



Deliverable D33.3
Factsheet Number #3
Software prototype name:
“Feedback Management Service”

Document Owner:	Ioan Toma(UIBK), Iker Larizgoitia(UIBK)
Contributors:	Alex Oberhauser(UIBK), Christoph Fuchs(UIBK), Martin Kammerlander(UIBK), Corneliu Stanciu(UIBK)
Dissemination:	Public
Contributing to:	WP33
Date:	31/03/2013
Revision:	1.0

List of Abbreviations

FM	Feedback Management
-----------	---------------------

Table of Contents

1	AVAILABILITY AND CONTACTS	4
2	ARCHITECTURE AND FUNCTIONALITIES	5
3	TECHNICAL INFORMATION.....	7
3.1	TECHNICAL DETAILS - MODELS	7
3.2	SERVICE LICENSE	8
3.3	THIRD PARTY LICENSES	8
4	TECHNICAL MANUAL	9
4.1	FEEDBACK MANAGEMENT SERVICE INSTALLATION	9
4.2	FEEDBACK MANAGEMENT SERVICE API	9
4.2.1	<i>Authentications API.....</i>	<i>9</i>
4.2.2	<i>Channels API.....</i>	<i>9</i>
4.2.3	<i>Common Weaver Models.....</i>	<i>10</i>
4.2.4	<i>Feedbacks API.....</i>	<i>10</i>
4.2.5	<i>Images API.....</i>	<i>11</i>
4.2.6	<i>Links API.....</i>	<i>11</i>
4.2.7	<i>Microblogs posts API.....</i>	<i>11</i>
4.2.8	<i>Publications API.....</i>	<i>12</i>
4.2.9	<i>Remotes API.....</i>	<i>12</i>
4.2.10	<i>Videos API.....</i>	<i>13</i>
5	USER MANUAL.....	14
6	FUTURE PLANS.....	17

1 Availability and Contacts

This table describes how to reach the prototype and the contact person in case of questions.

Version	1.0
Availability	https://dacodi.sti2.at/
Accompanying specification and design document	D33.3 FI Utility Services first prototype – M18
Source control	svn://repo.nimbus-ware.comMSEE/SP3/WP33/D33.3/trunk/MSEE_WP33_FeedbackManagement
Contact person	Ioan Toma – STI Innsbruck ioan.toma@sti2.at

2 Architecture and Functionalities

The Feedback Management Service enables enterprises to collect customers’ and partners’ comments and feedback from various communication channels, trying to understand market trends and estimate customer satisfaction to improve their products and services. The Feedback Management Service provides an automated, unified publication method in a multichannel environment as well as the automatic aggregation of the feedback provided in the channels.

In short, the Feedback Management Service enables fast and easy one-click publishing and collection of feedback on a multitude of channels, hiding technology complexity behind a user-friendly interface. The architecture of the Feedback Management Service is available in Figure 1.

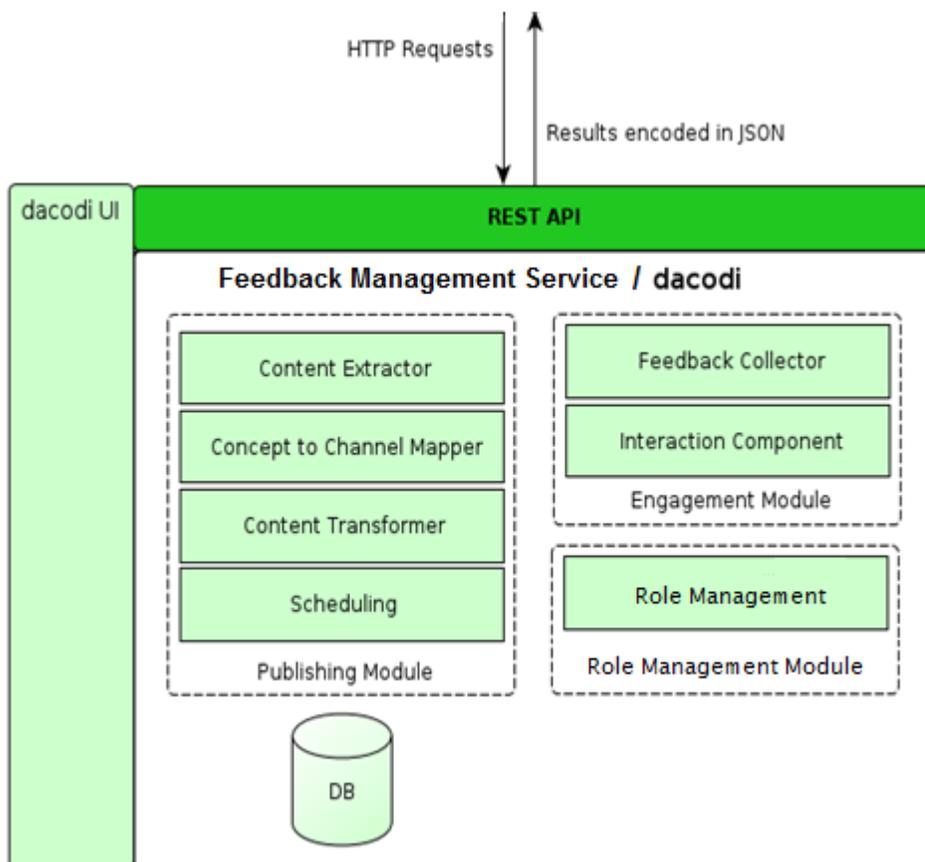


Figure 1 – Feedback Management Service Architecture

From a functional point of view the **Feedback Management Service**, branded as **dacodi**, contains the following modules:

- Role Management** – this module provides role management functionality for those that are using the Feedback Management Service. Two roles are envisioned and supported for the moment i.e. system *user* and *administrator*. A *user* is allowed to use the functionality of the service. An *administrator* has additional rights being able to administrate (add, delete, etc.) user accounts and has an overview about the health of the service.

- **Publication** – this module is responsible for the publication of content into channels. The Publication module is also responsible for adaptation of content to be published to the specific requirements of the different channels. The adaption is made on the base of the channel capabilities, e.g. shortened messages are posted to microblog platform, such as twitter. To achieve scalability the Feedback Management Service uses a message-oriented architecture to implement an asynchronous publication process. For this purpose the Advanced Message Queuing Protocol, short AMQP¹ is used.
- **Feedback Collection** – this module is responsible for the collection of various feedbacks across multiple channels. Feedback can have different forms. It can either be textual (comments, replies, etc.), a certain amount of positive (like, +1, thumbs up, etc.) or negative feedback (thumbs down, bury, down vote, etc.), or some other measurement of a user’s response to a published item. The Feedback Management Service can collect all these different types of feedback using various polling strategies. The Feedback Collection module is part of a larger module, called Engagement Module that includes also support for interaction and engagement with users via the Interaction Component. However the engagement part is not available in the current prototype.

The Feedback Management Service and its components are provided as a RESTfull service. We also provide a **Front-end Web application UI (dacodi UI)** that enables a user friendly access to the functionality offered by the Feedback Management Service including publication, feedback collection and role management.

The Feedback Management Service supports for the time being five channels i.e. Facebook, Twitter, LinkedIn, Flickr and YouTube.

¹ <http://amqp.org/>

3 Technical Information

3.1 Technical details - Models

To realize the outlined architecture, the objects that are handled by Feedback Management Service have the following models. These models are persistent in the database (DB) and used throughout the outlined components shown in Figure 1. The following models are used:

- **User models** a user in dacodi which can be uniquely identified by an account name (email) or identification token.
- **Authentication models** a single authentication of a user on a specific platform (e.g. facebook) which is usually given in the form of an authentication token-secret pair (e.g. oauth_token and oauth_secret in the case of the OAuth authentication protocol).
- **Channel models** a specific channel where content can be published or retrieved. A channel is directly linked to an adapter - for dacodi to make use of a channel, the related adapter has to be implemented.
- **ConceptToChannelMapping models** the central component of the channel selection process which is a Concept-to-Channel Mapper. This mapping maps each concept to the appropriate channels.
- **Remote models** a remote entity which was published in a channel - by dacodi or any other means. Every common weaver model instance has potentially multiple remote representations or remotes. The most important value of those Remotes is the remote_id, which uniquely identifies the remote instance on the external platform. Example: An image is represented as a common weaver model instance in our system. If it is uploaded to flickr and facebook, it has two remotes: one for flickr and one for facebook.
- **Feedback models** information that are generic and applicable to any form of feedback.
- **Publication models** an information item that was published via dacodi.
- **CommonWeaverModel models** the entities as described in the Common Weaver.

Based on the architecture defined in Figure 1, we now explain how they are addressed in the implementation:

- **Content creation** is a semi-automated process done by the users of the system. For example manufacturing enterprises will specify content items that they want to be disseminated in terms of new products and services offered, etc.
- **Channel selection:** Once an information item is sent to dacodi, based on the information item type (e.g. a business event), a fitting channel for the information item will be selected (e.g. business event is announced on LinkedIn but not Facebook). The central component of the channel selection process is an internal (Concept-to-Channel) Mapper, which maps each concept to the appropriate channels.
- **Content adaptation:** For every channel the information item has to fit in, a transformation might be necessary. For example: A business event might include

fields such as short title, long title, description, start date, end date, location and venue. Further, there might be an accompanying image which represents the event - like a poster. A channel which only takes short text messages (Twitter, for example) can't handle all those fields. Thus, the information item has to be transformed into a common representation from which the transformation to the different channels can happen. This common representation is what we call the Common Weaver Model (CWM).

3.2 Service license

Different parts for the Feedback Management are provided under different licenses. The service itself is provided under a property license being closed source. The Front-end Web application UI is open source being licensed under the LGPL license².

3.3 Third party licenses

Not available.

² <http://www.gnu.org/copyleft/lesser.html>

4 Technical Manual

4.1 Feedback Management Service Installation

A public instance of the service is installed and available to be used at: <https://dacodi.sti2.at/api/>.

4.2 Feedback Management Service API

The functionality of the Feedback Management Service is provided as RESTful service. The Feedback Management Service is implemented using Ruby on Rails³. For developers that want to use the service an API is provided and is available at: <https://dacodi.sti2.at/api/>. Examples on how to use the API are provided as well for each available method.

The following subsections describe the current API of the Feedback Management Service.

4.2.1 Authentications API

The Authentications API can be used to return all Authentications for the authenticated user. More details about the Authentications API including examples on how to use it are available at: <https://dacodi.sti2.at/api/v1/authentications.html>. The API includes the following methods:

- **GET /api/authentications**
Returns all Authentications for the authenticated user
Supported Formats: json

Parameters:

Parameter name	Description
Id (optional)	Value: Must be a number

- **GET /api/authentications/:id**
Returns a single Authentication for the authenticated user
Supported Formats: json

Parameters:

Parameter name	Description
Id (optional)	Value: Must be a number

4.2.2 Channels API

The Channels API can be used to return all Channels for the authenticated user. More details about the Channels API including examples on how to use it are available at: <https://dacodi.sti2.at/api/v1/channels/index.html>. The API includes the following methods:

- **GET /api/channels**
Returns all Channels for the authenticated user
Supported Formats: json

Parameters:

³ <http://rubyonrails.com/>

Parameter name	Description
Id (optional)	Value: Must be a number

- GET /api/authentications/:id**
 Returns a single Channel for the authenticated user
Supported Formats: json
Parameters:

Parameter name	Description
Id (optional)	Value: Must be a number

4.2.3 Common Weaver Models

The Common weaver models API can be used to return all common weaver models for the authenticated user. A Common Weaver Model (CVM) is the common representation of an information item from which the transformation to the different channels can happen. More details about the Common Weaver Models API including examples on how to use it are available at: https://sesa-dev.seekda.com/api/v1/common_weaver_models.html. The API includes the following methods:

- GET /api/common_weaver_models**
Supported Formats: json
Parameters:

Parameter name	Description
-	-

4.2.4 Feedbacks API

The Feedbacks API can be used to return all Feedbacks received for the current user. More details about the Feedbacks API including examples on how to use it are available at: <https://sesa-dev.seekda.com/api/v1/feedbacks.html>. The API includes the following methods:

- GET /api/feedbacks**
 Returns all Feedbacks of the current user
Supported Formats: json
Parameters:

Parameter name	Description
Id (optional)	Value: Must be a number

- GET /api/feedbacks/:id**
 Retrieves a single feedback
Supported Formats: json
Parameters:

Parameter name	Description
Id (optional)	Value: Must be a number

4.2.5 Images API

The Images API can be used to handle images, namely to publish into and to retrieve them various channels. More details about the Images API including examples on how to use it are available at: <https://dacodi.sti2.at/api/v1/images.html>. The API includes the following methods:

- **GET /api/images/:id**
Retrieves a single Image
Supported Formats: json
Parameters:

Parameter name	Description
Id (optional)	Value: Must be a number

- **POST /api/images**
Publishes an Image
Supported Formats: json
Parameters:

Parameter name	Description
-	-

4.2.6 Links API

The Links API can be used to retrieve links that are published by the authenticated user. More details about the Links API including examples on how to use it are available at: <https://dacodi.sti2.at/api/v1/links.html>. The API includes the following methods:

- **GET /api/links/:id**
Retrieves a single Link
Supported Formats: json
Parameters:

Parameter name	Description
Id (optional)	Value: Must be a number

4.2.7 Microblogs posts API

The Microblog posts API can be used to retrieve microblog post for the authenticated user. More details about the Microblogs posts API including examples on how to use it are available at: https://dacodi.sti2.at/api/v1/microblog_posts.html. The API includes the following methods:

- **GET /api/microblog_posts/:id**
Returns a single MicroblogPost for the authenticated user
Supported Formats: json
Parameters:

Parameter name	Description
Id (optional)	Value: Must be a number

4.2.8 Publications API

The Publications API is a core API of the Feedback Management Service that can be used to retrieve publications made by the authenticated user. More details about the Publications API including examples on how to use it are available at: <https://dacodi.sti2.at/api/v1/publications.html>. The API includes the following methods:

- **GET /api/publications**

Returns all Publications for the authenticated user

Supported Formats: json

Parameters:

Parameter name	Description
-	-

- **GET /api/publications/:id**

Retrieves a single Publication

Supported Formats: json

Parameters:

Parameter name	Description
Id (optional)	Value: Must be a number

- **GET /api/publications/:id-status**

Retrieves the Publication's status

Supported Formats: json

Parameters:

Parameter name	Description
Id (optional)	Value: Must be a number

4.2.9 Remotes API

Information items that have been published - either via our system, via the platform directly, or via 3rd party apps - are called Remotes (remote information items).

The Remotes API can be used to retrieve all remotes of the authenticated user. More details about the Remotes API including examples on how to use it are available at: <https://dacodi.sti2.at/api/v1/remotes.html>. The API includes the following methods:

- **GET /api/remotes**

Retrieves all Remotes of the current user

Supported Formats: json

Parameters:

Parameter name	Description
-	-

- **GET /api/remotes/:id**
Retrieves a single Remote
Supported Formats: json
Parameters:

Parameter name	Description
Id (optional)	Value: Must be a number

4.2.10 Videos API

The Videos API can be used to handle videos namely to publish into and to retrieve them various channels. More details about the Images API including examples on how to use it are available at: <https://dacodi.sti2.at/api/v1/images.html>. The API includes the following methods:

- **GET /api/videos/:id**
Retrieves a single Video
Supported Formats: json
Parameters:

Parameter name	Description
Id (optional)	Value: Must be a number

- **POST /api/videos**
Publishes a Video
Supported Formats: json
Parameters:

Parameter name	Description
-	-

5 User Manual

As part of the Feedback Management Service, a user friendly Front-end Web application is provided in order to use the service. The rest of this section gives an overview on how one can use the service from the Front-end Web application.

In order to use the service a user has to sign up for an account. If the user has already an account, she/he can login in (see Figure 2 and Figure 3).



Welcome

Welcome to dacodi! You need to [sign up](#) or [sign in](#) in order to play around with this prototype.

Figure 2 - Welcome page



Sign in

Email

Password

[Forgot your password?](#)

Remember me

Don't have an account? [Sign up here.](#)

Figure 3 - Sign in

Once login the user should add its social channels accounts that will enable her/him to publish on these channels. As mentioned before the Feedback Management Service supports for the time being five channels i.e. Facebook, Twitter, LinkedIn, Flickr and YouTube. The user can or more of each of these accounts. The accounts can be also removed if the user does not want any more to publish and retrieve feedback through these accounts. Please note that one should add accounts only once, as the accounts will be available without any need of additional setup. Figure 4 shows the manage accounts page.

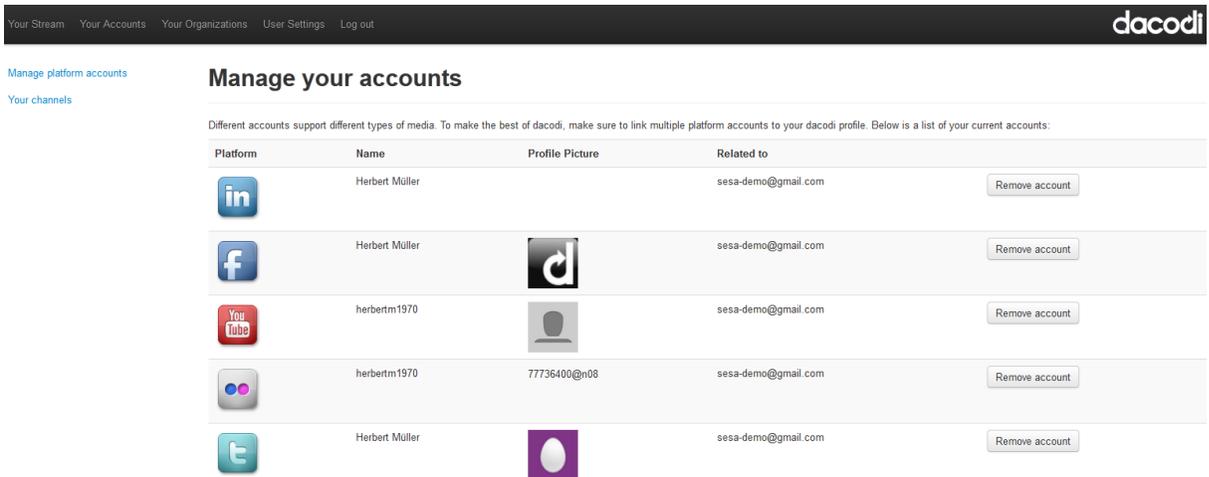


Figure 4 - Manage accounts

Once the accounts are setup the user can create the information item she/he wants to publish. She/he can publish a microblog post, a link, a video, an image or a combination of those. Figure 5 shows how one can publish a link about the MSEE project.

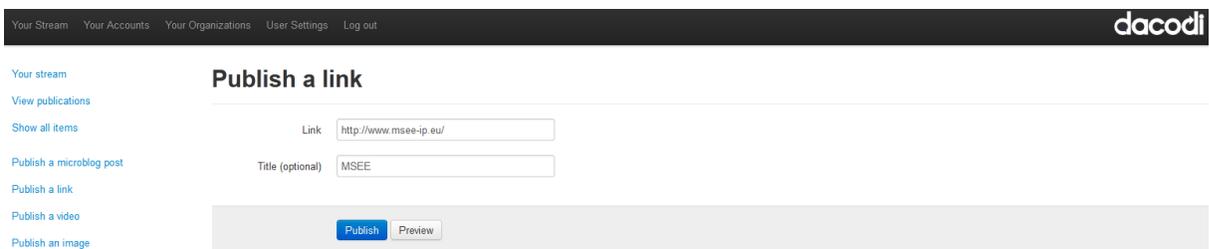


Figure 5 - Publish a link

In the current implementation the information items will be published on all the channels that the user has registered previously. The result of publishing the link about MSEE in the previous step on Facebook, one of the available channels, is shown in Figure 6.

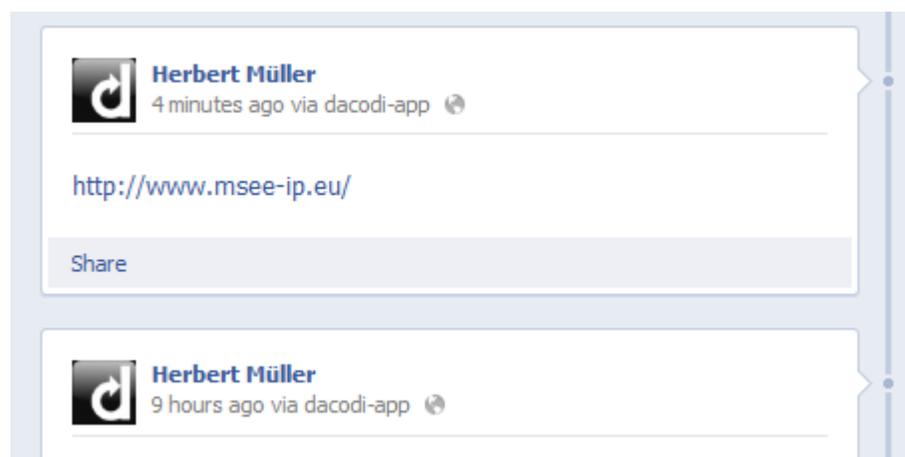


Figure 6 - Result of publishing in Facebook

In the Front-end Web application the user is able to see if the publishing phase was successful and on which channels (see Figure 7).

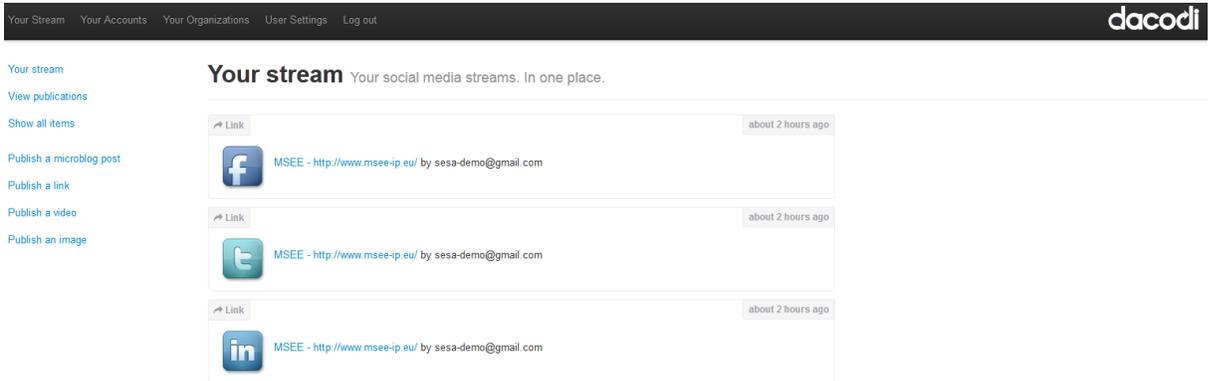


Figure 7 - Stream page containing published items

The user is also able to see the feedback he received regarding its published item in terms of comments, likes, dis-likes, views and reposts (see Figure 8).

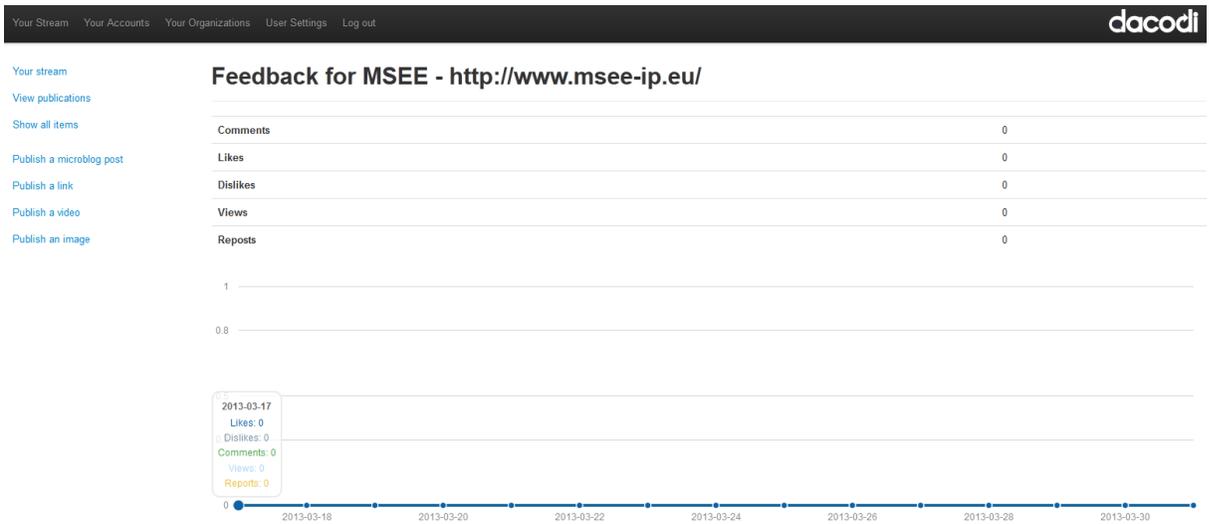


Figure 8 - Feedback for a published item

6 Future Plans

The next steps for the Feedback Management Service will be focused on testing and improving current implementation, adding new channels relevant for the manufacturing enterprises and improving the Front-end UI.