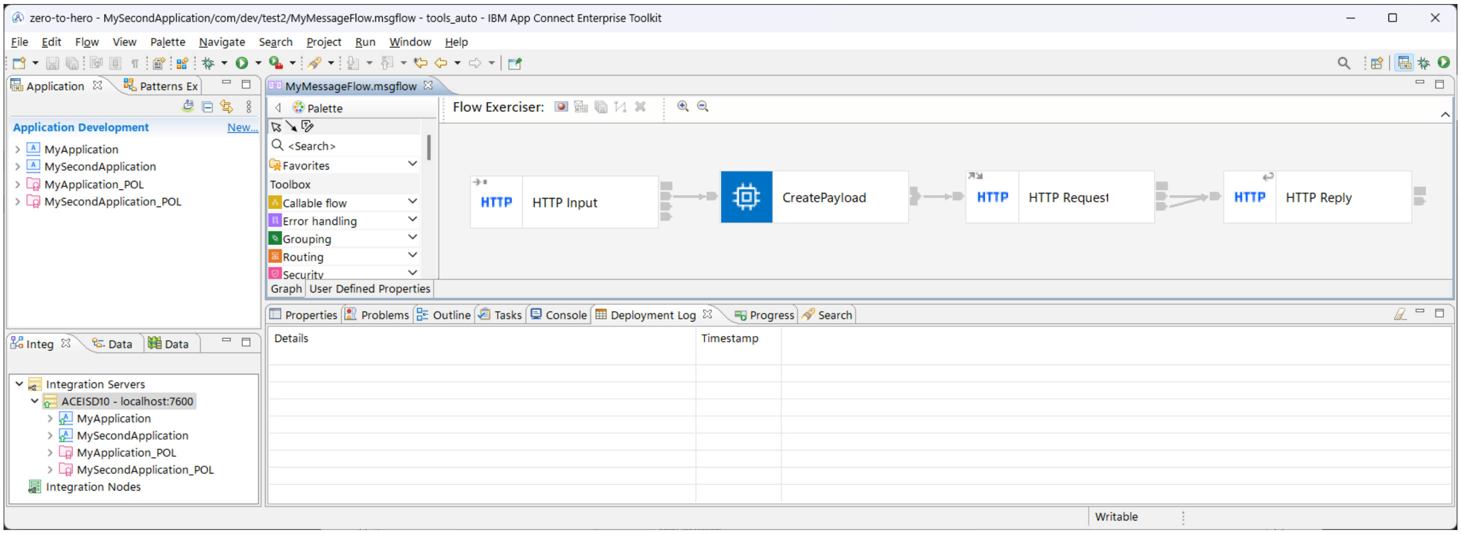
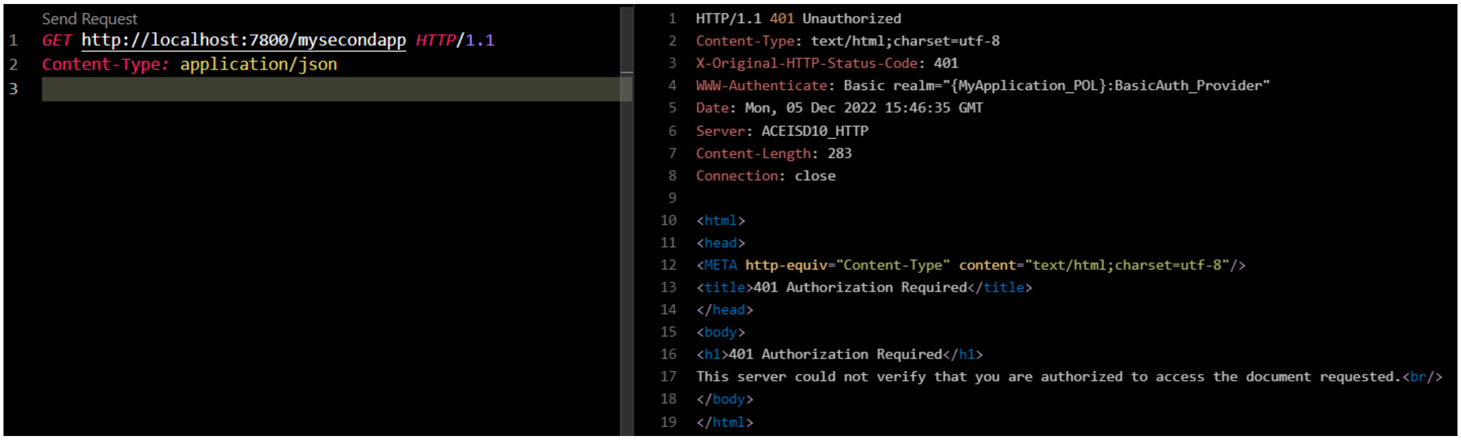
How to invoke a Basic Auth secured HTTP Service in IBM ACE

This document demonstrates the capability of invoking Basic Auth secured HTTP(s) service by injecting Auth Headers in runtime. Although Basic Auth is not generally used in production environments (securing the APIs is majorly handled by API Management Tools like IBM API Connect), it is worth learning.

Step 1: MyApplication is a Basic Auth protected REST service. Created a new MySecondApplication which will invoke MyApplication. Note: HTTP Request would require the Basic Auth in headers.



First, we invoke the service without introducing Auth. This should result into error.



Send Request

```
GET http://localhost:7800/mysecondapp HTTP/1.1
Content-Type: application/json

```

```
HTTP/1.1 401 Unauthorized
Content-Type: text/html;charset=utf-8
X-Original-HTTP-Status-Code: 401
WWW-Authenticate: Basic realm="{MyApplication_POL}:BasicAuth_Provider"
Date: Mon, 05 Dec 2022 15:46:35 GMT
Server: ACEISD10_HTTP
Content-Length: 283
Connection: close

<html>
<head>
<META http-equiv="Content-Type" content="text/html;charset=utf-8"/>
<title>401 Authorization Required</title>
</head>
<body>
<h1>401 Authorization Required</h1>
This server could not verify that you are authorized to access the document requested.<br/>
</body>
</html>
```

Step 2: Writing BasicAuth in ESQL to add in the HTTPRequestHeader **[NOT RECOMMENDED].** Writing password in plain text is not the recommended way (although it works).

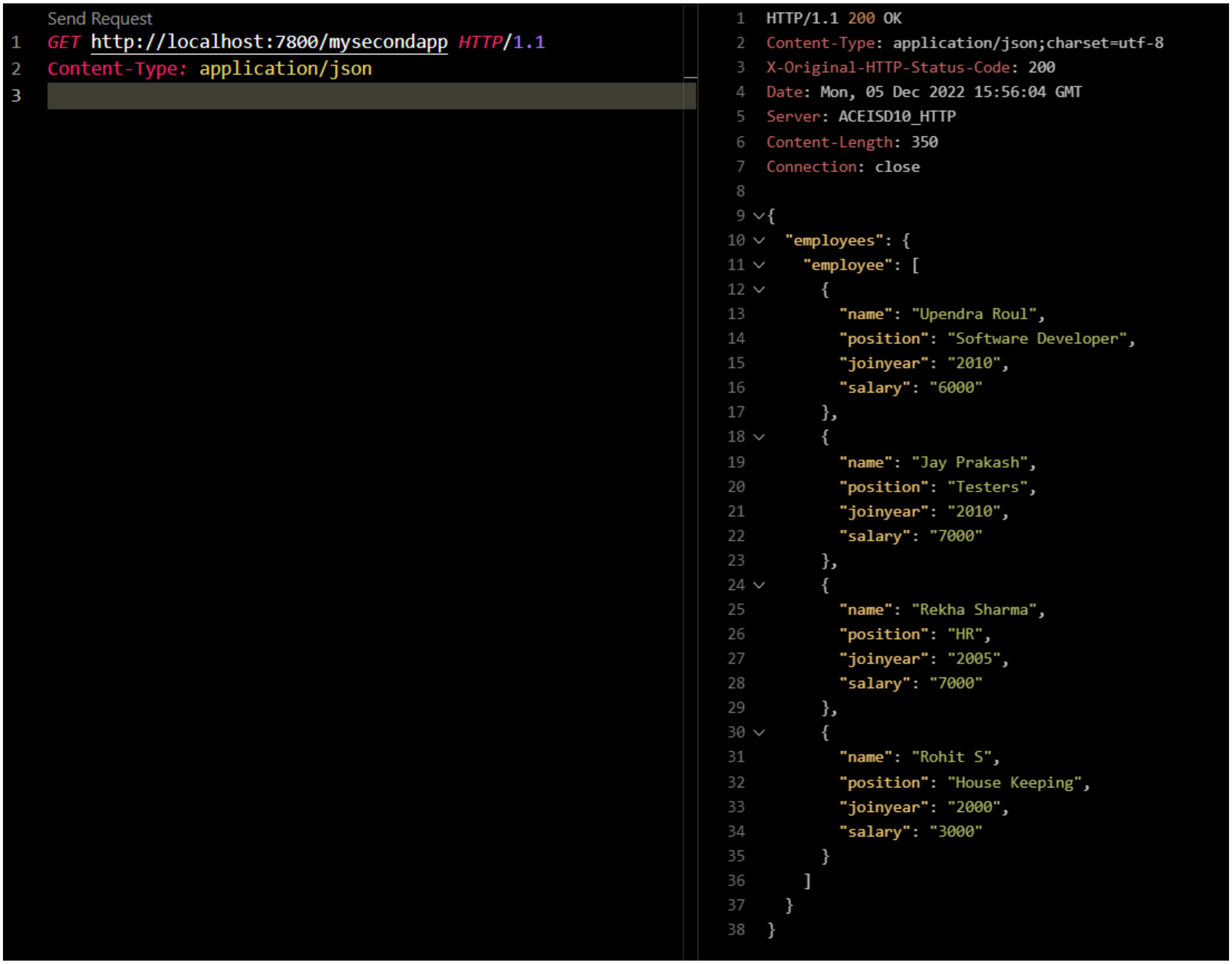
```
CREATE FUNCTION Main() RETURNS BOOLEAN
BEGIN

    -- Adding Auth Headers
    DECLARE username CHARACTER 'myusername';
    DECLARE password CHARACTER 'mypassword';
    CREATE NEXTSIBLING OF OutputRoot.Properties DOMAIN 'HTTPRequestHeader';
    SET OutputRoot.HTTPRequestHeader.Authorization = 'Basic ' || BASE64ENCODE(username || ':' || password);

    -- Creating XML Payload
    CALL createXMLPayload();

    RETURN TRUE;
END;
```

Re-deply the application, this time the application should work fine!



```
Send Request
GET http://localhost:7800/mysecondapp HTTP/1.1
Content-Type: application/json

```

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8
X-Original-HTTP-Status-Code: 200
Date: Mon, 05 Dec 2022 15:56:04 GMT
Server: ACEISD10_HTTP
Content-Length: 350
Connection: close

{
  "employees": {
    "employee": [
      {
        "name": "Upendra Roul",
        "position": "Software Developer",
        "joinyear": "2010",
        "salary": "6000"
      },
      {
        "name": "Jay Prakash",
        "position": "Testers",
        "joinyear": "2010",
        "salary": "7000"
      },
      {
        "name": "Rekha Sharma",
        "position": "HR",
        "joinyear": "2005",
        "salary": "7000"
      },
      {
        "name": "Rohit S",
        "position": "House Keeping",
        "joinyear": "2000",
        "salary": "3000"
      }
    ]
  }
}
```

Now, since writing passwords in plain-text is not the recommended way, we'll comment the piece of code and try to inject the same the Auth using Policy!

Step 3: Comment the code in ESQL that injects the username and password

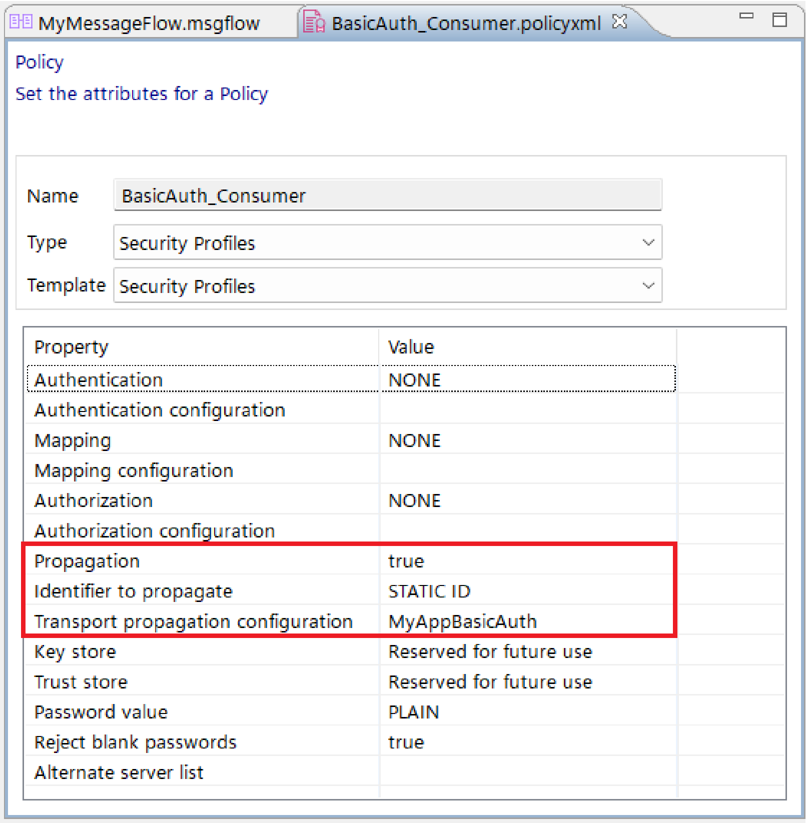
```
CREATE FUNCTION Main() RETURNS BOOLEAN
BEGIN

    -- Adding Auth Headers
    -- DECLARE username CHARACTER 'myusername';
    -- DECLARE password CHARACTER 'mypassword';
    -- CREATE NEXTSIBLING OF OutputRoot.Properties DOMAIN 'HTTPRequestHeader';
    -- SET OutputRoot.HTTPRequestHeader.Authorization = 'Basic ' || BASE64ENCODE(username || ':' || password);

    -- Creating XML
    CALL createXMLPayload();

    RETURN TRUE;
END;
```

Step 4: Create a policy for the consumer. Note the Identified to propagate is now set to STATIC ID, Propagation is set to true and Transport propagation configuration is the Security Identifier (last time we saved the credentials in server.conf.yaml, this time we replaced it using mqsisetdbparms). We cannot use credentials stored in server.conf.yaml as it will throw an expection: **BIP2769E** which says *The transportPropagationConfig property of the security profile must be set to the name of of a security identity that has been defined using the* ***mqsisetdbparms*** *or* ***mqsicredentials*** *command.*



Policy

Set the attributes for a Policy

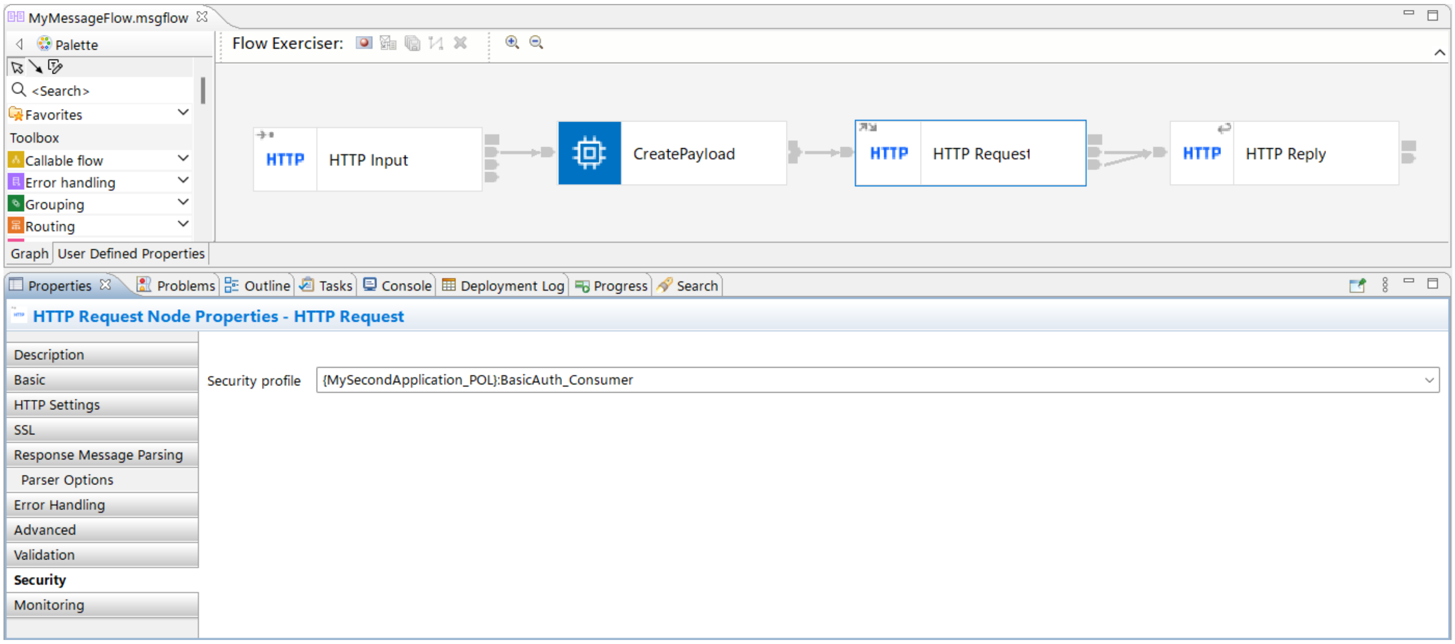
Name: BasicAuth_Consumer

Type: Security Profiles

Template: Security Profiles

Property	Value
Authentication	NONE
Authentication configuration	
Mapping	NONE
Mapping configuration	
Authorization	NONE
Authorization configuration	
Propagation	true
Identifier to propagate	STATIC ID
Transport propagation configuration	MyAppBasicAuth
Key store	Reserved for future use
Trust store	Reserved for future use
Password value	PLAIN
Reject blank passwords	true
Alternate server list	

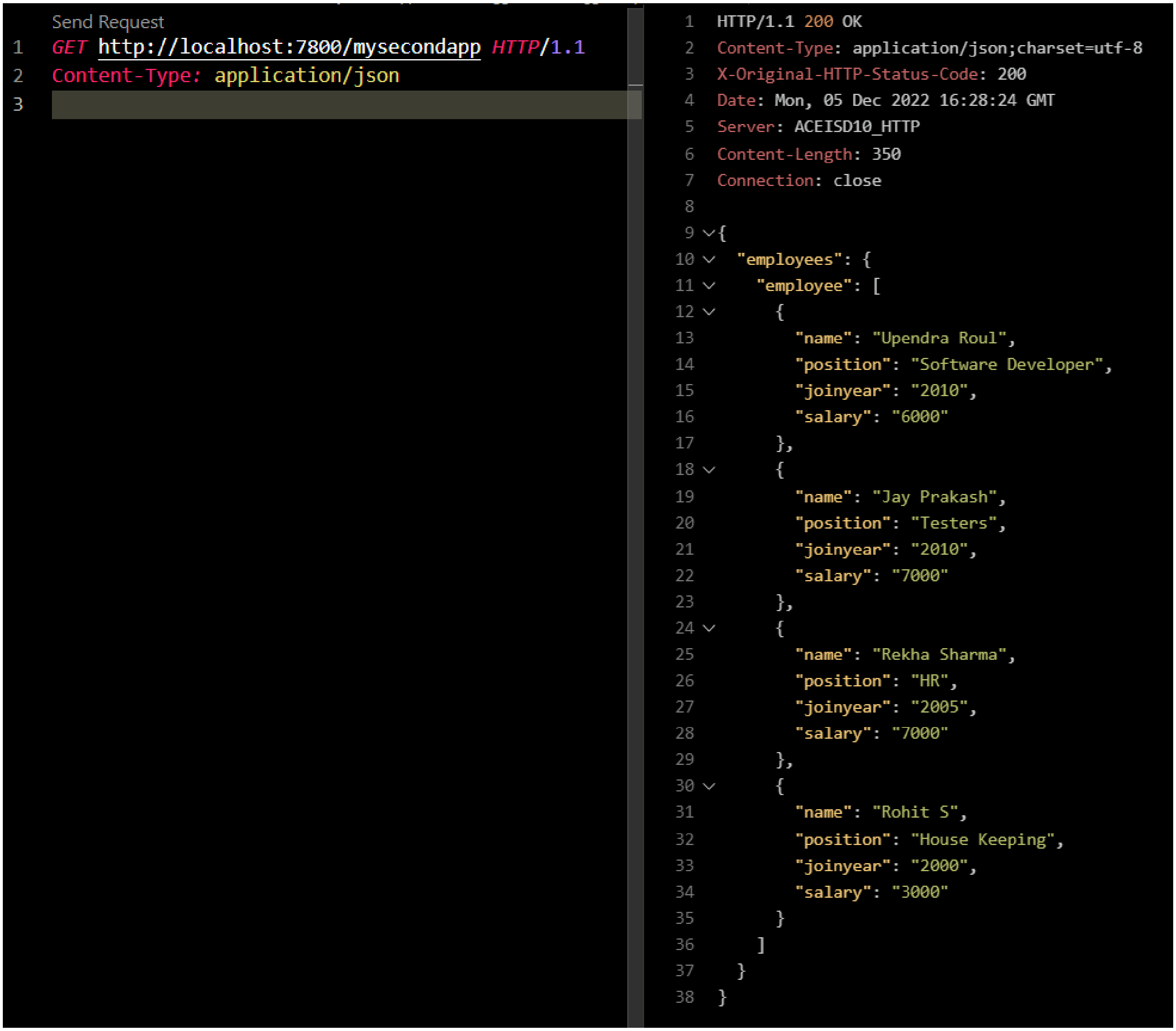
Step 5: In the Application, select the HTTP Request Node, go to Security tab and configure the Security profile as shown. The name must be configured as *{PolicyProject}:PolicyName*. Once the Security profile is configured, re-deploy the application (make sure the Policy is still deployed).



HTTP Request Node Properties - HTTP Request

Security profile: {MySecondApplication_POL}:BasicAuth_Consumer

Step 6: Redeploying the application and testing. It should be successful.



```
GET http://localhost:7800/mysecondapp HTTP/1.1
Content-Type: application/json

```

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8
X-Original-HTTP-Status-Code: 200
Date: Mon, 05 Dec 2022 16:28:24 GMT
Server: ACEISD10_HTTP
Content-Length: 350
Connection: close

{
  "employees": {
    "employee": [
      {
        "name": "Upendra Roul",
        "position": "Software Developer",
        "joinyear": "2010",
        "salary": "6000"
      },
      {
        "name": "Jay Prakash",
        "position": "Testers",
        "joinyear": "2010",
        "salary": "7000"
      },
      {
        "name": "Rekha Sharma",
        "position": "HR",
        "joinyear": "2005",
        "salary": "7000"
      },
      {
        "name": "Rohit S",
        "position": "House Keeping",
        "joinyear": "2000",
        "salary": "3000"
      }
    ]
  }
}
```