

Extending the API Key concept

One capability that API Keys exposed, though it was probably unintentional, was sharing files with others who do not have a Clowder ID. This can be accomplished by using the `/api/files/{fileid}/blob?key=[apikey]` pattern. While this is really tempting to use, this creates a significant security risk. The ability of being able share (send a URL) files with non-Clowder users is actually pretty compelling. It basically allows a limited drop-box type functionality.

My proposal is to embrace this concept while mitigating some of the risks. One way to accomplish this is to have the concept of "Access Keys" (not sold on the name) that can be created and associated to a Dataset or File object. When you create the relationship between the Access Key and the Object you would specify an expiration date or number of uses. Access Keys can ONLY be used to download files and can not be used in any other way (this could require a special API to be created to support their specific use). Once an Access Key expires or is been used the specified amount of times it disabled itself (or deletes itself?). The security works basically the same as API Keys, in that for the object...you inherit the rights of the issuer for that object. Since it is for download only, edit and delete are not a concern.

If more security is needed, a password could be added that the user would be challenged with and have to provide. I am not a huge fan of this simply because in my experience people typically send the link and password in the same email which defeats the purpose.

Implementation considerations

1. API Keys could borrow some of the features being proposed, such as being tied to a specific object(s) and an expiration date.
2. A UI link could be exposed on objects to "Create shareable link" that pops up a dialog to ask expiration date/usage count and stuff the link in the paste buffer, making it really easy for users.
3. Profiles could list both API Keys and Access Keys and allow them to be managed together.