Email Deposit

Introduction

This is the page with technical specs for the development of email deposit, an entry in the Open Repositories 2015 ideas challenge. While this page describes technical considerations, it links out to different use cases where the end user needs are being described.

Goal

The goal of all email deposit related features and functionality, is to drastically lower the effort to contribute bitstreams (files, publications, ...) to the repository, by leveraging email as the medium for main and single medium of interaction with the provider of the file.

Use Cases

For now, there are three different use cases that are being tracked, which will all need/rely on a basic infrastructure that can receive emails, extract metadata and/or files from them, and feed that information into the repository.

End User - Email Deposit of bitstream for automatically captured metadata

End User - Email Deposit of bitstream for archived metadata-only item

End User - Deposit entire item by email

Design challenges

Before we can start coding on this, there are a number of design challenges that should be addressed.

Source for email recipient addresses

In use cases

End User - Email Deposit of bitstream for automatically captured metadata

End User - Email Deposit of bitstream for archived metadata-only item

it is assumed that email addresses can be referred from or extracted from item metadata.

In an initial implementation, we may have to go with a single approach, that could be further extended on later on.

The DSpace 5 ORCID integration comes with an authority cache, where additional information can be stored about specific metadata values, in this case, the author. We could assume for the initial implementation that email address will be a field in the authority cache.

This challenge could also be solved on the level of a specific institution, matching values for author names on already existing items with the institutions staff registry, and getting the email addresses from there.

Security

There are a number of security concerns with the idea of automated, email based deposit flows. Those concerns that can't be addressed or solved should become very clear disclaimers in future documentation.

Authenticity of deposit

Anyone who sends an email can claim any email address as the "sent from" value. There are server side methods (SPF - domain level) and sender side methods (signing emails with pgp keys) that want to solve this for email in general. However, the current state of the art is that the vast majority of emails sent today are still without that kind of infrastructure.

What this means for this proposal is that when an email comes in from charles.xavier@myinstitution.edu, we may not be able to trust completely that this email is really sent by this person.

Unencrypted transfer

The proposed functionality can potentially not apply for the deposit of sensitive/confidential files, as attaching the files to an email and sending them is unencrypted. While this is not a problem for files that are meant to be open access anyway, we should not be recommending an email based deposit approach for sensitive/confidential files.

Develop as an add-on or as a DSpace feature

On the long term, it would be very neat to have this entire codebase/tool as an independent add-on, because it could make it easier for implementers in other repository systems to copy the code/approach, as opposed to dig into the DSpace core code in order to find the code for these features.

However on the short term the current DSpace REST-API may not be able support all of the required actions + the problem below with on-behalf of submission might be tricky to do from an external add-on.

Resolve conflicts with current DSpace logic

On behalf-of submission: EPerson vs Author in metadata

In DSpace, all submitted items have to be associated with an EPerson, which is the DSpace "account" object. A challenge arises in the fact that a certain author may not yet have a corresponding EPerson object in DSpace, at the point when the functionality finds his email address. This is the first challenge.

A second challenge comes up when professor Charles Xavier already has an EPerson in DSpace. Even though we could neatly link his EPerson to the incoming email, the Email Deposit functionality can't authenticate as that EPerson because Charles has no way (yet) for providing his approval to email deposit to create items on his behalf.

We should look at how the SWORD integration deals with this: the SWORD2 protocol supports on-behalf-of deposits, but it's unclear at this point how DSpace deals with this.

This is also linked with the aforementioned security problem where we can't guarantee authenticity of deposits by merely looking at the sender's email address.

A target collection needs to be set right at the initiation of a submission

DSpace allows different collections to set different rules on mandatory/optional fields etc for each collection. This is a key reason why collection selection needs to happen right at the start and can't be changed later on.

If an entirely new item is being deposited via email, we need to be able to set a selected collection at the start. One way to tackle this is that the collection to deposit in is a config parameter, either system wide (all incoming email deposits enter the same collection) or in the profile/EPerson of the depositing user: as a user, you define what your collection of choice is where email deposits enter.

Parsing target collection names or IDs from the emails itself would be a more challenging approach.

When you create a deposit, DSpace assumes you have agreed with the collection's license

This is more of an "inform the user" or policy challenge than a real technical one. In the default DSpace submission workflow, the user needs to agree to the collection's license in the last step.

For the usecase End User - Email Deposit of bitstream for archived metadata-only item, someone already accepted the license when the metadata only record was created originally, and the submitted files will become subject to that license as well.

For End User - Email Deposit of bitstream for automatically captured metadata it's less obvious, because the item to which the file will be attached will likely be in the workflow, and the license still has to be granted.

Again, the easiest and most straight forward way around it would be to inform the user that by sending his file to for email deposit, he/she implicitly accepts the terms of the license. We should take proper care of the wording for this in the specific emails that invite people to deposit.