User permissions

After user is authenticated with success and got the information about the particular HelixTrack Core instance that it belongs to, permissions information is obtained from that HelixTrack Core instance and returned as the part of JWT token's payload.

Each user will have a list of permissions. The following example illustrates regular user with its permissions:

```
{
  "permissions": [
    {
      "permission_id":         "string",
      "permission_context_id": "string"
    }
  ]
}
```

Permission ID

Permission IDs are connected to the one of the following permissions (with each the proper access level numeric value is associated):

- `READ` : Allowed reading of the context, access level = 1
- `CREATE`: Allowed insertion into the context, access level = 2
- `UPDATE`: Allowed modification of the context, access level = 3
- `DELETE`: Allowed removal of the context, access level = 5
- `ALL` : Allowed to perform all operations on the context access level = 5

Note: The `DELETE` and `ALL` permissions are the same one with the different naming.

Permission context ID

Permission context IDs are connected to the one of the following contexts:

- `node`: Access to all nodes
- `node.NODE_ID`: Access to the node
- `system_info`: Access to the system information for the particular node
- `extension` Access to the system's extensions
- `audit` Access to the system's audit data
- `reports` Access to the system's reports
- `account`: Access to the accounts
- `account.ACCOUNT_ID`: Access to the account
- `extension.account.ACCOUNT_ID` Access to the accounts's extensions
- `audit.account.ACCOUNT_ID` Access to the accounts's audit data
- `reports.account.ACCOUNT_ID` Access to the account's reports
- `organization`: Access to the organizations (requires access to the account)

- `organization.ORGANIZATION_ID`: Access to the organization (requires access to the account)
- `extension.organization.ORGANIZATION_ID` Access to the organization's extensions
- `audit.organization.ORGANIZATION_ID` Access to the organization's audit data
- `reports.organization.ORGANIZATION_ID` Access to the organization's reports
- `team`: Access to the teams (requires access to the organization)
- `team.TEAM_ID`: Access to the team (requires access to the organization)
- `project`: Access to the projects (requires access to the organization)
- `project.PROJECT_ID`: Access to the project (requires access to the organization)
- `extension.project.PROJECT_ID` Access to the project's extensions
- `audit.project.PROJECT_ID` Access to the project's audit data
- `reports.project.PROJECT_ID` Access to the project's reports

Permission contexts hierarchy

- `node`
    - `node.NODE_ID`
        - `system_info`
        - `extension`
        - `audit`
        - `reports`
        - `account`
            - `account.ACCOUNT_ID`
                - `extension.account.ACCOUNT_ID`
                - `audit.account.ACCOUNT_ID`
                - `reports.account.ACCOUNT_ID`
                - `organization`
                    - `organization.ORGANIZATION_ID`
                        - `extension.organization.ORGANIZATION_ID`
                        - `audit.organization.ORGANIZATION_ID`
                        - `reports.organization.ORGANIZATION_ID`
                        - `team`
                            - `team.TEAM_ID`
                        - `project`
                            - `project.PROJECT_ID`
                                - `extension.project.PROJECT_ID`
                                - `audit.project.PROJECT_ID`
                                - `reports.project.PROJECT_ID`

How do the user permissions work?

For each context where we want to perform certain operation we will verify if that operation is possible to perform by evaluating the following rules:

- Do I have access to the context? If we have access to the context or to a parent context (higher in the hierarchy) the access is granted.
    - No, reject.
    - Yes, lets go to the next check step.

- Do I have propper permission access level? Each operation that we want to execute requires certain level. Let's say that we want to read the content of the context. We need level >= 1. User has the level of 2 (creation granted). That means that it is allowed to read as well.
    - No, reject.
    - Yes, perform the desired operation.

Permissions engine

Each core operation must be verified against the permissions engine. For example, executing the operation for obtaining the list of projects:

- Method start
- Obtain the list of permissions
- Request the approval from the permissions engine
- Obtaining the result: success or failure
- If success, execute and return the result
- If failure, abort the executiom.

Evaluating the permissions

To evaluate the permission for the operation the name of the system entity, access level and the JWT token is provided:

```
{
  "entity": "project",
  "access_level": 1,
  "jwt": "jwt_token_value"
}
```

Each system entity is mapped to the proper permission context. For example, the `project` system entity is mapped to the `project` permissions context. The permissions engine has the information about all system mappings. If the evaluation completes with success the proper payload is returened:

```
{
  "code":                   -1,
  "errorMessage":           "string",
  "errorMessageLocalised":  "string"
}
```

For success the code with value of `0` is returned.