping

Cancel

The result of imported answers/comments:



[USER150FIRSTNAME LASTNAME](#) (Customer)



10 days ago

Etiam volutpat dui eget malesuada dignissim.

[Upvote](#) · [Reply](#) · [Select as Best](#)



[USER200FIRSTNAME LASTNAME](#) (Customer)



10 days ago

Aliquam id erat iaculis, sodales felis vitae, laoreet leo.

[Upvote](#) · [Reply](#) · [Select as Best](#)



Write an answer...

We help our customers succeed with Salesforce and Community Cloud.

Our experience, prebuilt templates and development expertise enable you to benefit from Salesforce and Community Cloud's powerful services quickly and cost effectively. Integrating your customer services and support functions into Salesforce and Community Cloud cuts costs and creates exciting new ways to do business better and more profitably.

Working with us can help you.

- Provide better customer service
- Accelerate your sales
- Build valuable communities
- Better connect with your customers

Our extensive experience of developing and implementing CRM, Salesforce and Communities related projects for companies and Salesforce partners means that we can deliver your next project successfully.



**Biggest Community components
provider on AppExchange**



**Award-winning Salesforce
Community EMEA Break Out
Partner**



**Over 30,000 installations of
our Community components**

Contact us now to discuss how we can help you succeed!



advancedcommunities.com

We Work 30 Churchill Place, Canary Wharf, London E14 5RE

We Work SOMA, 156 2nd Street, San Francisco, CA 94105 USA

Email: sales@advancedcommunities.com

Phone: +44 20 3051 0075