**RBAC with iControlREST on BIG-IP**

BIG-IP is able to use the built-in RBAC function. The operator function will be used in this example.

The right documentation can be found here: https://devcentral.f5.com/d/the-user-guide-for-the-icontrol-rest-interface-in-big-ip-version-1160?download=true

Chapter 1, paragraph: “About iControl and RBAC for user accounts”

The document explains how you can add users to the role of iControl\_REST\_API\_User. This small writing tries to give some guidance how you can use such a role in combination with a role of operator.

I am using Postman RESTclient for communicating with the BIG-IP (v12.1).

Note: Everywhere you see {{big\_ip\_a\_mgmt}} this should be your BIG-IP\_management\_IP. Or in Postman you could create an environmental value and stop retyping IP addresses. For those using Postman, the collection of used URIs has been added to this article and can be downloaded for your own use. (BIG-IP is able to use the built-in RBAC function. The operator function will be used in this example.

The right documentation can be found here: https://devcentral.f5.com/d/the-user-guide-for-the-icontrol-rest-interface-in-big-ip-version-1160?download=true

Chapter 1, paragraph: “About iControl and RBAC for user accounts”

The document explains how you can add users to the role of iControl\_REST\_API\_User. This small writing tries to give some guidance how you can use such a role in combination with a role of operator.

I am using Postman RESTclient for communicating with the BIG-IP (v12.1).

Note: Everywhere you see {{big\_ip\_a\_mgmt}} this should be your BIG-IP\_management\_IP. Or in Postman you could create an environmental value and stop retyping IP addresses. For those using Postman, the collection of used URIs has been added to this article and can be downloaded for your own use, here: <https://github.com/gwolfis/iControl>

If you want to have logging running allong (recommended), you should start a SSH session and use the following commands: (from the "About log files" in the above mentioned document)

* /var/log/restjavad-audit.0.log - shows all authentications to the iControl REST service. This is an ordered list of every REST call.
* /var/log/restjavad.0.log - contains information about connections to the iControl REST service, such as errors returned.
* /var/log/icrd - shows the actions of the icrd process, which manages the threads for processing the REST calls.
* /var/log/ltm - contains messages from mcpd, a process called by icrd that manages the system configuration.
* Use standard Unix commands to work with these files, such as tail, grep, and less.

Use basic authentication with admin rights to accomplish the following through Postman:

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/auth/user #Use admin to view all users on the BIG-IP

For the next task you might want to copy some of the response body to tweak your own non-admin user.

POST https://{{big\_ip\_a\_mgmt}}/mgmt/tm/auth/user # Create a user and define partition access and role

{

"name": "testuser",

"password": "test",

"partitionAccess": [

{

"name": "all-partitions",

"role": "operator"

}

]

}

If you can't wait (like me), log into the BIG-IP with the just created user and test. Did it work?

GET https://{{big\_ip\_a\_mgmt}}/mgmt/shared/authz/users # To get the properties of the just created user account.

You need the selflink properties of 'testuser' to be copied and placed in the iControl\_REST\_API\_User group

the part to copy should look something like this:

"selfLink": "https://localhost/mgmt/shared/authz/users/testuser"

GET https://{{big\_ip\_a\_mgmt}}/mgmt/shared/authz/roles # Shows all available roles and you should search for iControl\_REST\_API\_User

Next step is to PATCH this user to the role group of iControl\_REST\_API\_User.

PATCH https://{{big\_ip\_a\_mgmt}}/mgmt/shared/authz/roles/iControl\_REST\_API\_User # Add testuser to the iControl\_REST\_API\_User group by copy and pasting underneath section in the body of the PATCH request.

{

"userReferences":[{"link": "https://localhost/mgmt/shared/authz/users/testuser"}]

}

The response should include:

{

"name": "iControl\_REST\_API\_User",

"userReferences": [

{

"link": "https://localhost/mgmt/shared/authz/users/testuser"

},

----output omited---

Test if the user is allowed to use iControlREST, change the basic auth settings into your created test account.

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/ltm # Did it work? You should see something like this

{

"kind": "tm:ltm:ltmcollectionstate",

"selfLink": "https://localhost/mgmt/tm/ltm?ver=12.1.0",

"items": [

{

"reference": {

"link": "https://localhost/mgmt/tm/ltm/auth?ver=12.1.0"

}

},

{

"reference": {

"link": "https://localhost/mgmt/tm/ltm/classification?ver=12.1.0"

}

},

----output omited---

Since we are in operator mode we try to view a couple of things and modify poolmember states.

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/ltm/virtual # shows all configured virtual servers

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/ltm/pool # Shows all configured pools

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/ltm/node # Shows all nodes

As an operator you are allowed to change the node state.

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/ltm/pool/pool\_http/members # Show all poolmembers of pool pool\_http

Select a poolmember of choice to change state

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/ltm/pool/pool\_http/members/~Common~10.128.20.11:80

A poolmember has three states:

- Enabled

- Disabled (existing sessions will continue, but no new sessions will be accepted)

- Force Offline (existing sessions will be dropped immediately, no new sessions accepted)

Note1: There is a fourth state, but this one appears when the node is unavailable and is not relevant in this use case.

Note2: Also with a node you will see the same states, but you want to manage the state on poolmember level instead of node level, there is a difference… (read about poolmembers and nodes in F5 [documentation](https://support.f5.com/kb/en-us/products/big-ip_ltm/manuals/product/ltm-basics-12-1-0.html) if you don’t know.)

The states of the node are controlled by two parameters called "session" and "state"

Session: provides the function if existing sessions are allowed to continue, although the state is down (disabled mode), or should be dropped straight away (force offline)

State: facilitates if the node is up or down

Enabled in TMSH:

modify ltm node 10.128.20.11 session user-enabled state user-up #(GUI: mark = green round)

Disabled in TMSH:

modify ltm node 10.128.20.11 session user-disabled state user-up #(GUI: mark = black round)

Force offline in TMSH:

modify ltm node 10.128.20.11 session user-disabled state user-down #(GUI: mark = black diamond)

So how does this work with iControlREST:

PATCH https://{{big\_ip\_a\_mgmt}}/mgmt/tm/ltm/pool/pool\_http/members/~Common~10.128.20.11:80

{

"session":"user-enabled",

"state": "user-down"

}

Since you know the different values to change session and state, use them to play around and check the status of the respected node once you have done a modification.

You could go through the same for other roles and start exploring how to use them with iControl, the setup is the same.

So what if you want to clean up a bit since the user is no longer needed:

DELETE https://{{big\_ip\_a\_mgmt}}/mgmt/shared/authz/users/testuser

(don’t forget to change your basic authentication to admin first before trying to delete the user)

Confirm by doing the following checks

GET https://{{big\_ip\_a\_mgmt}}/mgmt/shared/authz/roles/iControl\_REST\_API\_User # Check if the non-admin user is removed.

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/auth/user # Check if the non-admin user is removed.

I hope that this helps in gaining understanding how a BIG-IP is able to add a non-admin user to the iControl\_REST\_API\_User role.

If you want to have logging running allong (recommended), you should start a SSH session and use the following commands: (from the "About log files" in the above mentioned document)

* /var/log/restjavad-audit.0.log - shows all authentications to the iControl REST service. This is an ordered list of every REST call.
* /var/log/restjavad.0.log - contains information about connections to the iControl REST service, such as errors returned.
* /var/log/icrd - shows the actions of the icrd process, which manages the threads for processing the REST calls.
* /var/log/ltm - contains messages from mcpd, a process called by icrd that manages the system configuration.
* Use standard Unix commands to work with these files, such as tail, grep, and less.

Use basic authentication with admin rights to accomplish the following through Postman:

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/auth/user #Use admin to view all users on the BIG-IP

For the next task you might want to copy some of the response body to tweak your own non-admin user.

POST https://{{big\_ip\_a\_mgmt}}/mgmt/tm/auth/user # Create a user and define partition access and role

{

"name": "testuser",

"password": "test",

"partitionAccess": [

{

"name": "all-partitions",

"role": "operator"

}

]

}

If you can't wait (like me), log into the BIG-IP with the just created user and test. Did it work?

GET https://{{big\_ip\_a\_mgmt}}/mgmt/shared/authz/users # To get the properties of the just created user account.

You need the selflink properties of 'testuser' to be copied and placed in the iControl\_REST\_API\_User group

the part to copy should look something like this:

"selfLink": "https://localhost/mgmt/shared/authz/users/testuser"

GET https://{{big\_ip\_a\_mgmt}}/mgmt/shared/authz/roles # Shows all available roles and you should search for iControl\_REST\_API\_User

Next step is to PATCH this user to the role group of iControl\_REST\_API\_User.

PATCH https://{{big\_ip\_a\_mgmt}}/mgmt/shared/authz/roles/iControl\_REST\_API\_User # Add testuser to the iControl\_REST\_API\_User group by copy and pasting underneath section in the body of the PATCH request.

{

"userReferences":[{"link": "https://localhost/mgmt/shared/authz/users/testuser"}]

}

The response should include:

{

"name": "iControl\_REST\_API\_User",

"userReferences": [

{

"link": "https://localhost/mgmt/shared/authz/users/testuser"

},

----output omited---

Test if the user is allowed to use iControlREST, change the basic auth settings into your created test account.

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/ltm # Did it work? You should see something like this

{

"kind": "tm:ltm:ltmcollectionstate",

"selfLink": "https://localhost/mgmt/tm/ltm?ver=12.1.0",

"items": [

{

"reference": {

"link": "https://localhost/mgmt/tm/ltm/auth?ver=12.1.0"

}

},

{

"reference": {

"link": "https://localhost/mgmt/tm/ltm/classification?ver=12.1.0"

}

},

----output omited---

Since we are in operator mode we try to view a couple of things and modify poolmember states.

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/ltm/virtual # shows all configured virtual servers

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/ltm/pool # Shows all configured pools

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/ltm/node # Shows all nodes

As an operator you are allowed to change the node state.

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/ltm/pool/pool\_http/members # Show all poolmembers of pool pool\_http

Select a poolmember of choice to change state

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/ltm/pool/pool\_http/members/~Common~10.128.20.11:80

A poolmember has three states:

- Enabled

- Disabled (existing sessions will continue, but no new sessions will be accepted)

- Force Offline (existing sessions will be dropped immediately, no new sessions accepted)

Note1: There is a fourth state, but this one appears when the node is unavailable and is not relevant in this use case.

Note2: Also with a node you will see the same states, but you want to manage the state on poolmember level instead of node level, there is a difference… (read about poolmembers and nodes in F5 [documentation](https://support.f5.com/kb/en-us/products/big-ip_ltm/manuals/product/ltm-basics-12-1-0.html) if you don’t know.)

The states of the node are controlled by two parameters called "session" and "state"

Session: provides the function if existing sessions are allowed to continue, although the state is down (disabled mode), or should be dropped straight away (force offline)

State: facilitates if the node is up or down

Enabled in TMSH:

modify ltm node 10.128.20.11 session user-enabled state user-up #(GUI: mark = green round)

Disabled in TMSH:

modify ltm node 10.128.20.11 session user-disabled state user-up #(GUI: mark = black round)

Force offline in TMSH:

modify ltm node 10.128.20.11 session user-disabled state user-down #(GUI: mark = black diamond)

So how does this work with iControlREST:

PATCH https://{{big\_ip\_a\_mgmt}}/mgmt/tm/ltm/pool/pool\_http/members/~Common~10.128.20.11:80

{

"session":"user-enabled",

"state": "user-down"

}

Since you know the different values to change session and state, use them to play around and check the status of the respected node once you have done a modification.

You could go through the same for other roles and start exploring how to use them with iControl, the setup is the same.

So what if you want to clean up a bit since the user is no longer needed:

DELETE https://{{big\_ip\_a\_mgmt}}/mgmt/shared/authz/users/testuser

(don’t forget to change your basic authentication to admin first before trying to delete the user)

Confirm by doing the following checks

GET https://{{big\_ip\_a\_mgmt}}/mgmt/shared/authz/roles/iControl\_REST\_API\_User # Check if the non-admin user is removed.

GET https://{{big\_ip\_a\_mgmt}}/mgmt/tm/auth/user # Check if the non-admin user is removed.

I hope that this helps in gaining understanding how a BIG-IP is able to add a non-admin user to the iControl\_REST\_API\_User role.