Unit 4

Security

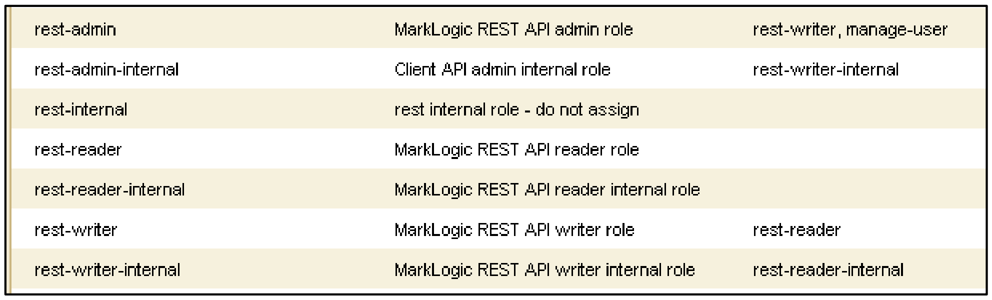
Create Users and Roles

Test Document Permissions

Exercise 1: Create Users and Roles

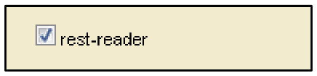
In this exercise we will use the MarkLogic Admin interface to view the configuration of the pre-defined roles used by the REST API (and therefore the Node.js API). We will then create users and assign them the appropriate roles. These users will be utilized as we write our application code in upcoming exercises.

1. Open a browser and navigate to **http://localhost:8001**
2. On the left hand navigation column, click **Security→Roles**.
3. Scroll down to find the roles associated with the REST API:



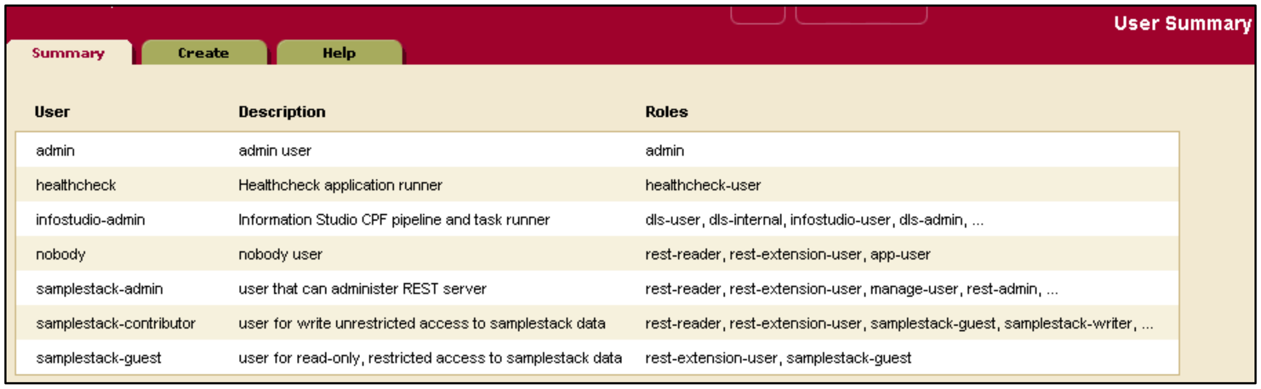
rest-admin	MarkLogic REST API admin role	rest-writer, manage-user
rest-admin-internal	Client API admin internal role	rest-writer-internal
rest-internal	rest internal role - do not assign	
rest-reader	MarkLogic REST API reader role	
rest-reader-internal	MarkLogic REST API reader internal role	
rest-writer	MarkLogic REST API writer role	rest-reader
rest-writer-internal	MarkLogic REST API writer internal role	rest-reader-internal

4. Click on the role named "**rest-writer**".
5. View the definition of the rest-writer role.
6. Notice that the **rest-writer** role is configured to inherit the **rest-reader** role:



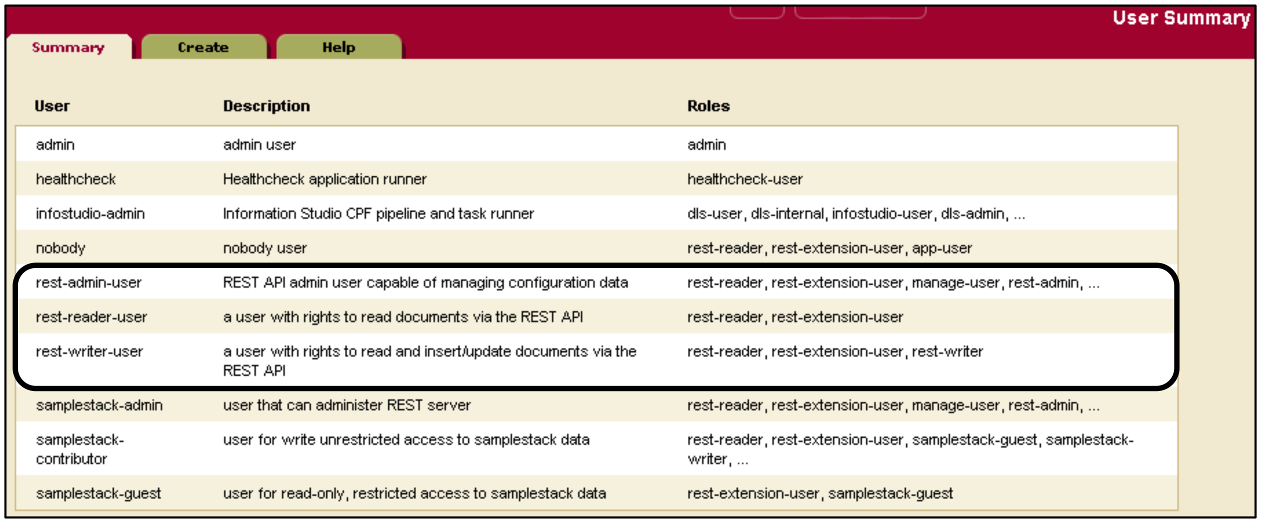
✓ rest-reader

7. This is an example of the hierarchical nature of roles. Any user assigned the **rest-writer** role will automatically inherit the permissions defined in the **rest-reader** role.
8. On the left hand navigation column, click **Security→Users**.
9. Notice the users that currently exist at this point in time:



User	Description	Roles
admin	admin user	admin
healthcheck	Healthcheck application runner	healthcheck-user
infostudio-admin	Information Studio CPF pipeline and task runner	dls-user, dls-internal, infostudio-user, dls-admin, ...
nobody	nobody user	rest-reader, rest-extension-user, app-user
samplestack-admin	user that can administer REST server	rest-reader, rest-extension-user, manage-user, rest-admin, ...
samplestack-contributor	user for write unrestricted access to samplestack data	rest-reader, rest-extension-user, samplestack-guest, samplestack-writer, ...
samplestack-guest	user for read-only, restricted access to samplestack data	rest-extension-user, samplestack-guest

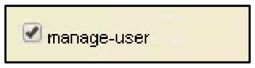
10. Our objective is to create three new users for our application. When completed, our users should be defined as shown:



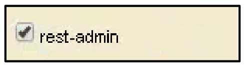
User	Description	Roles
admin	admin user	admin
healthcheck	Healthcheck application runner	healthcheck-user
infostudio-admin	Information Studio CPF pipeline and task runner	dls-user, dls-internal, infostudio-user, dls-admin, ...
nobody	nobody user	rest-reader, rest-extension-user, app-user
rest-admin-user	REST API admin user capable of managing configuration data	rest-reader, rest-extension-user, manage-user, rest-admin, ...
rest-reader-user	a user with rights to read documents via the REST API	rest-reader, rest-extension-user
rest-writer-user	a user with rights to read and insert/update documents via the REST API	rest-reader, rest-extension-user, rest-writer
samplestack-admin	user that can administer REST server	rest-reader, rest-extension-user, manage-user, rest-admin, ...
samplestack-contributor	user for write unrestricted access to samplestack data	rest-reader, rest-extension-user, samplestack-guest, samplestack-writer, ...
samplestack-guest	user for read-only, restricted access to samplestack data	rest-extension-user, samplestack-guest

11. We will now create the **rest-admin-user**.
12. From the Administration tool, click **Configure→Security→Users→Create** tab.
13. Provide the following user definition:
    a. Username = rest-admin-user
    b. Description = a concise logical description of the user. For example:
        i. REST API admin user capable of modifying the REST instance configuration
    c. Password = training
        i. Please keep the password all lower-case so we are all on the same page!
    d. Check the following roles. This will setup the user to inherit the permissions defined in each of these roles:

i. manage-user


☑ manage-user

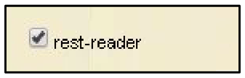
ii. rest-admin


☑ rest-admin

iii. rest-writer (inherited automatically when you assign the rest-admin role)

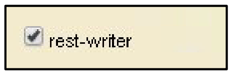
iv. rest-reader (inherited automatically when you assign the rest-writer role)

14. Click **OK** when finished defining this user.
15. We will now create the **rest-reader-user**.
16. From the Administration tool, click **Configure→Security→Users→Create** tab.
17. Provide the following user definition:
    a. Username = rest-reader-user
    b. Description = a user with rights to read documents via the REST API
    c. Password = training
        i. Please keep the password all lower-case so we are all on the same page!
    d. This user should inherit the permissions defined in each of these roles:
        i. rest-reader


☑ rest-reader

18. Click **OK** when finished defining this user.
19. We will now create the **rest-writer-user**.
20. From the Administration tool, click **Configure→Security→Users→Create** tab.
21. Provide the following user definition:
    a. Username = rest-writer-user
    b. Description = a user with rights to read and insert/update documents via the REST API
    c. Password = training
        i. Please keep the password all lower-case so we are all on the same page!
    d. This user should inherit the permissions defined in each of these roles:

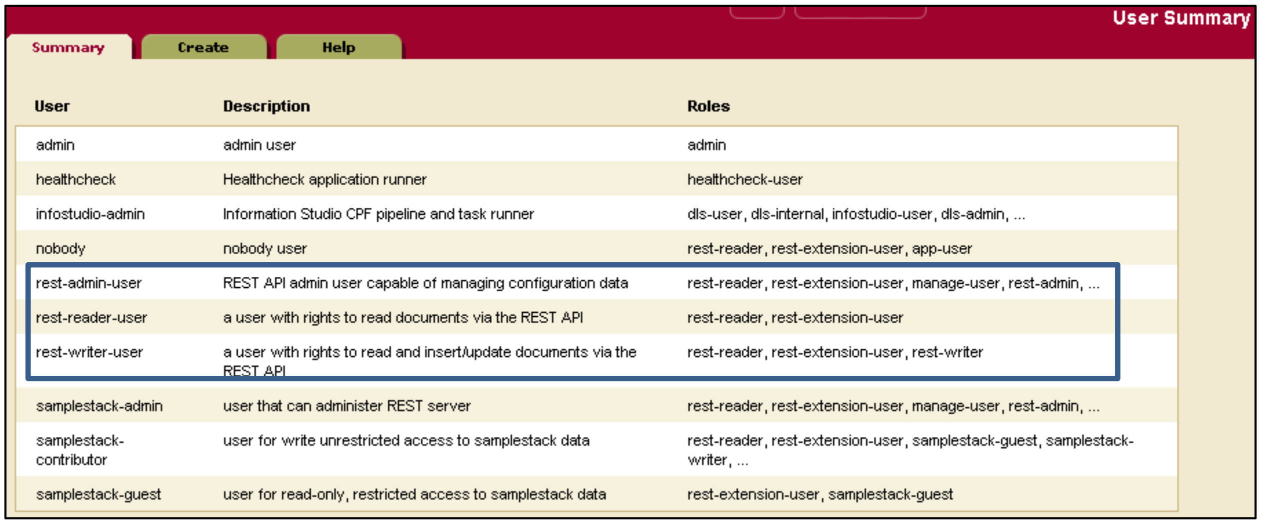
i. rest-writer



ii. rest-reader (inherited automatically when you assign the rest-writer role)

22. Click **OK** when finished defining this user.

23. Validate that your user summary is configured as shown:



User	Description	Roles
admin	admin user	admin
healthcheck	Healthcheck application runner	healthcheck-user
infostudio-admin	Information Studio CPF pipeline and task runner	dls-user, dls-internal, infostudio-user, dls-admin, ...
nobody	nobody user	rest-reader, rest-extension-user, app-user
rest-admin-user	REST API admin user capable of managing configuration data	rest-reader, rest-extension-user, manage-user, rest-admin, ...
rest-reader-user	a user with rights to read documents via the REST API	rest-reader, rest-extension-user
rest-writer-user	a user with rights to read and insert/update documents via the REST API	rest-reader, rest-extension-user, rest-writer
samplestack-admin	user that can administer REST server	rest-reader, rest-extension-user, manage-user, rest-admin, ...
samplestack-contributor	user for write unrestricted access to samplestack data	rest-reader, rest-extension-user, samplestack-guest, samplestack-writer, ...
samplestack-guest	user for read-only, restricted access to samplestack data	rest-extension-user, samplestack-guest

Exercise 2: Test Document Permissions

In this exercise we will use our newly created users to run some tests in order to better understand document level permissions.

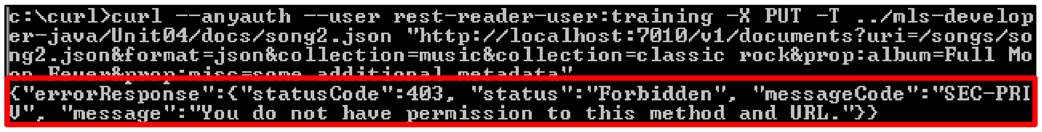
1. Navigate to **C:\mls-developer-node\Unit04** and open **4-2a (read a document).txt**.
2. Study the curl statement we will execute. Note that you are performing a GET request and that the user is now defined as the "rest-reader-user" that you just created:

```
curl --anyauth --user rest-reader-user:training -X GET
"http://localhost:7010/v1/documents?uri=/songs/song2.json"
```

3. From the command line, navigate to **c:\curl**
4. Run the curl command.
5. Note that you are able to read the document successfully. This is because you authenticated as a user who has the "rest-reader" role, and that role has a read permission on the specified document.
6. Now, let's see what happens when you try to perform an update as "rest-reader-user".
7. Navigate to **C:\mls-developer-node\Unit04** and open **4-2b (update a document).txt**.
8. Study the curl statement we will execute. Note that you are performing a PUT request and that the user is still defined as the "rest-reader-user" that you just created. Also take notice that the URI specified is "/songs/song2.json". Because this URI already exists in our database, we are attempting to perform an update here:

```
curl --anyauth --user rest-reader-user:training -X PUT -T
../mls-developer-node/Unit04/docs/song2.json
"http://localhost:7010/v1/documents?uri=/songs/song2.json&format=json&collection=music&co
llection=classic rock&prop:album=Full Moon Fever&prop:misc=some additional metadata"
```

9. From the command line, navigate to **c:\curl** and run the curl command.
10. Notice that you now receive an error response. This is because the "rest-reader-user" does not have a role with an update permission on the specified document:



```
c:\curl>curl --anyauth --user rest-reader-user:training -X PUT -T ../mls-develop
er-java/Unit04/docs/song2.json "http://localhost:7010/v1/documents?uri=/songs/so
ng2.json&format=json&collection=music&collection=classic rock&prop:album=Full Mo
on Fever&prop:misc=some additional metadata"
{"errorResponse":{"statusCode":403, "status":"Forbidden", "messageCode":"SEC-PRI
V", "message":"You do not have permission to this method and URL."}}
```

11. Change the curl command to use the "rest-writer-user" that you created.
12. You should now be able to successfully perform the update.