Db2 13 for z/OS

REST Services

Last updated: 2024-08-27

Notes

Before using this information and the product it supports, be sure to read the general information under "Notices" at the end of this information.

Subsequent editions of this PDF will not be delivered in IBM Publications Center. Always download the latest edition from IBM Documentation.

2024-08-27 edition

This edition applies to Db2® 13 for z/OS® (product number 5698-DB2®), Db2 13 for z/OS Value Unit Edition (product number 5698-DBV), and to any subsequent releases until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Specific changes are indicated by a vertical bar to the left of a change. A vertical bar to the left of a figure caption indicates that the figure has changed. Editorial changes that have no technical significance are not noted.

Contents

About this information

Throughout this information, "Db2" means "Db2 13 for z/OS". References to other Db2 products use complete names or specific abbreviations.

Important: To find the most up to date content for Db2 13 for z/OS, always use IBM® Documentation or download the latest PDF file from PDF format manuals for Db2 13 for z/OS (Db2 for z/OS in IBM Documentation).

Most documentation topics for Db2 13 for z/OS assume that the highest available function level is activated and that your applications are running with the highest available application compatibility level, with the following exceptions:

- The following documentation sections describe the Db2 13 migration process and how to activate new capabilities in function levels:
  - Migrating to Db2 13 (Db2 Installation and Migration)
  - What's new in Db2 13 (Db2 for z/OS What's New?)
  - Adopting new capabilities in Db2 13 continuous delivery (Db2 for z/OS What's New?)
- FL 500 A label like this one usually marks documentation changed for function level 500 or higher, with a link to the description of the function level that introduces the change in Db2 13. For more information, see How Db2 function levels are documented (Db2 for z/OS What's New?).

The availability of new function in Db2 13 depends on the type of enhancement, the activated function level, and the application compatibility levels of the applications. For a list of all available function levels in Db2 13, see Db2 13 function levels (Db2 for z/OS What's New?).

Function level 100
Db2 starts at function level 100 (V13R1M100) during migration to Db2 13, and fallback and coexistence with Db2 12 in data sharing remain possible. Many new capabilities in Db2 13 remain unavailable. For more information, see Function level 100 (for migrating to Db2 13 - May 2022) (Db2 for z/OS What's New?).

Function level 500
Activating function level 500 (V13R1M500) prevents coexistence with and fallback to Db2 12. Function level 500 is also the first opportunity for applications to use many of the new capabilities in Db2 13. However, new capabilities that depend on Db2 13 catalog changes remain unavailable. For more information, see Function level 500 (for migrating to Db2 13 - May 2022) (Db2 for z/OS What's New?).

Function level 501
Function level 501 (V13R1M501) is the first opportunity after migration to Db2 13 for applications to use new features and capabilities that depend on catalog changes in Db2 13. For more information, see Function level 501 (Db2 13 installation or migration - May 2022) (Db2 for z/OS What's New?).

Some virtual storage and optimization enhancements take effect in function level 100. Optimization enhancements become available after full prepare of the SQL statements, depending on the statement type:

- For static SQL statements, after bind or rebind of the package.
- For non-stabilized dynamic SQL statements, immediately, unless the statement is in the dynamic statement cache.
- For stabilized dynamic SQL statements, after invalidation, free, or changed application compatibility level.

Who should read this information

This information is primarily intended for people who are responsible for developing REST APIs in a Db2 for z/OS environment. It assumes that the user is familiar with the following concepts:

- The basic concepts and facilities of Db2 for z/OS environment
- The basic concepts of Structured Query Language (SQL)

Db2 Utilities Suite for z/OS

Important: Db2 Utilities Suite for z/OS is available as an optional product. You must separately order and purchase a license to such utilities, and discussion of those utility functions in this publication is not intended to otherwise imply that you have a license to them.

Db2 13 utilities can use the DFSORT program regardless of whether you purchased a license for DFSORT on your system. For more information about DFSORT, see https://www.ibm.com/support/pages/dfsort.

Db2 utilities can use IBM Db2 Sort for z/OS as an alternative to DFSORT for utility SORT and MERGE functions. Use of Db2 Sort for z/OS requires the purchase of a Db2 Sort for z/OS license. For more information about Db2 Sort for z/OS, see Db2 Sort for z/OS documentation.

Related concepts

Db2 utilities packaging (Db2 Utilities)

Terminology and citations

When referring to a Db2 product other than Db2 for z/OS, this information uses the product's full name to avoid ambiguity.

The following terms are used as indicated:

Db2
Represents either the Db2 licensed program or a particular Db2 subsystem.

IBM OMEGAMON® for Db2 Performance Expert on z/OS
Refers to any of the following products:

- IBM IBM OMEGAMON for Db2 Performance Expert on z/OS
- IBM Db2 Performance Monitor on z/OS
- IBM Db2 Performance Expert for Multiplatforms and Workgroups
- IBM Db2 Buffer Pool Analyzer for z/OS

C, C++, and C language
Represent the C or C++ programming language.

CICS®
Represents CICS Transaction Server for z/OS.

IMS
Represents the IMS Database Manager or IMS Transaction Manager.

MVS™
Represents the MVS element of the z/OS operating system, which is equivalent to the Base Control Program (BCP) component of the z/OS operating system.

RACF®
Represents the functions that are provided by the RACF component of the z/OS Security Server.

Accessibility features for Db2 for z/OS

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in z/OS products, including Db2 for z/OS. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers and screen magnifiers.
- Customization of display attributes such as color, contrast, and font size

Tip: IBM Documentation (which includes information for Db2 for z/OS) and its related publications are accessibility-enabled for the IBM Home Page Reader. You can operate all features using the keyboard instead of the mouse.

Keyboard navigation

For information about navigating the Db2 for z/OS ISPF panels using TSO/E or ISPF, refer to the *z/OS TSO/E Primer*, the *z/OS TSO/E User's Guide*, and the *z/OS ISPF User's Guide*. These guides describe how to navigate each interface, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Related accessibility information

IBM and accessibility

See the *IBM Accessibility Center* at http://www.ibm.com/able for more information about the commitment that IBM has to accessibility.

How to send your comments about Db2 for z/OS documentation

Your feedback helps IBM to provide quality documentation.

Send any comments about Db2 for z/OS and related product documentation by email to db2zinfo@us.ibm.com.

To help us respond to your comment, include the following information in your email:

- The product name and version
- The address (URL) of the page, for comments about online documentation
- The book name and publication date, for comments about PDF manuals
- The topic or section title
- The specific text that you are commenting about and your comment

Related concepts

About Db2 13 for z/OS product documentation (Db2 for z/OS in IBM Documentation)

Related reference

PDF format manuals for Db2 13 for z/OS (Db2 for z/OS in IBM Documentation)

How to read syntax diagrams

Certain conventions apply to the syntax diagrams that are used in IBM documentation.

Apply the following rules when reading the syntax diagrams that are used in Db2 for z/OS documentation:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

  The ►►—— symbol indicates the beginning of a statement.

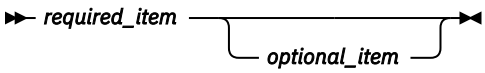
  The ——► symbol indicates that the statement syntax is continued on the next line.

  The ►—— symbol indicates that a statement is continued from the previous line.

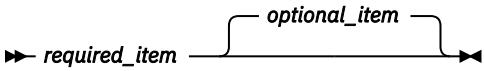
  The ——►◄ symbol indicates the end of a statement.
- Required items appear on the horizontal line (the main path).

- Optional items appear below the main path.

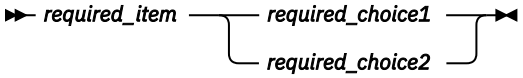
  

  If an optional item appears above the main path, that item has no effect on the execution of the statement and is used only for readability.

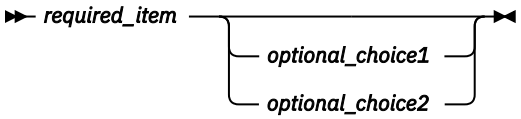
  

- If you can choose from two or more items, they appear vertically, in a stack.

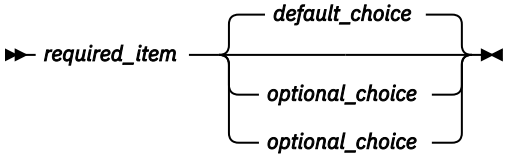
  If you *must* choose one of the items, one item of the stack appears on the main path.

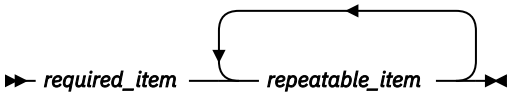
  If choosing one of the items is optional, the entire stack appears below the main path.

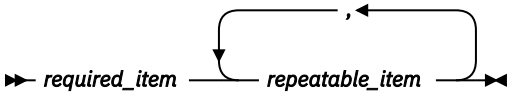
  If one of the items is the default, it appears above the main path and the remaining choices are shown below.

- An arrow returning to the left, above the main line, indicates an item that can be repeated.

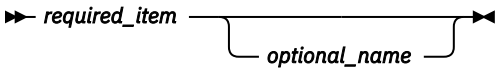
  If the repeat arrow contains a comma, you must separate repeated items with a comma.

  A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Sometimes a diagram must be split into fragments. The syntax fragment is shown separately from the main syntax diagram, but the contents of the fragment should be read as if they are on the main path of the diagram.

fragment-name



- For some references in syntax diagrams, you must follow any rules described in the description for that diagram, and also rules that are described in other syntax diagrams. For example:
  - For *expression*, you must also follow the rules described in Expressions (Db2 SQL).
  - For references to *fullselect*, you must also follow the rules described in fullselect (Db2 SQL).
  - For references to *search-condition*, you must also follow the rules described in Search conditions (Db2 SQL).
- With the exception of XPath keywords, keywords appear in uppercase (for example, FROM). Keywords must be spelled exactly as shown.
- XPath keywords are defined as lowercase names, and must be spelled exactly as shown.
- Variables appear in all lowercase letters (for example, *column-name*). They represent user-supplied names or values.
- If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

Related concepts

Commands in Db2 (Db2 Commands)
Db2 online utilities (Db2 Utilities)
Db2 stand-alone utilities (Db2 Utilities)

Chapter 1. Enabling Db2 REST services

Before you can use Db2 as a REST service provider, you must create a database, a table space, a table, and an index that are required by the services.

Procedure

To enable the creation of Db2 REST services:

1. Apply the following APARs:
   - APAR PI70652. This APAR is the minimum level that is required.
   - APAR PI98649, which enables versioning for Db2 REST services.
   - Any subsequent available Db2 REST services maintenance.
2. Configure and run sample job DSNTIJRS in the SDSNSAMP library to create database DSNSVCDB, table space DSNSVCTS, table SYSIBM.DSNSERVICE, and index SYSIBM.DSNSVC01.

   Db2 uses the SYSIBM.DSNSERVICE table to describe REST services and associate them with corresponding packages.
3. Configure and run sample job DSNTIJR2 to enable Db2 REST service versioning support.

Related reference

Db2 REST services table (SYSIBM.DSNSERVICE) (Db2-supplied user tables)

Chapter 2. REST services versioning

Db2 REST service versioning support gives you the ability to define and use multiple versions of a REST service concurrently. Each version of a REST service can be created using a different SQL statement, different input and output parameters, and different bind options. This model provides the flexibility to create and deploy new versions of REST services to satisfy new application requirements without breaking existing applications.

Consider using Db2 REST service versioning if any of the following applies to you:

- You need to keep the existing version of a REST service available while you create another version.
- You want your applications and consumers of a REST service to use a specific version of a REST service.
- You have a REST service with versions using different SQL statements, different inputs and output parameters, and bind options.
- You want Db2 REST service discovery and invocation package authorizations to be common across multiple versions of a REST service.
- You want to specify a version string on Db2 REST services.

REST services created prior to enabling REST versioning support are version-less, and have the following attributes:

- A version-less REST service definition is the default service version for the <collection id>.<service name>.
- No other versions of a version-less <collection id>.<service name> can be created.
- A version-less REST service will have an empty-string (“”) value for the VERSION column in both SYSPACKAGE and DSNSERVICES tables.
- Version-less REST services will continue to be referenced using the default version URI: /services/<collection id>/<service name>

REST services created after enabling the REST versioning support will always have an assigned version, and have following attributes:

- The initial Db2 REST service that is created for a given <collection id>.<service name> is set as the default service version.
- The default service version of a Db2 REST service can be changed using the REBIND PACKAGE (DSN) subcommand with the RESTSERVICEDEFAULT bind option (Db2 Commands).
- The default service version can be referenced using either the explicit version URI: /services/<collection id>/<service name>/<version> or the default service version URI: /services/<collection id>/<service name>
- Any REST service created without a user specified version will be created using the default version value, V1.
- A valid REST service version has one to 64 characters, each of which is an uppercase letter, lowercase letter, numerals (0-9), the underscore (“_”) character, the dash (“-“) character, the period (“.”) character, the ampersand (“@”) character, the pound sign (“#”) character, or the dollar sign (“$”) character.
- The default service version of a REST service cannot be dropped or freed when other versions of that service exist.
- One or more versions of a REST service can exist at any point in time at the current server, but only one version of a REST service is considered the default version. When you first create a REST service, that initial version is considered the default version of the REST service.

Enabling REST services versioning support

Db2 REST service versioning support is enabled by running sample installation job DSNTIJR2. If you are using Db2 data sharing, the required maintenance must be installed and active on all members of the data sharing group. For more information, see the instructions and comments provided in the sample job and Chapter 1, “Enabling Db2 REST services,” on page 1.

REST service URI format with versioning

A new Db2 REST service URI format is needed to support the use, discovery, and invocation of versioned REST services.

The basic Db2 REST service URI format is /services/*<collection-id>*/*<service-name>*. If the *<collection id>* portion is omitted, then the default “SYSIBMSERVICE” collection ID is used. This basic Db2 REST service URI will continue to work for all existing Db2 REST services and will be fully supported even after the Db2 REST services versioning support is enabled.

Once the Db2 REST services versioning support is enabled, an additional Db2 REST URI format will be available for versioned REST services. The new URI format will support the version identifier in the following format: /services/*<collection-id>*/*<service-name>*/*<version-id>*

With this change, the valid Db2 REST service URI variations and mappings can be done in three ways.

Specified Variables	URI Format	Explanation
Service Name	/services/*<service-name>*	SYSIBMSERVICE, the default collection ID and the default service version of REST service <service-name> is referenced.
Collection and service name	/services/*<collection-id>*/*<service-name>*	The default service version of the REST service <collection-id>/<service-name> is referenced.
Collection, service name, and version	/services/*<collection-id>*/*<service-name>*/*<version-id>*	The fully qualified REST service and version is referenced.

The default version for a REST service is the first REST service created for the given <collection-id>/<service-name>. A later version of the REST service can be selected as the default using the DSN REBIND PACKAGE subcommand with the RESTSERVICEDEFAULT bind option. The default version can only be dropped or freed when there are no other versions.

Example use of versioning

The Db2 REST service, *MyServices.createCustomer*, is created using the default version V1, and the SQL statement:

```
INSERT INTO CUSTOMER (FNAME, LNAME) VALUES(:firstName, :lastName)
```

Using the Db2 REST service manager API, the default version V1 of the *MyServices.createCustomer* service could be created by issuing a POST request to the createService API with the following HTTP request body:

```
{
 "requestType": "createService",
 "sqlStmt": "INSERT INTO CUSTOMER (FNAME, LNAME) VALUES(:firstName, :lastName)",
 "collectionID": "MyServices",
 "serviceName": "createCustomer",
 "description": "Create a new customer profile entry.",
}
```

Because this is the first service with the name *MyServices.createCustomer*, this version of the service is the default version. The service could be invoked using the default version URI of /services/*MyServices*/*createCustomer*, or by using the version specific URI of /services/*MyServices*/*createCustomer*/*V1*.

A new version of the *MyServices.createCustomer* service is needed to include the customer's e-mail address. The new version of the *MyServices.createCustomer* is created with "V2" specified as the version and the SQL statement:

```
INSERT INTO CUSTOMER (FNAME, LNAME, EMAIL) VALUES(:firstName, :lastName, :emailAddress)
```

Using the Db2 REST service manager API, version V2 of the *MyServices.createCustomer* service could be created by issuing a POST request to the createService API with the following HTTP request body:

```
{
 "requestType": "createService",
 "sqlStmt": "INSERT INTO CUSTOMER (FNAME, LNAME, EMAIL)
VALUES(:firstName, :lastName, :emailAddress)",
 "collectionID": "MyServices",
 "serviceName": "createCustomer",
 "description": "Create a new customer profile entry with customer name and email.",
 "version": "V2"
}
```

The V2 version of *MyServices.createCustomer* service is invoked by using the version specific URI of /services/*MyServices*/*createCustomer*/*V2*. There is no impact to any users of version V1 of the *MyServices.createCustomer service*. In addition, the new V2 version of *MyServices.createCustomer* shares all of the same package authorizations that were defined for version V1 of the *MyServices.createCustomer* service.

Later, a new V3 version of the *MyServices.createCustomer* service is needed so that the CUSTNUM (customer number) generated column value can be returned to the caller when creating a new customer account. The new version V3 of the *MyServices.createCustomer* is created using the SQL statement:

```
SELECT CUSTNUM AS :customerAcctNum FROM
FINAL TABLE (INSERT INTO CUSTOMER (FNAME, LNAME, EMAIL)
VALUES(:firstName, :lastName, :emailAddress))
```

Using the Db2 REST service manager API, version V3 of the *MyServices.createCustomer* service could be created by issuing a POST request to the createService API with the following HTTP request body:

```
{
 "requestType": "createService",
 "sqlStmt": "SELECT CUSTNUM AS :customerAcctNum FROM
            FINAL TABLE (INSERT INTO CUSTOMER (FNAME, LNAME, EMAIL)
            VALUES(:firstName, :lastName, :emailAddress))",
 "collectionID": "MyServices",
 "serviceName": "createCustomer",
 "description": "Create a new customer profile entry with customer name and email, returning
the new customer number.",
 "version": "V3"
}
```

The new V3 version of *MyServices.createCustomer* service is invoked by using the version specific URI of /services/*MyServices*/*createCustomer*/*V3*, and shares the same Db2 package authorizations as V1 and V2.

Related tasks

Rebinding a package (Db2 Application programming and SQL)

Chapter 3. Authorization of REST service users

You can use a Db2 REST profile in the RACF DSNR resource class to manage a user's access to the Db2 REST service APIs.

To authorize access, issue the RACF RDEFINE command to define a Db2 REST protected access profile in an active DSNR resource class. A REST protected access profile has a name in the form of *subsystem.environment*, where *subsystem* is the name of a Db2 subsystem and *environment* denotes the REST environment. For example, you can issue the following command to define profile DB2T.REST for subsystem DB2T with a universal access authority of NONE:

```
RDEFINE DSNR (DB2T.REST) OWNER(DB2OWNER) UACC(NONE)
```

You can then issue the RACF PERMIT command to give users access to profile DB2T.REST. For example, you can issue the following command to authorize user DB2USER1 access to profile DB2T.REST:

```
PERMIT DB2T.REST CLASS(DSNR) ID(DB2USER1)  ACCESS(READ)
```

Db2USER1 can now access Db2 by using Db2 REST service APIs.

Related tasks

Establishing RACF protection for Db2 (Managing Security)

Chapter 4. Authentication of REST service requests

Db2 uses HTTP basic authentication or client certificate authentication to authenticate all HTTP REST service connection requests.

Db2 REST HTTP basic authentication

Db2 uses HTTP basic authentication when you specify the user ID and password or a RACF PassTicket in the HTTP Authorization header. The HTTP basic authentication credentials can be in clear text or Base64 encoding. Db2 uses the IBM System Authorization Facility (SAF) to authenticate the HTTP basic authentication credentials.

Db2 REST client certificate authentication

Certificate authentication is used when the following conditions are true:

- The connection to Db2 is a secure connection that uses HTTPS and AT-TLS.
- A client certificate that is registered with RACF or another SAF-compliant security product is presented to Db2.

Chapter 5. Db2 REST services trusted context and trusted connection support

If Db2 trusted context support is enabled, Db2 REST requests exploit it.

The way in which Db2 uses a trusted context depends on the types of authentication that are used to establish the REST connection:

- If either HTTP basic authentication or client certificate authentication, but not both, is used, Db2 establishes a connection in the following way:
  1. Db2 performs authentication and authorization processing using the presented credentials.
  2. If authentication is successful, Db2 searches for a trusted context definition with attributes that match the REST request connection attributes.
  3. If a matching trusted context is found, that trusted context is associated with the Db2 REST request, thereby creating a trusted connection.
  4. If a matching trusted context is not found, the Db2 REST request runs on a normal Db2 connection.
- If both HTTP basic authentication and client certificate authentication are used, Db2 establishes a connection in the following way:
  1. Db2 performs authentication and authorization processing using the client certificate credentials.
  2. If authentication is successful, Db2 uses the client certificate information to establish a trusted connection.
  3. After the trusted connection is established, Db2 performs authentication and authorization processing using the HTTP basic authentication credentials.
  4. If authentication is successful, Db2 uses the HTTP basic authentication credentials to perform a switch-user operation on the trusted connection.

  If any of the previous steps are unsuccessful, Db2 rejects the Db2 REST request and issues a security failure error.

Chapter 6. Security of REST service connections

Db2 supports HTTPS REST service requests by using the z/OS Communications Server IP Application Transparent Transport Layer Security (AT-TLS) capability. The policy-driven AT-TLS support is configured in the TCP/IP stack and performs the TLS check on behalf of Db2 by invoking the z/OS system SSL feature in the TCP layer.

To support HTTPS requests, you must use a secure port for SSL connections. Make sure that the DDF TCP/IP SQL Listener service is capable of listening to a secondary secure port for inbound SSL connections. DDF verifies that all connections to Db2 through the secure port are protected by AT-TLS policies.

Related concepts

Encrypting your data with Secure Socket Layer (SSL) support (Managing Security)

Chapter 7. Creating a Db2 REST service

You can use the Db2 REST service manager API, or createService API, to create a new user-defined service if you have the required authority. You can also issue the BIND SERVICE subcommand to create a new REST service.

Before you begin

Before you can create a service, you must apply APARs and create database objects that are required by the services. For instructions, see Chapter 1, “Enabling Db2 REST services,” on page 1.

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 SERVICE subcommand (DSN) (Db2 Commands) for information about required authorization.

Procedure

- To create a service, issue an HTTP or HTTPS POST request through a REST client with the following URI:

  ```
  POST https://<host>:<port>/services/DB2ServiceManager
  ```

- Set the HTTP header `Accept` and `Content-Type` fields to `application/json` for the request.
- Optional: Set any Db2 client information related HTTP header fields with values that will be assigned to the Db2 client information special registers. For more information, see Chapter 19, “Management of REST service client information,” on page 45.
- Specify the create service parameters using JSON format key/value pairs in the HTTP body for the request. The *requestType*, *sqlStmt*, *collection*, *serviceName*, *description*, and *version* JSON keys are case sensitive. General service create bind option keys are case insensitive. The *requestType*, *sqlStmt*, and *serviceName* parameters are required. To use the version create service parameter, REST service versioning must be enabled. Specify the create service HTTP body content:

  ```
  {
   "requestType": "createService",
   "sqlStmt": "<sqlStatement>",
   "collectionID": "<serviceCollectionID>",
   "serviceName": "<serviceName>",
   "description": "<serviceDescription>",
   "version": "<versionIdentifier>",
   "<bindOption>": "<bindOptionValue>",
  ...
   "<bindOption>": "<bindOptionValue>"
  }
  ```

  where

  – `createService` indicates that you request to create a new service.

  – *<sqlStatement>* is the SQL statement that you include in the new service. For each new service, you can embed a single CALL, DELETE, INSERT, SELECT, TRUNCATE, UPDATE, or WITH SQL statement. When you create the service, Db2 also binds a package that is used to invoke the service.

    Db2 adds a new row in the user-defined SYSIBM.DSNSERVICE catalog table for the service and saves the bound package in the directory. Db2 also sets the HOSTLANG column in the SYSIBM.SYSPACKAGE and SYSIBM.SYSPACKCOPY tables to 'R' to mark the package for the REST API.

  – *<serviceCollectionID>* is the collection identifier of the package that is associated with the new service. The *serviceCollectionID* property is optional. If you specify *<serviceCollectionID>*, Db2

names the package in the form of *collectionID.serviceName*. Otherwise, Db2 uses the default form of SYSIBMSERVICE.*serviceName*. The maximum length is 128 bytes.

If *<serviceCollectionID>* is a delimited identifier, the following characters are accepted: If *<serviceName>* is a delimited identifier, it can contain following characters: uppercase letters (A–Z), lowercase letters (a–z), numerals (0–9), underscore (_), at sign (@), pound sign (#), and dollar sign ($).

– *<serviceName>* is the name of the new service that you create. The maximum length is 128 bytes.

  If *<serviceName>* is a delimited identifier, the following characters are accepted characters: uppercase letters (A–Z), lowercase letters (a–z), numerals (0–9), underscore (_), at sign (@), pound sign (#), and dollar sign ($).

– *<serviceDescription>* is a brief description of the new service. The *serviceDescription* property is optional. If provided, Db2 stores the value in the DESCRIPTION column of the SYSIBM.DSNSERVICE catalog table.

– *<versionIdentifier>* is an optional version identifier for the service being created. This is only valid if REST service versioning support is enabled. The maximum length is 64 characters.

  If *<versionIdentifier>* is a delimited identifier, it can contain the following characters: If *<versionIdentifier>* is a delimited identifier, the following characters are accepted: uppercase letters (A–Z), lowercase letters (a–z), numerals (0–9), underscore (_), at sign (@), pound sign (#), and dollar sign ($).

– *<bindOption>* is the option that you specify for binding the package that is associated with the new service. The *bindOption* property is optional. All bind options supported by the BIND SERVICE subcommand, except DESCRIPTION, NAME, SQLDDNAME, SQLENCODING, and VERSION, apply to the `createService` API. The `createService` API uses the *serviceName*, *description*, and *version* parameters, instead of the NAME, DESCRIPTION, and VERSION bind options of the BIND SERVICE subcommand. See BIND SERVICE subcommand (DSN) (Db2 Commands) for supported bind options. See BIND and REBIND options for packages, plans, and services (Db2 Commands) for details about the bind options.

Example

This example shows how to create service `simpleSelect1` for OWNER DB2GRP1 and QUALIFIER USRT002 and includes setting the *Db2-Client-ApplName* and *Db2-Client-WrkStnName* HTTP request header fields to the values "Customer Service" and "CS Laptop-47", respectively. Enter the following URI to start an HTTPS POST request:

```
POST https://host:port/services/DB2ServiceManager
```

Specify the following HTTP header fields for the request:

```
Accept:application/json
Content-Type:application/json
Db2-Client-ApplName:Customer Service
Db2-Client-WrkStnName:CS Laptop-47
```

Specify the following HTTP body for the request:

```
{
 "requestType": "createService",
 "sqlStmt": "SELECT DEPTNAME FROM DSN8B10.DEPT WHERE LOCATION = :LOCATION",
 "collectionID": "SYSIBMSERVICE",
 "serviceName": "simpleSelect1",
 "description": "Select department name based on location.",
 "owner": "DB2GRP1",
 "qualifier": "USRT002"
}
```

Db2 returns the following response in JSON:

```
{
 "StatusCode":201,
```

```
"StatusDescription":"DB2 Rest Service simpleSelect1 was created successfully.",
"URL": "http://<host>:<port>/services/SYSIBMSERVICE/simpleSelect1/V1"
"StatusOptions": {
 "Applied":[
   {
    "ACTION": "ADD"
  },
   {
    "VALIDATE": "RUN"
  },
   {
    "EXPLAIN": "NO"
  },
   {
    "ISOLATION": "CS"
  },
   {
    "RELEASE": "COMMIT"
  },
   {
    "OWNER": "DB2GRP1"
  },
   {
    "QUALIFIER": "USRT002"
  },
   {
    "APREUSE": ""
  },
   {
    "APCOMPARE": ""
  },
   {
    "BUSTIMESENSITIVE": "YES"
  },
   {
    "SYSTIMESENSITIVE": "YES"
  },
   {
    "ARCHIVESENSITIVE": "YES"
  },
   {
    "APPLCOMPAT": "V12R1M500"
  },
   {
    "DESCSTAT": "YES"
  },
   {
    "SQLERROR": "NOPACKAGE"
  },
   {
    "CURRENTDATA": "NO"
  },
   {
    "DEGREE": "1"
  },
   {
    "NODEFER": "PREPARE"
  },
   {
    "REOPT": "NONE"
  },
   {
    "IMMEDWRITE": "NO"
  },
   {
    "DBPROTOCOL": "DRDA"
  },
   {
    "OPTHINT": ""
  },
   {
    "ENCODING": "UNICODE(01208)"
  },
   {
    "CONCURRENTACCESSRESOLUTION": ""
  },
   {
    "PATH": ""
  },
   {
    "QUERYACCELERATION": ""
  },
```

```
    {
      "GETACCELARCHIVE": ""
    },
    {
      "ACCELERATOR": ""
    },
}
```

The response shows that service `simpleSelect1` was successfully created. Since version was not specified, the version of the service created defaults to V1.

If necessary, you can issue the **REBIND PACKAGE** command to modify the options associated with service `simpleSelect1` .

Related concepts

Management of REST service client information

Db2 DDF implicitly defines special registers and a global variable to store client information about a valid HTTP REST service request. The client information is externalized in accounting records and the output of the -DISPLAY command.

Related tasks

Dropping a Db2 REST service

You can use the Db2 REST service manager API to drop a user-defined service if you have the required authority. Dropping a service removes the service, frees its associated package, and deletes the corresponding row from the SYSIBM.DSNSERVICE catalog table. You can also issue the FREE SERVICE subcommand to drop a REST service.

Related reference

BIND SERVICE subcommand (DSN) (Db2 Commands)

BIND and REBIND options for packages, plans, and services (Db2 Commands)

Chapter 8. Invoking a Db2 REST service

You can invoke a Db2 REST service through a RESTful HTTP client if you have the required authority.

Before you begin

You must have one of the following privileges or authorities to invoke a service:

- Execute privilege on the package for the service
- Ownership of the service
- SYSADM or SYSCTRL authority
- System DBADM.

Procedure

- To invoke a service, issue an HTTP or HTTPS POST request through a REST client with the following URI format:

  ```
  POST https://<host>:<port>/services/<serviceCollectionID>/<serviceName>/<version>
  ```

  where

  - *<serviceName>* is the name of the service to be invoked.
  - *<serviceCollectionID>* is the collection-id for the service to be invoked.
  - *<version>* is the version identifier for REST service version to be invoked. The *<version>* portion of the URI is optional and is only valid to be used after REST service versioning has been enabled. If the *<version>* portion of the URI is not specified, then the default service version will be invoked.
- Set the HTTP header `Accept` and `Content-Type` fields to `application/json` for the request.
- Optional: Set any Db2 client information related HTTP header fields with values that will be assigned to the Db2 client information special registers. For more information, see Chapter 19, “Management of REST service client information,” on page 45.
- Specify the required REST service input parameters using JSON format key/value pairs in the HTTP body for the request.

Example

This example shows how to invoke the default version of the service `SYSIBMSERVICE.simpleSelect1`, which has a single input String parameter named LOCATION. This example includes setting the *Db2-Client-ApplName* and *Db2-Client-WrkStnName* HTTP request header fields to the values "Customer Service" and "CS Laptop-47", respectively. Enter the following URI to start an HTTPS POST request:

```
POST https://<host>:<port>/services/SYSIBMSERVICE/simpleSelect1
```

Specify the following HTTP header fields for the request:

```
Accept:application/json
Content-Type:application/json
Db2-Client-ApplName:Customer Service
Db2-Client-WrkStnName:CS Laptop-47
```

Specify the following HTTP body for the request:

```
{
 "LOCATION": "<location>"
}
```

Db2 returns the following response in JSON:


```
{
 "ResultSet Output":
  [
   "DEPTNAME": "SPIFFY COMPUTER SERVICE DIV."
  },
  {
   "DEPTNAME": "PLANNING"
  },
  {
   "DEPTNAME": "INFORMATION CENTER"
  },
  {
   "DEPTNAME": "DEVELOPMENT CENTER"
  },
  {
   "DEPTNAME": "MANUFACTURING SYSTEMS"
  },
  {
   "DEPTNAME": "ADMINISTRATION SYSTEMS"
  },
  {
   "DEPTNAME": "SUPPORT SERVICES"
  },
  {
   "DEPTNAME": "OPERATIONS"
  },
  {
   "DEPTNAME": "SOFTWARE SUPPORT"
  },
  {
   "DEPTNAME": "BRANCH OFFICE F2"
  },
  {
   "DEPTNAME": "BRANCH OFFICE G2"
  },
  {
   "DEPTNAME": "BRANCH OFFICE H2"
  },
  {
   "DEPTNAME": "BRANCH OFFICE I2"
  },
  {
   "DEPTNAME": "BRANCH OFFICE J2"
  },
 "StatusCode": 200,
 "StatusDescription": "Execution Successful."
}
```


The response shows that service `simpleSelect1` was successfully invoked.

Related concepts

Management of REST service client information

Db2 DDF implicitly defines special registers and a global variable to store client information about a valid HTTP REST service request. The client information is externalized in accounting records and the output of the -DISPLAY command.

Related tasks

Creating a Db2 REST service

You can use the Db2 REST service manager API, or createService API, to create a new user-defined service if you have the required authority. You can also issue the BIND SERVICE subcommand to create a new REST service.

Chapter 9. Dropping a Db2 REST service

You can use the Db2 REST service manager API to drop a user-defined service if you have the required authority. Dropping a service removes the service, frees its associated package, and deletes the corresponding row from the SYSIBM.DSNSERVICE catalog table. You can also issue the FREE SERVICE subcommand to drop a REST service.

Before you begin

When you drop a service, you must have a privilege that allows the runner to free or drop the associated package. For required authorization, see FREE SERVICE command (DSN) (Db2 Commands).

Procedure

- To drop a service, issue an HTTP or HTTPS POST request through a REST client with the following URI:

  ```
  POST https://<host>:<port>/services/DB2ServiceManager
  ```

- Set the HTTP header `Accept` and `Content-Type` fields to `application/json` for the request.
- Optional: Set any Db2 client information related HTTP header fields with values that will be assigned to the Db2 client information special registers. For more information, see Chapter 19, “Management of REST service client information,” on page 45.
- Specify the following HTTP body for the request:

  ```
  {
   "requestType": "dropService",
   "collectionID": "<serviceCollectionID>",
   "serviceName": "<serviceName>",
   "version": "<version-id>"
  }
  ```

  where

  - `dropService` indicates that you request to remove a user-defined service.
  - *<serviceCollectionID>* is the identifier of the package that is associated with the service.
  - *<serviceName>* is the name of the service that you want to drop.
  - *<version-id>* is the version of the service you want to drop. If you omit the *<version-id>*, the default version of the REST service is deleted.

Example

This example shows how to drop the V1 version of service `simpleSelect1`, including setting the *Db2-Client-ApplName* and *Db2-Client-WrkStnName* HTTP request header fields to the values "Customer Service" and "CS Laptop-47", respectively. Enter the following URI to start an HTTPS POST request:

```
POST https://<host>:<port>/services/DB2ServiceManager
```

Specify the following HTTP header fields for the request:

```
Accept:application/json
Content-Type:application/json
Db2-Client-ApplName:Customer Service
Db2-Client-WrkStnName:CS Laptop-47
```

Specify the following HTTP body for the request:

```
{
 "requestType": "dropService",
 "collectionID": "SYSIBMSERVICE",
```

```
 "serviceName": "simpleSelect1",
 "version": "V1"
}
```

Db2 returns the following response in JSON:

```
{
 "StatusCode":200,
 "StatusDescription": "DB2 Rest Service SYSIBMSERVICE.simpleSelect1.(V1) was dropped
successfully."
}
```

The response shows that service `simpleSelect1` was successfully removed.

Related concepts

Management of REST service client information

Db2 DDF implicitly defines special registers and a global variable to store client information about a valid HTTP REST service request. The client information is externalized in accounting records and the output of the -DISPLAY command.

Related tasks

Creating a Db2 REST service

You can use the Db2 REST service manager API, or createService API, to create a new user-defined service if you have the required authority. You can also issue the BIND SERVICE subcommand to create a new REST service.

Related reference

FREE SERVICE command (DSN) (Db2 Commands)

BIND and REBIND options for packages, plans, and services (Db2 Commands)

Chapter 10. Freeing inactive packages for Db2 REST service

Removing the unneeded inactive package copies for Db2 REST services is a good practice, which can help to provide storage relief.

About this task

Db2 native REST services are created by invoking the REST service manager API with a create Service request, or by using the DSN BIND SERVICE command. This action inserts a row that define this service in he REST services table and creates an application package.

The application package for a REST service can then be rebound using various options by using the DSN REBINDPACKAGE command. During rebind, inactive package copies, for example the original and previous copies, can be generated based on the PLANMGMT option. Phased-out copies of packages can be generated if already running threads have the package allocated when the rebind occurs. Invalid package copies can occur when a dependent object is changed or dropped. These types of inactive packages can build up over time when many rebinds are done.

Procedure

To free only the inactive package copies for a Db2 REST service, use the DSN subcommand FREE PACKAGE.

For example, the following command frees all original, previous, and phased-out package copies for a REST service, regardless of whether they are invalid. The REST service definition and active package are unaffected.

```
FREE PACKAGE("SYSIBMSERVICE"."simpleSelect1".(V1)) PLANMGMTSCOPE(INACTIVE)
```

Related tasks

Creating a Db2 REST service
You can use the Db2 REST service manager API, or createService API, to create a new user-defined service if you have the required authority. You can also issue the BIND SERVICE subcommand to create a new REST service.

Related reference

FREE PACKAGE subcommand (DSN) (Db2 Commands)
Db2 REST services table (SYSIBM.DSNSERVICE) (Db2-supplied user tables)

Chapter 11. Discovering all Db2 REST services

You can use the Db2 REST service discover API to discover all available services if you have the required authority.

Before you begin

You must have one of the following privileges or authorities to discover all Db2 REST services:

- Execute privilege on the package for the service
- Ownership of the service
- SYSADM or SYSCTRL authority
- System DBADM.

Procedure

- To discover all services, enter a URI to start either an HTTPS POST or GET request:
  - `GET https://<host>:<port>/services/`
  - `POST https://<host>:<port>/services/DB2ServiceDiscover`
- Set the HTTP header `Accept` and `Content-Type` fields to `application/json` for the request.
- Optional: Set any Db2 client information related HTTP header fields with values that will be assigned to the Db2 client information special registers. For more information, see Chapter 19, “Management of REST service client information,” on page 45.

Example

This example shows how to discover all services and includes setting the *Db2-Client-ApplName* and *Db2-Client-WrkStnName* HTTP request header fields to the values "Customer Service" and "CS Laptop-47", respectively. Enter the following URI to start an HTTPS POST request:

```
POST  https://<host>:<port>/services/DB2ServiceDiscover
```

Specify the following HTTP header fields for the request:

```
Accept:application/json
Content-Type:application/json
Db2-Client-ApplName:Customer Service
Db2-Client-WrkStnName:CS Laptop-47
```

Db2 returns a response in JSON like the following:

```
{
    "DB2Services": [
        {
            "serviceName": "DB2ServiceDiscover",
            "serviceCollectionID": null,
            "version": null,
            "isDefaultVersion": true,
            "serviceStatus": "started",
            "serviceDescription": "DB2 service to list all available services.",
            "serviceProvider": "db2service-1.0",
            "serviceURL": "<host>:<port>/services/DB2ServiceDiscover"
        },
        {
            "serviceName": "DB2ServiceManager",
            "serviceCollectionID": null,
            "version": null,
            "isDefaultVersion": true,
            "serviceStatus": "started",
            "serviceDescription": "DB2 service to create, drop, or alter a user defined
```

```
service.",
            "serviceProvider": "db2service-1.0",
            "serviceURL": "<host>:<port>/services/DB2ServiceManager"
        },
        {
            "serviceName": "callSPBigOuts",
            "serviceCollectionID": "SYSIBMSERVICE",
            "version": "",
            "isDefaultVersion": true,
            "serviceStatus": "started",
            "serviceDescription": "Call SP SYSADM.BigOuts. Multiple Varchar(32704) OUTPUT
PARMS.",
            "serviceProvider": "db2service-1.0",
            "serviceURL": "<host>:<port>/services/SYSIBMSERVICE/callSPBigOuts",
            "alternateServiceURL": "<host>:<port>/services/SYSIBMSERVICE/callSPBigOuts"
        },
        {
            "serviceName": "callSPDeptUpdate",
            "serviceCollectionID": "SYSIBMSERVICE",
            "version": "V1",
            "isDefaultVersion": true,
            "serviceStatus": "started",
            "serviceDescription": "Call SP SYSADM.DeptUpdateSP, which takes single input
NewLocation parm.",
            "serviceProvider": "db2service-1.0",
            "serviceURL": "<host>:<port>/services/SYSIBMSERVICE/callSPDeptUpdate/V1",
            "alternateServiceURL": "<host>:<port>/services/SYSIBMSERVICE/callSPDeptUpdate"
        },
        {
            "serviceName": "callSPRestT20",
            "serviceCollectionID": "SYSIBMSERVICE",
            "version": "",
            "isDefaultVersion": true,
            "serviceStatus": "started",
            "serviceDescription": "Call SP SYSADM.RestT20. Uses LITERALS for the input and
vchar, OUT int and vchar.",
            "serviceProvider": "db2service-1.0",
            "serviceURL": "<host>:<port>/services/SYSIBMSERVICE/callSPRestT20",
            "alternateServiceURL": "<host>:<port>/services/SYSIBMSERVICE/callSPRestT20"
        }
    ],
}
```

Related concepts

Management of REST service client information

Db2 DDF implicitly defines special registers and a global variable to store client information about a valid HTTP REST service request. The client information is externalized in accounting records and the output of the -DISPLAY command.

Related tasks

Discovering details of a Db2 REST service

You can use the Db2 REST service discover API to discover details of a service if you have the required authority.

Chapter 12. Discovering details of a Db2 REST service

You can use the Db2 REST service discover API to discover details of a service if you have the required authority.

Before you begin

You must have one of the following privileges or authorities to discover details of a Db2 REST service:

- Execute privilege on the package for the service
- Ownership of the service
- SYSADM or SYSCTRL authority
- System DBADM.

Procedure

- To discover service details, issue an HTTP or HTTPS GET request through a REST client with the following URI:

  ```
  GET https://<host>:<port>/services/<serviceCollectionID>/<serviceName>/<version>
  ```

  where

  – *<serviceName>* is the name of the new service that you specify.
  – *<serviceCollectionID>* is the identifier of the package that is associated with the new service.
  – *<version>* is the version identifier for REST service version to be discovered. The *<version>* portion of the URI is optional and is only valid to be used after REST service versioning has been enabled. If the *<version>* portion of the URI is not specified, then the default service version will be discovered.
- Set the HTTP header `Accept` and `Content-Type` fields to `application/json` for the request.
- Optional: Set any Db2 client information related HTTP header fields with values that will be assigned to the Db2 client information special registers. For more information, see Chapter 19, “Management of REST service client information,” on page 45.

Example

This example shows how to discover details of the "default service version" for the service `simpleSelect1` in the `simpleTests` collection ID and includes setting the *Db2-Client-ApplName* and *Db2-Client-WrkStnName* HTTP request header fields to the values "Customer Service" and "CS Laptop-47", respectively. Enter the following URI to start an HTTPS GET request:

```
GET https://<host>:<port>/services/simpleTests/simpleSelect1
```

Specify the following HTTP header fields for the request:

```
Accept:application/json
Content-Type:application/json
Db2-Client-ApplName:Customer Service
Db2-Client-WrkStnName:CS Laptop-47
```

Db2 returns the following response in JSON:

```
{
    "simpleSelect1":{
        "serviceURL":"<host>:<port>/services/simpleTests/simpleSelect1/V1",
        "serviceStatus":"started",
        "ResponseSchema":{
            "$schema":"http://json-schema.org/draft-04/schema#",
            "description":"Service simpleSelect1 invocation HTTP response body",
            "properties":{
```

```
                "StatusCode":{
                    "minimum":100,
                    "description":"Service invocation HTTP status code",
                    "maximum":600,
                    "multipleOf":1,
                    "type":"integer"
                },
                "ResultSet Output":{
                    "items":{
                        "description":"ResultSet Row",
                        "properties":{
                            "DEPTNAME":{
                                "maxLength":36,
                                "description":"VARCHAR(36)",
                                "type":"string"
                            }
                        },
                        "required":[
                            "DEPTNAME"
                        ],
                        "type":"object"
                    },
                    "type":"array"
                },
                "StatusDescription":{
                    "description":"Service invocation status description",
                    "type":"string"
                }
            },
            "required":[
                "StatusDescription",
                "StatusCode",
                "ResultSet Output"
            ],
            "type":"object"
        },
        "isDefaultVersion":true,
        "RequestSchema":{
            "$schema":"http://json-schema.org/draft-04/schema#",
            "description":"Service simpleSelect1 invocation HTTP request body",
            "properties":{
                "LOCATION":{
                    "maxLength":16,
                    "description":"Nullable CHAR(16)",
                    "type":[
                        "null",
                        "string"
                    ]
                }
            },
            "required":[
                "LOCATION"
            ],
            "type":"object"
        },
        "alternateServiceURL":"<host>:<port>/services/simpleTests/simpleSelect1",
        "serviceProvider":"db2service-1.0",
        "serviceDescription":"Select deptname based on location.",
        "version":"V1",
        "serviceCollectionID":"simpleTests",
        "serviceName":"simpleSelect1"
    }
}
```


Related concepts

Management of REST service client information

Db2 DDF implicitly defines special registers and a global variable to store client information about a valid HTTP REST service request. The client information is externalized in accounting records and the output of the -DISPLAY command.

Related tasks

Discovering all Db2 REST services

You can use the Db2 REST service discover API to discover all available services if you have the required authority.

Chapter 13. Copying Db2 REST services

You can use the DSN BIND SERVICE subcommand with the COPY option to copy existing REST services to either a local or remote Db2 server.

Before you begin

- See BIND SERVICE subcommand (DSN) (Db2 Commands) for information about required authorization to issue the BIND SERVICE subcommand. Additionally, you must hold the COPY privilege or its equivalent at the local server.
- The local server must be at least Db2 12 for z/OS with REST services versioning enabled. See Chapter 2, “ REST services versioning,” on page 3.
- When copying to another location, the remote server must be at least Db2 12 for z/OS with REST services versioning enabled.
- Only versioned services can be copied. Copying version-less services is not supported.

Procedure

- To copy a REST service, specify the new collection-id using the SERVICE option. Use the COPY and COPYVER options to specify the *collection-id*, *service name*, and *version-id* of an existing REST service you want to copy from. Other bind options may be optionally specified, see BIND SERVICE subcommand (DSN) (Db2 Commands) for more options information.

Examples

Example 1: To copy the existing REST service "simpleTests"."simpleSelect1" with a *version-id* of V1, to another *collection-id*, "finalTests", on the local server, issue the following DSN subcommand.

```
BIND SERVICE("finalTests") COPY("simpleTests"."simpleSelect1") COPYVER(V1)
```

Example 2: The following example copies the existing REST service "payroll"."getEmployeeSalary" with a *version-id* of Ver2, to a remote server with the *location-name* PRODSYS. The new service has the same *collection-id* as the original service. The QUALIFIER option is also specified for the new service which determines the implicit qualifier for unqualified names of tables, views, indexes, and aliases contained in the service created on the PRODSYS system. Lastly, the DESCRIPTION option is used to describe the newly copied service.

```
BIND SERVICE(PRODSYS."payroll") COPY("payroll"."getEmployeeSalary") COPYVER(Ver2)
QUALIFIER(PRODQUAL) DESCRIPTION('This is the PRODSYS copy of getEmployeeSalary')
```

Chapter 14. Enabling CORS support for Db2 REST services

You can enable Cross-Origin Resource Sharing (CORS) support for Db2 to permit a web page or application to access remote content from a different domain (or port) than the site that the web page was loaded from. You can enable Db2 REST services to use the HTTP Cross-Origin Resource Sharing (CORS) protocols, including support for the CORS "pre-flight" HTTP OPTIONS verb and CORS HTTP request/response header fields.

Before you begin

Apply the PTF for APAR PH59837.

If you want to use the REST CORS functionality before the availability of the RACF module ICHRRCDX update that delivers the new DSNRAUTH class definition, your z/OS RACF security administrator can temporarily create the DSNRAUTH class using the RACF dynamic class descriptor table (CDT) support. For more information, see Chapter 16, “Creating a temporary DSNRAUTH class by using the RACF dynamic class descriptor table,” on page 39.

About this task

Cross-Origin Resource Sharing (CORS) is a protocol standard for permitting a web page or application to access remote content from a different domain (or port) than the site that the web page was loaded from. For example, assume that a user loads a page from the “origin” site at mynode.ibm.com. The downloaded web page includes client-side content (such as Java Script) which invokes a Db2 native REST service using site `db2server.ibm.com:446`. The call to the Db2 REST service triggers the CORS protocols because the Db2 REST service site is different than the “origin” site where which the web page was originally loaded.

Db2 REST services supports the HTTP Cross-Origin Resource Sharing (CORS) protocols, including support for the CORS "pre-flight" HTTP OPTIONS verb and CORS HTTP request/response header fields.

The configuration and management of the Db2 REST CORS origin authorization rules are implemented using a new z/OS RACF RESOURCE CLASS (DSNRAUTH) and associated RACF generic or discrete resource profiles to represent the allowed remote (origin) sites.

The CORS origin checking is managed as a system wide Db2 setting which is independent of the "end-user" that is driving the CORS request. So, the authorization ID associated with the DDF address space (*ssnm*DIST) started task is used for the CORS origin resource authorization check.

Procedure

To enable and permit Db2 REST CORS support for a specific origin, complete the following steps.

1. Activate the RACF resource class DSNRAUTH, with ACTIVE and RACLISTED.

   Optionally, if you intend to use generic Db2 REST service CORS profiles, also enable GENERIC for the DSNRAUTH resource class.
2. Create one or more RACF resource profiles to represent the CORS origin hosts that are allowed by the Db2 system.

   An origin host for Db2 REST service CORS uses a naming convention, where an example is `DSNCORS.DB2A.COM.SOMESERVER.WWW`, with the following format:

   ```
   DSNCORS.ssid|group-attach.normalized-origin-value
   ```

   ssid|group-attach
   For Db2 non-data sharing, the Db2 subsystem name (SSID) value is used. For Db2 data sharing, the Db2-group-attachment-name value is used.

normalized-origin-value
The format and content of the normalized-origin-value portion of a RACF Db2 CORS resource check depends on the type of the input origin value that Db2 receives in the REST request. Db2 REST CORS supports the input origin values in any of the following representations, with the option to specify a port:

- Regular hostname values, with the option to include a port.
- Internet Protocol Version 4 (IPv4) addresses, with the option to include a port.
- Internet Protocol Version 6 (IPv6) addresses, with the option to include a port. If the port is included, the IP address portion is enclosed within square brackets ('[' and ']').
- Special Internet Protocol Version 4 “mapped” Internet Protocol Version 6 dual stack (Hybrid dual-stack IPv4-mapped IPv6) addresses, with the option to include a port. If a port is included, the IP address portion must be enclosed within square brackets ('[' and ']').

For more information and examples, see Chapter 15, “Db2 REST services CORS resource naming conventions,” on page 35.

3. Permit the authorization ID associated with the Db2 DDF (*ssid*DIST) started task address space READ access to the just created resource profile.
4. Refresh the DSNRAUTH RACLIST by issuing the following command:

```
SETROPTS RACLIST(DSNRAUTH) REFRESH
```

Examples

Example 1

Enable Db2 REST CORS access on stand-alone Db2 subsystem DB2A, with a DB2ADIST started task ID of SYSDSP, from host origin "www.mybank.com":

```
RDEFINE DSNRAUTH DSNCORS.DB2A.COM.MYBANK.WWW UACC(NONE)
PERMIT DSNCORS.DB2A.COM.MYBANK.WWW CLASS(DSNRAUTH) ACCESS(READ) ID(SYSDSP)
SETROPTS RACLIST(DSNRAUTH) REFRESH
```

Example 2

Enable Db2 REST CORS access on Db2 data-sharing group with group attach name DB2G, where all members use the same DDF (*ssid*DIST) started task ID of SYSDSP for, from any host origin with ".org” as the top level domain:

```
RDEFINE DSNRAUTH DSNCORS.DB2G.ORG.** UACC(NONE)
PERMIT DSNCORS.DB2G.ORG.** CLASS(DSNRAUTH) ACCESS(READ) ID(SYSDSP)
SETROPTS RACLIST(DSNRAUTH) REFRESH
```

Example 3

Enable Db2 REST CORS access on stand-alone Db2 subsystem DB2D, with a DB2DDIST started task ID of SYSDSP, from any host origin under the "internal.myco.com" sub-domain:

```
RDEFINE DSNRAUTH DSNCORS.DB2D.COM.MYCO.INTERNAL.** UACC(NONE)
PERMIT DSNCORS.DB2D.COM.MYCO.INTERNAL.** CLASS(DSNRAUTH) ACCESS(READ) ID(SYSDSP)
SETROPTS RACLIST(DSNRAUTH) REFRESH
```

Example 4

Enable Db2 REST CORS access on stand-alone Db2 subsystem DB2A, with a DB2ADIST started task ID of SYSDSP, from host origin value "http://192.168.1.100":

```
RDEFINE DSNRAUTH DSNCORS.DB2A.0000.0000.0000.0000.0000.0000.192.168.001.100 UACC(NONE)
PERMIT DSNCORS.DB2A.0000.0000.0000.0000.0000.0000.192.168.001.100 CLASS(DSNRAUTH) -
ACCESS(READ) ID(SYSDSP)
SETROPTS RACLIST(DSNRAUTH) REFRESH
```

Related concepts

Initialization (RACF Access Control Module Guide)

Db2 REST services CORS resource naming conventions

Db2 REST services cross-origin resource sharing (CORS) support is enabled by activating and RACLISTing the RACF Resource DSNRAUTH class, and then creating one or more RACF resource profiles to represent the CORS origin hosts that are allowed by the Db2 system.

Chapter 15. Db2 REST services CORS resource naming conventions

Db2 REST services cross-origin resource sharing (CORS) support is enabled by activating and RACLISTing the RACF Resource DSNRAUTH class, and then creating one or more RACF resource profiles to represent the CORS origin hosts that are allowed by the Db2 system.

The resource naming convention used to represent allowed CORS origin hosts for Db2 REST services uses the following format:

```
DSNCORS.ssid|group-attach.normalized-origin-value
```

ssid|group-attach
For Db2 non-data sharing, the Db2 subsystem name (SSID) value is used. For Db2 data sharing, the Db2-group-attachment-name value is used.

normalized-origin-value

The format and content of the *normalized-origin-value* portion of a RACF Db2 CORS resource check depends on the type of the input origin value that Db2 receives in the REST request. Db2 REST CORS supports input origin values in any of the following representations:

Regular hostname

A regular hostname is made up of a sequence of labels, concatenated with dots. Labels can only contain ASCII a-z, A-Z, digits 0-9, hyphen-minus char ('-'), or underbar ('_') characters. If any other characters are detected in the origin value, they are replaced with the underbar (underscore) character. Origin values containing internationalized domain name (IDN) addresses are not supported for Db2 CORS usage.

The following are examples of regular hostname origin values, which can include an optional port:

- `http://www.ibm.com`
- `http://server1.example.org`
- `http://mynode.server.ibm.com:8080`

For origin values that are regular hostname values, the *normalized-origin-value* portion is fully qualified domain name of the remote origin site, in reverse and in uppercase. More specifically, the *normalized-origin-value* portion is generated as follows:

- The maximum DSNRAUTH resource length is 246 characters. Therefore, the maximum origin host length that can be used is between 233 and 236 characters, depending on the length of the *ssid|group-attach* portion of the CORS resource name. Origin values that exceed this maximum length will be truncated by removing least-significant leading characters from the original host origin value.
- A regular hostname has the component labels in a least-significant to most-significant order. In order to provide the ability to effectively use profile wildcarding, the ordering is reversed so that the hostname is represented in a most-significant to least-significant order.
- The reordered hostname is then converted to all uppercase characters.
- This reversal of hostname values supports use wildcard values for creating generic resource profiles that match multiple hostname values within a sub-domain.

For example, in the *normalized-origin-value*, the input origin value `http://www.ibm.com` is converted to `COM.IBM.WWW`.

IPv4 address

An Internet Protocol Version 4 (IPv4) address, which may include an optional port, as shown in the following examples:

- `http://192.168.1.100`
- `http://91.123.4.56`
- `http://10.4.15.200:993`

For origin values that are IPv4 literal addresses, the *normalized-origin-value* portion uses a modified IPv4-Mapped IPv6 format. The modified format uses prefix `0000.0000.0000.0000.0000.0000`, rather than the IPv4-mapped version prefix of `0000.0000.0000.0000.0000.FFFF`. Using this alternate prefix would allow users to define unique REST CORS profiles for IPv4 vs. IPv4-Mapped IPv6 addresses. The IPv4 normalized-origin-value is generated as follows:

- Six segments of hex digits, followed by four decimal octets.
- The leading six segments use the value `0000.0000.0000.0000.0000.0000`.
- The four decimal octets are the IPv4 address portion, where each decimal octet is represented using three digits, expanded to include leading zeros if necessary.
- The IP decimal octets are already in the desired most-significant to least-significant order, so no reordering of the IP address is needed.
- Total length of 45 characters.

For example, an input origin value using IPv4 literal address `http://91.123.4.56` is converted to the normalized-origin-value of `0000.0000.0000.0000.0000.0000.091.123.004.056`.

IPv6 address

An Internet Protocol Version 6 (IPv6) address, as shown in the following examples. If the optional port is included, the IP address portion must be enclosed within square brackets ('[' and ']').

- `http://fd20:db8:3333:4444:5555:6666:7777:8888`
- `http://fd12:3456:789a:1::1`
- `http://[fd12:3456:789a:1::1]:993`

For origin values that are IPv6 literal addresses, the normalized-origin-value portion is generated as follows:

- The IPv6 address is expanded to the full eight segment representation.
- Each segment uses four uppercase hexadecimal digits each, expanding to include leading zeros if necessary.
- The colon (':') separators between the segments are replaced with period ('.') characters.
- The IP hexadecimal segments are already in most-significant to least-significant order, so no reordering of the IP address is needed.
- Total length of 36 characters.

For example, an input origin value using IPv6 literal address `http://fd12:3456:789a:1::1` is converted to the normalized-origin-value of `FD12.3456.789A.0001.0000.0000.0000.0001`.

Hybrid dual-stack IPv4-mapped IPv6 address

A special Internet Protocol Version 4 “mapped” Internet Protocol Version 6 dual stack address. If the optional port is included, the IP address portion must be enclosed within square brackets ('[' and ']'). See the following examples:

- `http://::ffff:10.4.15.200`
- `http://::ffff:192.168.1.100`
- `http://[::ffff:192.168.1.100]:993`

For origin values that are Hybrid dual-stack IPv4-mapped IPv6 literal addresses, the normalized-origin-value portion is generated as follows:

- Expanded to use the full IPv4-Mapped IPv6 format.
- Six segments of hex digits, followed by four decimal octets.

- The leading six segments will use the value `0000.0000.0000.0000.0000.FFFF`.
- The four decimal octets will be the IPv4 address portion, where each decimal octet will be represented using three digits, expanding to include leading zeros if necessary.
- The IPv4 decimal octets are already in the desired most-significant to least-significant order, so no reordering of the IP address is needed.
- Total of 45 characters in length

For example, an input origin value using IPv6 literal address `http://::ffff:192.168.1.100` is converted to the normalized-origin-value of `0000.0000.0000.0000.0000.FFFF.192.168.001.100`.

Using wildcard values

You can use wildcard values to create RACF generic resource profiles that match for multiple hostname values within a sub-domain. When using wildcard values, the recommendation is to use hostname values to reference the origin site. For any origin IP addresses that you define, the recommendation is to create a discrete profile (without wildcard values) for each specific origin IP address. The Db2 CORS IP address support is limited to valid Internet Protocol address formats that are described previously.

Related tasks

Enabling CORS support for Db2 REST services

You can enable Cross-Origin Resource Sharing (CORS) support for Db2 to permit a web page or application to access remote content from a different domain (or port) than the site that the web page was loaded from. You can enable Db2 REST services to use the HTTP Cross-Origin Resource Sharing (CORS) protocols, including support for the CORS "pre-flight" HTTP OPTIONS verb and CORS HTTP request/response header fields.

Chapter 16. Creating a temporary DSNRAUTH class by using the RACF dynamic class descriptor table

If you want to use the REST CORS functionality before the availability of the RACF module ICHRRCDX update that delivers the new DSNRAUTH class definition, your z/OS RACF security administrator can temporarily create the DSNRAUTH class using the RACF dynamic class descriptor table (CDT) support. Unlike the RACF static CDT, use of the RACF dynamic CDT support has the benefit of not requiring a re-IPL of z/OS.

Before you begin

Apply the PTF for APAR PH59837.

Before you use the approach described here, consider the following important facts:

- You must use the exact class options specified below in order to ensure that your temporary RACF dynamic CDT DSNRAUTH class definition will be compatible and consistent with the official RACF delivered ICHRRCDX defined DSNRAUTH class. Failure to use these specified class options in your temporary dynamic CDT DSNRAUTH class definition may prevent your ability to migrate to the official RACF delivered ICHRRCDX defined DSNRAUTH class.
- It is expected that you will remove any temporary DSNRAUTH CDT definition when the RACF module ICHRRCDX update delivering the formal DSNRAUTH class is available.
- The RACF dynamic CDT support is intended to only be used for creating user-defined class definitions, which have documented naming and attribute restrictions to avoid collisions with official IBM class definitions. Using the dynamic CDT support to define a class that does not follow the documented naming and attribute restrictions is allowed, but will result in warning messages if these naming or attribute restrictions are violated. Therefore, the RACF dynamic CDT support can be successfully used to create the DSNRAUTH class definition, but the following expected warning messages will be issued and must be ignored:
  - `Warning: Class name DSNRAUTH does not contain a national character nor a number.`
  - `Warning: The POSIT value is not within the recommended ranges for installation use. The valid ranges are 19-56 and 128-527.`

Procedure

To define the DSNRAUTH class in the RACF dynamic CDT, complete the following steps:

1. Issue RACF RDEFINE commands with the options shown in the following example

```
RDEFINE CDT DSNRAUTH UACC(NONE)
   CDTINFO(
      CASE(ASIS) DEFAULTRC(8) DEFAULTUACC(NONE)
      MACPROCESSING(NORMAL) FIRST(ALPHA,NUMERIC,NATIONAL,SPECIAL)
      GENERIC(ALLOWED) GENLIST(DISALLOWED) KEYQUALIFIERS(0)
      MAXLENX(246) MAXLENGTH(246) OPERATIONS(NO)
      OTHER(ALPHA,NUMERIC,NATIONAL,SPECIAL) POSIT(610)
      PROFILESALLOWED(YES) RACLIST(ALLOWED) MACPROCESSING(NORMAL)
      SIGNAL(YES) SECLABELSREQUIRED(NO))
```

2. Issue the following SETROPTS command.

```
SETROPTS CLASSACT(CDT) RACLIST(CDT) REFRESH
```

Related tasks

Enabling CORS support for Db2 REST services

You can enable Cross-Origin Resource Sharing (CORS) support for Db2 to permit a web page or application to access remote content from a different domain (or port) than the site that the web page was loaded from. You can enable Db2 REST services to use the HTTP Cross-Origin Resource Sharing (CORS) protocols, including support for the CORS "pre-flight" HTTP OPTIONS verb and CORS HTTP request/response header fields.

Related reference

RACF RDEFINE command (define general resource profile) (Security Server RACF Command Language Reference)

RACF SETROPTS (Set RACF options) (Security Server RACF Command Language Reference)

Overview of the RACF class descriptor table (CDT) (z/OS Security Server RACF Security Administrator's Guide)

Chapter 17. Troubleshooting REST service requests

If Db2 cannot successfully execute an HTTP service request, an HTTP status code is provided in the response header and body that indicates the general category for the failure. The response body also includes a StatusDescription field that might provide additional information about the reason for the failure.

The following table describes the HTTP status codes for common error conditions.

Table 1. Common HTTP status codes for REST service error conditions

HTTP status code	Description
HTTP 500 (Internal Server Error)	Indicates that the server could not fulfill a request. In most cases, the HTTP code is accompanied by a Db2 SQL code that provides more details about the error condition.
HTTP 400® (Bad Request)	Indicates a problem with an input parameter, such as a missing required input parameter, that is detected by the Db2 DDF native REST code before executing the Db2 SQL statement. This code is also used for many DB2ServiceManager failures (for example, Create/Drop service) and DB2DiscoverService failures (discover service/discover service details), which are typically caused by incorrect or missing inputs.
HTTP 401 (Unauthorized)	Indicates that the user could not be successfully authenticated.
HTTP 403 (Forbidden)	Indicates that the user might not have the required permissions to access a resource.

The following example shows a request in which the input for the "HIREDATE" parameter value, which targets a Db2 DATE column, fails because the input was not provided in a valid DATE format:

```
POST http://hostname.ibm.com:446/services/SYSIBMSERVICE/inDateConvertError
Request body:
{
"HIREDATE":"NOT_A_DATE"
}
```

The response to this request includes an HTTP status code followed by the StatusDescription section, which provides details about the error condition. In this case, a Db2 SQL code is also included.

```
HTTP reply msg Header Fields:{null=[HTTP/1.1 500 Internal Server Error], Content-Language=[en-
US],
Date=[Fri, 31 Mar 2017 18:23:28 GMT], Content-Length=[229], Content-Type=[application/json;
charset=UTF-8],
X-Powered-By=[DB2 for z/OS], Server=[DB2 DDF Native REST, STLEC1]}
Response body:
{
"StatusCode":500,
"StatusDescription":"Service SYSIBMSERVICE.inDateConvertError execution failed due to SQL
error,
SQLCODE=-180, SQLSTATE=22007, Message=THE DATE, TIME, OR TIMESTAMP VALUE *N IS INVALID Error
Location:DSNLJXUS:6"
}
```

Related concepts

Db2 codes (Db2 Codes)

Chapter 18. Monitoring of REST service connections and threads by using profile tables

You can monitor the connections and threads of HTTP REST service requests by setting special registers and global variables in the profiles.

Db2 system profile monitoring can use the following filtering criteria to monitor Db2 RESTful services requests:

- IP address or domain name, in the LOCATION column.
- Product identifier, in the PRDID column. The current PRDID value used for Db2 native REST services is:
  - 'HTP01010' for non-secure HTTP URL connections
  - 'HTS01010' for HTTPS/secure connections.
- Role and authorization identifier, in both ROLE and AUTHID columns.
- Role, in the ROLE column only.
- Authorization identifier, in the AUTHID column only.
- Collection identifier and package name (Service name), in both COLLID and PKGNAME columns.
- Collection identifier, in the COLLID column only.
- Package name (Service name), in the PKGNAME column only.
- Client application name, in the CLIENT_APPLNAME column.
- Client user identifier, in the CLIENT_USERID column.
- Client workstation name, in the CLIENT_WRKSTNNAME column.

Other criteria combinations are not accepted.

System monitor profile exception queuing is not supported for Db2 native REST requests. If a system monitor profile EXCEPTION level threshold is met for an incoming Db2 native REST request, the REST request will be rejected.

Related concepts

Management of REST service client information
Db2 DDF implicitly defines special registers and a global variable to store client information about a valid HTTP REST service request. The client information is externalized in accounting records and the output of the -DISPLAY command.

Related tasks

Monitoring remote connections by using profile tables (Db2 Administration Guide)
Monitoring threads by using profile tables (Db2 Administration Guide)
Monitoring and controlling Db2 by using profile tables (Db2 Administration Guide)

Chapter 19. Management of REST service client information

Db2 DDF implicitly defines special registers and a global variable to store client information about a valid HTTP REST service request. The client information is externalized in accounting records and the output of the -DISPLAY command.

Db2 allows a subset of characters from the first 128 Unicode code points to be stored as client information. See the **Allowable characters for client information** section for a complete list of allowable characters. When processing a REST service request, Db2 stores the client information in the following special registers:

CURRENT CLIENT_ACCTNG

Contains the value of the client accounting string. The Db2 REST services maximum supported length of the CURRENT CLIENT_ACCTNG value is 255 characters.

A user specified value can be set using the HTTP request header field *Db2-Client-Acctng*.

The Db2 generated default value contains the string of an HTTP REST GET or POST method that is concatenated with the name of a qualified service in the form of *<collectionID.serviceName>*, as in `GET SYSIBMSERVICE.simpleSelect1` or `POST SYSIBMSERVICE.simpleSelect1`. *<serviceName>* might be truncated.

CURRENT CLIENT_APPLNAME

The Db2 REST services maximum supported length of the CURRENT CLIENT_APPLNAME value is 128 characters.

A user specified value can be set using the HTTP request header field *Db2-Client-ApplName*.

The Db2 REST generated default value is the service name string in the service request.

CURRENT CLIENT_CORR_TOKEN

The Db2 REST services maximum supported length of the CURRENT CLIENT_CORR_TOKEN is 128 characters.

A user specified value can be set using the HTTP request header field *X-Correlation-ID* or the HTTP request header field *Db2-Client-Corr-Token*. If both HTTP *X-Correlation-ID* and *Db2-Client-Corr-Token* request header fields are present in the request, then the *Db2-Client-Corr-Token* field value is used.

The Db2 REST generated default value is the logical unit of work identifier or LUWID string that is generated for the thread without the commit count for the service request. The correlation token can be used to correlate any downstream connections that are used by the service.

CURRENT CLIENT_USERID

The Db2 REST services maximum supported length of the CURRENT CLIENT_USERID is 128 characters.

A user specified value can be set using the HTTP request header field *Db2-Client-Userid*.

The Db2 REST generated default value is the authentication ID string. The authorization ID is used to establish an HTTP connection for the service request.

CURRENT CLIENT_WRKSTNNAME

The Db2 REST services maximum supported length of the CURRENT CLIENT_WRKSTNNAME is 255 characters.

A user specified value can be set using the HTTP request header field *User-Agent* or *Db2-Client-WrkStnName*. If both *User-Agent* or *Db2-Client-WrkStnName* request header fields are present in the

request, then the *Db2-Client-WrkStnName* is used. Many REST clients set the *User-Agent* header value to a default string that represents the name and version of the REST client that is being used.

The Db2 REST generated default value is the client IP address string. The IP address is associated with the client that sent the REST service request.

Db2 also stores information about the HTTP REST service request in the following built-in global variable:

SYSIBM.CLIENT_IPADDR
Contains the value of the client IP address string.

The IP address is associated with the client that sent the REST service request.

Allowable characters for client information values

All Db2 HTTP request header client information values must be provided in UTF-8 encoding and consist only of a subset of characters from the first 128 Unicode code points. Specifically, only the following single byte Unicode characters are allowed in Db2 REST services client information values:

Description	Character	Unicode code point
Upper Case Alpha	A-Z	X'41'-X'5A'
Lower Case Alpha	a-z	X'61'-X'7A'
Numeric	0-9	X'30'-X'39'
Blank	(sp)	X'20'
Number Sign/Pound Sign	#	X'23'
Dollar Sign	$	X'24'
Percent Sign	%	X'25'
Ampersand	&	X'26'
Left Parenthesis	(	X'28'
Right Parenthesis	)	X'29'
Asterisk	*	X'2A'
Plus Sign	+	X'2B'
Comma	,	X'2C'
Hyphen/Minus	-	X'2D'
Period	.	X'2E'
Slash/Solidus	/	X'2F'
Colon	:	X'3A'
Semi-Colon	;	X'3B'
Less Than Sign	<	X'3C'
Equal	=	X'3D'
Greater Than Sign	>	X'3E'
At Sign	@	X'40'
Underscore	_	X'5F'

Any disallowed Unicode character code points found in the input client information values will be replaced by the period (x'2E') character. Although Db2 REST services client information values allow the use

of several variant characters, such as #, @, $, and others, it is suggested that users use only the invariant characters when possible. This suggestion is to avoid any unexpected display issues of the client information values after translation to the Db2 system EBCDIC CCSID. The Db2 client information values may be included in various Db2 console messages, trace records, and other external representations, which might not display as expected if variant characters are used in the client information settings.

Related concepts

Variant characters (Db2 Internationalization Guide)

Related reference

Special registers (Db2 SQL)

Chapter 20. Classification of REST services

You can use the z/OS Workload Manager (WLM) support to define performance objectives for a Db2 REST service and classify the service by the authorization ID or any other classification attribute.

Important: If classification rules are not available for some or all of the services, Db2 assigns the default SYSOTHER class to classify those services. A service of the SYSOTHER class is not defined with a performance goal and is not treated as important as a service with a discretionary goal.

Related concepts

z/OS performance options for Db2 (Db2 Performance)

Related tasks

Classifying Db2 DDF threads in WLM (Db2 Performance)

Related reference

Defining WLM classification rules

Chapter 21. Display of REST service locations and threads

You can issue the -DISPLAY THREAD command to identify and display threads that actively execute Db2 REST services. The correlation ID for a thread that processes a service is set to “DB2_REST” while the application name is the name of the service that is invoked.

The following is a sample output of the -DISPLAY THREAD command which identifies "DB2A" as the active thread and "`simpleSelect1`" as the application.

```
00- 17.39.04          -DB2ADIS THREAD(*) DET
  - 17.39.04 STC00179  DSNV401I  -DB2A DISPLAY THREAD REPORT FOLLOWS -
  - 17.39.04 STC00179  DSNV402I  -DB2A ACTIVE THREADS -
  - NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
  - DB2A     RA *     0 028.DBAA  02 SYSOPR            0071     7
  -  V437-WORKSTATION=Mozilla/5.0 (Windows NT 6.1; WOW64; rv:45.0) Gecko/2
  -       0100101 Firefox/45.0
  -       USERID=user001
  -       APPLICATION NAME=simpleSelect1
  -  V441-ACCOUNTING=POST SYSIBMSERVICE.simpleSelect1
  -  V442-CRTKN=::FFFF:host.port.D149FF032587
  -  V445-G934E317.FC7D.D149FF032587=7 ACCESSING DATA FOR
  -   (  1)::FFFF:host
  -  V447--INDEX SESSID           A ST TIME
  -  V448--(  1) port          R2 1624617373942
  - DISCONN  DA *     0 NONE          NONE     DISTSERV 0071     9
  -  V471-USIBMSY.SYEC1DB2.D149FF2F06EA=9
  - DISPLAY ACTIVE REPORT COMPLETE
  - 17.39.04 STC00179  DSN9022I  -DB2A DSNVDT '-DIS THREAD' NORMAL COMPLETION
```

Related reference

-DISPLAY THREAD command (Db2) (Db2 Commands)

Chapter 22. Db2 REST services with high availability

To ensure that your Db2 REST service requests are processed with high availability, the target of the REST service requests must be a Db2 data sharing group.

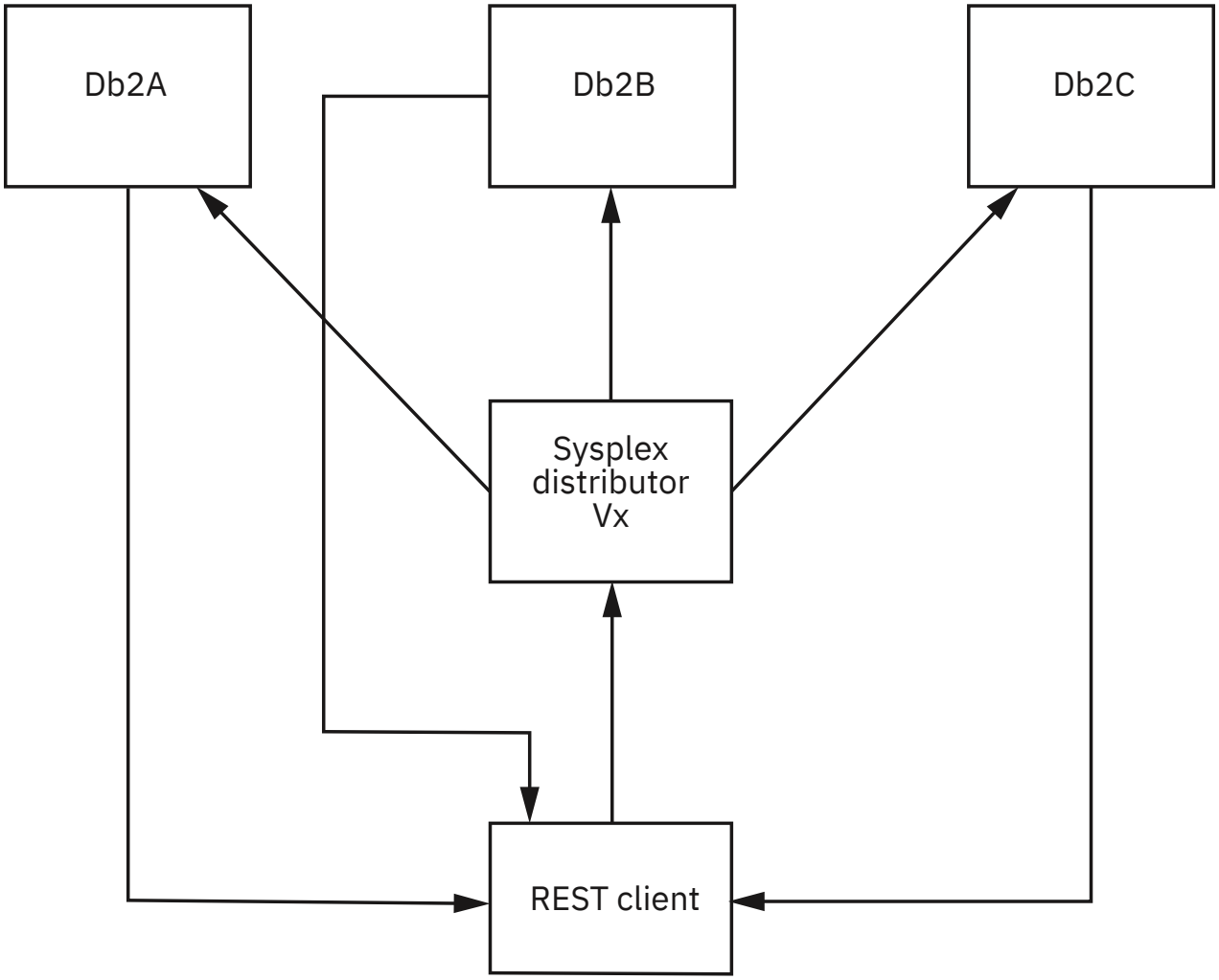
You need to satisfy the following requirements when you configure the Db2 data sharing group for REST services with high availability:

- Use a group IP address so that any available member of the group can be accessed to satisfy a REST request.
- Use the same TCP port value for all members of the data sharing group. This requirement is true for a Db2 data sharing group that is accessed by any distributed clients, and not just for REST request clients.

For a z/OS environment, a Db2 group IP address should be a *distributed dynamic virtual IP address* (DDVIPA). A DDVIPA is a sysplex distributor IP address that is managed by z/OS Communications Server. A DDVIPA provides the technology to balance socket connection requests to a set of servers. Those servers listen for inbound connection requests on a common TCP port. If at least one of the members of the data sharing group is available, a client that uses the sysplex distributor IP address (or its defined DNS domain name) can establish a connection to a member of the data sharing group. A DDVIPA does not balance the REST requests that are sent to a member of the data sharing group. After a path has been established for a connection, for as long as the connection persists, all REST requests that use the connection follow the established path.

Example: Processing of REST requests with high availability

The following figure shows a Db2 for z/OS data sharing environment that supports REST requests with high availability.



The data sharing group has three members: DB2A, DB2B, and DB2C. Assume that the environment is configured as follows:

- The data sharing members are all listening on port 446.
- DDVIPA Vx is configured within the z/OS sysplex to handle requests for port 446.
- The REST client is configured to request a REST request against a server with IP address Vx and TCP port 446.

A REST request is processed in the following way:

1. A new socket connection request is sent from the REST client to sysplex distributor Vx.
2. Sysplex distributor Vx determines whether data sharing member DB2A, DB2B, or DB2C receives the connection request.
3. The data sharing member that the sysplex distributor chose in the previous step builds and sends a reply directly to the REST client.
4. For a persistent connection, each REST client request that uses the connection is sent to sysplex distributor Vx. Sysplex distributor Vx forwards the request to the previously selected data sharing member, and that data sharing member sends its reply directly to the client.

   Db2 closes a persistent connection that remains idle for more than approximately 15 seconds. A persistent connection is considered to be idle when it is waiting for a new REST request from the client.

Related concepts

Sysplex distributor (z/OS Communications Server: IP Configuration Guide)

Information resources for Db2 for z/OS and related products

You can find the online product documentation for Db2 13 for z/OS and related products in IBM Documentation.

For all online product documentation for Db2 13 for z/OS, see IBM Documentation (https://www.ibm.com/docs/en/db2-for-zos/13).

For other PDF manuals, see PDF format manuals for Db2 13 for z/OS (https://www.ibm.com/docs/en/db2-for-zos/13?topic=zos-pdf-format-manuals-db2-13).

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785 US

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785 US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as shown below:

© (*your company name*) (*year*).
Portions of this code are derived from IBM Corp. Sample Programs.
© Copyright IBM Corp. (*enter the year or years*).

Programming interface information

This information is intended to help you to plan for and administer Db2 13 for z/OS. This information also documents General-use Programming Interface and Associated Guidance Information and Product-sensitive Programming Interface and Associated Guidance Information provided by Db2 13 for z/OS.

General-use Programming Interface and Associated Guidance Information

General-use Programming Interfaces allow the customer to write programs that obtain the services of Db2 13 for z/OS.

General-use Programming Interface and Associated Guidance Information is identified where it occurs by the following markings:

GUPI General-use Programming Interface and Associated Guidance Information... GUPI

Product-sensitive Programming Interface and Associated Guidance Information

Product-sensitive Programming Interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of this IBM software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive Programming Interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs by the following markings:

PSPI Product-sensitive Programming Interface and Associated Guidance Information... PSPI

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered marks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at: http://www.ibm.com/legal/copytrade.shtml.

Linux® is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions:

Applicability: These terms and conditions are in addition to any terms of use for the IBM website.

Personal use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights: Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Statement at http://www.ibm.com/privacy.

Glossary

The glossary is available in IBM Documentation

For definitions of Db2 for z/OS terms, see Db2 glossary (Db2 Glossary).

Index

T

U

Product Number: 5698-DB2
5698-DBV