

Db2 for z/OS REST and Hybrid Cloud Lab Documentation

Lab 1: Db2 for z/OS Native REST Services

Created for Db2 for z/OS REST and Hybrid Cloud Wildfire Workshop

Date: September 10, 2024

Version: 2.0

Authors:

Victoria Felt

Keziah Knopp

Isabella Reyes

Contents

OVERVIEW	4
LAB 1 DB2 NATIVE REST SERVICES	6
1.1 LAB PREPARATIONS AND DISCOVERING SERVICES	7
1.1.1 <i>Launching VSCode and Discover Db2 REST Services</i>	7
1.2 CREATING A DB2 REST SERVICE	11
1.2.1 <i>Confirm the creation of the Db2 REST Services</i>	14
1.3 EXECUTE THE DB2 REST SERVICE	18
1.3.1 <i>Display the Db2 Native REST service JSON schema information</i>	19
1.3.2 <i>Executing the Db2 REST service</i>	27
1.5 VERSIONING DB2 REST SERVICES	31
1.5.1 <i>Discover the Versioned Service</i>	31
1.5.2 <i>View the JSON schema of the service</i>	33
1.5.3 <i>Execute the versioned service</i>	35
1.5.4 <i>Create new versions of the service</i>	37
1.5.5 <i>Execute the new versioned services</i>	39
1.6 SUMMARY	43

Overview

Since IBM Db2 for z/OS version 11, has the ability to be a REST (**R**epresentational State **T**ransfer) service provider. In this workshop, you will learn how quickly Db2 stored procedures and SQL statements can be enabled as REST services, and support REST interaction with your web, mobile and cloud applications.

Db2 REST services are fully integrated into the Db2 distributed data facility (DDF). All Db2 REST services are managed natively within the Db2 subsystem/member. The Db2 native REST service solution leverages the existing DDF capabilities for authentication, authorization, client information management, service classification, system profiling, and service monitoring. Db2 defines a REST service as a package. Each package contains a single static SQL statement and is recorded in the SYSIBM.DSNSERVICE catalog table. The following SQL statements are supported: CALL, DELETE, INSERT, SELECT, TRUNCATE, UPDATE and WITH.

Db2 provides REST APIs to create, discover and manage user-defined REST services in Db2. Db2 REST user-defined services are invoked by the POST method only. The z/OS Connect API Editor can reassign this POST method. For example, in Lab 2 you will use the z/OS Connect API Editor to map the Db2 REST service's POST method to the appropriate GET method.

The purpose of the following lab is to provide the information necessary to create and execute Db2 REST services using VSCode and relevant extensions.

Introduction

This introduction provides the system requirements and instructions to create Db2 REST services and APIs. The following labs are presented in a series of instructions using Db2 and the Db2 installation verification program's (IVP) "EMP" table created in job DSNTEJ1. The IVP EMP table should be available on customer systems to practice with local subsystems.

The following concepts are covered in this workshop:




- Create a Db2 REST service using an existing stored procedure and/or a SQL statement

Software inventory used in this workshop to complete the exercises

1. IBM Db2 12 for z/OS
2. Microsoft Visual Studio Code (VSCode)
3. A REST API interface – VSCode extension Thunder Client will be used in the exercises

Icons

The following symbols appear in this document at places where additional guidance is available.

Icon	Purpose	Explanation
	Important!	This symbol calls attention to a particular step or command. For example, it might alert you to type a command carefully because it is case sensitive.
	Information	This symbol indicates information that might not be necessary to complete a step but is helpful or good to know.
	Troubleshooting	This symbol indicates that you can fix a specific problem by completing the associated troubleshooting information.

Lab 1 Db2 Native REST Services

This exercise covers creating a Db2 REST service using the Db2 BIND command using an update provided in Db2 PTF UI51748 and APAR PI98649 (PTF UI584231 or UI58425). The Db2 REST service will use a Db2 native SQL stored procedure or a SQL statement. When creating the service, you have the option to use an already existing stored procedure, or a simple SQL statement. If you would like to use both, two separate batch jobs must be submitted. A Db2 REST service is restricted to only one stored procedure or SQL statement.

Db2 z/OS REST services do support HTTPS, but requires z/OS “Application Transparent – Transport Level Security” (AT-TLS).

**Important!**

In this lab you have the option to create a Db2 Native REST service that uses a simple SQL statement or an existing stored procedure, or both.

If you would like to create both, you must send two separate batch jobs.

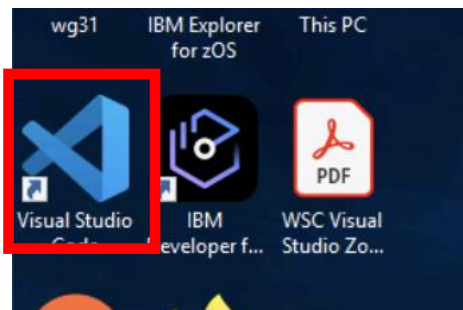
1.1 Lab Preparations and Discovering Services

Db2 REST services can be created and managed using VSCode with the Db2 for z/OS Developer Extension and REST debugging tools. In this step, you will set up a REST API VSCode Extension for use in testing and executing the REST services that you will be creating using the Db2 BIND subcommand. In this lab, we will be using “Thunder Client” as our REST API extension; there are other available REST API extensions available in the VSCode Extensions Marketplace.

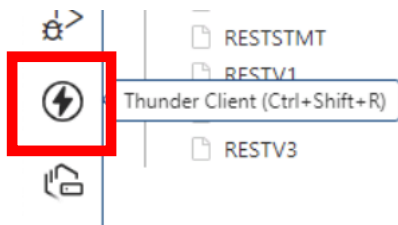
1.1.1 Launching VSCode and Discover Db2 REST Services

In this exercise, you will use the tools available for internet download to work with REST APIs. In this section, you will use a REST API Extension in VSCode to issue REST requests to execute the Db2 REST Services.

- __ 1. Double click on the VSCode icon on the desktop:

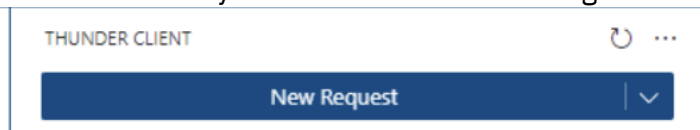


- __ 2. When VSCode launches, navigate to the left-hand side panel and click on the Thunder Client



extension.

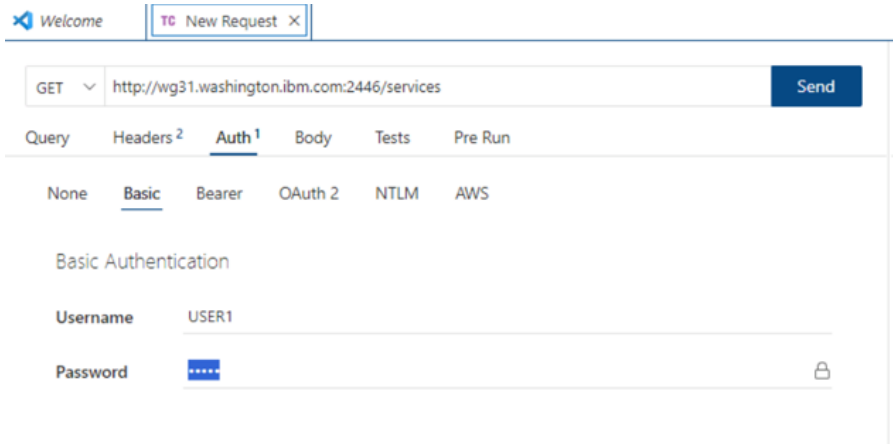
- __ 3. Click a “New Request” and update the request with the information below to discover all Db2 REST services currently installed in the Db2 catalog.



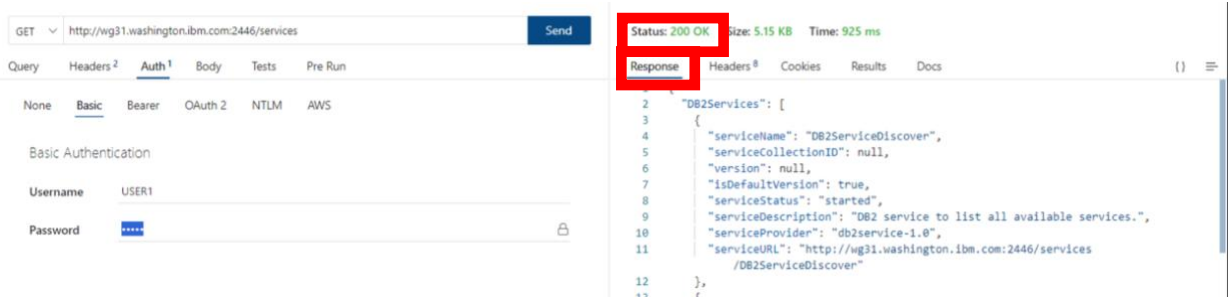
- A. Set the REST Method to: **GET**
 B. Add the Db2 discover API to the REST URL: **http://wg31.washington.ibm.com:2446/services**
 __i. Db2 DNS name = **wg31.washington.ibm.com**
 __ii. Db2 port = **2446**
 __iii. Db2 Discover URI = **/services**

- __ 4. Select “**Auth**” from the header and “**Basic**” in the sub-header to enter the following information:
 A. Username: **USER1**

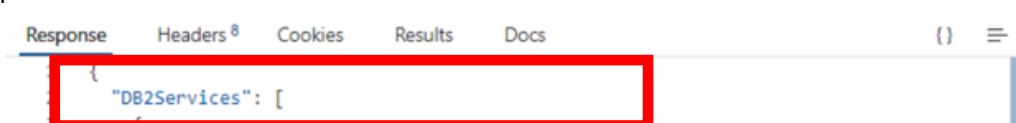
B. Password: **USER1**



- __ 5. Click **SEND** to send the request.
- __ 6. Discover Response. Confirm the proper output below:
 - A. The first items displayed for review are the response status, stats, and **response** body. The status **200 OK** will be the normal successful return code for Db2 services.



- B. The response headers information may be viewed, for analysis, by selecting from the options presented.



- C. Return to the response body if you viewed the response headers. Scroll down to see the Db2 services installed. Db2 services: "DB2ServiceDiscover" and "DB2ServiceManager" are for administration and system management.

Status: 200 OK Size: 5.15 KB Time: 925 ms

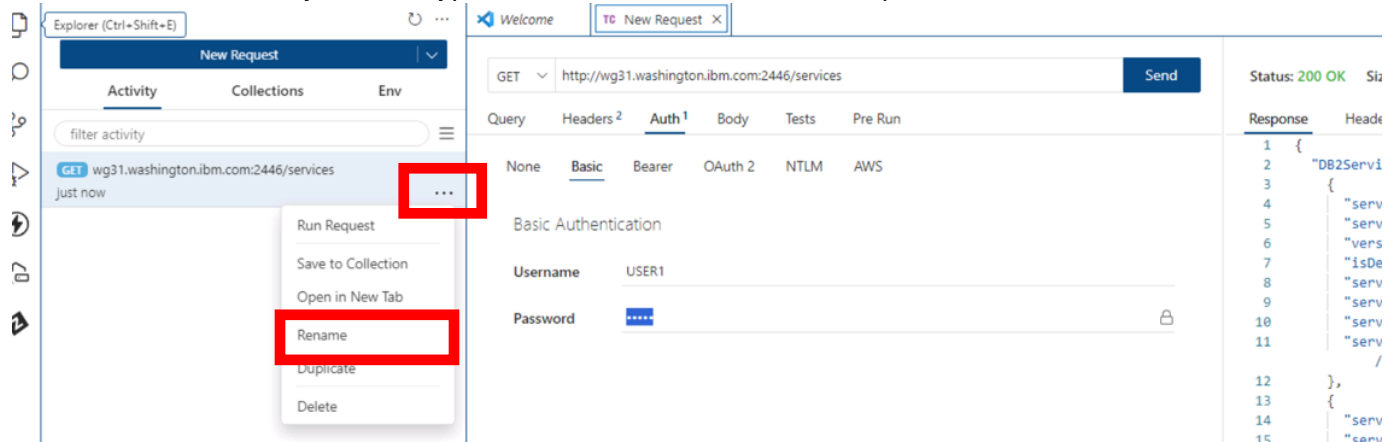
Response Headers⁸ Cookies Results Docs {} ≡

```

2  "DB2Services": [
3
4  "serviceName": "DB2ServiceDiscover",
5  "serviceCollections": null,
6  "version": null,
7  "isDefaultVersion": true,
8  "serviceStatus": "started",
9  "serviceDescription": "DB2 service to list all available services.",
10 "serviceProvider": "db2service-1.0",
11 "serviceURL": "http://wg31.washington.ibm.com:2446/services
    /DB2ServiceDiscover"
12 ],
13
14 "serviceName": "DB2ServiceManager",
15 "serviceCollections": null,
16 "version": null,
17 "isDefaultVersion": true,
18 "serviceStatus": "started",
19 "serviceDescription": "DB2 service to create, drop, or alter a user defined
    service.",
20 "serviceProvider": "db2service-1.0",
21 "serviceURL": "http://wg31.washington.ibm.com:2446/services/DB2ServiceManager"
22 }

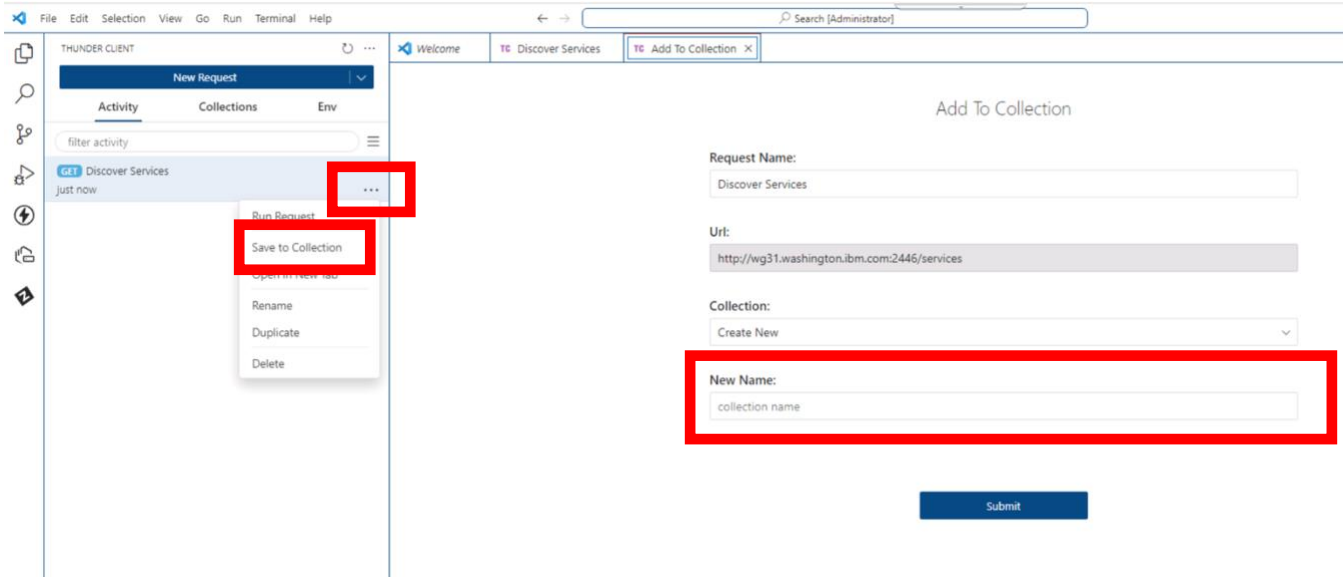
```

- __ 7. Rename this request to “Discover Services.” Hover over the request name for the three dots to appear. Click “Rename”. After you have typed “Discover Services” in the top bar, hit enter to save it.

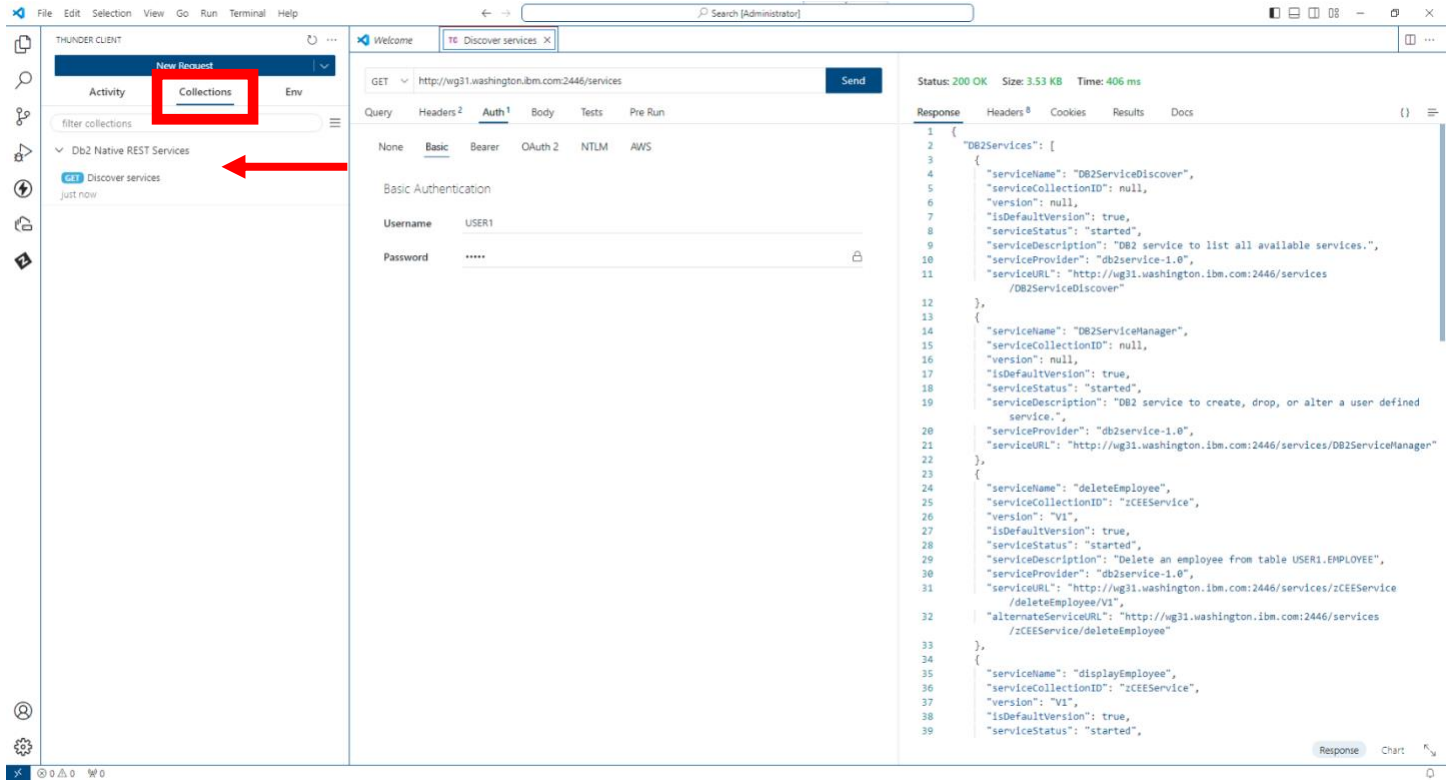


- __ 8. Click the same menu icon, “...”, next to the service and select “Save to Collection.”

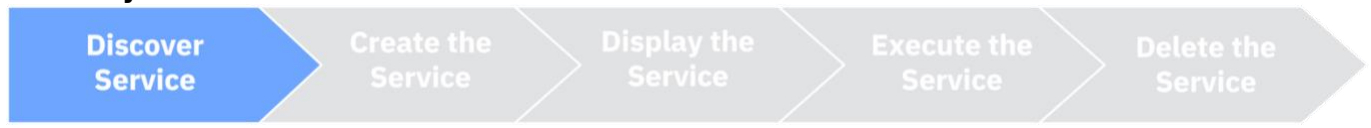
- 9. In the Collection drop menu, select “Create New”. In the “New Name” box that appears, title the collection “Db2 Native REST Services” and hit **Submit**.



- 10. You may close the “Add to Collection” window. To view your request in the Collection, click the “Collections” tab next to “Activity”.



Summary: In this section we launched VSCode and authenticated our collection with Basic Authentication.



1.2 Creating A Db2 REST Service

The following sections demonstrate how quickly a Db2 REST service can be created using a Db2 stored procedure and/or a SQL statement. Creating REST services from stored procedures allows existing business logic to be used by new applications in the API economy. The DB2 BIND subcommand will be used to create the Db2 user-defined REST service, and the REST service will be stored in the Db2 catalog (table DSNSERVICE) ready for use. VSCode will be used to test the execution of the Db2 REST service.

When you create a service, Db2 identifies you – or the authorization ID that you use – as the default owner of the service. Therefore, you must have the required privileges to create a service and BIND the associated package into a collection. For example, you must be authorized to execute the SQL statement that is embedded in the service. See [BIND PACKAGE \(DSN\) page](#) in IBM Documentation for information regarding the privileges and authorities that you must have to create a package for a Db2 REST service.

To reiterate, in this section, you will be using a JCL batch job to create a Db2 Native REST Service using VSCode.



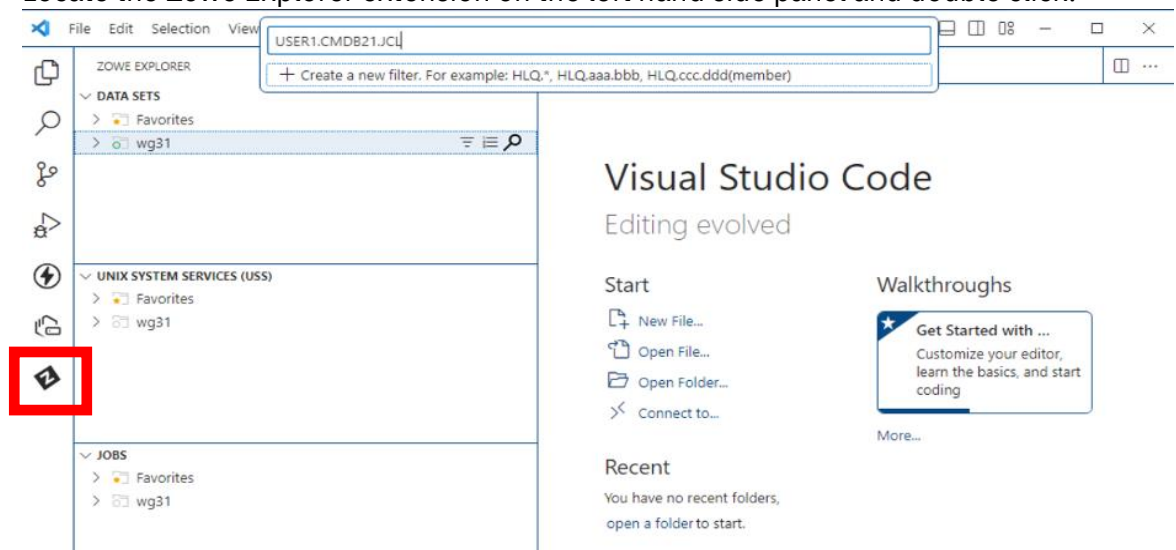
Important!

In this lab you have the option to create a Db2 Native REST service that uses an existing stored procedure, **or** a simple SQL statement, **or** both.

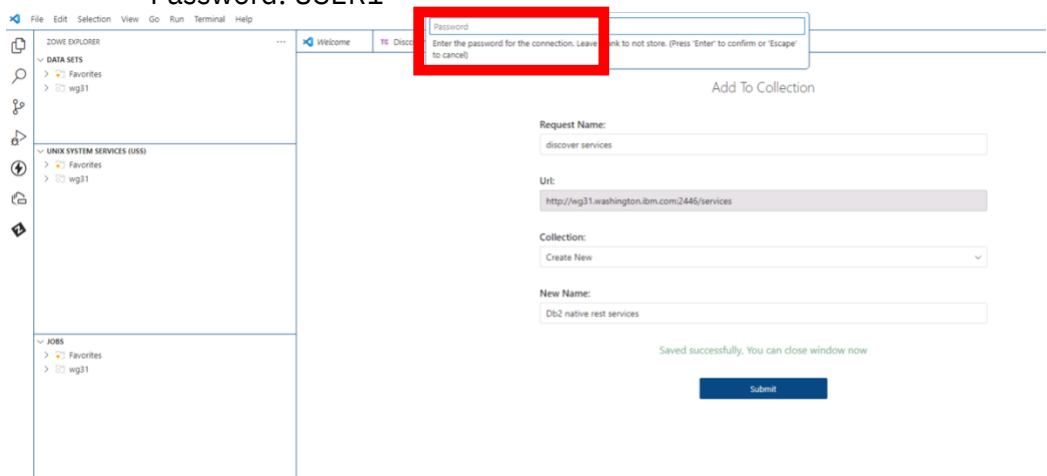
If you would like to create **both**, you must send two *separate* batch jobs by following and executing the steps below twice; once with the JCL calling the stored procedure, and the second time with the JCL for the simple SQL statement.

The following steps show you how to complete creating a Db2 REST service using VSCode. Though, creating this service could also be accomplished in a TSO 3270 emulator or green screen.

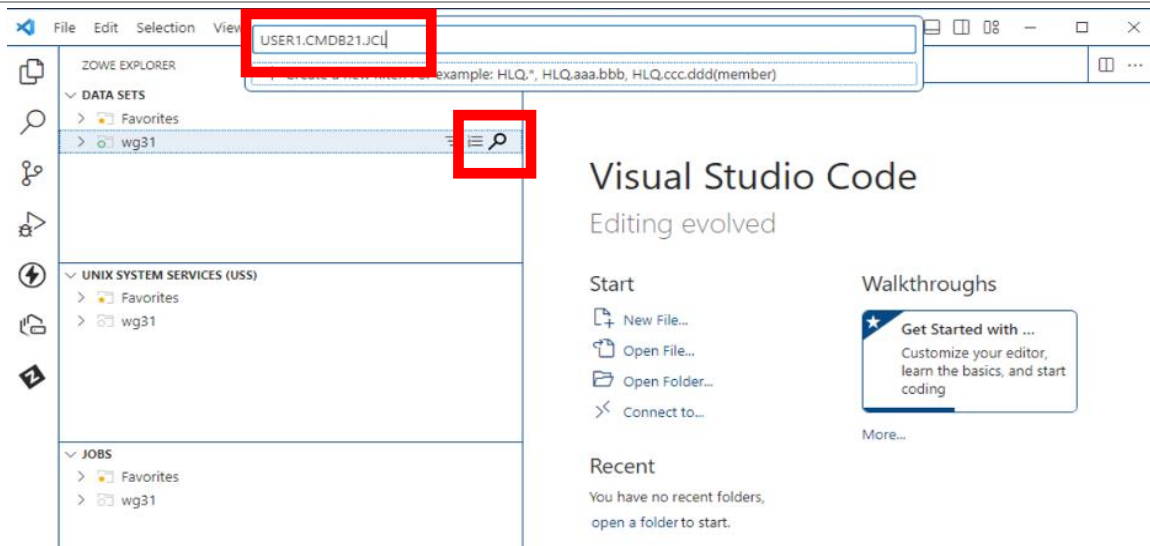
- __ 1. Locate the Zowe Explorer extension on the left hand side panel and double click.



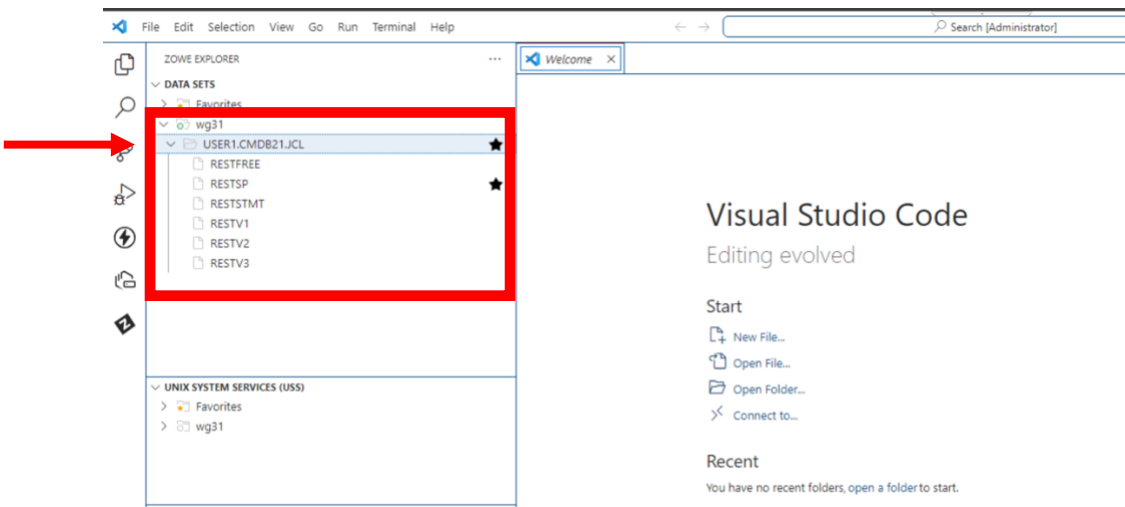
- __ 2. In the Data Sets subsection, locate the magnifying glass search icon and click on it.
 - a. If prompted, use the following credentials then press enter on your keyboard:
 Username: USER1
 Password: USER1



- __ 3. To view the appropriate data set, type in the qualifier of "USER1.CMDB21.JCL" and hit enter.

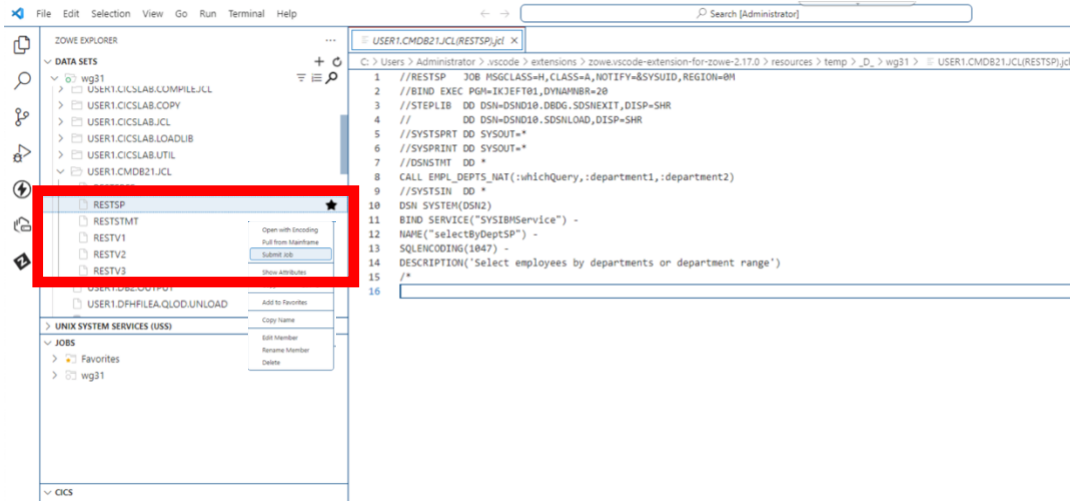


- __ 4. Locate the jobs for creating the different Db2 REST Services. On the left side click on the arrow next to USER1.CMDB21.JCL to expand the datasets with the jobs.



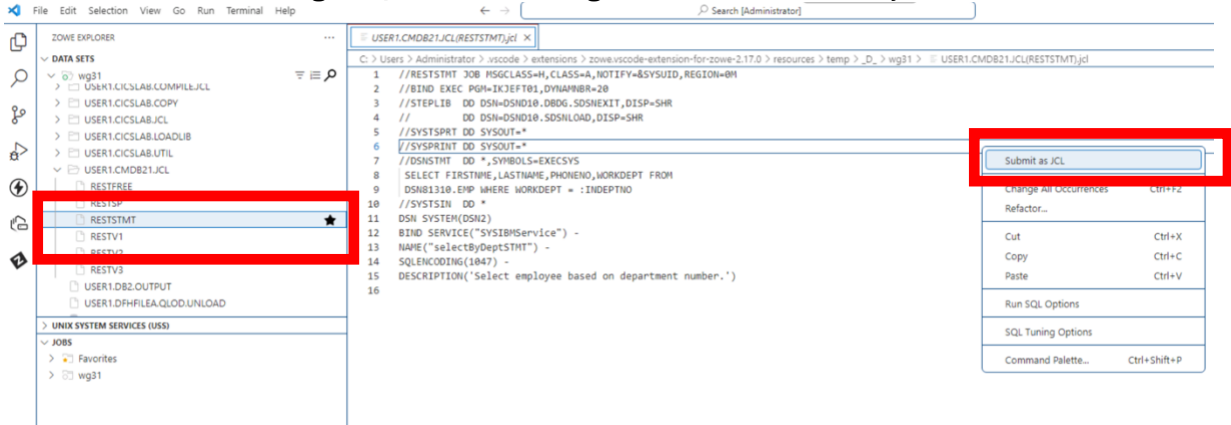
- __ 5. Now, you can create any of the services using the provided JCL. You may choose one or both options listed below. You can double click on each job to view the JCL in z/OS Explorer, and they are also listed below for your reference.

A. To create a service calling the stored procedure, right click on RESTSP.jcl and select **Submit**.



OR

B. To create a service using a SQL Statement, right click on RESTSTMT.jcl and select **Submit**.



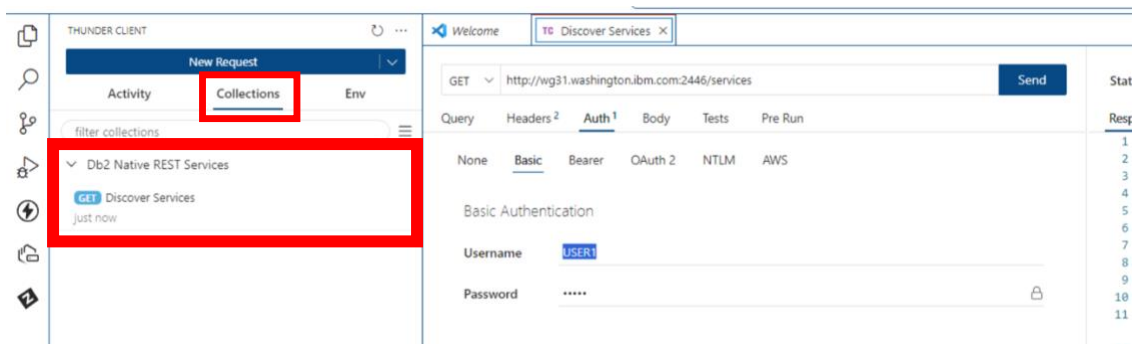
Note: Expect a Completion Code (CC) of 0000 for successful completion. Click on the job ID hyperlink to verify the CC.



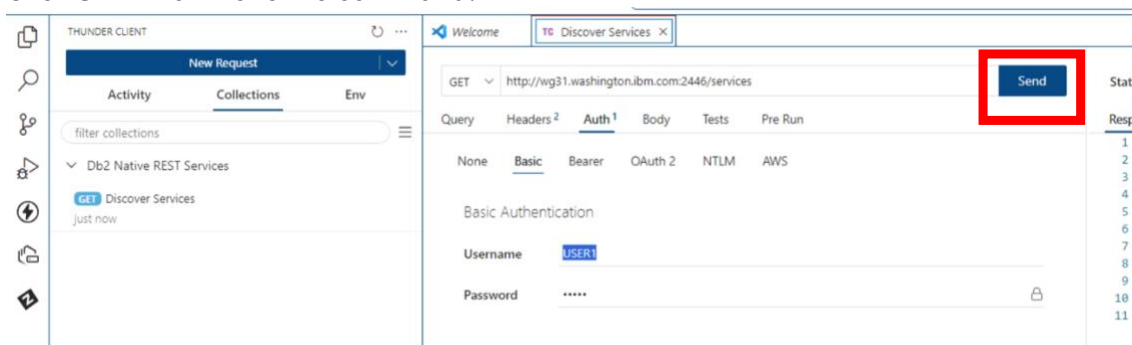
1.2.1 Confirm the creation of the Db2 REST Services

This section is optional because receiving a Completion Code (CC) of 0000 confirms the successful creation of your service. However, it is useful to “re-discover” the services installed on Db2 to see any updates made to the subsystem. Return to the Thunder Client extension in VSCode that we used in Step 1.1.1.

1. Use the “Discover Services” request again to confirm the creation of the Db2 service.
 - A. Open the “Discover Services” request in the Thunder Client VSCode extension, found in the “Db2 Native REST Services” collection.



- B. Click **SEND** to invoke the command.



Troubleshooting



If the request fails, returning a status code of anything other than **200 OK**, go back to [section 1.1.1](#) and confirm that the request is filled out correctly; namely the correct REST Method and URI.

Also, read the StatusCode and StatusDescription to troubleshoot and debug the error.

Be careful when saving any changes made to the request.

- C. Scroll down and confirm that the “selectByDeptSP” and/or “selectByDeptSTMT” is in the discover response output body.

```

Status: 200 OK Size: 5.15 KB Time: 925 ms
Response Headers Cookies Results Docs
1 {
2   "DB2Services": [
3     {
4       "serviceName": "DB2ServiceDiscover",
5       "serviceCollectionID": null,
6       "version": null,
7       "isDefaultVersion": true,
8       "serviceStatus": "started",
9       "serviceDescription": "DB2 service to list all available services.",
10      "serviceProvider": "db2service-1.0",
11      "serviceURL": "http://wg31.washington.ibm.com:2446/services
12        /DB2ServiceDiscover"
13    },
14    {
15      "serviceName": "DB2ServiceManager",
16      "serviceCollectionID": null,
17      "version": null,
18      "isDefaultVersion": true,
19      "serviceStatus": "started",
20      "serviceDescription": "DB2 service to create, drop, or alter a user defined
21        service.",
22      "serviceProvider": "db2service-1.0",
23      "serviceURL": "http://wg31.washington.ibm.com:2446/services/DB2ServiceManager"
24    },
25    {
26      "serviceName": "selectByDeptSP",
27      "serviceCollectionID": "SYSIBMService",
28      "version": "V1",
29      "isDefaultVersion": true,
30      "serviceStatus": "started",
31      "serviceDescription": "Select employees by departments or department range",
32      "serviceProvider": "db2service-1.0",
33      "serviceURL": "http://wg31.washington.ibm.com:2446/services/SYSIBMService
34        /selectByDeptSP/V1",
35      "alternateServiceURL": "http://wg31.washington.ibm.com:2446/services
36        /SYSIBMService/selectByDeptSP"
37    },
38    {
39      "serviceName": "selectByDeptSTMT",
40      "serviceCollectionID": "SYSIBMService",
41      "version": "V1",
42      "isDefaultVersion": true,
43      "serviceStatus": "started",
44      "serviceURL": "http://wg31.washington.ibm.com:2446/services/SYSIBMService
45        /selectByDeptSTMT/V1",
46      "alternateServiceURL": "http://wg31.washington.ibm.com:2446/services
47        /SYSIBMService/selectByDeptSTMT"
48    }
49  ]
50 }

```

The ServiceURL listed below (as well as above in the JSON) contains the URL used to display the JSON request and response schemas and execute the Db2 REST service. The service URL will be used to display the Db2 services’ metadata. You may enter the URL in Firefox (not as RESTClient) to view the schema.

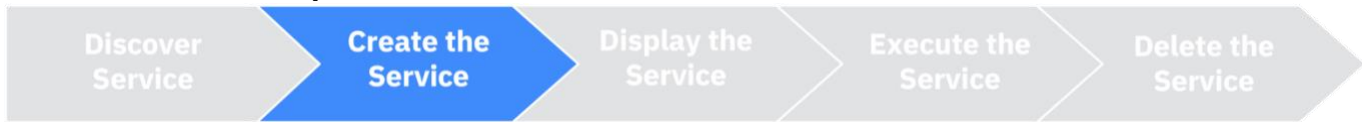
Stored Procedure:

- <http://wg31.washington.ibm.com:2446/services/SYSIBMService/selectByDeptSP>
- The Uniform Resource Identifier (URI) is: /services/SYSIBMService/selectByDeptSP

SQL Statement:

- <http://wg31.washington.ibm.com:2446/services/SYSIBMService/selectByDeptSTMT>
- The Uniform Resource Identifier (URI) is: /services/SYSIBMService/selectByDeptSTMT

Summary: In this section, we confirmed the creation of the REST service using our previous “Discover Services on DB2M” request.



1.3 Execute the Db2 REST service

The first thing to do before executing the Db2 REST service is to review the JSON request and response schema information. The JSON request schema provides the information needed to execute the Db2 REST service, and the response schema provides the information being sent from Db2.

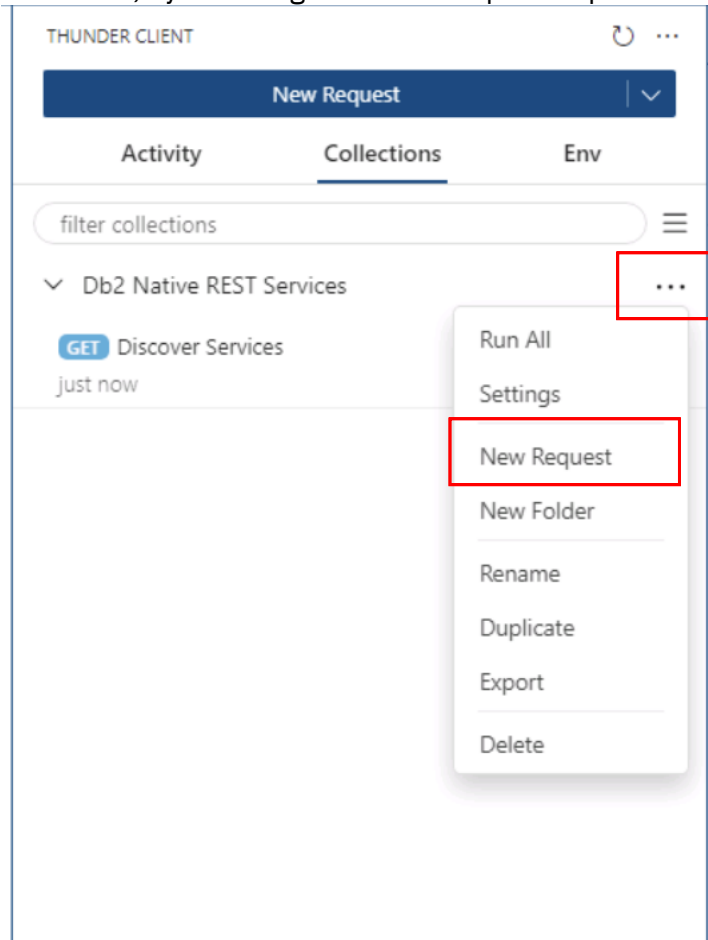
For the service using the stored procedure – “selectByDeptSP” – it is important to take note of a property of the response schema. A Db2 REST service that calls a stored procedure, returns a result set that Db2 calls a “dynamic anonymous result set”. The reason there is an “anonymous result set” is because the Db2 stored procedure can provide various output results, which is determined based on runtime input. The term “anonymous result sets” indicates that in order to determine what the result set output will look like, the Db2 service must be executed and tested to definitively know the result set. You will view this property in step 6A below. In contrast, you will not see this property in the response schema for the service using a simple SQL statement because the result set output is guaranteed based on the runtime input parameters.

Db2 SQL statements and stored procedures with output parameters will have their JSON response fields included in the JSON response schema, because that information is included in the Db2 catalog.

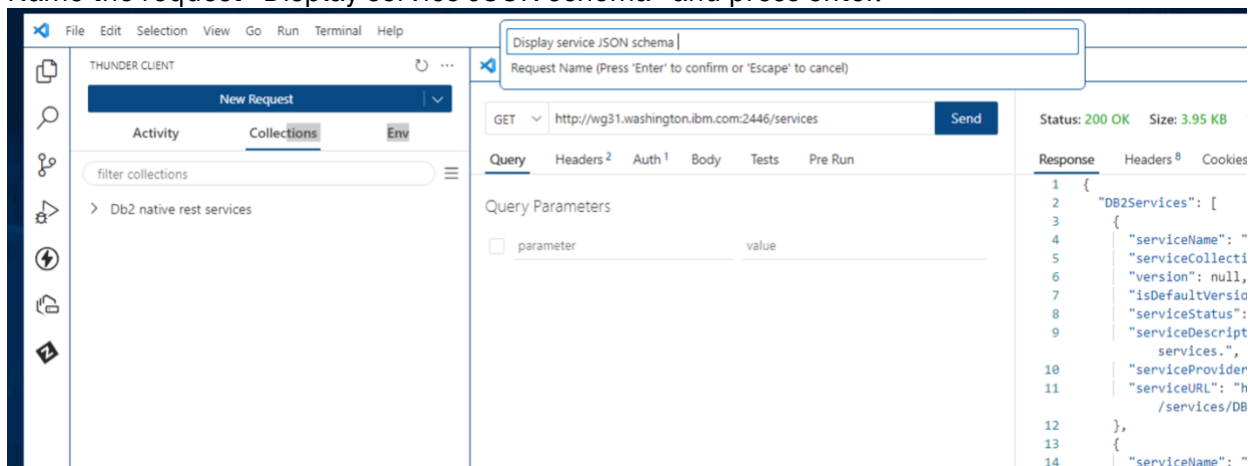
1.3.1 Display the Db2 Native REST service JSON schema information

In this section, regardless of which service you created – “selectByDeptSP” and/or “selectByDeptSTMT” – you will only create one request that can be used for each option. The request information required only differs in the **Db2 service URI**, which is detailed in [step 5B](#) below.

- __ 1. Return to the Thunder Client VSCode Extension. Create a new request in the “Db2 Native REST” Collection, by selecting the “New Request” option within the collection’s menu.



- __ 2. Name the request “Display service JSON schema” and press enter.



- __ 3. Update the request with the information below to view the created service’s JSON schema.

- A. Set the REST Method to: GET
- B. To view the JSON schema information for a specific service, the appropriate corresponding Db2 service URI must be used. Select the schema for the corresponding service creation option that you chose in **Step 5 of 1.2**.

To view the JSON schema for the service calling a Stored Procedure fill in the request body information with the URI found in step i below.

To view the JSON schema for the service calling a SQL Statement fill in the request body information with the URI found in step ii below.

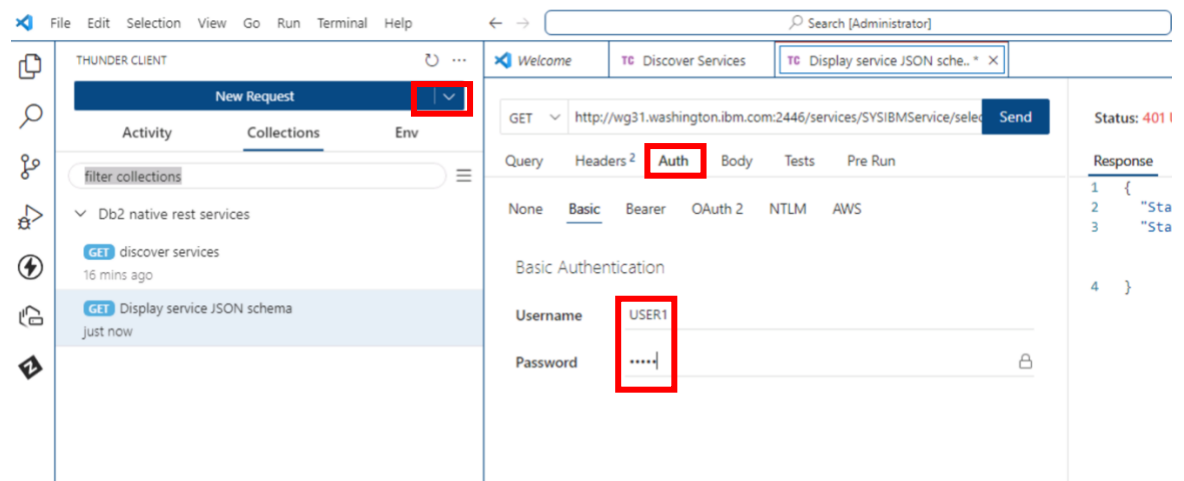
i. **FOR STORED PROCEDURE**

<http://wg31.washington.ibm.com:2446/services/SYSIBMSERVICE/selectByDeptSP>

- (1) Db2 DNS name = wg31.washington.ibm.com
- (2) Db2 port = 2446
- (3) Db2 service URI = /services/SYSIBMSERVICE/selectByDeptSP

Select “Auth” from the header and “Basic” in the sub-header to enter the following information:

Username: USER1
 Password: USER1



OR

FOR SQL STATEMENT

<http://wg31.washington.ibm.com:2446/services/SYSIBMSERVICE/selectByDeptSTMT>

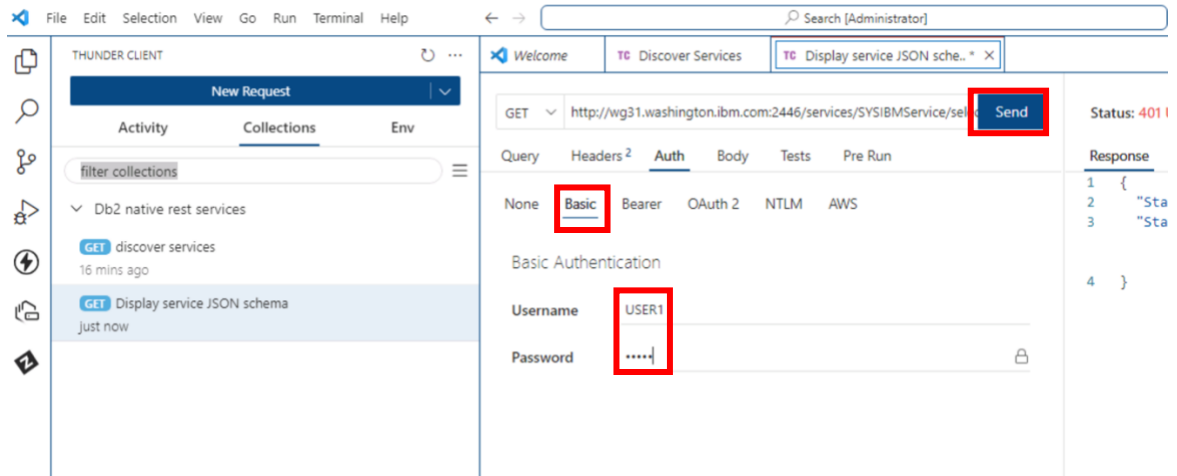
- (1) Db2 DNS name = **wg31.washington.ibm.com**
- (2) Db2 port = **2446**

(3) Db2 service URI = **/services/SYSIBMSERVICE/selectByDeptSTMT**

Select “**Auth**” from the header and “**Basic**” in the sub-header to enter the following information:

Username: USER1

Password: USER1



- __ 4. Click **SEND** to invoke the command. If you created both services and would like to view both schemas, send one request, view the response. Then update the Db2 service URI with the other URI, and send the request again to view the response.
- __ 5. The Db2 REST service schema information will be displayed in the response. Confirm the following output below for the services that you created, corresponding to which option you chose in Step 5 of 1.2.

The output of the service using the **Stored Procedure** is found in **step A** below.

The output for the service using the **SQL Statement** is found in **step B** below.

A. FOR STORED PROCEDURE: “selectByDeptSP” schema information

- i. Listed below are highlighted sections with the request schema fields and data types for the **stored procedure**. In order to send a successful REST request to Db2, this schema indicates what must be included in the request body in order to execute the service. Review the required fields and their expected corresponding data types below. You will use this information later to execute the service.
 - (1) whichQuery = integer
 - (2) department1 = string
 - (3) department2 = string

Response	Headers ⁸	Cookies	Results	Docs	{}	:
12	"RequestSchema": {					
13	"\$schema": "http://json-schema.org/draft-04/schema#",					
14	"type": "object",					
15	"properties": {					
16	"whichQuery": {					
17	"type": [
18	"null",					
19	"integer"					
20],					
21	"multipleOf": 1,					
22	"minimum": -2147483648,					
23	"maximum": 2147483647,					
24	"description": "Nullable INTEGER"					
25	}					
26	"department1": {					
27	"type": [
28	"null",					
29	"string"					
30],					
31	"maxLength": 3,					
32	"description": "Nullable CHAR(3)"					
33	}					
34	"department2": {					
35	"type": [
36	"null",					
37	"string"					
38],					
39	"maxLength": 3,					
40	"description": "Nullable CHAR(3)"					
41	}					
42	}					
43	"required": [
44	"whichQuery",					
45	"department1",					
46	"department2"					
47]					
48	"description": "Service selectByDeptSP invocation HTTP request body"					

From this information we can imagine what a request body should entail. For example, here is a sample JSON request that we could use to invoke the service:

```
{
  "whichQuery": 2,
  "department1": "A00",
  "department2": "E00"
}
```

- ii. Scroll down in the schema and notice the JSON **response** schema result set type, "Anonymous ResultSets". As described at the beginning of the section, this indicates that the number of results sets to be returned is ambiguous, due to the fact that the results are dependent on the input parameters the stored procedure.

```

49   },
50   "ResponseSchema": {
51     "$schema": "http://json-schema.org/draft-04/schema#",
52     "type": "object",
53     "properties": {
54       "ResultSet 1 Output": {
55         "description": "Stored Procedure ResultSet 1 Data",
56         "type": "array",
57         "items": {
58           "description": "ResultSet Row",
59           "type": "object"
60         }
61       },
62       "Anonymous ResultSets": {
63         "type": "integer",
64         "multipleOf": 1,
65         "minimum": 0,
66         "maximum": 1,
67         "description": "Number of Anonymous ResultSets"
68       },
69       "StatusDescription": {
70         "type": "string",
71         "description": "Service invocation status description"
72       },
73       "StatusCode": {
74         "type": "integer",
75         "multipleOf": 1,
76         "minimum": 100,
77         "maximum": 600,
78         "description": "Service invocation HTTP status code"
79       }
80     },
81     "required": [
82       "StatusDescription",

```

- iii. Reviewing other sections of the **response** schema, you will see what data will be returned from Db2 and how you will receive that data. Even though we know that the number of results sets will vary, due to the “Anonymous ResultSet” property, the result set output will be returned as an array of items that will be of the type “object”.

```

49   },
50   "ResponseSchema": {
51     "$schema": "http://json-schema.org/draft-04/schema#",
52     "type": "object",
53     "properties": {
54       "ResultSet 1 Output": {
55         "description": "Stored Procedure ResultSet 1 Data",
56         "type": "array",
57         "items": {
58           "description": "ResultSet Row",
59           "type": "object"
60         }
61       }
62     }
63   }

```

OR

B. FOR SQL STATEMENT: “selectByDeptSTMT” schema information

- i. Listed below are highlighted sections with the request schema field names and data types for the SQL statement. In order to send a successful REST request to the Db2, this schema indicates what must be included in the request body in order to execute the service. Review the required fields and their expected corresponding data types below. You will use this information later in the lab to execute the service.

INDEPTNO = string

```
11     "serviceStatus": "started",
12     "RequestSchema": {
13       "$schema": "http://json-schema.org/draft-04/schema#",
14       "type": "object",
15       "properties": {
16         "INDEPTNO": {
17           "type": [
18             "null",
19             "string"
20           ],
21           "maxLength": 3,
22           "description": "Nullable CHAR(3)"
23         }
24       },
25       "required": [
26         "INDEPTNO"
27       ],
28       "description": "Service selectByDeptSTMT invocation HTTP request
        body"
```

From this information we can imagine what a request body should entail. For example, here is a sample JSON request that we could use to invoke the service:

```
{"INDEPTNO": "A00"}
```

- ii. Scroll down further in the schema and notice the JSON **response** schema result set type, DOES NOT include the type “Anonymous ResultSets”. As described at the beginning of the section, this is because this service uses a simple SQL statement and the result set output is known. Instead, the schema indicates that the requested set of data will be returned.


```
Status: 200 OK Size: 1.63 KB Time: 49 ms

Response Headers8 Cookies Results Docs {}

29 },
30 "ResponseSchema": {
31   "$schema": "http://json-schema.org/draft-04/schema#",
32   "type": "object",
33   "properties": {
34     "ResultSet Output": {
35       "type": "array",
36       "items": {
37         "type": "object",
38         "properties": {
39           "FIRSTNAME": {
40             "type": "string",
41             "maxLength": 12,
42             "description": "VARCHAR(12)"
43           },
44           "LASTNAME": {
45             "type": "string",
46             "maxLength": 15,
47             "description": "VARCHAR(15)"
48           },
49           "PHONENO": {
50             "type": [
51               "null",
52               "string"
53             ],
54             "maxLength": 4,
55             "description": "Nullable CHAR(4)"
56           },
57           "WORKDEPT": {
58             "type": [
59               "null",
60               "string"
61             ],
62             "maxLength": 3,
```

- iii. Reviewing other sections of the response schema, you will see what data will be returned from Db2 and how you will receive that data. Selecting an employee by department will return an array of object types that indicate the employee's first name, middle initial, last name, phone number, and department code.

```
30   "responseschema": {
31     "$schema": "http://json-schema.org/draft-04/schema#",
32     "type": "object",
33     "properties": {
34       "ResultSet Output": {
35         "type": "array",
36         "items": {
37           "type": "object",
38           "properties": {
39             "FIRSTNAME": {
40               "type": "string",
41               "maxLength": 12,
42               "description": "VARCHAR(12)"
43             },
44             "LASTNAME": {
45               "type": "string",
46               "maxLength": 15,
47               "description": "VARCHAR(15)"
48             },
49             "PHONENO": {
50               "type": [
51                 "null",
52                 "string"
53               ],
54               "maxLength": 4,
55               "description": "Nullable CHAR(4)"
56             },
57             "WORKDEPT": {
58               "type": [
59                 "null",
60                 "string"
61               ],
62               "maxLength": 3,
63               "description": "Nullable CHAR(3)"
64             }
65           },
66           "required": [
```

__ 6. Upon reviewing the schema information for the service(s), now know what is required to execute the service(s) successfully, and what the result set output should look like when it is returned.

Summary: In this section, we viewed the JSON schema of the service(s) that we created in section 1.3.



1.3.2 Executing the Db2 REST service

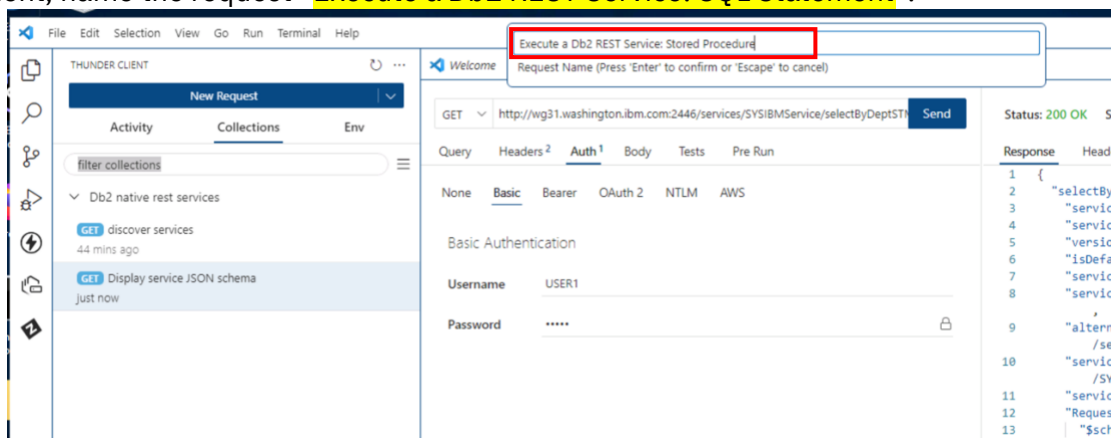
In this exercise you will execute the Db2 REST service(s) that has(have) been created in [section 1.3](#). If you created both services and would like to execute them both in this section, for organization and clarity, execute the following steps twice. Only at step 5 should you follow the specific instructions for your request.

For the create request, you will add two custom headers to the request. The headers indicate that the REST service is using JSON for the information transmitted. The Header Fields:

- The Accept field will be set to `application/json`
 - Defines that JSON will be used for the response
- The Content-Type will be set to `application/json`
 - Defines that JSON will be used for the request body

___ 1. Create a new request in the “Db2 Native REST Services” Collection, as detailed in [section 1.3.1 step 1](#).

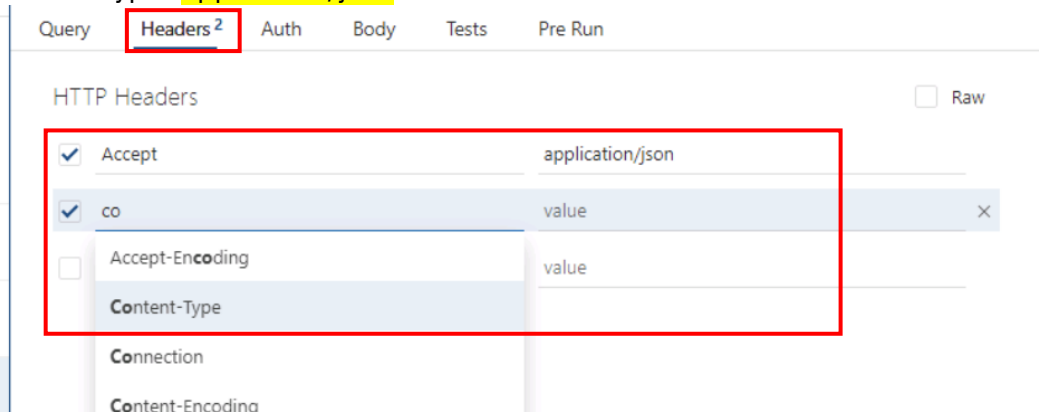
___ 2. For stored procedure, name the request “Execute a Db2 REST Service: Stored Procedure”. For SQL statement, name the request “Execute a Db2 REST Service: SQL Statement”.



___ 3. Navigate to the **Headers** tab in your newly created request to populate the appropriate REST Header fields.

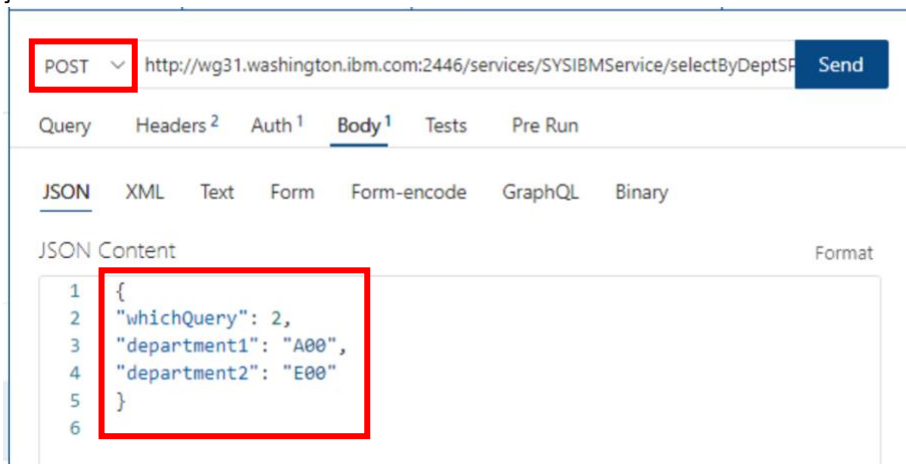
- A. In the Key column type “Accept”, and in its corresponding Value column type “application/json”.

- B. Add another header by typing “Content-Type” in the Key column, and in its corresponding Value column type “application/json.”



4. If the service was created using a stored procedure, follow the steps in step A. If the service was created using a SQL statement, follow the steps in step B.
- A. **FOR STORED PROCEDURE:** Update the request with the information below to execute the Db2 service we created in section 1.2. The required input parameters we reviewed in section 1.4.1 resides in the JSON body.
- i. Set the REST Method to: **POST**
 - ii. Add the Db2 REST URL:
<http://wg31.washington.ibm.com:2446/services/SYSIBMSERVICE/selectByDeptSP>
 - (1) Db2 DNS name = **wg31.washington.ibm.com**
 - (2) Db2 port = **2446**
 - (3) Db2 Service URI = **/services/SYSIBMSERVICE/selectByDeptSP**
 - iii. Create the request JSON body by entering all the fields needed to execute the Db2 Service we created in section 1.3 (“selectByDeptSP”), provided below. Click on the **Body** tab and then the JSON button.


```
{
"whichQuery": 2,
"department1": "A00",
"department2": "E00"
}
```



- (1) The which query variable is an integer, so no quotation marks are present.

- (2) The fields “department1” and “department2” have string variables and include quotation marks.

Information



We know this is the information that the REST Service is expecting because we reviewed the JSON schema for **selectByDeptSP** in [section 1.3.1](#).

An application developer would use the same process in order to use the service in their application.

- B. **FOR SQL STATEMENT:** Update the request with the information below to execute the Db2 service we created in section 1.3. The required input parameters we reviewed in [section 1.4.1](#) resides in the JSON body.
- i. Set the REST Method to: **POST**
 - ii. Add the Db2 REST URL:
<http://wg31.washington.ibm.com:2446/services/SYSIBMSERVICE/selectByDeptSTMT>
 - (1) Db2 DNS name = **wg31.washington.ibm.com**
 - (2) Db2 port = **2446**
 - (3) Db2 Service URI = **/services/SYSIBMSERVICE/selectByDeptSTMT**
 - iii. Create the request JSON body by entering all the fields needed to execute the Db2 Service we created in section 1.3 (“selectByDeptSTMT”), provided below. Click on the **Body** tab and then the JSON tab.

```
{
  "INDEPTNO": "A00"
}
```

- (1) The INDEPTNO variable is a character string that maps to WORKDEPT.

Information



We know this is the information that the REST Service is expecting because we reviewed the JSON schema for **selectByDeptSTMT** in [section 1.3.1](#).

An application developer would use the same process in order to use the service in their application.

- __ 5. Whether you are executing the request using the provided stored procedure or a simple SQL statement, click **SEND** to invoke the command. In step 7 below, review the result set output corresponding to your service.
- __ 6. Execute service response. **Confirm the output below.**
- A. The first item for review is the status code should be **200 OK**. The status code **200 OK** will be the normal successful return code for Db2 services.

Status: **200 OK** Size: 3.65 KB Time: 70 ms

Response Headers⁸ Cookies Results Docs

```

1  {
2  "Output Parameters": {},
3  "ResultSet 1 Output": [
4    {
5      "employeeNumber": "000010",
6      "firstName": "CHRISTINE",
7      "middleInitial": "I",
8      "lastName": "HAAS",
9      "department": "A00",
10     "phoneNumber": "3978"
11   },
12   {

```

- i. The next item for review is the result set output that contains the employee information available in the response **Body** tab.
- (1) FOR STORED PROCEDURE: Scrolling down to review the entire response body, we can see that the employees belonging to the work departments within the range A00 to E00 have been returned, which corresponds to the input parameters that we sent in the request body when we executed the service “selectByDeptSP”.
- (2) FOR SQL STATEMENT: Scrolling down to review the entire response body, we can see that the employees belonging to the work department A00 have been returned, which corresponds to the input parameters that we sent in the request body when we executed the service “selectByDeptSTMT”.

Summary: In this section we executed the REST Service(s) that we created in step 1.3 In order to execute it properly, we viewed the JSON schema corresponding to the service created (selectByDeptSP and/or selectByDeptSTMT) and then executed the service in a request.

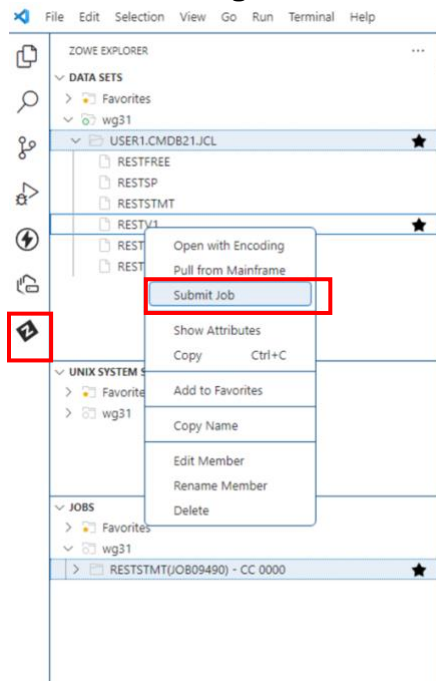


1.5 Versioning Db2 REST Services

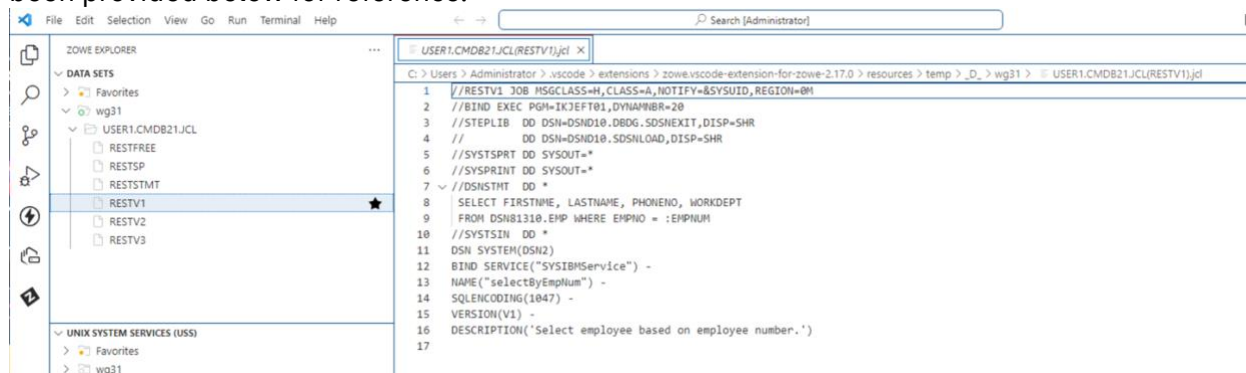
In this section of the lab, we will explore a Db2 REST Service versioning example. A versioned Db2 Native REST Service – using a SQL statement – has been provided for you, which you will view and then create two new versions and examine the different output result sets. To understand the service, we will view the service's schema, execute the service, and then create the new services.

1.5.1 Discover the Versioned Service

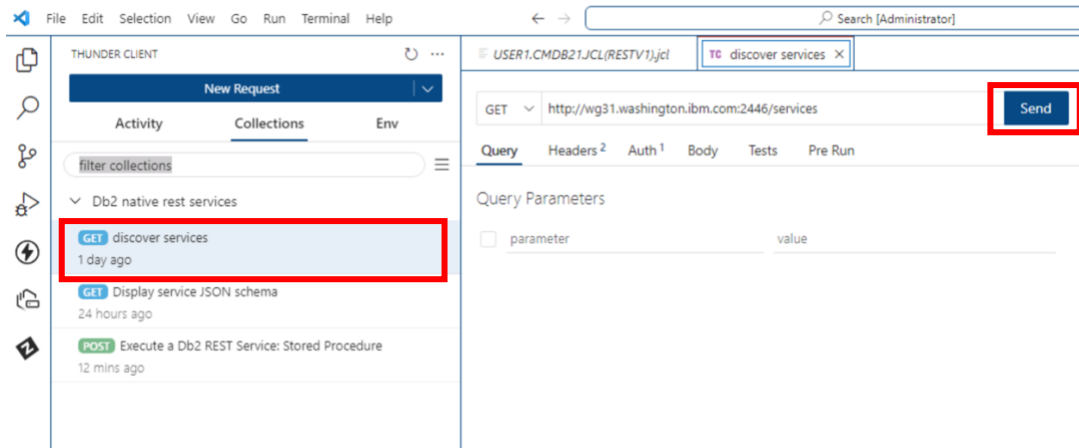
1. Navigate back to the Zowe extension to locate the JCL for the versions of your Db2 REST services. To create the “selectByEmpNum” service, right click on RESTV1 and click Submit to create the service. Do the same thing for RESTV2 and RESTV3 as well.



2. Review the JCL before submitting to understand the differences between each version. The JCL has been provided below for reference.



3. Navigate back to the Thunder client. Use the “Discover Services” request again to discover the versioned “selectByEmpNum” service after having submitted the RESTV1 JCL. Click **SEND** to invoke the command.



4. Scroll down and find the versioned service “selectByEmpNum” in the discover response output body. Review the version number and different parameters, next we will view the service schema located at the serviceURL.

```

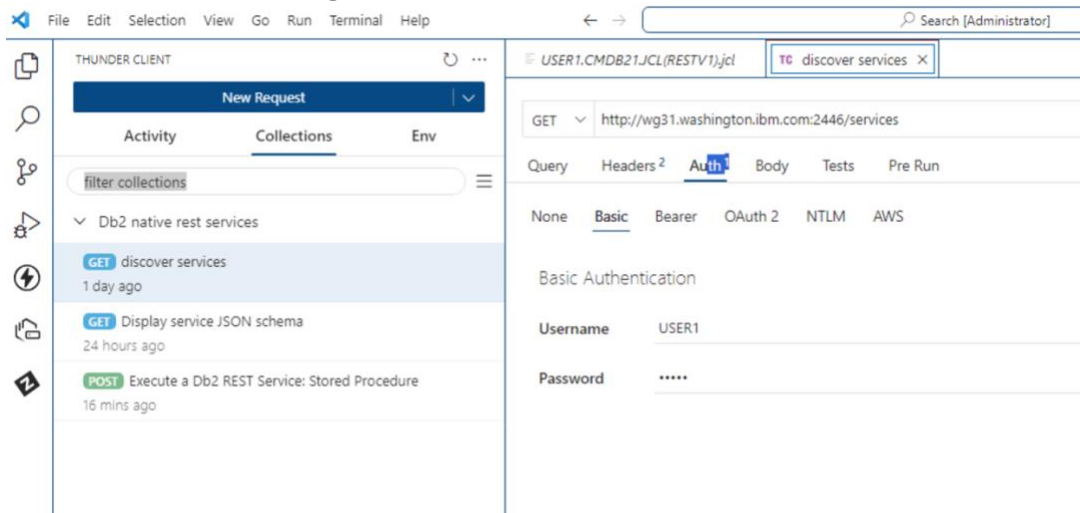
44     },
45     {
46         "serviceName": "selectByEmpNum",
47         "serviceCollectionID": "SYSIBMService",
48         "version": "V1",
49         "isDefaultVersion": true,
50         "serviceStatus": "started",
51         "serviceDescription": "Select employee based on employee number
        .",
52         "serviceProvider": "db2service-1.0",
53         "serviceURL": "http://wg31.washington.ibm.com:2446/services
        /SYSIBMService/selectByEmpNum/V1",
54         "alternateServiceURL": "http://wg31.washington.ibm.com:2446
        /services/SYSIBMService/selectByEmpNum"
55     }

```

Summary: Here we quickly discovered the information for the provided service “selectByEmpNum”.

1.5.2 View the JSON schema of the service

- __ 1. Use the “Display service JSON schema” request again to learn more about the versioned service “selectByEmpNum”.
 - A. Open the “**Display service JSON schema**” request again in the “Db2 Native REST Services” collection from side navigation bar if you had previously closed the request.

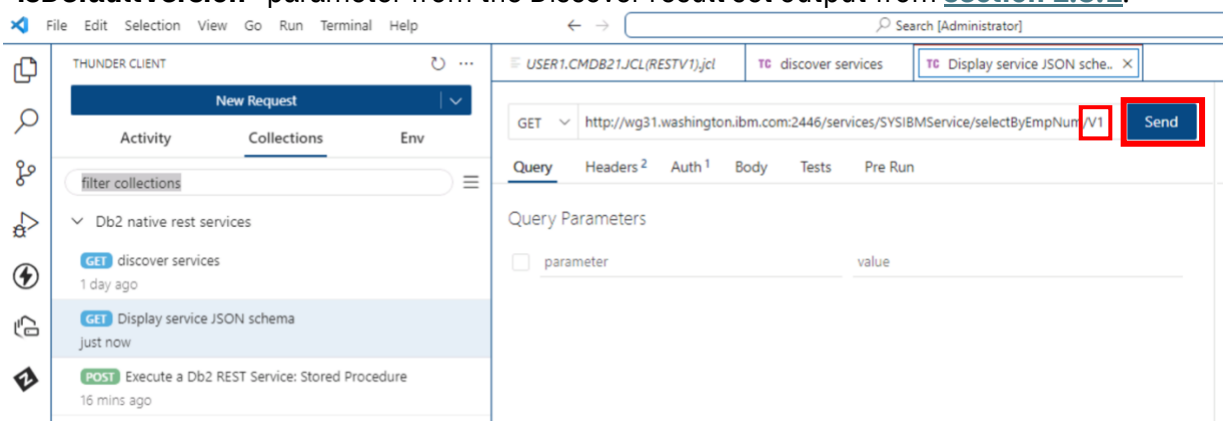


- __ 2. Update the request with the information below to view the versioned service’s JSON schema.
 - A. The REST method should remain GET.
 - B. Add the “selectByEmpNum” serviceURL:

<http://wg31.washington.ibm.com:2446/services/SYSIBMSERVICE/selectByEmpNum/V1>

- i. Db2 DNS name = wg31.washington.ibm.com
- ii. Db2 port = 2446
- iii. Db2 service URL = /services/SYSIBMSERVICE/selectByEmpNum/
- iv. Service version = **V1**

Notice that the version number – **V1** – is included. If this parameter is not included in the URL, then the default version of the service will be used, which is indicated by the “**isDefaultVersion**” parameter from the Discover result set output from [section 1.5.1](#).



- __ 3. Click SEND to invoke the command.
- __ 4. The Db2 REST service schema information will be displayed in the response. Confirm the following output below for the services that you created.
 - A. Listed below are highlighted sections with the **request** schema fields and data types for the **selectByEmpNum/V1** service. In order to send a successful REST request to Db2, this schema

indicates what must be included in the request body in order to execute the service. Review the required fields and their expected corresponding data types below. You will use this information later to execute the service.

- i. employeeNumber = string

```

Status: 200 OK Size: 1.61 KB Time: 64 ms

Response Headers Cookies Results Docs {}
1 {
2   "selectByEmpNum": {
3     "serviceName": "selectByEmpNum",
4     "serviceCollectionID": "SYSIBMSERVICE",
5     "version": "V1",
6     "isDefaultVersion": true,
7     "serviceProvider": "db2service-1.0",
8     "serviceDescription": "Select employee based on employee
9     number.",
10    "alternateServiceURL": "http://wg31.washington.ibm.com:2446
11    /services/SYSIBMSERVICE/selectByEmpNum",
12    "serviceURL": "http://wg31.washington.ibm.com:2446/services
13    /SYSIBMSERVICE/selectByEmpNum/V1",
14    "serviceStatus": "started",
15    "RequestSchema": {
16      "$schema": "http://json-schema.org/draft-04/schema#",
17      "type": "object",
18      "properties": {
19        "EMPNUM": {
20          "type": [
21            "null",
22            "string"
23          ],
24          "maxLength": 6,
25          "description": "Nullable CHAR(6)"
26        }
27      },
28      "required": [
29        "EMPNUM"
30      ],
31      "description": "Service selectByEmpNum invocation HTTP
32      request body"
33    }
34  }
35 }

```

From this information we can imagine what a request body should entail. For example, here is a sample JSON request that we could use to invoke the service:

```

{
  "EMPNUM": "000010"
}

```

- B. Reviewing other sections of the **response** schema, you will see what data will be returned from Db2 and how you will receive that data.

Summary: In this section, we reviewed the JSON schema for the service selectByEmpNum, to understand how to execute the service and see how the data will be returned in the response.

1.5.3 Execute the versioned service

__ 1. To execute the versioned service, you may use either of the “Execute a Db2 REST Service: ...” requests – Stored Procedure or SQL Statement.

A. Open one of the “Execute a Db2 REST Service: ...” requests again in the “Db2 Native REST Services” collection from side navigation bar if you had previously closed the request.

__ 2. Update the request with the information below to execute the versioned service.

A. The REST Method should remain: **POST**

B. Add the “selectByEmpNum” serviceURL:

<http://wg31.washington.ibm.com:2446/services/SYSIBMSERVICE/selectByEmpNum/V1>

i. Db2 DNS name = **wg31.washington.ibm.com**

ii. Db2 port = **2446**

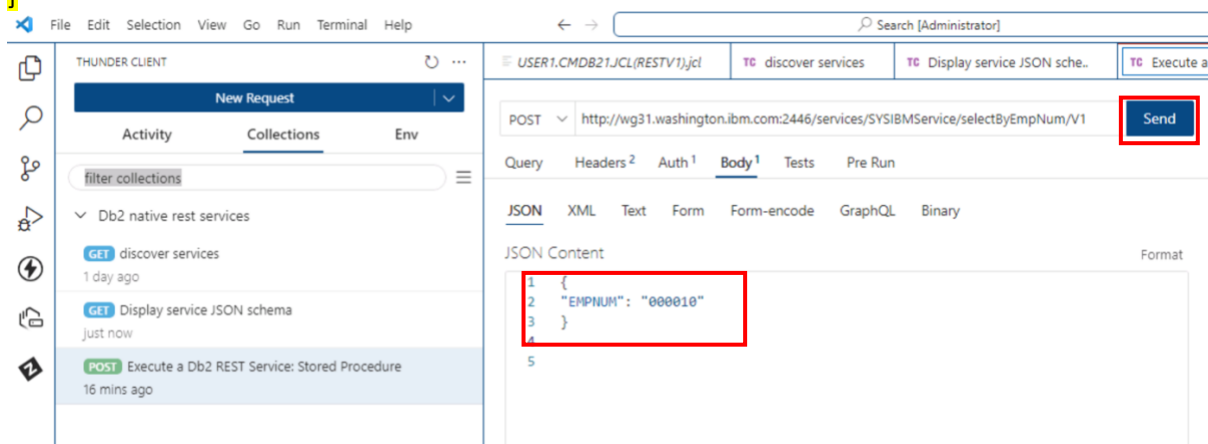
iii. Db2 service URL = **/services/SYSIBMSERVICE/selectByEmpNum/**

iv. Service version = **V1**

Notice that the version number – **V1** – is included. If this parameter is not included in the URL, then the default version of the service will be used, which is indicated by the “**isDefaultVersion**” parameter from the Discover result set output in [section 1.6.1 step 1C](#).

C. Add the service’s required parameters to the JSON Body of the request:

```
{
  "EMPNUM": "000010"
}
```



__ 3. Click SEND to invoke the request.

__ 4. Execute Service Response. **Confirm the proper output below:**

A. The first item for review is the status code should be 200 OK. The status code 200 OK will be the normal successful return code for Db2 services.

- B. The next item for review is the result set output that contains the employee information available in the response Body tab.



The screenshot shows a REST client interface with the following details:

- Status: 200 OK (highlighted in a red box)
- Size: 161 Bytes
- Time: 64 ms
- Response tab is selected, showing a JSON body:

```
1 {
2   "ResultSet Output": [
3     {
4       "FIRSTNAME": "CHRISTINE",
5       "LASTNAME": "HAAS",
6       "PHONENO": "3978",
7       "WORKDEPT": "A00"
8     }
9   ],
10  "StatusCode": 200,
11  "StatusDescription": "Execution Successful"
12 }
```

- i. Upon executing the service we can see that employee information pertaining to the employee number that was sent to Db2 with the REST request was returned. In the following section, we will create new versions of this same service such that more information will be returned, by selecting data with more complex SQL statements.

Summary: In this section we executed the service that was provided – selectByEmpNum – to understand the service’s default version behavior, and result set output. In the following section we will create two new versions, that will expand upon this result set.

1.5.4 Create new versions of the service

1. To create a new version of the “selectByEmpNum” service, use the provided JCL to submit batch jobs in the same way you completed [section 1.3](#) about executing services.

A. Navigate back to Zowe.

B. Locate the JCL for the versioned services under USER1.CMDB21.JCL.

```

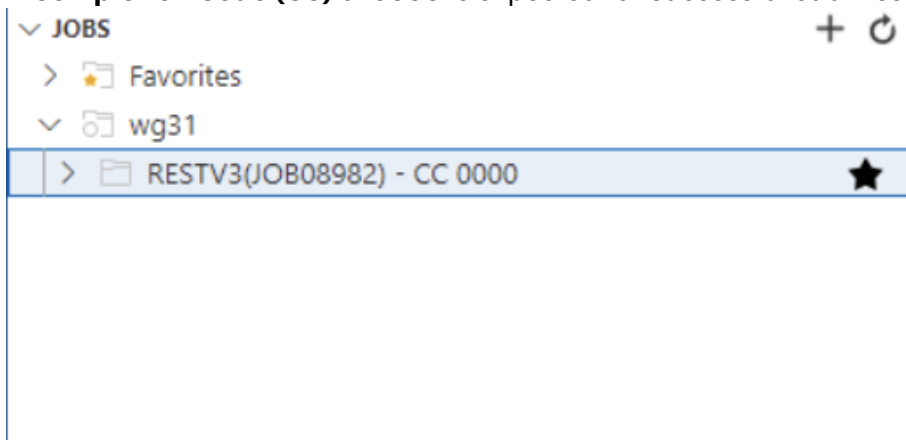
1 //RESTV2 JOB MSGCLASS=M,CLASS=M,NOTIFY=BSYSUID,REGION=M
2 //RIND EXEC PGM=IKJEFT01,DYNAMIBR=20
3 //STEPLIB DD DSN=DSN010.D,DISP=SDR
4 // DD DSN=DSN010.D,DISP=SDR
5 //SYSPRINT DD SYSOUT=*
6 //SYSPRINT DD SYSOUT=*
7 //DSNEXIT DD *
8 SELECT E.FIRSTNM, E.LASTNM, E.PHONO, E.WORDEPT,
9 H.LASTNM AS MANAGER FROM DSNB1210.EMP E, DSNB1210.EMP H,
10 DSNB1210.DPT D WHERE E.EMPNO = :EMPNO AND E.WORDEPT = D.DPTNO
11 and D.HRNO = H.EMPNO
12 //SYSJOB DD *
13 DSN SYSTEM(DSN2)
14 BIND SERVICE("SYSBMSERVICE") -
15 NAME("selectByEmpNum") -
16 SQLCODE(1847) -
17 VERSION(V2) -
18 DESCRIPTION("Select employee based on employee number, includes the emp-
19 loyee's manager.")
20
  
```

2. Right click on RESTV2 and RESTV3, and click **Submit** to create the services. Review the JCL before submitting to understand the differences between each version. The JCL has been provided below for reference.

```

1 //RESTV3 JOB MSGCLASS=M,CLASS=M,NOTIFY=BSYSUID,REGION=M
2 //RIND EXEC PGM=IKJEFT01,DYNAMIBR=20
3 //STEPLIB DD DSN=DSN010.D,DISP=SDR
4 // DD DSN=DSN010.D,DISP=SDR
5 //SYSPRINT DD SYSOUT=*
6 //SYSPRINT DD SYSOUT=*
7 //DSNEXIT DD *
8 SELECT E.FIRSTNM, E.LASTNM, E.PHONO, E.WORDEPT,
9 H.LASTNM AS MANAGER, H.PHONO AS PPHONO FROM DSNB1210.EMP
10 E, DSNB1210.EMP H, DSNB1210.DPT D WHERE E.EMPNO = :EMPNO AND
11 E.WORDEPT = D.DPTNO and D.HRNO = H.EMPNO
12 //SYSJOB DD *
13 DSN SYSTEM(DSN2)
14 BIND SERVICE("SYSBMSERVICE") -
15 NAME("selectByEmpNum") -
16 SQLCODE(1847) -
17 VERSION(V3) -
18 DESCRIPTION("Select employee based on employee number, includes the emp-
19 loyee's manager.")
20
  
```

- __ 3. When the status message with the job ID pops up, click the hyperlink to verify the job ran successfully. A **Completion Code (CC)** of **0000** is expected for successful submission.

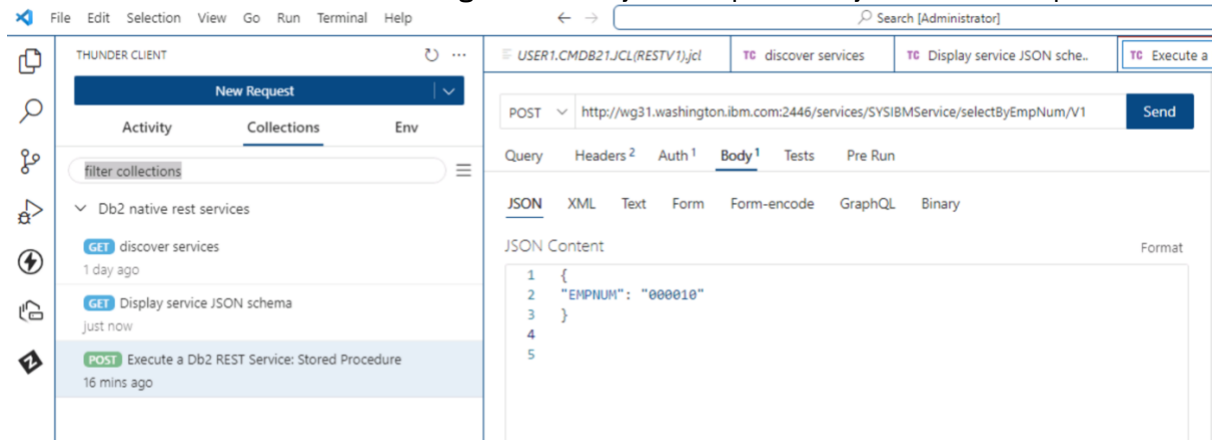


Summary: In this section we created two new versions of the existing service “selectByEmpNum” using the provided JCL. Version 2 returns the employee’s manager in the result set, and version 3 includes the employee’s manager and the manager’s phone number in the result set. In the following section, we will see how these services compare when executed.

1.5.5 Execute the new versioned services

__ 1. To execute the versioned services, you may use either of the “Execute a Db2 REST Service: ...” requests – Stored Procedure or SQL Statement.

- A. Open one of the “Execute a Db2 REST Service: ...” requests again in the “Db2 Native REST Services” collection from side navigation bar if you had previously closed the request.



__ 2. Update the request with the information below to execute the second version of the “selectByEmpNum” service that includes the employee’s manager in the response.

- A. The REST Method should remain: **POST**
- B. Add the “selectByEmpNum” serviceURL:

<http://wg31.washington.ibm.com:2446/services/SYSIBMSERVICE/selectByEmpNum/V2>

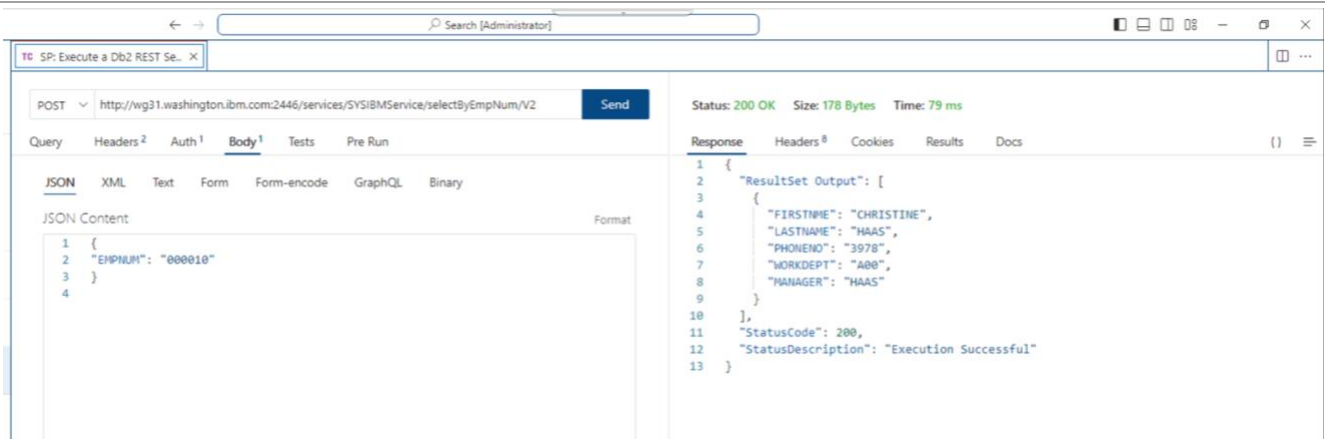
- i. Db2 DNS name = **wg31.washington.ibm.com**
- ii. Db2 port = **2446**
- iii. Db2 service URL = **/services/SYSIBMSERVICE/selectByEmpNum/**
- iv. Service version = **V2**

*****DO NOT FORGET TO UPDATE THE VERSION NUMBER*****

If this parameter is not included in the URL, then the default version of the service will be executed, which is indicated by the “isDefaultVersion” parameter from the Discover result set output in [section 1.5.1 step 3](#).

C. Add the services required parameters to the JSON Body of the request:

```
{
  "EMPNUM": "000010"
}
```



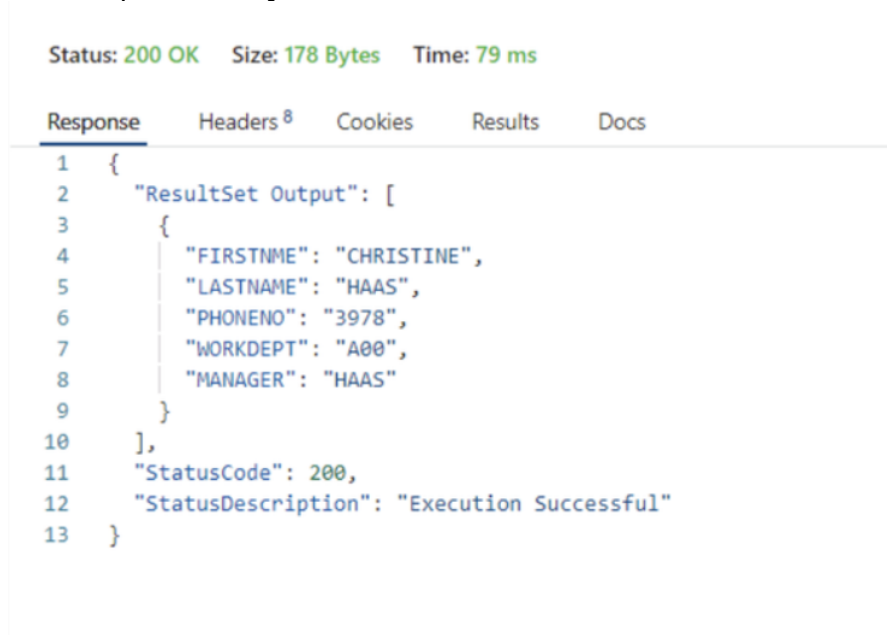
i. The EMPNUM variable is a character string that maps to EMPNO.

__ 3. Click **SEND** to invoke the request.

__ 4. Execute Service Response. Confirm the proper output below:

A. The first item for review is the status code should be **200 OK**. The status code **200 OK** will be the normal successful return code for Db2 services.

B. The next item for review is the result set output that contains the employee information available in the response **Body** tab.



i. Upon executing the service, we can see that now the manager's last name is included in the result set output when compared to the execution of the original service V1.

__ 5. Upon successful execution of the second version of the service, execute the third version of the service that includes the manager's phone number in the response. In the "Execute a Db2 REST Service: ..." request, update the fields with the following information:

- A. The REST Method should remain: **POST**
- B. Add the “selectByEmpNum” serviceURL:

http://wg31.washington.ibm.com:2446/services/SYSIBMSERVICE/selectByEmpNum/V3

- i. Db2 DNS name = **wg31.washington.ibm.com**
- ii. Db2 port = **2446**
- iii. Db2 service URL = **/services/SYSIBMSERVICE/selectByEmpNum/**
- iv. Service version = **V3**

*****DO NOT FORGET TO UPDATE THE VERSION NUMBER*****

If this parameter is not included in the URL, then the default version of the service will be executed, which is indicated by the “**isDefaultVersion**” parameter from the Discover result set output in [section 1.5.1](#).

- C. Add the service’s required parameters to the JSON Body of the request:

```
{
  "EMPNUM": "000010"
}
```

- 1. The EMPNUM variable is a character string that maps to EMPNO.

__ 6. Click **SEND** to invoke the request.

__ 7. Execute Service Response. **Confirm the proper output below:**

- A. The first item for review is the status code should be **200 OK**. The status code **200 OK** will be the normal successful return code for Db2 services.
- B. The next item for review is the result set output that contains the employee information available in the response **Body** tab.

```
Status: 200 OK Size: 196 Bytes Time: 43 ms
Response Headers Cookies Results Docs
1 {
2   "ResultSet Output": [
3     {
4       "FIRSTNAME": "CHRISTINE",
5       "LASTNAME": "HAAS",
6       "PHONENO": "3978",
7       "WORKDEPT": "A00",
8       "MANAGER": "HAAS",
9       "NGRPHONE": "3978"
10    }
11  ],
12  "StatusCode": 200,
13  "StatusDescription": "Execution Successful"
14 }
```

- i. Upon executing the service, we can see that now the manager’s last name AND phone number is included in the result set output when compared to the execution of the original service V1.

Summary: In this section we executed the newly created versions of the service that was provided - selectByEmpNum. By employing this technique, there can be multiple services with the same name that perform similar tasks depending on what the desired output is.

1.6 Summary

During this lab, you learned how quickly a stored procedure and/or a SQL statement could be used to create a Native Db2 REST Service. Upon reviewing the Db2 system information and the sample Db2 Stored Procedure, you accomplished the following using the REST client – Postman – to manage, view, and execute the Native REST Services:

1. Used the discover API to view the installed services on DB2.
2. Created a service using a stored procedure and/or a SQL statement.
3. Viewed the service(s) schema(s) to understand the required parameters needed to execute the service(s), as well as understand how the service would behave when called.
4. Executed the service using the information available in the service(s) schema(s).
5. Finally, explore the versioning capability of Db2 REST Services.