Multiple identities in thread design doc

Objective

Moderators sometimes need to take action in their official capacity on a thread where they already inhabit an anonymous identity. Currently the only way to do this is for the moderator to reveal themselves as such using their current identity, thus deanonymizing themselves. Instead, we want them to be able to post as their “mod role” identity regardless of other assigned identities.

Design goals

- We don’t want *everyone* to be able to post as multiple identities in a thread.
- Should “moderators” be able to post as multiple identities with any “identity role” they have or just with specific roles?
    - Similar to how specific roles have a “post_as” permission, they should have a “override_identity” permission (name TBD) which allows role-havers to use that role even on threads where they already have an identity.
- Moderators should be able to switch back and forth between anonymous and role identities.
- “Moderators” should be able to assume a random identity after they already used their moderator identity in a thread
    - We need to check whether all identities they currently have assigned have the overridable permission
- We don’t want to allow multiple *random* identities as part of this design
- If a moderator has multiple roles that can override identities, then the moderator should be able to post as each of those identities independently

    - For example, if someone is both “admin” and “mod” they would be able to switch between the two

- Can anyone see that someone is occupying two identities within the same thread?
    - This could be useful for audit purposes in case the multi-identity permission is being used “maliciously”
    - The admin can know this is happening from the database
    - We’re not going to have any surfacing of this information right now
- What about multiple people having the same role? Can they post as the same identity?
    - This is currently possible (I think), but whether its allowed is independent from whether the identity can be used when one is already assigned
    - If we want to change this, we should do it as its own design

Types of identities after changes

After these changes we’ll have 3 types of identities:

1. Random identities
2. Non-overriding role identities
3. Overriding role identities

Design

Current design

- The identity of a user in a thread is stored in the **user_thread_identities** table
- Queries that return threads, contributions or comments join with that table and surfaces the identity of the author as a **secret_identity**

    - This search returns the places where the **user_thread_identities** table is being used.

- Currently, the getThreadDetails function will check if the user has an identity in the thread, and whether the user is requesting to post as a role. If the user does not have an identity in the thread, it will either assign them the requested role or a random one if no role is requested
    - It would be nice to take this change to clean up getThreadDetails because it's written in a very confusing way.
- Requests to create a post/comment/thread with a fixed identity will have the identity_id parameter set to the external id of the role. Requests without the identity_id are considered as either random generation (if no identity for the user was specified) or "same as before" (if there is already an identity).
    - We won't need to modify these endpoints because their signature will work even with multiple identities. We may need to change the semantic though, because we need to specify the identity we want if there are multiple in the thread.
    - Currently the frontend doesn't show the roles dropdown when an identity is already assigned.
- When fetching thread or feed data, the identity of the author of every post/comment is set in the `secret_identity` field.
    - This doesn't need any update. The only thing that will potentially change is that there may be posts marked as "own" (so the current user's) that have different identities.
- When the user goes to create a post/comment/thread we look at the `posting_identities` array returned by the `boards/:board_id/` endpoint, and (if the user doesn't already have an identity associated in the thread), we display the additional identities in the identity selector.
    - We will need to indicate which identities have the ability to override an already existing one.

Changes to database/sql queries

- We maintain the **user_thread_identities** table as is. The first time a user posts in a thread, that's where their identity is registered.
- If we get an override request, then the identity will be written to a **user_thread_identity_overrides** table, which will have the following schema:

  ```
  user_thread_identity_overrides {
    id,
    post_id,
    comment_id,
    identity_id,
    role_id,
    CHECK (post_id is not null or comment_id is not null),
    CHECK (identity_id is not null or role_id is not null),
  }
  ```

  - We should check if CHECK has a "xor" option
- All the SQL queries that use **user_thread_identities** will need to change to fetch the identity of individual post/comments if it exists
- We will need to add a **can_override_identity_as** permission similar to **post_as** permission for role
- We will need to change the bobadex stats queries to also check for identities assigned in **user_thread_identity_overrides**

Changes to server

- Make `getThreadDetails` check whether the user already has an identity associated with the thread, and in that case rather than blocking the creation of a new identity, check if all the identities associated with the user have the **can_override_identity_as** permission, and only in that case allow an additional identity.
  - Overriding roles can always be assigned

    - If the user already has one or more identities and they're all overriding, they can always be assigned the new identity
    - If the user already has one or more identities and one of them is not overriding, they cannot be assigned an additional identity (unless that identity is overriding)
    - **In short:** a user in a thread can have 0 or more *overriding* identities, and 0 or 1 *non-overriding* identities
- Make sure that the posting endpoints always interpret a request without a specific identity specified as getting a new random identity (or reusing the same). To post as a role, the user will *always* have to specify that specific role.
    - **Note:** it would be nice to always send up the id of an identity that is already assigned, even if that identity is a random one rather than a role one. Unfortunately, we aren't currently returning these IDs as part of our thread data. Once we do that, we should make this change.

Changes to API endpoints

- When we get the posting identities from the board metadata, we should know which ones are overridable by adding a **can_ovveride** property.

Changes to frontend

- Whenever the user has already a role identity assigned in a thread make sure to always send it up to the server with the request, or it will be interpreted as "create new random identity" if there's no requested identity associated.
- If there is an identity already assigned to the thread, still display an option to select all the identities with the **can_override** property
- In places where **user_identity** is surfaced for threads (like here), that must become an array.

Changes to components

- (stretch) In the identities selector dropdown, it would be nice to mark whether an identity has already been used in the thread or if the user would be using that in that thread for the first time.