Summary notes for ABC API Authorization Design

This document is related to TDP task #XYZ.

ABC API usage

- The ABC API will be accessed within the XY network only.
- The ABC API is intended to be used as “Point to Point” by ABC web application.

ABC API Security scope

Based on the received information, **authentication** is not in our scope and we should focus only on SOAP and REST requests **authorization** for this particular task.

Requests/headers/fields encryption is not expected to be delivered within the scope of the sprint XY.

Available approaches overview

There are available following main authorization check approaches:

- Programmatically - Authorization checks in code with structures like if/else blocks.
    - An easy and flexible way which would extend existing implementation.
- Annotations - Authorization annotations can be attached to controller methods.
    - Annotations can be checked via related Interceptors. In our case specific interceptors for both SOAP and REST Controllers would need to be configured.
    - Methods and Permissions identification would be needed based on javax.servlet.http.HttpServletRequest and javax.interceptor.InvocationContext classes.
    - In case of authorization fail, the response can be provided directly to javax.servlet.http.HttpServletResponse
    - Interceptors must be properly invoked for all API requests – path configuration needed especially for REST requests.
- Existing frameworks like Apache Shiro or Okta.
    - They offer mature solutions which require just additional roles and permissions configuration and mapping of framework authorization exceptions into our result status.

Selected approach for implementation

Based on requirements, current implementation and team agreement there will be delivered following solution:

- Existing Task and Category related permissions defined in existing DB table user_permission will be applied and checked.
    - Permissions are based on Task and Category resources combination with related create/update/view operations.
- The ABC API specific records will be created in existing user_group, user_role tables.
- Authorization check will be done programmatically and explicitly in the ABCService class, which provides a common implementation for both SOAP and REST controllers.

Implementation

Interface

- ABCAuthorization#hasPermission(userGroup, String requiredPermission)
- ABCAuthorization#hasPermission(userGroup, String… requiredPermissions)

SQL scripts and new DB records

1. Authorization check query:

```
select count(up.permission_id) from user_permission up
  JOIN user_role_permission urp on up.permission_id = urp.permission_id
  JOIN user_role ur on ur.role_id = urp.role_id
  JOIN user_group_role ugr on ugr.role_id = ur.role_id
  JOIN user_group ug on ug.group_id = ugr.group_id
  where ug.group_name = :groupName and up.permission_id in (permission1, …, permissionN)
```

2. New groups, roles and permissions mapping will be defined in following files in XY branch:

    …

    2.1 New user_role and user_group records:

        - ABC_api_full_access_role and ABC_api_full_access_group

UserGroup propagation

UserGroup value needed for authorization check will be propagated following way:

- REST API requests: Will be added HTTP header: Client-User-Group
- SOAP API requests: WSHeader type will be extended by WSClientUserGroup element.

Authorization process and response

An operation will be allowed in a case that checked userGroup has available required permission(s), otherwise “Forbidden” status is returned based on API_v1_statusCodes_contract document.

API methods, permissions and identified Services for authorization check

Following methods and permissions will be subject to authorization check.

Authorization check will be implemented on Controller level and based on the latest agreement, all required permissions need to be assigned to specific userGroup in order to be such userGroup authorized to perform particular API method.

Methods:	SOAP – createTasks	REST – POST /tasks
Required permission(s):	createTask	

Methods:	SOAP – updateTasks	REST – PUT /tasks
Required permission(s):	updateTask	

Methods:	SOAP – updateCategories	REST – PUT /categories
Required permission(s):	updateCategory	

Methods:	SOAP – getCategory, searchCategories	REST – GET /categories/categoryName, POST /search/categories
Required permission(s):	viewCategory	

Methods:	SOAP – getTask, searchTasks	REST – GET /tasks/taskNumber, POST /search/tasks,
Required permission(s):	viewTask	

Methods:	SOAP – updateState	REST – POST /state
Required permission(s):	updateCategory, updateTask	