Key actions should all be “secured” using a generic mechanism to ensure we can easily audit the access and prevent any data or confidentiality breach.

Concept:

* All actions have a key
* Actions can be grouped
* Individual actions or groups are granted permissions according to a permission table
* Permissions are granted on a role level (sponsor, brand, partner, team or agent)

For example, a permission table could look like:

* GUID
* permission\_key
  + shall either refer to a unique action in the frontend or to a group;
  + the key shall be meaningful (text) and follow some naming conventions, for example
    - ACTION\_READ\_XXX
    - ACTION\_UPDATE\_XXX
    - ACTION\_DELETE\_XXX
    - GROUP\_UPDATE\_XXX
* permission\_admin: allow | block | null
* permission\_sponsor: allow | block | null
* permission\_brand: allow | block | null
* permission\_partner: allow | block | null
* permission\_team: allow | block | null
* permission\_agent: allow | block | null
* creation\_date (auto)

And the permission\_groups table could look like

* ID
* action\_key
* group\_key
* creation\_date (auto)

In case any permission is “block”, then action should be prevented. An action may result in several entries in the permission table in case the action is in 1 or more groups. Ex:

* ACTION\_DELETE\_ABC -> sposor = null
* GROUP\_SPONSOR\_DELETE\_TOPIC1 -> sposor = block (with ACTION\_DELETE\_ABC being part of this group)
* GROUP\_SPONSOR\_GENERIC\_TOPIC2 -> sposor = allow (with ACTION\_DELETE\_ABC being part of this group)
* BLOCK

There should be an independent module that handles the permission lookup to ensure the logic is centralized and can be updated in a single place. In other words, no specific logic shall be coded at action level in the frontend, every action shall call the same and central lookup method, with the action key as parameter (the role should also not be hardcoded be clearly be a variable)

Actions \*not\* allowed should also not be shown in the frontend