Project Acronym:	**CATALYST**
Project Full Title:	**Collective Applied Intelligence and Analytics for Social Innovation**
Grant Agreement:	**6611188**
Project Duration:	**24 months (Oct. 2013 - Sept. 2015)**

D3.2 Social network interconnection layer

Deliverable Status:	**Final**
File Name:	**CATALYST_ D3.2.pdf**
Due Date:	**May 2014 (M8)**
Submission Date:	**June 2014 (M9)**
Dissemination Level:	**Public**
Task Leader:	**Imagination for People**

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n°6611188

The CATALYST project consortium is composed of:

SO	Sigma Orionis	France
I4P	Imagination for People	France
OU	The Open University	United Kingdom
UZH	University of Zurich	Switzerland
EN	Euclid Network	United Kingdom
CSCP	Collaborating Centre on Sustainable Consumption and Production	Germany
Purpose	Purpose Europe	United Kingdom
Wikitalia	Wikitalia	Italy

The CATALYST project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n°6611188

Revision Control

Version	Author	Date	Status
0.1	Benoit Gregoire (I4P)	June 6, 2014	Draft
1.0	Stéphanie Albiéro (Sigma)	June 11, 2014	Quality check and submission to the EC

The CATALYST project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n°6611188

Executive summary

The present document is a deliverable of the CATALYST project, funded by the European Commission's Directorate-General for Communications Networks, Content & Technology (DG CONNECT), under its 7th EU Framework Program for Research and Technological Development (FP7).

This deliverable is an outcome of Task 3.2. The main goal of this task was to develop the technical framework necessary to create deliberation groups that span more than one forum or social network.

This includes the software necessary for conversations to flow bi-directionally across networks. It also includes performing the necessary format adaptation for the resulting messages to be both technically compatible (ex: HTML or not) and culturally compatible (ex: length, management of hyperlinks) with the destination networks.

The main features include the ability to automatically import posts, comments and messages from other social media platforms; the ability to post back to the original platform from within Assembl; giving posts a permanent URL so they can be referred to across the web; and recomposing posts from a social network into threads, whether the original platform allowed it or not. The system has been tested successfully through email and mailing lists, which is believed to be the hardest case. It is anticipated that further cases are comparatively simpler.

The CATALYST project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n°6611188

Table of Contents

Executive summary 4

Table of Contents 5

1. The Assembl messaging abstraction layer 6

2. High Level Architecture 7

2.1 PostSource class and its derivatives: communications with remote servers 8

2.2 The Post class and its derivatives: managing format differences 9

2.3 The AgentProfile class and its derivatives: management of identities 10

3. Conclusion and Future direction 12

List of Tables and Figures 14

The CATALYST project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n°6611188

1. The Assembl messaging abstraction layer

The Assembl messaging abstraction layer is a core piece of the Assembl architecture. It is designed to enable two-way interactions with any messaging platform or social network that allows chronological discussions between individuals to take place. It presents these messages in a uniform interface within Assembl and will eventually make them available to the other CATALYST tools.

The activities include:

- Automatically importing posts, comments or messages from external social media platforms;
- Posting back to external platforms from within Assembl;
- Giving posts a permanent, stable URL if they do not already have one so they can be referred to across the web and harvested from using other catalyst tools;
- Recomposing posts from social networks (reply relationships) to allow for threading, whether the original platform supports limited threading levels or no threads at all.

The CATALYST project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n°6611188

2. High Level Architecture

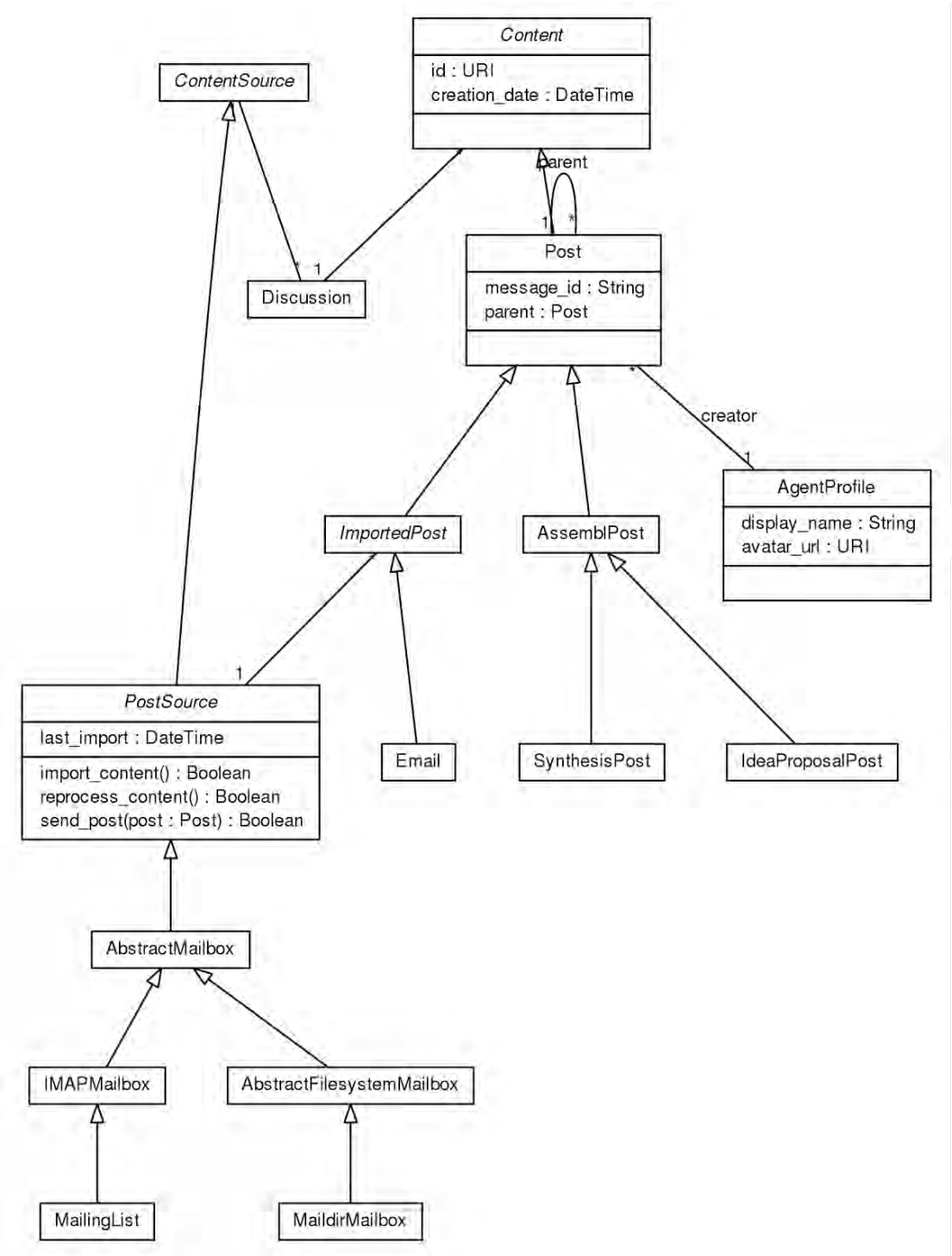

Figure 1: Assembl simplified diagram (processing of posts)

While the Assembl frontend has knowledge of the different types of Posts, dealing with the difference is largely a backend concern.

The CATALYST project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n°6611188

2.1 PostSource class and its derivatives: communications with remote servers

These classes are mainly concerned with supporting the importing of chronological Posts from multiple sources. Their activities include:

Managing the information necessary to make a source available to the discussion

For example, in the case of IMAP servers (that manage emails) the IMAPMailbox class stores the address and directory where the messages can be found, and associates it with a Discussion. A given Discussion can receive Posts from more than one PostSource.

Performing the interactions necessary to retrieve posts from the source server

The class instances for a specific server type need to know how to speak the API or protocol(s) supported by that server. So far the following have been implemented:

- IMAP servers, with or without encryption. IMAP servers are the most common means of accessing emails. Using an IMAP server allows Assembl to access any system and allows users to subscribe to an email to receive all messages.
- Mailing lists. This allows Assembl to be aware of the semantics of a mailing list, namely that every user can be reached through a single email destination and that no duplicate messages should be sent there.
- Mailboxes stored in the maildir format. This allows users to access past discussions for harvesting purposes.
- Messages posted directly on the Assembl web interface

Starting with emails may seem surprising in a "social network interconnection layer", but there are two important reasons to start there.

The first reason is simply that we already have multiple communities using mailing lists. Migrating these communities to Assembl with the least disruption possible is the goal.

The second reason is that mailing lists are clearly the most complex of chronological discussions:

- Every message can have a different encoding
- Every message can have a different natural language
- A single message can have multiple formats (HTML and plain text)
- Messages can have attachments
- Message threads can be unlimited in depth, and the reply relationships can be represented in multiple ways
- There is enormous variability in message length in practice (social network comments are typically more uniform within a single network)
- Messages have a subject separate from their body text

Any abstraction layer that can deal with such variability should easily deal with the simpler and more homogenous messages from social networks (one API, relatively uniform length, very limited HTML support, single format, limited threading level).

Performing efficient pooling of the original server

It is necessary to receive new messages efficiently (placing minimal stress on the involved servers). This means requesting only messages not already imported where the server supports this function, and not pooling too often.

It is also necessary, however to receive messages in a timely fashion (as close to real-time as possible).

The CATALYST project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n°6611188

While not the core subject of this report, it is important to understand that Assembl maintains a real-time socket to all of its open clients on the web, allowing it to transmit real-time updates, in a manner similar to Google docs (one can see what changes other users have made without reloading the page). This includes messages.
To reach this performance requirement, there are two situations must be avoided:

- We cannot trigger a verification of every source every time a user load the interface. Not only would this create undue stress on the remote server (possibly leading to blacklisting of the Catalyst tool), but if there are indeed messages to process, it would delay the response to the user (importing many messages may take several seconds).
- We cannot wait for someone to load the interface to import new messages, as this would cause everyone to receive messages after a possibly large delay, and all in a large batch instead of continuously.

To solve these two problems, the task of managing the import operations of the PostSources is implemented using a Celery distributed task queue (http://www.celeryproject.org/). When a message is imported, the source subsequently communicates with a 0MQ (http://zeromq.org/) message queue, which notifies the clients (the end user's browsers) immediately.

Sending a message or notification back to the source

It is important to be able to either post a message back to the original conversation (not always possible, for example Facebook does not allow more than one level of replies), or send a personal notification. The most common use case is when a user responds to a message originating from a different system. We want the original author to be notified.

2.2 The Post class and its derivatives: managing format differences

These classes are mostly responsible for managing the differences in format and capabilities of the different Post types.

Making the format adaptations required to represent the post in a uniform format.

This includes:

1. Converting character encoding
2. Choosing between formats if there are either multiple versions of the base format, or even multiple representations sent at the same time (example: a single email can contain both a plain text representation, and a HTML one)

Avoiding duplicates

The type of harvesting Assembl performs could very easily create infinite loops, in which Assembl retrieves message A on server 1, sends a reply as message B back to server 1, and then receives message B from server 1 (which is expected). If Assembl does not detect that the message is the same the user sent, despite any format shifting that may have occurred, there will be a duplicate message posted to the conversation.
Worse, if Assembl is configured to mirror every message on every server, an infinite loop is created.

Inferring response relationships

Assembl can recompose the message thread (who replied to who), in order to use it to display a threaded view (the only efficient view for harvesting, without it, it is very difficult to sort out who is replying to who):

The CATALYST project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n°6611188

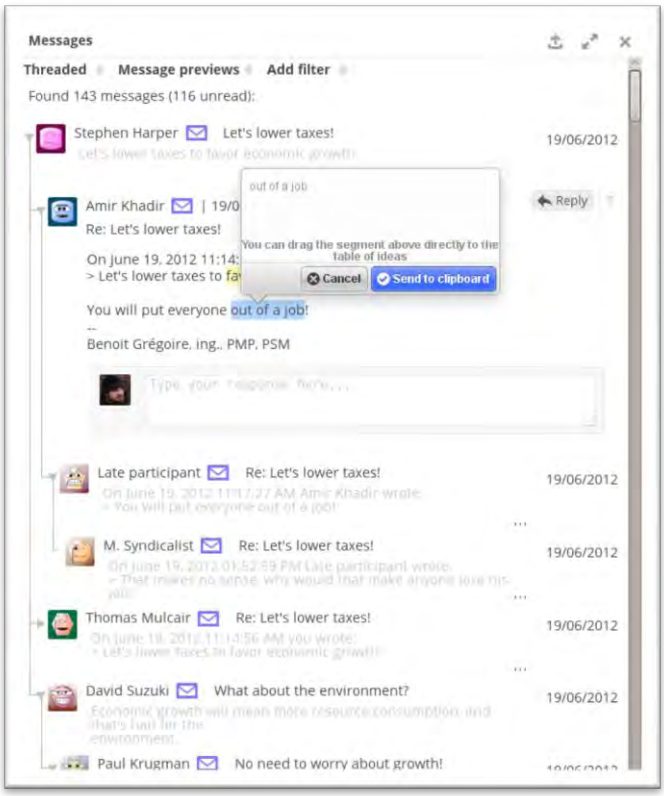

Figure 2: Threaded message view in Assembl

In the future we will use this function to generate notifications of replies across networks (sometimes this will be the only interaction allowed by social networks).

2.3 The AgentProfile class and its derivatives: management of identities

Having unique identities is a difficult problem and has become a practical concern for Assembl. The program needs to recognise that a person logged in to Assembl is the same person no matter if he signs up with a Google account, through Facebook, or via an email.

In practice this requires architecting the system is such a way that a user can claim and verify multiple accounts (such as a email account), and have them linked to a unique profile. This is rare, but not unique:

The CATALYST project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n°6611188

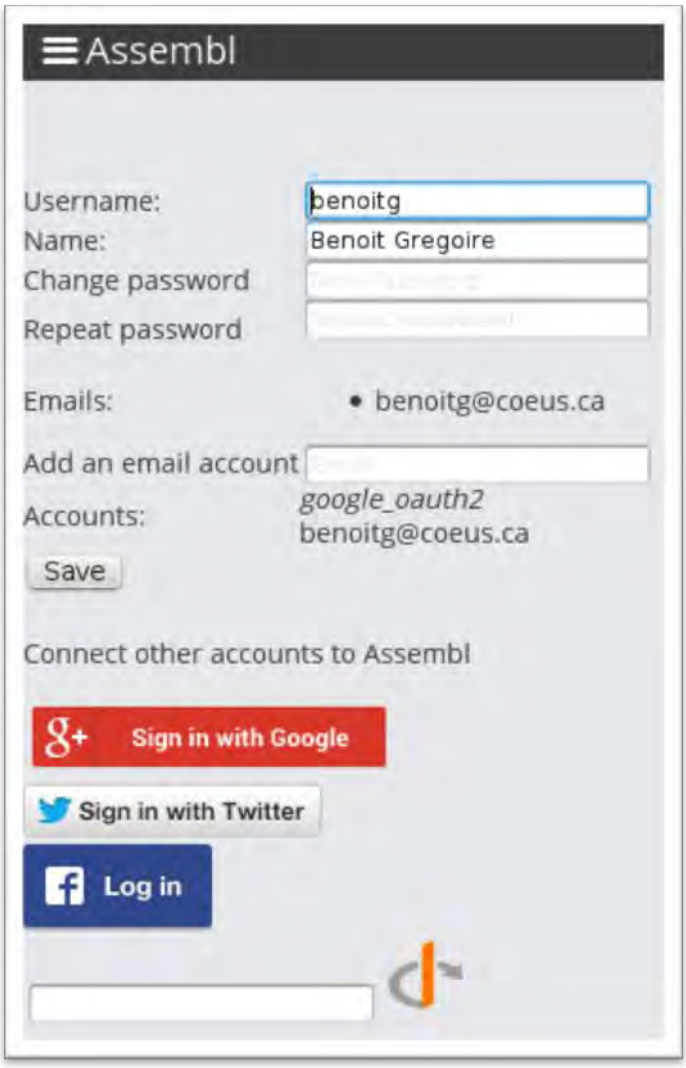

Figure 3: Account merging

What is unique is that Assembl can do this merge AFTER the first messages have been received. If a user adds an email account, any message sent previously from that account will be shown as belonging to the user.

The CATALYST project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n°6611188

3. Conclusion and Future direction

As outlined in task 3.2, we started with writing the abstractions necessary to make messages flow bi-directionally. This work has been successful.

- Participants can post messages via email, and respond through the web and vice-versa
- Messages flow in real time (Participants on the web see messages arrive without reloading their page).
- Messages (even those submitted via) receive a URL and can be referred to on the Web.
- Threading and response relationships are kept despite network shifting (ex: someone responding by email to a message from the web and vice-versa).
- This technical framework has been tested through connections with mailing lists.
    - This enables users to participate in the debate through an interface they are already comfortable with (Gmail for example) if they do not wish to migrate to Assembl itself. Existing communities who may be using email already are able to keep the same habits but at the same time, thus increasing the impact of their contributions.

As planned in task 3.2 the transformation work is still in its infancy at this stage. So far:

- Assembl can transform a plain text message in a representation suitable for the web (dealing with line-breaks, and other plain text formatting)
- Assembl can sanitise HTML-only emails to display them in the interface.
- Assembl can send a web message back to a mailing list.

The features outlined in this document are for the most part difficult to show visually. Most of the code involved in the functionalities outlined in this document can be found in the assembl/assembl/models directory.

Development on Assembl is completely open, and activity can be followed here:

- Code browser: https://github.com/ImaginationForPeople/assembl
- Commit history: https://github.com/ImaginationForPeople/assembl/commits/develop

While not usable to demonstrate most of the functionalities in this document (a connection to external network is required), a sandbox is generally available at the following address: http://assembl.coeus.ca/sandbox/ to play with the interface (anyone can create an account).

- If you open an account, you can see the real-time update feature in action by opening a second browser window on the discussion, and moving an idea or posting a message.

Future directions

Our original hypothesis that platform balkanization is a problem has not been invalidated in any way, and receiving all messages to organise in a coherent way in Assembl does seem to work.

We already learned one lesson however: even in the case of a mailing list, re-sending all messages and syntheses to everyone on the list is not as useful as anticipated:

- On the one hand, users of one forum do not necessarily want to receive a message or notification every time something is posted on another forum on the same general subject. The number of messages can differ widely, and the user may not yet know the participants.
- On the other hand, users DO want to receive a notification when somebody directly responds to them, regardless of where the response comes from.
- Assembl has the potential to allow users to follow (receive notifications of all replies to a given message) on more specialised sub-topics within the discussion. This may allow for the building of a critical mass that is necessary for collective intelligence to emerge on the given subject.

The CATALYST project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n°6611188

Answering these opposing needs requires much more sophisticated and configurable notification logic. For the rest of this task, we plan to spend more time on this aspect than planned, and less on supporting actual mirroring between the many social networks.

For example, for Facebook (the elephant in the room of social network interconnection), it has proven surprisingly difficult to find a compelling use case for bi-directional mirroring. On the other hand, allowing a single synthesis, visualisation or message to be shared on Facebook, and then having the responses to that post on Facebook be imported back in Assembl may be a more effective group building tool than simple social sharing. This process would require notifications for users if other participants react to the post.

In the immediate future, this task will focus on additional work on message transformation, to improve ergonomy in the web interface and make the outputs of the other tasks (visualisations, syntheses) available in mail notifications.

The CATALYST project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n°6611188

List of Tables and Figures

Figure 1: Assembl simplified diagram (processing of posts) 7
Figure 2: Threaded message view in Assembl 10
Figure 3: Account merging 11

The CATALYST project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n°6611188