Yammer

Yammer

Available as of Camel 2.12

The Yammer component allows you to interact with the Yammer enterprise social network. Consuming messages, users, and user relationships is supported as well as creating new messages.

Yammer uses OAuth 2 for all client application authentication. In order to use camel-yammer with your account, you'll need to create a new application within Yammer and grant the application access to your account. Finally, generate your access token. More details are at https://developer.yammer.com/v1.0/docs/authentication

Maven users will need to add the following dependency to their pom.xml for this component:

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-yammer</artifactId>
    <version>${camel-version}</version>
</dependency>
```

URI format

```
yammer:[function]?[options]
```

YammerComponent

The yammer component can be configured with the Yammer account settings which are mandatory to configure before using. You can also configure these options directly in the endpoint.

Option	Description
consumerKey	The consumer key
consumerSecret	The consumer secret
accessToken	The access token

Consuming messages

The camel-yammer component provides several endpoints for consuming messages:

URI	Description
`yammer:messages?[options]`	All public messages in the user's (whose access token is being used to make the API call) Yammer network. Corresponds to "All" conversations in the Yammer web interface.
`yammer:my_feed?[options]`	The user's feed, based on the selection they have made between "Following" and "Top" conversations.
`yammer:algo?[options]`	The algorithmic feed for the user that corresponds to "Top" conversations, which is what the vast majority of users will see in the Yammer web interface.
`yammer:following?[options]`	The "Following" feed which is conversations involving people, groups and topics that the user is following.

`yammer:sent?[options]`	All messages sent by the user.
`yammer:private?[options]`	Private messages received by the user.
`yammer:received?[options]`	**Camel 2.12.1:** All messages received by the user

URI Options for consuming messages

Name	Default Value	Description
useJson	`false`	Set to true if you want to use raw JSON rather than converting to POJOs.
delay	`5000`	in milliseconds
consumerKey	`null`	Consumer Key. Can also be configured on the `YammerComponent` level instead.
consumerSecret	`null`	Consumer Secret. Can also be configured on the `YammerComponent` level instead.
accessToken	`null`	Access Token. Can also be configured on the `YammerComponent` level instead.
limit	`-1`	Return only the specified number of messages. Works for threaded=true and threaded=extended.
threaded	`null`	threaded=true will only return the first message in each thread. This parameter is intended for apps which display message threads collapsed. threaded=extended will return the thread starter messages in order of most recently active as well as the two most recent messages, as they are viewed in the default view on the Yammer web interface.
olderThan	`-1`	Returns messages older than the message ID specified as a numeric string. This is useful for paginating messages. For example, if you're currently viewing 20 messages and the oldest is number 2912, you could append "?olderThan=2912 to your request to get the 20 messages prior to those you're seeing.
newerThan	`-1`	Returns messages newer than the message ID specified as a numeric string. This should be used when polling for new messages. If you're looking at messages, and the most recent message returned is 3516, you can make a request with the parameter "?newerThan=3516 to ensure that you do not get duplicate copies of messages already on your page.

Message format

All messages by default are converted to a POJO model provided in the org.apache.camel.component.yammer.model package. The original message coming from yammer is in JSON. For all message consuming & producing endpoints, a Messages object is returned. Take for example a route like:

```
from("yammer:messages?consumerKey=aConsumerKey&consumerSecret=aConsumerSecretKey&accessToken=aAccessToken").to
("mock:result");
```

and lets say the yammer server returns:

```
{
        "messages":[
                {
                        "replied_to_id":null,
                        "network_id":7654,
                        "url":"https://www.yammer.com/api/v1/messages/305298242",
                        "thread_id":305298242,
                        "id":305298242,
                        "message_type":"update",
                        "chat_client_sequence":null,
                        "body":{
                                "parsed":"Testing yammer API...",
                                "plain":"Testing yammer API...",
```

```
                                "rich":"Testing yammer API..."
                        },
                        "client_url":"https://www.yammer.com/",
                        "content_excerpt":"Testing yammer API...",
                        "created_at":"2013/06/25 18:14:45 +0000",
                        "client_type":"Web",
                        "privacy":"public",
                        "sender_type":"user",
                        "liked_by":{
                                "count":1,
                                "names":[
                                        {
                                                "permalink":"janstey",
                                                "full_name":"Jonathan Anstey",
                                                "user_id":1499642294
                                        }

                                ]

                        },
                        "sender_id":1499642294,
                        "language":null,
                        "system_message":false,
                        "attachments":[

                        ],
                        "direct_message":false,
                        "web_url":"https://www.yammer.com/redhat.com/messages/305298242"
                },
                {
                        "replied_to_id":null,
                        "network_id":7654,
                        "url":"https://www.yammer.com/api/v1/messages/294326302",
                        "thread_id":294326302,
                        "id":294326302,
                        "message_type":"system",
                        "chat_client_sequence":null,
                        "body":{
                                "parsed":"(Principal Software Engineer) has [[tag:14658]] the redhat.com
network. Take a moment to welcome Jonathan.",
                                "plain":"(Principal Software Engineer) has #joined the redhat.com network. Take
a moment to welcome Jonathan.",
                                "rich":"(Principal Software Engineer) has #joined the redhat.com network. Take
a moment to welcome Jonathan."
                        },
                        "client_url":"https://www.yammer.com/",
                        "content_excerpt":"(Principal Software Engineer) has #joined the redhat.com network.
Take a moment to welcome Jonathan.",
                        "created_at":"2013/05/10 19:08:29 +0000",
                        "client_type":"Web",
                        "sender_type":"user",
                        "privacy":"public",
                        "liked_by":{
                                "count":0,
                                "names":[

                                ]

                        }
                }
                ]

        }
```


Camel will marshal that into a Messages object containing 2 Message objects. As shown below there is a rich object model that makes it easy to get any information you need:

```
        Exchange exchange = mock.getExchanges().get(0);
        Messages messages = exchange.getIn().getBody(Messages.class);

        assertEquals(2, messages.getMessages().size());
        assertEquals("Testing yammer API...", messages.getMessages().get(0).getBody().getPlain());
        assertEquals("(Principal Software Engineer) has #joined the redhat.com network. Take a moment to
welcome Jonathan.", messages.getMessages().get(1).getBody().getPlain());
```

That said, marshaling this data into POJOs is not free so if you need you can switch back to using pure JSON by adding the useJson=false option to your URI.

Creating messages

To create a new message in the account of the current user, you can use the following URI:

```
yammer:messages?[options]
```

The current Camel message body is what will be used to set the text of the Yammer message. The response body will include the new message formatted the same way as when you consume messages (i.e. as a Messages object by default).

Take this route for instance:

```
from("direct:start").to("yammer:messages?
consumerKey=aConsumerKey&consumerSecret=aConsumerSecretKey&accessToken=aAccessToken").to("mock:result");
```

By sending to the direct:start endpoint a "Hi from Camel!" message body:

```
        template.sendBody("direct:start", "Hi from Camel!");
```

a new message will be created in the current user's account on the server and also this new message will be returned to Camel and converted into a Messages object. Like when consuming messages you can interrogate the Messages object:

```

        Exchange exchange = mock.getExchanges().get(0);
        Messages messages = exchange.getIn().getBody(Messages.class);

        assertEquals(1, messages.getMessages().size());
        assertEquals("Hi from Camel!", messages.getMessages().get(0).getBody().getPlain());
```

Retrieving user relationships

The camel-yammer component can retrieve user relationships:

```
yammer:relationships?[options]
```

URI Options for retrieving relationships

Name	Default Value	Description
useJson	`false`	Set to true if you want to use raw JSON rather than converting to POJOs.
delay	`5000`	in milliseconds
consumerKey	`null`	Consumer Key. Can also be configured on the `YammerComponent` level instead.
consumerSecret	`null`	Consumer Secret. Can also be configured on the `YammerComponent` level instead.
accessToken	`null`	Access Token. Can also be configured on the `YammerComponent` level instead.
userId	`current user`	To view the relationships for a user other than the current user.

Retrieving users

The camel-yammer component provides several endpoints for retrieving users:

URI	Description
`yammer:users?[options]`	Retrieve users in the current user's Yammer network.
`yammer:current?[options]`	View data about the current user.

URI Options for retrieving users

Name	Default Value	Description
useJson	`false`	Set to true if you want to use raw JSON rather than converting to POJOs.
delay	`5000`	in milliseconds
consumerKey	`null`	Consumer Key. Can also be configured on the `YammerComponent` level instead.
consumerSecret	`null`	Consumer Secret. Can also be configured on the `YammerComponent` level instead.
accessToken	`null`	Access Token. Can also be configured on the `YammerComponent` level instead.

Using an enricher

It is helpful sometimes (or maybe always in the case of users or relationship consumers) to use an enricher pattern rather than a route initiated with one of the polling consumers in camel-yammer. This is because the consumers will fire repeatedly, however often you set the delay for. If you just want to look up a user's data, or grab a message at a point in time, it is better to call that consumer once and then get one with your route.

Lets say you have a route that at some point needs to go out and fetch user data for the current user. Rather than polling for this user over and over again, use the pollEnrich DSL method:

```
from("direct:start").pollEnrich("yammer:current?
consumerKey=aConsumerKey&consumerSecret=aConsumerSecretKey&accessToken=aAccessToken").to("mock:result");
```

This will go out and fetch the current user's User object and set it as the Camel message body.

See Also

- Configuring Camel
- Component
- Endpoint
- Getting Started