

# Amazon RDS for Db2

Amazon RDS supports DB instances that run the following editions of IBM Db2:

- Db2 Standard Edition
- Db2 Advanced Edition

Amazon RDS supports DB instances that run the following versions of Db2:

- Db2 11.5

For more information about minor version support, see [Db2 on Amazon RDS versions](#).

Before creating a DB instance, complete the steps in the [Setting up your Amazon RDS environment](#) section of this user guide. When you create a DB instance using your master user, the user gets DBADM authority, with some limitations. Use this user for administrative tasks such as creating additional database accounts. You can't use SYSADM, SYSCTRL, SYSMAINT instance-level authority, or SECADM database-level authority.

You can create the following:

- DB instances
- DB snapshots
- Point-in-time restores
- Automated storage backups
- Manual storage backups

You can use DB instances running Db2 inside a virtual private cloud (VPC). You can also add features to your Amazon RDS for Db2 DB instance by enabling various options. Amazon RDS supports Multi-AZ deployments for RDS for Db2 as a high availability, failover solution.

 **Important**

To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances. It also restricts access to certain system procedures and tables that need elevated

privileges. You can access your database using standard SQL clients such as IBM Db2 CLP. However, you can't access the host directly by using Telnet or Secure Shell (SSH).

## Topics

- [Overview of Db2 on Amazon RDS](#)
- [Prerequisites for creating an Amazon RDS for Db2 DB instance](#)
- [Multiple databases on an Amazon RDS for Db2 DB instance](#)
- [Connecting to your Db2 DB instance](#)
- [Securing Amazon RDS for Db2 DB instance connections](#)
- [Administering your Amazon RDS for Db2 DB instance](#)
- [Integrating an Amazon RDS for Db2 DB instance with Amazon S3](#)
- [Migrating data to Amazon RDS for Db2](#)
- [Amazon RDS for Db2 federation](#)
- [Working with replicas for Amazon RDS for Db2](#)
- [Options for Amazon RDS for Db2 DB instances](#)
- [External stored procedures for Amazon RDS for Db2](#)
- [Known issues and limitations for Amazon RDS for Db2](#)
- [Amazon RDS for Db2 stored procedure reference](#)
- [Amazon RDS for Db2 user-defined function reference](#)
- [Troubleshooting for Amazon RDS for Db2](#)

## Overview of Db2 on Amazon RDS

You can read the following sections to get an overview of Db2 on Amazon RDS.

## Topics

- [Amazon RDS for Db2 features](#)
- [Db2 on Amazon RDS versions](#)
- [Amazon RDS for Db2 licensing options](#)
- [Amazon RDS for Db2 instance classes](#)

- [Amazon RDS for Db2 default roles](#)
- [Amazon RDS for Db2 parameters](#)
- [EBCDIC collation for Db2 databases on Amazon RDS](#)
- [Local time zone for Amazon RDS for Db2 DB instances](#)

## Amazon RDS for Db2 features

Amazon RDS for Db2 supports most of the features and capabilities of the IBM Db2 database. Some features might have limited support or restricted privileges. For more information about the Db2 database features for specific Db2 versions, see the [IBM Db2 documentation](#).

You can filter new Amazon RDS features on the [What's New with Database?](#) page. For **Products**, choose **Amazon RDS**. Then, you can search by using keywords such as **Db2 2023**.

 **Note**

The following lists aren't exhaustive.

### Topics

- [Supported features in RDS for Db2](#)
- [Unsupported features in RDS for Db2](#)

## Supported features in RDS for Db2

RDS for Db2 supports features that include features that are native to IBM Db2 and features that are core to Amazon RDS.

### Features native to IBM Db2

RDS for Db2 supports the following Db2 database features:

- Creation of a standard database that uses a customer-defined code set, collation, page size, and territory. Use the Amazon RDS [rdsadmin.create\\_database](#) stored procedure.
- Addition, deletion, or modification of local users and groups. Use the Amazon RDS stored procedures for [Stored procedures for granting and revoking privileges for RDS for Db2](#).

- Creation of roles with the Amazon RDS [`rdsadmin.create\_role`](#) stored procedure.
- Support for standard row-organized tables.
- Support for analytic workload for column-organized tables.
- Ability to define Db2-compatibility features such as Oracle and MySQL.
- Support for Java-based external stored procedures.
- Support for data encryption in transit by using SSL/TLS.
- Monitoring the status of a database (ALIVE, DOWN, STORAGE\_FULL, UNKNOWN, and STANDBY\_CONNECTABLE).
- Restoration of a customer-provided offline or online Linux (LE) database. Use Amazon RDS stored procedures for [Stored procedures for databases for RDS for Db2](#).
- Application of customer-provided Db2 archive logs to keep the database synchronized with self-managed Db2 databases. Use Amazon RDS stored procedures for [Stored procedures for databases for RDS for Db2](#).
- Support for Db2 instance-level and database-level auditing.
- Support for homogeneous federation.
- Ability to load a table from data files in Amazon Simple Storage Service (Amazon S3).
- Authorizations granted to users, groups or roles, such as CONNECT, SYSMON, ACCESSCTRL, DATAACCESS, SQLADM, WLMADM, EXPLAIN, LOAD, or IMPLICIT\_SCHEMA.
- Creation of multiple databases.

 **Note**

An RDS for Db2 DB instance can contain up to 50 databases. For more information, see [the section called “Multiple Db2 databases”](#).

## Features core to Amazon RDS

RDS for Db2 supports the following core Amazon RDS features:

- Custom parameter groups to assign to DB instances
- Creation, modification, and deletion of DB instances
- Restoration of a self-managed Db2 offline or online Linux (LE) database backup

**Note**

To be able to restore your backup, don't provide a name for your database when you create a DB instance. For more information, see [Creating an Amazon RDS DB instance](#).

- Support of gp3, io2, and io1 storage types
- Use of AWS Managed Microsoft AD for Kerberos authentication, and LDAP group authorization for RDS for Db2
- Modification of security groups, ports, instance types, storage, backup retention periods, and other settings for existing Db2 instances
- Deletion protection for DB instances
- Cross-Region point-in-time recovery (PITR), including for encrypted backups
- Use of AWS Key Management Service (AWS KMS) for storage encryption and encryption at rest
- Multi-AZ DB instances with one standby for high availability
- Reboots of DB instances
- Updates to master passwords
- Restoration of DB instances to a specific time
- Backup and restoration of DB instances by using storage-level backups
- Start and stop of DB instances
- Maintenance of DB instances
- Same-Region and cross-Region standby and read replicas

## Unsupported features in RDS for Db2

RDS for Db2 doesn't support the following Db2 database features:

- SYSADM, SECADM, and SYSMAINT access for the master user.
- External stored procedures written in C, C++, or Cobol.
- Multiple Db2 DB instances on a single host.
- External GSS-API plugins for authentication.
- External third-party plugins to back up or restore Db2 databases.
- Multi-node massively parallel processing (MPP), such as IBM Db2 Warehouse.

- IBM Db2 pureScale.
- Manual setup of High Availability Disaster Recovery (HADR) for RDS for Db2.

### Note

Amazon RDS supports and manages HADR for RDS for Db2 through replicas. For more information, see [Working with replicas for Amazon RDS for Db2](#).

RDS for Db2 supports Multi-AZ deployments, cross-Region automated backups, and replication. For more information, see [Multi-AZ DB instance deployments for Amazon RDS](#) and [Replicating automated backups to another AWS Region](#).

- Native database encryption.
- Heterogeneous federation to Informix, Sybase, and Teradata. For more information, see [the section called “Federation”](#).
- Creation of non-fenced routines and migration of existing non-fenced routines by backing up and restoring data. For more information, see [Non-fenced routines](#).
- Creation of new non-automatic storage tablespaces. For more information, see [Non-automatic storage tablespaces during migration](#).
- External tables.

## Db2 on Amazon RDS versions

For Db2, version numbers take the form of *major.minor.build.revision*, for example, 11.5.9.0.sb00000000.r1. Our version implementation matches that of Db2.

### **major**

The major version number is both the integer and the first fractional part of the version number, for example, 11.5. A version change is considered major if the major version number changes—for example, going from version 11.5 to 12.1.

### **minor**

The minor version number is both the third and fourth parts of the version number, for example, 9.0 in 11.5.9.0. The third part indicates the Db2 modpack, for example, 9 in 9.0. The fourth part indicates the Db2 fixpack, for example, 0 in 9.0. A version change is considered minor if either the Db2 modpack or the Db2 fixpack changes—for example, going from version

11.5.9.0 to 11.5.9.1, or from 11.5.9.0 to 11.5.10.0, with exceptions to provide catalog table updates. (Amazon RDS takes care of these exceptions.)

## build

The build number is the fifth part of the version number, for example, sb00000000 in 11.5.9.0.sb00000000. A build number where the number portion is all zeroes indicates a standard build. A build number where the number portion isn't all zeroes indicates a special build. A build number changes if there is a security fix or special build of an existing Db2 version. A build number change also indicates that Amazon RDS automatically applied a new minor version.

## revision

The revision number is the sixth part of the version number, for example, r1 in 11.5.9.0.sb00000000.r1. A revision is an Amazon RDS revision to an existing Db2 release. A revision number change indicates that Amazon RDS automatically applied a new minor version.

## Topics

- [Supported Db2 minor versions on Amazon RDS](#)
- [Supported Db2 major versions on Amazon RDS](#)

## Supported Db2 minor versions on Amazon RDS

The following table shows the minor versions of Db2 11.5 that Amazon RDS currently supports.

### Note

Dates with only a month and a year are approximate and are updated with an exact date when it's known.

Db2 engine version	IBM release date	RDS release date
11.5.9.0	15 November 2023	27 November 2023

You can specify any currently supported Db2 version when creating a new DB instance. You can specify the major version (such as Db2 11.5) and any supported minor version for the specified

major version. If no version is specified, Amazon RDS defaults to a supported version, typically the most recent version. If a major version is specified but a minor version is not, Amazon RDS defaults to a recent release of the major version that you have specified. To see a list of supported versions, as well as defaults for newly created DB instances, use the [describe-db-engine-versions](#) AWS Command Line Interface (AWS CLI) command.

For example, to list the supported engine versions for Amazon RDS for Db2, run the following AWS CLI command. Replace *region* with your AWS Region.

For Linux, macOS, or Unix:

```
aws rds describe-db-engine-versions \
--filters Name=engine,Values=db2-ae,db2-se \
--query "DBEngineVersions[].{Engine:Engine, EngineVersion:EngineVersion,
DBParameterGroupFamily:DBParameterGroupFamily}" \
--region region
```

For Windows:

```
aws rds describe-db-engine-versions ^
--filters Name=engine,Values=db2-ae,db2-se ^
--query "DBEngineVersions[].{Engine:Engine, EngineVersion:EngineVersion,
DBParameterGroupFamily:DBParameterGroupFamily}" ^
--region region
```

This command produces output similar to the following example:

```
[  
 {  
   "Engine": "db2-ae",  
   "EngineVersion": "11.5.9.0.sb00000000.r1",  
   "DBParameterGroupFamily": "db2-ae-11.5"  
 },  
 {  
   "Engine": "db2-se",  
   "EngineVersion": "11.5.9.0.sb00000000.r1",  
   "DBParameterGroupFamily": "db2-se-11.5"  
 }  
 ]
```

The default Db2 version might vary by AWS Region. To create a DB instance with a specific minor version, specify the minor version during DB instance creation. You can determine the default version for an AWS Region for db2-ae and db2-se database engines by running the `describe-db-engine-versions` command. The following example returns the default version for db2-ae in US East (N. Virginia).

For Linux, macOS, or Unix:

```
aws rds describe-db-engine-versions \
--default-only --engine db2-ae \
--query "DBEngineVersions[].[Engine:Engine, EngineVersion:EngineVersion,
DBParameterGroupFamily:DBParameterGroupFamily]" \
--region us-east-1
```

For Windows:

```
aws rds describe-db-engine-versions ^
--default-only --engine db2-ae ^
--query "DBEngineVersions[].[Engine:Engine, EngineVersion:EngineVersion,
DBParameterGroupFamily:DBParameterGroupFamily]" ^
--region us-east-1
```

This command produces output similar to the following example:

```
[  
 {  
   "Engine": "db2-ae",  
   "EngineVersion": "11.5.9.0.sb00000000.r1",  
   "DBParameterGroupFamily": "db2-ae-11.5"  
 }  
]
```

With Amazon RDS, you control when to upgrade your Db2 instance to a new major version supported by Amazon RDS. You can maintain compatibility with specific Db2 versions, test new versions with your application before deploying in production, and perform major version upgrades at times that best fit your schedule.

When automatic minor version upgrade is enabled, Amazon RDS automatically upgrades your DB instances to new Db2 minor versions as they are supported by Amazon RDS. This patching occurs during your scheduled maintenance window. You can modify a DB instance to enable or disable automatic minor version upgrades.

Except for Db2 versions 11.5.9.1 and 11.5.10.0, automatic upgrades to new Db2 minor version includes automatic upgrades to new builds and revisions. For 11.5.9.1 and 11.5.10.0, manually upgrade minor versions.

If you opt out of automatically scheduled upgrades, you can manually upgrade to a supported minor version release by following the same procedure as you would for a major version update. For information, see [Upgrading a DB instance engine version](#).

## Supported Db2 major versions on Amazon RDS

RDS for Db2 major versions are available under standard support at least until IBM end of support (base) for the corresponding IBM version. The following table shows the dates that you can use to plan your testing and upgrade cycles. If Amazon extends support for an RDS for Db2 version for longer than originally stated, we plan to update this table to reflect the later date.

You can use the following dates to plan your testing and upgrade cycles.

### Note

Dates with only a month and a year are approximate and are updated with an exact date when it's known.

You can view the major versions of your Db2 databases by running the [describe-db-major-engine-versions](#) AWS CLI command or by using the [DescribeDBMajorEngineVersions](#) RDS API operation.

Db2 major version	IBM release date	RDS release date	IBM end of support (Standard and Advanced Edition)	IBM end of support (extended)	
Db2 11.5	27 June 2019	27 November 2023	30 April 2027	30 April 2031	

# Amazon RDS for Db2 licensing options

Amazon RDS for Db2 has two licensing options: bring your own license (BYOL) and Db2 license through AWS Marketplace.

## Topics

- [Bring your own license \(BYOL\) for Db2](#)
- [Db2 license through AWS Marketplace](#)
- [Switching between Db2 licenses](#)

## Bring your own license (BYOL) for Db2

In the BYOL model, you use your existing Db2 database licenses to deploy databases on Amazon RDS. Verify that you have the appropriate Db2 database license for the DB instance class and Db2 database edition that you want to run. You must also follow IBM policies for licensing IBM database software in the cloud computing environment.

### Note

Multi-AZ DB instances are cold standbys because the Db2 database is installed but not running. Standbys aren't readable, running, or serving requests. For more information, see [IBM Db2 licensing information](#) on the IBM website.

In this model, you continue to use your active IBM support account, and you contact IBM directly for Db2 database service requests. If you have an Support account with case support, you can contact Support for Amazon RDS issues. Amazon Web Services and IBM have a multi-vendor support process for cases that require assistance from both organizations.

Amazon RDS supports the BYOL model for Db2 Standard Edition and Db2 Advanced Edition.

## Topics

- [IBM IDs for bring your own license \(BYOL\) for Db2](#)
- [Adding IBM IDs to a parameter group for RDS for Db2 DB instances](#)
- [Integrating with AWS License Manager](#)

## IBM IDs for bring your own license (BYOL) for Db2

In the BYOL model, you need your IBM Customer ID and your IBM Site ID to create, modify, or restore RDS for Db2 DB instances. You must create a custom parameter group with your IBM Customer ID and your IBM Site ID *before* you create an RDS for Db2 DB instance. For more information, see [Adding IBM IDs to a parameter group for RDS for Db2 DB instances](#). You can run multiple RDS for Db2 DB instances with different IBM Customer IDs and IBM Site IDs in the same AWS account or AWS Region.

 **Important**

If we can't verify your license by your IBM Customer ID and your IBM Site ID, we might terminate any DB instances running with these unverified licenses.

If you're a new IBM Db2 customer, you must first purchase a Db2 software license from [IBM](#). After you purchase a Db2 software license, you will receive a Proof of Entitlement from IBM, which lists your IBM Customer ID and your IBM Site ID.

If you're an existing IBM Db2 customer, you can find your IBM Customer ID and your IBM Site ID on your Proof of Entitlement certificate from IBM.

You can also find your IBM Customer ID and your IBM Site ID in your [IBM Passport Advantage Online](#) account. After your log in, you can view both IDs on either the main page or the Software downloads page.

### Adding IBM IDs to a parameter group for RDS for Db2 DB instances

Because you can't modify default parameter groups, you must create a custom parameter group and then modify it to include the values for your IBM Customer ID and your IBM Site ID. For information about parameter groups, see [DB parameter groups for Amazon RDS DB instances](#).

 **Important**

You must create a custom parameter group with your IBM Customer ID and your IBM Site ID *before* you create an RDS for Db2 DB instance.

Use the parameter settings in the following table.

Parameter	Value
rds.ibm_customer_id	<your IBM Customer ID>
rds.ibm_site_id	<your IBM Site ID>
ApplyMethod	immediate , pending-reboot

These parameters are dynamic, which means that any changes to them take effect immediately and that you don't need to reboot the DB instance. If you don't want the changes to take effect immediately, you can set ApplyMethod to pending-reboot and schedule these changes to be made during a maintenance window.

You can create and modify a custom parameter group by using the AWS Management Console, the AWS CLI, or the Amazon RDS API.

## Console

### To add your IBM Customer ID and your IBM Site ID to a parameter group

1. Create a new DB parameter group. For more information about creating a DB parameter group, see [Creating a DB parameter group in Amazon RDS](#).
2. Modify the parameter group that you created. For more information about modifying a parameter group, see [Modifying parameters in a DB parameter group in Amazon RDS](#).

## AWS CLI

### To add your IBM Customer ID and your IBM Site ID to a parameter group

1. Create a custom parameter group by running the [create-db-parameter-group](#) command.

Include the following required options:

- `--db-parameter-group-name` – A name for the parameter group that you are creating.
- `--db-parameter-group-family` – The Db2 engine edition and major version. Valid values: db2-se-11.5, db2-ae-11.5.
- `--description` – A description for this parameter group.

For more information about creating a DB parameter group, see [Creating a DB parameter group in Amazon RDS](#).

2. Modify the parameters in the custom parameter group that you created by running the [modify-db-parameter-group](#) command.

Include the following required options:

- `--db-parameter-group-name` – The name of the parameter group that you created.
- `--parameters` – An array of parameter names, values, and the application methods for the parameter update.

For more information about modifying a parameter group, see [Modifying parameters in a DB parameter group in Amazon RDS](#).

## RDS API

### To add your IBM Customer ID and your IBM Site ID to a parameter group

1. Create a custom DB parameter group by using the Amazon RDS API [CreateDBParameterGroup](#) operation.

Include the following required parameters:

- `DBParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

For more information about creating a DB parameter group, see [Creating a DB parameter group in Amazon RDS](#).

2. Modify the parameters in the custom parameter group that you created by using the RDS API [ModifyDBParameterGroup](#) operation.

Include the following required parameters:

- `DBParameterGroupName`
- `Parameters`

For more information about modifying a parameter group, see [Modifying parameters in a DB parameter group in Amazon RDS](#).

Now you are ready to create a DB instance and attach the custom parameter group to the DB instance. For more information, see [Creating an Amazon RDS DB instance](#) and [Associating a DB parameter group with a DB instance in Amazon RDS](#).

## Integrating with AWS License Manager

To aid in monitoring RDS for Db2 license usage in the BYOL model, [AWS License Manager](#) integrates with RDS for Db2. License Manager supports tracking of RDS for Db2 engine editions based on virtual CPUs (vCPUs). You can also use License Manager with AWS Organizations to manage all of your organizational accounts centrally.

To track license usage of your RDS for Db2 DB instances, you must create self-managed licenses. You can create self-managed licenses by using the AWS Management Console, the AWS License Manager CLI, and the AWS License Manager API. Or, you can automate the creation of self-managed licenses with AWS CloudFormation and Terraform templates.

RDS for Db2 resources that match the product information filter are automatically associated with the self-managed license. Discovery of RDS for Db2 DB instances can take up to 24 hours.

The following table shows available values for the Engine Edition product information filter for RDS for Db2.

Value	Description
db2-se	Db2 Standard Edition
db2-ae	Db2 Advanced Edition

## Topics

- [Terminology](#)
- [Creating a self-managed license in AWS License Manager](#)
- [Automating the creation of self-managed licenses in AWS License Manager with templates](#)
- [Settings for creating self-managed licenses](#)

## Terminology

This page uses the following terminology when discussing the Amazon RDS integration with AWS License Manager.

### Self-managed license

Self-managed license is a term used in AWS License Manager. The Amazon RDS console refers to the license as an AWS License Manager configuration. A self-managed license contains licensing rules based on the terms of your enterprise agreements. The rules that you create determine how AWS processes commands that consume licenses. While creating a self-managed license, work closely with your organization's compliance team to review your enterprise agreements. For more information, see [Self-managed licenses in License Manager](#).

### Creating a self-managed license in AWS License Manager

You can create a self-managed license by using the AWS Management Console, the AWS License Manager CLI, and the AWS License Manager API.

#### Note

If you create an RDS for Db2 DB instance by using the AWS Management Console, you will create a self-managed license by entering a name for the license. Then Amazon RDS associates the DB instance with this license. (In the Amazon RDS console, this license is referred to as an AWS License Manager configuration.) If you want to create an RDS for Db2 DB instance by using the AWS License Manager CLI or AWS License Manager API, you must first create a self-managed license with the following steps. The same situation applies to restoring an RDS for Db2 DB instance to a point in time or from a snapshot.

## Console

### To create a self-managed license to track the license usage of your RDS for Db2 DB instances

1. Go to <https://console.aws.amazon.com/license-manager/>.
2. Create a self-managed license.

For instructions, see [Create a self-managed license](#) in the *AWS License Manager User Guide*.

Add a rule for an **RDS Product Information Filter** in the **Product Information** panel.

For more information, see [ProductInformation](#) in the *AWS License Manager API Reference*.

## AWS License Manager CLI

### Note

This procedure uses an AWS License Manager CLI command.

To create a self-managed license by using the AWS CLI, run the AWS License Manager [create-license-configuration](#) command. Use the `--cli-input-json` or `--cli-input-yaml` options to pass the options to the command.

For more information, see [the section called “Settings for creating self-managed licenses”](#).

The following command creates a self-managed license for Db2 Standard Edition.

```
aws license-manager create-license-configuration --cli-input-json file://rds-db2-se.json
```

The following JSON is the content of the `rds-db2-se.json` file used in the previous command.

```
{
    "Name": "rds-db2-se",
    "Description": "RDS Db2 Standard Edition",
    "LicenseCountingType": "vCPU",
    "LicenseCountHardLimit": false,
    "ProductInformationList": [
        {
            "ResourceType": "RDS",
            "ProductInformationFilterList": [
                {
                    "ProductInformationFilterName": "Engine Edition",
                    "ProductInformationFilterValue": ["db2-se"],
                    "ProductInformationFilterComparator": "EQUALS"
                }
            ]
        }
    ]
}
```

For more information about product information, see [Automated discovery of resource inventory](#) in the *AWS License Manager User Guide*.

For more information about the --cli-input parameter, see [Generating AWS CLI skeleton and input parameters from a JSON or YAML input file](#) in the *AWS CLI User Guide*.

## AWS License Manager API



This procedure uses an AWS License Manager API command.

To create a self-managed license, use the [CreateLicenseConfiguration](#) AWS License Manager API operation with the following required parameters:

- Name
- LicenseCountingType
- ProductInformationList
- ResourceType
- ProductInformationFilterList
- ProductInformationFilterName
- ProductInformationFilterValue
- ProductInformationFilterComparator

For more information about the parameters, see [the section called “Settings for creating self-managed licenses”](#).

### Automating the creation of self-managed licenses in AWS License Manager with templates

You can automate the creation of self-managed licenses by using AWS CloudFormation and Terraform templates.

The following example AWS CloudFormation template creates self-managed licenses for Db2 Standard Edition on RDS for Db2. For a template for Db2 Advanced Edition, update the values for Name, Description, and ProductInformationFilter.

```
AWSTemplateFormatVersion: "2010-09-09"
Description: CloudFormation template to create a License Configuration for Db2 Standard Edition on RDS for Db2.
```

#### Resources:

```
Db2LicenseConfiguration:
  Type: "AWS::LicenseManager::LicenseConfiguration"
  Properties:
    Name: "rds-db2-se"
    Description: "Db2 Standard Edition on RDS for Db2"
    LicenseCountingType: "vCPU"
    LicenseCountHardLimit: false
    ProductInformationList:
      - ResourceType: "RDS"
        ProductInformationFilterList:
          - ProductInformationFilterName: "Engine Edition"
        ProductInformationFilterValue:
          - "db2-se"
        ProductInformationFilterComparator: "EQUALS"
```

For more information about using AWS CloudFormation with Amazon RDS, see [Creating Amazon RDS resources with AWS CloudFormation](#).

The following example Terraform template creates self-managed licenses for Db2 Standard Edition on RDS for Db2. Replace *us-east-1* with your AWS Region. For a template for Db2 Advanced Edition, update the values for name, description, and product\_information\_filter.

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_licensemanager_license_configuration" "rds_db2_license_config" {
  name              = "rds-db2-se"
  description       = "Db2 Standard Edition on RDS for Db2"
  license_counting_type = "vCPU"
  license_count_hard_limit = false

  product_information_list {
    resource_type = "RDS"

    product_information_filter {
      name      = "Engine Edition"
```

```
    comparator = "EQUALS"
    value      = ["db2-se"]
}
}
}
```

For more information about using Terraform and Amazon RDS, see [Using Terraform as an IaC tool for the AWS Cloud](#) and [Best practices for using the Terraform AWS Provider](#) in *AWS Prescriptive Guidance*.

## Settings for creating self-managed licenses

In the following table, you can find details about the settings for creating self-managed licenses by using the AWS License Manager CLI, the AWS License Manager API, an AWS CloudFormation template, and a Terraform template. The parameter name in the following table appears in the format of the name used in the AWS License Manager API and the AWS CloudFormation template.

Parameter name	Data type	Required	Description
Name	string	Yes	The name of the license configuration.
Description	string	No	The description of the license configuration.
LicenseCountingType	string	Yes	The dimension used to track the license inventory. Valid value: vCPU.
LicenseCountHardLimit	boolean	No	Indicates whether hard or soft license enforcement is used. Exceeding a hard limit blocks the launch of new instances.

Parameter name	Data type	Required	Description
ProductInformationList	array of objects	Yes	A list of product information for a license configuration.
ResourceType	string	Yes	The resource type. Valid value: RDS.
ProductInformationFilterList	array of objects	Yes	A list of product information filters for a license configuration.
ProductInformationFilterName	string	Yes	The name of the type of filter being declared. Valid value: Engine Edition.
ProductInformationFilterValue	array of strings	Yes	The value to filter on. You must only specify one value. Valid values: db2-se or db2-ae.
ProductInformationFilterComparator	string	Yes	The logical operator for ProductInformationFilterName . Valid value: EQUALS.

## Db2 license through AWS Marketplace

In the Db2 license through AWS Marketplace model, you pay an hourly rate to subscribe to Db2 licenses. This model helps you get started quickly with RDS for Db2 without needing to purchase licenses.

To use Db2 license through AWS Marketplace, you need an active AWS Marketplace subscription for the particular IBM Db2 edition that you want to use. If you don't already have one, [subscribe to AWS Marketplace](#) for that IBM Db2 edition.

Amazon RDS supports Db2 license through AWS Marketplace for IBM Db2 Standard Edition and IBM Db2 Advanced Edition.

## Topics

- [Terminology](#)
- [Payments and billing](#)
- [Subscribing to Db2 Marketplace listings and registering with IBM](#)
- [Obtaining a private offer](#)

## Terminology

This page uses the following terminology when discussing the Amazon RDS integration with AWS Marketplace.

### SaaS subscription

In AWS Marketplace, software-as-a-service (SaaS) products such as the pay-as-you-go license model adopt a usage-based subscription model. IBM, the software seller for Db2, tracks your usage and you pay only for what you use.

### Public offer

Public offers allow you to purchase AWS Marketplace products directly from the AWS Management Console.

### Private offer

Private offers are a purchasing program that allow sellers and buyers to negotiate custom prices and end user licensing agreement (EULA) terms for purchases in AWS Marketplace.

### Db2 Marketplace fees

Fees charged for the Db2 software license usage by IBM. These service fees are metered through AWS Marketplace and appear on your AWS bill under the AWS Marketplace section.

## Amazon RDS fees

Fees that AWS charges for the RDS for Db2 services, which excludes licenses when using AWS Marketplace for Db2 licenses. Fees are metered through the Amazon RDS service being used and appear on your AWS bill.

## Payments and billing

RDS for Db2 integrates with AWS Marketplace to offer hourly, pay-as-you-go licenses for Db2. The Db2 Marketplace fees cover the license costs of the Db2 software, and the Amazon RDS fees cover the costs of your RDS for Db2 DB instance usage. For information about pricing, see [Amazon RDS for Db2 pricing](#).

To stop these fees, you must delete any RDS for Db2 DB instances. In addition, you can remove your subscriptions to AWS Marketplace for Db2 licenses. If you remove your subscriptions without deleting your DB instances, Amazon RDS will continue to bill you for the use of the DB instances. For more information, see [the section called “Deleting a DB instance”](#).

You can view bills and manage payments for your RDS for Db2 DB instances that use Db2 license through AWS Marketplace in the [AWS Billing console](#). Your bills includes two charges: one for your usage of Db2 license through AWS Marketplace and one for your usage of Amazon RDS. For more information about billing, see [Viewing your bill](#) in the *AWS Billing and Cost Management User Guide*.

## Subscribing to Db2 Marketplace listings and registering with IBM

To use Db2 license through AWS Marketplace, you must use the AWS Management Console to complete the following two tasks. You can't complete these tasks through the AWS CLI or the RDS API.

### Note

If you want to create your DB instances by using the AWS CLI or the RDS API, you must complete these two tasks first.

## Topics

- [Task 1: Subscribe to Db2 in AWS Marketplace](#)

- [Task 2: Register your subscription with IBM](#)

## Task 1: Subscribe to Db2 in AWS Marketplace

To use Db2 license with AWS Marketplace, you need to have an active AWS Marketplace subscription for Db2. Because subscriptions are associated with a specific IBM Db2 edition, you need to subscribe to Db2 in AWS Marketplace for each edition of Db2 that you want to use: [IBM Db2 Advanced Edition](#), [IBM Db2 Standard Edition](#). For information about AWS Marketplace subscriptions, see [SaaS usage-based subscriptions](#) in the *AWS Marketplace Buyer Guide*.

We recommend that you subscribe to Db2 in AWS Marketplace *before* you start to [create a DB instance](#).

## Task 2: Register your subscription with IBM

After you subscribe to Db2 in AWS Marketplace, complete the registration of your IBM order from the AWS Marketplace page for the type of Db2 subscription that you chose. On the AWS Marketplace page, choose **View purchase options**, and then choose **Set up your account**. You can register either with your existing IBM account or by creating a free IBM account.

### Obtaining a private offer

You can request an AWS Marketplace private offer for Db2 from IBM. For more information, see [Private offers](#) in the *AWS Marketplace Buyer Guide*.

#### Note

If you are an AWS Organizations user and received a private offer that was issued to your payer and member accounts, follow the procedure below to subscribe to Db2 directly on each account in your organization.

### To obtain a Db2 private offer

1. After a private offer has been issued, sign in to the AWS Marketplace Console.
2. Open the email with a Db2 private offer link.
3. Follow the link to directly access the private offer.

**Note**

Following this link before logging in to the correct account will result in a **Page note found (404)** error.

4. Review the terms and conditions.
5. Choose **Accept terms**.

**Note**

If an AWS Marketplace private offer is not accepted, the Db2 service fees from AWS Marketplace will continue to be billed at the public hourly rate.

6. To verify the offer details, select **Show details** in the product listing.

After you've completed the procedure, you can create your DB instance by following the steps in [the section called "Creating a DB instance"](#). In the AWS Management Console, under **License**, make sure that you choose **Through AWS Marketplace**.

## Switching between Db2 licenses

You can switch between Db2 licenses in RDS for Db2. For example, you can start with bring your own license (BYOL), and then switch to Db2 license through AWS Marketplace.

**Important**

If you want to switch to Db2 license through AWS Marketplace, make sure that you have an active AWS Marketplace subscription for the IBM Db2 edition that you want to use. If you don't, first [subscribe to Db2 in AWS Marketplace](#) for that Db2 edition, and then complete the restore procedure.

## Console

### To switch between Db2 licenses

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.

2. In the navigation pane, choose **Automated backups**.

The automated backups are displayed on the **Current Region** tab.

3. Choose the DB instance that you want to restore.

4. For **Actions**, choose **Restore to point in time**.

The **Restore to point in time** window appears.

5. Choose **Latest restorable time** to restore to the latest possible time, or choose **Custom** to choose a time.

If you chose **Custom**, enter the date and time you want to restore the instance to.

 **Note**

Times are shown in your local time zone, which is indicated by an offset from Coordinated Universal Time (UTC). For example, UTC-5 is Eastern Standard Time/Central Daylight Time.

6. For **DB engine**, choose the Db2 license you want to use.

7. For **DB instance identifier**, enter the name of the target restored DB instance. The name must be unique.

8. Choose other options as needed, such as DB instance class, storage, and whether you want to use storage autoscaling.

For information about each setting, see [Settings for DB instances](#).

9. Choose **Restore to point in time**.

For more information, see [Restoring a DB instance to a specified time for Amazon RDS](#).

## AWS CLI

To switch between Db2 licenses, run the [restore-db-instance-to-point-in-time](#) command. The following example restores the latest point-in-time version, sets the DB engine to IBM Db2 Advanced Edition, and sets the license model to Db2 license through AWS Marketplace.

You can specify other settings. For information about each setting, see [Settings for DB instances](#).

## Example

For Linux, macOS, or Unix:

```
aws rds restore-db-instance-to-point-in-time \
--source-db-instance-identifier my_source_db_instance \
--target-db-instance-identifier my_target_db_instance \
--use-latest-restorable-time \
--engine db2-ae \
--license-model marketplace-license
```

For Windows:

```
aws rds restore-db-instance-to-point-in-time ^
--source-db-instance-identifier my_source_db_instance ^
--target-db-instance-identifier my_target_db_instance ^
--use-latest-restorable-time ^
--engine db2-ae ^
--license-model marketplace-license
```

For more information, see [Restoring a DB instance to a specified time for Amazon RDS](#).

## RDS API

To switch between Db2 licenses, call the Amazon RDS API [RestoreDBInstanceToPointInTime](#) operation with the following parameters:

- `SourceDBInstanceIdentifier`
- `TargetDBInstanceIdentifier`
- `RestoreTime`
- `Engine`
- `LicenseModel`

For more information, see [Restoring a DB instance to a specified time for Amazon RDS](#).

## Amazon RDS for Db2 instance classes

The computation and memory capacity of a DB instance is determined by its instance class. The DB instance class you need depends on your processing power and memory requirements.

## Supported RDS for Db2 instance classes

The supported Amazon RDS for Db2 instance classes are a subset of the Amazon RDS DB instance classes. For the complete list of Amazon RDS instance classes, see [DB instance classes](#).

### Topics

- [Supported RDS for Db2 instance classes for Db2 Standard Edition](#)
- [Supported RDS for Db2 instance classes for Db2 Advanced Edition](#)

### Supported RDS for Db2 instance classes for Db2 Standard Edition

The following table lists all instance classes supported for the Db2 Standard Edition of Db2 database version 11.5.9.0. These instance classes are available for both bring your own license (BYOL) and Db2 license through AWS Marketplace.

Instance class type	Instance class
General purpose instance classes with 3rd generation Intel Xeon Scalable processors, SSD storage, and network optimization	db.m6idn.large–db.m6idn.8xlarge
General purpose instance classes powered by 3rd generation Intel Xeon Scalable processors	db.m6in.large–db.m6in.8xlarge
General purpose instance classes	db.m7i.large–db.m7i.8xlarge db.m6i.large–db.m6i.8xlarge
Memory optimized instance classes with local NVMe-based SSDs, powered by 3rd generation Intel Xeon Scalable processors	db.x2iedn.xlarge
Memory optimized instance classes powered by 3rd generation Intel Xeon Scalable processors	db.r6idn.large–db.r6idn.4xlarge db.r6in.large–db.r6in.4xlarge
Memory optimized instance classes	db.r7i.large–db.r7i.8xlarge

Instance class type	Instance class
	db.r6i.large–db.r6i.4xlarge
Burstable performance instance classes	db.t3.small–db.t3.2xlarge

## Supported RDS for Db2 instance classes for Db2 Advanced Edition

The following table lists all instance classes supported for the Db2 Advanced Edition of Db2 database version 11.5.9.0. These instance classes are available for both bring your own license (BYOL) and Db2 license through AWS Marketplace.

Instance class type	Instance class
General purpose instance classes with 3rd generation Intel Xeon Scalable processors, SSD storage, and network optimization	db.m6idn.large–db.m6idn.32xlarge
General purpose instance classes powered by 3rd generation Intel Xeon Scalable processors	db.m6in.large–db.m6in.32xlarge
General purpose instance classes	db.m6i.large–db.m7i.48xlarge db.m7i.large–db.m7i.48xlarge
Memory optimized instance classes with local NVMe-based SSDs, powered by 3rd generation Intel Xeon Scalable processors	db.x2iedn.xlarge–db.x2iedn.32xlarge
Memory optimized instance classes powered by 3rd generation Intel Xeon Scalable processors	db.r6idn.large–db.r6idn.32xlarge db.r6in.8xlarge–db.r6in.32xlarge
Memory optimized instance classes	db.r6i.large–db.r7i.48xlarge db.r7i.large–db.r7i.48xlarge

Instance class type	Instance class
Burstable performance instance classes	db.t3.small–db.t3.2xlarge

## Amazon RDS for Db2 default roles

RDS for Db2 adds the following six roles and grants them to the `master_user_role` with the `ADMIN` option. When the database is provisioned, RDS for Db2 grants `master_user_role` to the master user. The master user can in turn grant these roles to other users, groups, or roles with native GRANT statements by connecting to the database.

- **DBA** – RDS for Db2 creates this empty role with DATAACCESS authorization. The master user can add more authorizations or privileges to this role, and then grant the role to other users, groups, or roles.
- **DBA\_RESTRICTED** – RDS for Db2 creates this empty role. The master user can add privileges to this role, and then grant the role to other users, groups, or roles.
- **DEVELOPER** – RDS for Db2 creates this empty role with DATAACCESS authorization. The master user can add more authorizations or privileges to this role, and then grant the role to other users, groups, or roles.
- **ROLE\_NULLID\_PACKAGES** – RDS for Db2 grants EXECUTE privileges to this role on ALL NULLID packages that were bound by Db2 when `CREATE DATABASE` was run.
- **ROLE\_PROCEDURES** – RDS for Db2 grants EXECUTE privileges to this role on all SYSIBM procedures.
- **ROLE\_TABLESPACES** – RDS for Db2 grants USAGE privileges on tablespaces created by the `CREATE DATABASE` command.

## Amazon RDS for Db2 parameters

Amazon RDS for Db2 uses three types of parameters: database manager configuration parameters, registry variables, and database configuration parameters. You can manage the first two types through parameter groups and the last type through the [`rdsadmin.update\_db\_param`](#) stored procedure.

By default, an RDS for Db2 DB instance uses a DB parameter group that is specific to a Db2 database and DB instance. This parameter group contains parameters for the IBM Db2 database

engine, specifically the database manager configuration parameters and registry variables. For information about working with parameter groups, see [Parameter groups for Amazon RDS](#).

RDS for Db2 database configuration parameters are set to the default values of the storage engine that you have selected. For more information about Db2 parameters, see the [Db2 database configuration parameters](#) in the IBM Db2 documentation.

## Topics

- [Viewing the parameters in parameter groups](#)
- [Viewing all parameters with Db2 commands](#)
- [Modifying the parameters in parameter groups](#)
- [Modifying the database configuration parameters with Db2 commands](#)

## Viewing the parameters in parameter groups

The database manager configuration parameters and the registry variables are set in parameter groups. You can view the database manager configuration parameters and the registry variables for a specific Db2 version by using the AWS Management Console, the AWS CLI, or the RDS API.

### Console

#### To view the parameter values for a DB parameter group

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Parameter groups**.  
The DB parameter groups appear in a list.
3. Choose the name of the parameter group to see its list of parameters.

### AWS CLI

You can view the database manager configuration parameters and the registry variables for a Db2 version by running the [describe-engine-default-parameters](#) command. Specify one of the following values for the --db-parameter-group-family option:

- db2-ae-11.5
- db2-se-11.5

For example, to view the parameters for Db2 Standard Edition 11.5, run the following command:

```
aws rds describe-engine-default-parameters --db-parameter-group-family db2-se-11.5
```

This command produces output similar to the following example:

```
{  
    "EngineDefaults": {  
        "Parameters": [  
            {  
                "ParameterName": "agent_stack_sz",  
                "ParameterValue": "1024",  
                "Description": "You can use this parameter to determine the amount of  
memory that is allocated by Db2 for each agent thread stack.",  
                "Source": "engine-default",  
                "ApplyType": "static",  
                "DataType": "integer",  
                "AllowedValues": "256-32768",  
                "IsModifiable": false  
            },  
            {  
                "ParameterName": "agentpri",  
                "ParameterValue": "-1",  
                "Description": "This parameter controls the priority given to all  
agents and to other database manager instance processes and threads by the operating  
system scheduler. This priority determines how CPU time is allocated to the database  
manager processes, agents, and threads relative to other processes and threads running  
on the machine.",  
                "Source": "engine-default",  
                "ApplyType": "static",  
                "DataType": "integer",  
                "AllowedValues": "1-99",  
                "IsModifiable": false  
            },  
            ...  
        ]  
    }  
}
```

To list only the modifiable parameters for Db2 Standard Edition 11.5, run the following command:

For Linux, macOS, or Unix:

```
aws rds describe-engine-default-parameters \
    --db-parameter-group-family db2-se-11.5 \
    --query 'EngineDefaults.Parameters[?IsModifiable==`true`].
{ParameterName:ParameterName, DefaultValue:ParameterValue}'
```

For Windows:

```
aws rds describe-engine-default-parameters ^
    --db-parameter-group-family db2-se-11.5 ^
    --query 'EngineDefaults.Parameters[?IsModifiable==`true`].
{ParameterName:ParameterName, DefaultValue:ParameterValue}'
```

## RDS API

To view the parameter values for a DB parameter group, use the [DescribeDBParameters](#) operation with the following required parameter.

- DBParameterGroupName

## Viewing all parameters with Db2 commands

You can view the settings for database manager configuration parameters, database configuration parameters, and registry variables by using Db2 commands.

### To view the settings

1. Connect to your Db2 database. In the following example, replace *database\_name*, *master\_username*, and *master\_password* with your information.

```
db2 "connect to database_name user master_username using master_password"
```

2. Find the supported Db2 version.

```
db2 "select service_level, fixpack_num from table(sysproc.env_get_inst_info()) as
instanceinfo"
```

3. View the parameters for a specific Db2 version.

- View database manager configuration parameters by running the following command:

```
db2 "select cast(substr(name,1,24) as varchar(24)) as name, case  
      when value_flags = 'NONE' then '' else value_flags end flags,  
      cast(substr(value,1,64) as varchar(64)) as current_value  
   from sysibmadm.dbmcfg  
  order by name asc with UR"
```

- View all of your database configuration parameters by running the following command:

```
db2 "select cast(substr(name,1,24) as varchar(24)) as name, case  
      when value_flags = 'NONE' then '' else value_flags end flags,  
      cast(substr(value,1,64) as varchar(64)) as current_value  
   from table(db_get_cfg(null)) order by name asc, member asc with UR"
```

- View the currently set registry variables by running the following command:

```
db2 "select cast(substr(reg_var_name,1,50) as varchar(50)) as reg_var_name,  
      cast(substr(reg_var_value,1,50) as varchar(50)) as reg_var_value,  
      level from table(env_get_reg_variables(null))  
  order by reg_var_name,member with UR"
```

## Modifying the parameters in parameter groups

You can modify the database manager configuration parameters and the registry variables in custom parameter groups by using the AWS Management Console, the AWS CLI, or the RDS API. For more information, see [DB parameter groups for Amazon RDS DB instances](#).

### Console

#### To modify database manager configuration parameters and registry variables

- Create a custom parameter group. For more information about creating a DB parameter group, see [Creating a DB parameter group in Amazon RDS](#).
- Modify the parameters in that custom parameter group. For more information about modifying a parameter group, see [Modifying parameters in a DB parameter group in Amazon RDS](#).

## AWS CLI

### To modify database manager configuration parameters and registry variables

1. Create a custom parameter group by running the [create-db-parameter-group](#) command.

Include the following required options:

- `--db-parameter-group-name` – A name for the parameter group that you are creating.
- `--db-parameter-group-family` – The Db2 engine edition and major version. Valid values: db2-se-11.5, db2-ae-11.5.
- `--description` – A description for this parameter group.

For more information about creating a DB parameter group, see [Creating a DB parameter group in Amazon RDS](#).

2. Modify the parameters in the custom parameter group that you created by running the [modify-db-parameter-group](#) command.

Include the following required options:

- `--db-parameter-group-name` – The name of the parameter group that you created.
- `--parameters` – An array of parameter names, values, and the application methods for the parameter update.

For more information about modifying a parameter group, see [Modifying parameters in a DB parameter group in Amazon RDS](#).

## RDS API

### To modify database manager configuration parameters and registry variables

1. Create a custom DB parameter group by using the [CreateDBParameterGroup](#) operation.

Include the following required parameters:

- `DBParameterGroupName`
- `DBParameterGroupFamily`

- Description

For more information about creating a DB parameter group, see [Creating a DB parameter group in Amazon RDS](#).

2. Modify the parameters in the custom parameter group that you created by using the [ModifyDBParameterGroup](#) operation.

Include the following required parameters:

- DBParameterGroupName
- Parameters

For more information about modifying a parameter group, see [Modifying parameters in a DB parameter group in Amazon RDS](#).

## Modifying the database configuration parameters with Db2 commands

You can modify the database configuration parameters with Db2 commands.

### To modify the database configuration parameters

1. Connect to the rdsadmin database. In the following example, replace *master\_username* and *master\_password* with your information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Change the database configuration parameters by calling the `rdsadmin.update_db_param` stored procedure. For more information, see [rdsadmin.update\\_db\\_param](#).

```
db2 "call rdsadmin.update_db_param(  
      'database_name',  
      'parameter_to_modify',  
      'changed_value',  
      'restart_database' )"
```

## EBCDIC collation for Db2 databases on Amazon RDS

Amazon RDS for Db2 supports EBCDIC collation for Db2 databases. You can only specify an EBCDIC collation sequence for a database when you create the database by using the Amazon RDS [the section called “rdsadmin.create\\_database” stored procedure](#).

When you create an RDS for Db2 DB instance by using the AWS Management Console, AWS CLI, or RDS API, you can specify a database name. If you specify a database name, Amazon RDS creates a database with the default collation of SYSTEM. If you need to create a database with EBCDIC collation, don't specify a database name when you create a DB instance.

The collation for a database in RDS for Db2 is set at the time of creation and is immutable.

### To create a Db2 database with EBCDIC collation

1. If you don't have an RDS for Db2 DB instance, create a DB instance but don't specify a database name. You can create a DB instance by using the AWS Management Console, AWS CLI, or RDS API. For more information, see [Creating a DB instance](#).
2. Create a Db2 database and set the collation option to an EBCDIC value by calling the `rdsadmin.create_database` stored procedure. For more information, see [rdsadmin.create\\_database](#).

 **Important**

After you create a database using the stored procedure, you can't change the collation sequence. If you want a database to use a different collation sequence, drop the database by calling the [the section called “rdsadmin.drop\\_database” stored procedure](#). Then, create a database with the required collation sequence.

## Local time zone for Amazon RDS for Db2 DB instances

The time zone of an Amazon RDS DB instance running Db2 is set by default. The default is Coordinated Universal Time (UTC). To match the time zone of your applications, you can set the time zone of your DB instance to a local time zone instead.

You set the time zone when you first create your DB instance. You can create your DB instance by using the AWS Management Console, the RDS API, or the AWS CLI. For more information, see [Creating a DB instance](#).

If your DB instance is part of a Multi-AZ deployment, then when it fails over, its time zone remains the local time zone that you set.

You can restore your DB instance to a point in time that you specify. The time appears in your local time zone. For more information, see [Restoring a DB instance to a specified time for Amazon RDS](#).

Setting the local time zone on your DB instance has the following limitations:

- You can't modify the time zone of an existing Amazon RDS for Db2 DB instance.
- You can't restore a snapshot from a DB instance in one time zone to a DB instance in a different time zone.
- We strongly recommend that you don't restore a backup file from one time zone to a different time zone. If you restore a backup file from one time zone to another, then you must audit your queries and applications for the effects of the time zone change.

## Available time zones

You can use the following values for the time zone setting.

Zone	Time zone
Africa	Africa/Cairo, Africa/Casablanca, Africa/Harare, Africa/Lagos, Africa/Luanda, Africa/Monrovia, Africa/Nairobi, Africa/Tripoli, Africa/Windhoek
America	America/Araguaina, America/Argentina/Buenos_Aires, America/Asuncion, America/Bogota, America/Caracas, America/Chicago, America/Chihuahua, America/Cuiaba, America/Denver, America/Detroit, America/Fortaleza, America/Godthab, America/Guatemala, America/Halifax, America/Lima, America/Los_Angeles, America/Manaus, America/Matamoros, America/Mexico_City, America/Monterrey, America/Montevideo, America/New_York, America/Phoenix, America/Santiago, America/Sao_Paulo, America/Tijuana, America/Toronto
Asia	Asia/Amman, Asia/Ashgabat, Asia/Baghdad, Asia/Baku, Asia/Bangkok, Asia/Beirut, Asia/Calcutta, Asia/Damascus, Asia/Dhaka, Asia/Hong_Kong, Asia/Irkutsk, Asia/Jakarta, Asia/Jerusalem, Asia/Kabul, Asia/Karachi, Asia/Kathmandu, Asia/Kolkata, Asia/Krasnoyarsk, Asia/Magadan, Asia/Manila, Asia/Muscat, Asia/Novosibirsk, Asia/Rangoon, Asia/Riyadh, Asia/Seoul, Asia/

<b>Zone</b>	<b>Time zone</b>
	Shanghai, Asia/Singapore, Asia/Taipei, Asia/Tehran, Asia/Tokyo, Asia/Ulaanbaatar, Asia/Vladivostok, Asia/Yakutsk, Asia/Yerevan
Atlantic	Atlantic/Azores, Atlantic/Cape_Verde
Australia	Australia/Adelaide, Australia/Brisbane, Australia/Darwin, Australia/Eucla, Australia/Hobart, Australia/Lord_Howe, Australia/Perth, Australia/Sydney
Brazil	Brazil/DeNoronha, Brazil/East
Canada	Canada/Newfoundland, Canada/Saskatchewan
Etc	Etc/GMT-3
Europe	Europe/Amsterdam, Europe/Athens, Europe/Berlin, Europe/Dublin, Europe/Helsinki, Europe/Kaliningrad, Europe/London, Europe/Madrid, Europe/Moscow, Europe/Paris, Europe/Prague, Europe/Rome, Europe/Sarajevo, Europe/Stockholm
Pacific	Pacific/Apia, Pacific/Auckland, Pacific/Chatham, Pacific/Fiji, Pacific/Guam, Pacific/Honolulu, Pacific/Kiritimati, Pacific/Marquesas, Pacific/Samoa, Pacific/Tongatapu, Pacific/Wake
US	US/Alaska, US/Central, US/East-Indiana, US/Eastern, US/Pacific
UTC	UTC

# Prerequisites for creating an Amazon RDS for Db2 DB instance

The following items are prerequisites before creating a DB instance.

## Topics

- [Administrator account](#)
- [Additional considerations](#)

## Administrator account

When you create a DB instance, you must designate an administrator account for the instance. Amazon RDS grants DBADM authority to this local database administrator account.

The administrator account has the following characteristics, capabilities, and limitations:

- Is a local user and not an AWS account.
- Doesn't have Db2 instance-level authorities such as SYSADM, SYSMAINT, or SYSCTRL.
- Can't stop or start a Db2 instance.
- Can't drop a Db2 database if you specified the name when you created the DB instance.
- Has full access to the Db2 database including catalog tables and views.
- Can create local users and groups by using Amazon RDS stored procedures.
- Can grant and revoke authorities and privileges.

The administrator account can perform the following tasks:

- Create, modify, or delete DB instances.
- Create DB snapshots.
- Initiate point-in-time restores.
- Create automated backups of DB snapshots.
- Create manual backups of DB snapshots.
- Use other Amazon RDS features.

## Additional considerations

Before creating a DB instance, consider the following items:

- Each Amazon RDS for Db2 DB instance can host up to 50 Db2 databases. For more information, see [Multiple databases on an Amazon RDS for Db2 DB instance](#).
- Initial database name
  - If you don't provide a database name when you create a DB instance, Amazon RDS doesn't create a database.
  - Don't provide a database name under the following circumstances:
    - You want to modify the db2\_compatibility\_vector parameter. For more information, see [Setting the db2\\_compatibility\\_vector parameter](#).
- In the bring your own license (BYOL) model, you must first create a custom parameter group that contains your IBM Customer ID and your IBM Site ID. For more information, see [Bring your own license \(BYOL\) for Db2](#).
- In the Db2 license through AWS Marketplace model, you need an active AWS Marketplace subscription for the particular IBM Db2 edition that you want to use. If you don't already have one, [subscribe to Db2 in AWS Marketplace](#) for the IBM Db2 edition that you want to use. For more information, see [Db2 license through AWS Marketplace](#).

# Multiple databases on an Amazon RDS for Db2 DB instance

You can create multiple databases on a single RDS for Db2 DB instance by calling the [`rdsadmin.create\_database`](#) stored procedure. A single RDS for Db2 DB instance is limited to 50 databases. This number includes databases in both activated and deactivated states.

## Note

If you create multiple databases on an RDS for Db2 DB instance that was created before November 15, 2024, then you must reboot the DB instance to enable support for multiple databases.

By default, Amazon RDS activates databases when you create them. To optimize memory resources, you can deactivate databases that you use infrequently and then activate them later when needed. For more information, see [the section called “Deactivating a database”](#) and [the section called “Activating a database”](#).

The number of activated databases on a DB instance depends on the available memory resources on the server. Memory resources differ based on the DB instance class and the amount of memory configured for the database. For information about DB instance classes, see [the section called “DB instance classes”](#). For information about how to update the memory for an RDS for Db2 database, see [the section called “rdsadmin.update\\_db\\_param”](#).

We recommend that you choose a DB instance class that has 2 GB of memory for common database tasks, operating system requirements, and other Amazon RDS automation tasks such as backups. For more information about changing the DB instance class, see [the section called “Modifying a DB instance”](#).

In addition, IBM recommends a minimum of 1 GB of memory for each active database. For more information, see [Disk and memory requirements](#) in the IBM documentation.

You can calculate the maximum number of active databases a DB instance can have with the following formula:

```
Active database limit = (total server memory - 2 GB) / 1 GB
```

The following example shows the maximum number of active databases for a DB instance with a db.m6i.xlarge DB instance class:

```
Active database limit = (total server memory - 2 GB) / 1 GB  
                      = (16 GB - 2 GB) / 1 GB  
                      = 14 databases
```

When Amazon RDS recovers a database after a crash, it activates the database if it was previously active. In certain cases, such as when you modify a DB instance class to a lower memory configuration, there might be insufficient memory to activate all databases on the DB instance. In those cases, Amazon RDS activates databases in the order in which they were created.

 **Note**

Any databases that Amazon RDS can't activate because of insufficient memory remain in a deactivated state.

# Connecting to your Db2 DB instance

After Amazon RDS provisions your Amazon RDS for Db2 DB instance, you can use any standard SQL client application to connect to the DB instance. Because Amazon RDS is a managed service, you can't sign in as SYSADM, SYSCTRL, SECADM, or SYSMAINT.

You can connect to a DB instance that is running the IBM Db2 database engine by using IBM Db2 CLP, IBM CLPPlus, DBeaver, or IBM Db2 Data Management Console.

## Note

Connecting to a Db2 database can fail if your RDS for Db2 DB instance doesn't have sufficient memory. For more information, see [the section called "Database connection error".](#)

## Topics

- [Finding the endpoint of your Amazon RDS for Db2 DB instance](#)
- [Connecting to your Amazon RDS for Db2 DB instance with IBM Db2 CLP](#)
- [Connecting to your Amazon RDS for Db2 DB instance with IBM CLPPlus](#)
- [Connecting to your Amazon RDS for Db2 DB instance with DBeaver](#)
- [Connecting to your Amazon RDS for Db2 DB instance with IBM Db2 Data Management Console](#)
- [Considerations for security groups with Amazon RDS for Db2](#)

## Finding the endpoint of your Amazon RDS for Db2 DB instance

Each Amazon RDS DB instance has an endpoint, and each endpoint has the DNS name and port number for the DB instance. To connect to your Amazon RDS for Db2 DB instance with a SQL client application, you need the DNS name and port number for your DB instance.

You can find the endpoint for a DB instance by using the AWS Management Console or the AWS CLI.

## Console

### To find the endpoint of an RDS for Db2 DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the upper-right corner of the console, choose the AWS Region of your DB instance.
3. Find the DNS name and port number for your RDS for Db2 DB Instance.
  - a. Choose **Databases** to display a list of your DB instances.
  - b. Choose the RDS for Db2 DB instance name to display the instance details.
  - c. On the **Connectivity & security** tab, copy the endpoint. Also, note the port number. You need both the endpoint and the port number to connect to the DB instance.

The screenshot shows the 'Connectivity & security' tab of an RDS instance configuration page. The tab bar includes Connectivity & security, Monitoring, Logs & events, Configuration, and Maintenance & backups. The Connectivity & security tab is active. The instance name is 'database-1'. The 'Endpoint' field contains 'database-1.' followed by a redacted domain, and the 'Port' field contains '50000', both of which are highlighted with red boxes. The Networking section shows the Availability Zone as 'us-east-2a' and the VPC as 'vpc-' followed by a redacted identifier. The Security section shows 'VPC security groups' with 'default' selected and 'Active' checked, and 'Publicly accessible' set to 'Yes'. The Certificate authority is listed as 'rds-ca-2019'.

Endpoint & port	Networking	Security
Endpoint database-1. .amazonaws.com	Availability Zone us-east-2a	VPC security groups default Active
Port 50000	VPC vpc- Subnet group default-vpc-	Publicly accessible Yes
	Subnets	Certificate authority <a href="#">Info</a> rds-ca-2019

## AWS CLI

To find the endpoint of an RDS for Db2 DB instance, run the [describe-db-instances](#) command. In the following example, replace **database-1** with the name of your DB instance.

For Linux, macOS, or Unix:

```
aws rds describe-db-instances \
```

```
--db-instance-identifier database-1 \
--query 'DBInstances[]'.
{DBInstanceIdentifier:DBInstanceIdentifier,DBName:DBName,Endpoint:Endpoint}' \
--output json
```

For Windows:

```
aws rds describe-db-instances ^
--db-instance-identifier database-1 ^
--query 'DBInstances[]'.
{DBInstanceIdentifier:DBInstanceIdentifier,DBName:DBName,Endpoint:Endpoint}' ^
--output json
```

This command produces output similar to the following example. The Address line in the output contains the DNS name.

```
[
{
    "DBInstanceIdentifier": "database-1",
    "DBName": "DB2DB",
    "Endpoint": {
        "Address": "database-1.123456789012.us-east-2.amazonaws.com",
        "Port": 50000,
        "HostedZoneId": "Z20C4A7DETW6VH"
    }
}
```

## Connecting to your Amazon RDS for Db2 DB instance with IBM Db2 CLP

You can use a command line utility such as IBM Db2 CLP to connect to Amazon RDS for Db2 DB instances. This utility is part of IBM Data Server Runtime Client. To download the client from IBM Fix Central, see [IBM Data Server Client Packages Version 11.5 Mod 8 Fix Pack 0](#) in IBM Support.

### Topics

- [Terminology](#)
- [Installing the client](#)
- [Connecting to a DB instance](#)
- [Troubleshooting connections to your RDS for Db2 DB instance](#)

## Terminology

The following terms help explain commands used when [connecting to your RDS for Db2 DB instance](#).

### **catalog tcpip node**

This command registers a remote database node with a local Db2 client, which makes the node accessible to the client application. To catalog a node, you provide information such as the server's host name, port number, and communication protocol. The cataloged node then represents a target server where one or more remote databases reside. For more information, see [CATALOG TCPIP/TCPIP4/TCPIP6 NODE command](#) in the IBM Db2 documentation.

### **catalog database**

This command registers a remote database with a local Db2 client, which makes the database accessible to the client application. To catalog a database, you provide information such as the database's alias, the node on which it resides, and the authentication type needed to connect to the database. For more information, see [CATALOG DATABASE command](#) in the IBM Db2 documentation.

## Installing the client

After [downloading the package for Linux](#), install the client using root or administrator privileges.

### **Note**

To install the client on AIX or Windows, follow the same procedure but modify the commands for your operating system.

### To install the client on Linux

1. Run `./db2_install -f sysreq` and choose **yes** to accept the license.
2. Choose the location to install the client.
3. Run `clientInstallDir/instance/db2icrt -s client instance_name`. Replace *instance\_name* with a valid operating system user on Linux. In Linux, the Db2 DB instance name is tied to the operating system username.

This command creates a **sql1ib** directory in the home directory of the designated user on Linux.

## Connecting to a DB instance

To connect to your RDS for Db2 DB instance, you need its DNS name and port number. For information about finding them, see [Finding the endpoint](#). You also need to know the database name, master username, and master password that you defined when you created your RDS for Db2 DB instance. For more information about finding them, see [Creating a DB instance](#).

### To connect to an RDS for Db2 DB instance with IBM Db2 CLP

1. Sign in with the username that you specified during the IBM Db2 CLP client installation.
2. Catalog your RDS for Db2 DB instance. In the following example, replace *node\_name*, *dns\_name*, and *port* with a name for the node in the local catalog, the DNS name for your DB instance, and the port number.

```
db2 catalog TCPIP node node_name remote dns_name server port
```

### Example

```
db2 catalog TCPIP node remnode remote database-1.123456789012.us-east-1.amazonaws.com server 50000
```

3. Catalog the rdsadmin database and your database. This will allow you to connect to the rdsadmin database to perform some administrative tasks using Amazon RDS stored procedures. For more information, see [Administering your RDS for Db2 DB instance](#).

In the following example, replace *database\_alias*, *node\_name*, and *database\_name* with aliases for this database, the name of the node defined in the previous step, and the name of your database. *server\_encrypt* encrypts your username and password over the network.

```
db2 catalog database rdsadmin [ as database_alias ] at node node_name
authentication server_encrypt
```

```
db2 catalog database database_name [ as database_alias ] at node node_name
authentication server_encrypt
```

## Example

```
db2 catalog database rdsadmin at node remnode authentication server_encrypt  
db2 catalog database testdb as rdsdb2 at node remnode authentication server_encrypt
```

4. Connect to your RDS for Db2 database. In the following example, replace *rds\_database\_alias*, *master\_username*, and *master\_password* with the name of your database, the master username, and master password of your RDS for Db2 DB instance.

```
db2 connect to rds_database_alias user master_username using master_password
```

This command produces output similar to the following example:

### Database Connection Information

```
Database server      = DB2/LINUXX8664 11.5.9.0  
SQL authorization ID = ADMIN  
Local database alias = TESTDB
```

5. Run queries and view results. The following example shows a SQL statement that selects the database you created.

```
db2 "select current server from sysibm.dual"
```

This command produces output similar to the following example:

```
1  
-----  
TESTDB  
  
1 record(s) selected.
```

## Troubleshooting connections to your RDS for Db2 DB instance

If you receive the following NULLID error, it usually indicates that your client and RDS for Db2 server versions don't match. For supported Db2 client versions, see [Supported combinations of clients, drivers and server levels](#) in the IBM Db2 documentation.

```
db2 "select * from syscat.tables"  
SQL0805N Package "NULLID.SQLC2029 0X4141414141454A69" was not found.  
SQLSTATE=51002
```

After you receive this error, you must bind packages from your older Db2 client to a Db2 server version supported by RDS for Db2.

### To bind packages from an older Db2 client to a newer Db2 server

1. Locate the bind files on the client machine. Typically, these files are located in the **bnd** directory of the Db2 client's installation path and have the extension **.bnd**.
2. Connect to the Db2 server. In the following example, replace *database\_name* with the name of your Db2 server. Replace *master\_username* and *master\_password* with your information. This user has DBADM authority.

```
db2 connect to database_name user master_username using master_password
```

3. Run the bind command to bind the packages.
  - a. Navigate to the directory where the bind files exist on the client machine.
  - b. Run the bind command for each file.

The following options are required:

- **blocking all** – Binds all packages in the bind file in a single database request.
- **grant public** – Grants permission to public to execute the package.
- **sqlerror continue** – Specifies that the bind process continues even if errors occur.

For more information about the bind command see [BIND command](#) in the IBM Db2 documentation.

4. Verify that the bind was successful by either querying the syscat.package catalog view or checking the message returned after the bind command.

For more information, see [DB2 v11.5 Bind File and Package Name List](#) in IBM Support.

# Connecting to your Amazon RDS for Db2 DB instance with IBM CLPPlus

You can use a utility such as IBM CLPPlus to connect to an Amazon RDS for Db2 DB instance. This utility is part of IBM Data Server Runtime Client. To download the client from IBM Fix Central, see [IBM Data Server Client Packages Version 11.5 Mod 8 Fix Pack 0](#) in IBM Support.

## **Important**

We recommend that you run IBM CLPPlus on an operating system that supports graphical user interfaces such as macOS, Windows, or Linux with Desktop. If running headless Linux, use switch **-nw** with CLPPlus commands.

## Topics

- [Installing the client](#)
- [Connecting to a DB instance](#)

## Installing the client

After downloading the package for Linux, install the client.

## **Note**

To install the client on AIX or Windows, follow the same procedure but modify the commands for your operating system.

## To install the client on Linux

1. Run **./db2\_install**.
2. Run **clientInstallDir/instance/db2icrt -s client *instance\_name***. Replace *instance\_name* with a valid operating system user on Linux. In Linux, the Db2 DB instance name is tied to the operating system username.

This command creates a **sql1ib** directory in the home directory of the designated user on Linux.

## Connecting to a DB instance

To connect to your RDS for Db2 DB instance, you need its DNS name and port number. For information about finding them, see [Finding the endpoint](#). You also need to know the database name, master username, and master password that you defined when you created your RDS for Db2 DB instance. For more information about finding them, see [Creating a DB instance](#).

### To connect to an RDS for Db2 DB instance with IBM CLPPlus

1. Review the command syntax. In the following example, replace *clientDir* with the location where the client is installed.

```
cd clientDir/bin  
./clpplus -h
```

2. Configure your Db2 server. In the following example, replace *dsn\_name*, *database\_name*, *endpoint*, and *port* with the DSN name, database name, endpoint, and port for your RDS for Db2 DB instance. For more information, see [Finding the endpoint of your Amazon RDS for Db2 DB instance](#).

```
db2cli writecfg add -dsn dsn_name -database database_name -host endpoint -port port  
-parameter "Authentication=SERVER_ENCRYPT"
```

3. Connect to your RDS for Db2 DB instance. In the following example, replace *master\_username* and *dsn\_name* with the master username and DSN name.

```
./clpplus -nw master_username@dsn_name
```

4. A Java Shell window opens. Enter the master password for your RDS for Db2 DB instance.

#### Note

If a Java Shell window doesn't open, run `./clpplus -nw` to use the same command line window.

Enter password: \*\*\*\*\*

A connection is made and produces output similar to the following example:

**Database Connection Information :**

```
-----  
Hostname = database-1.abcdefghijkl.us-east-1.rds.amazonaws.com  
Database server = DB2/LINUXX8664 SQL110590  
SQL authorization ID = admin  
Local database alias = DB2DB  
Port = 50000
```

5. Run queries and view results. The following example shows a SQL statement that selects the database you created.

```
SQL > select current server from sysibm.dual;
```

This command produces output similar to the following example:

```
1
```

```
-----  
DB2DB  
SQL>
```

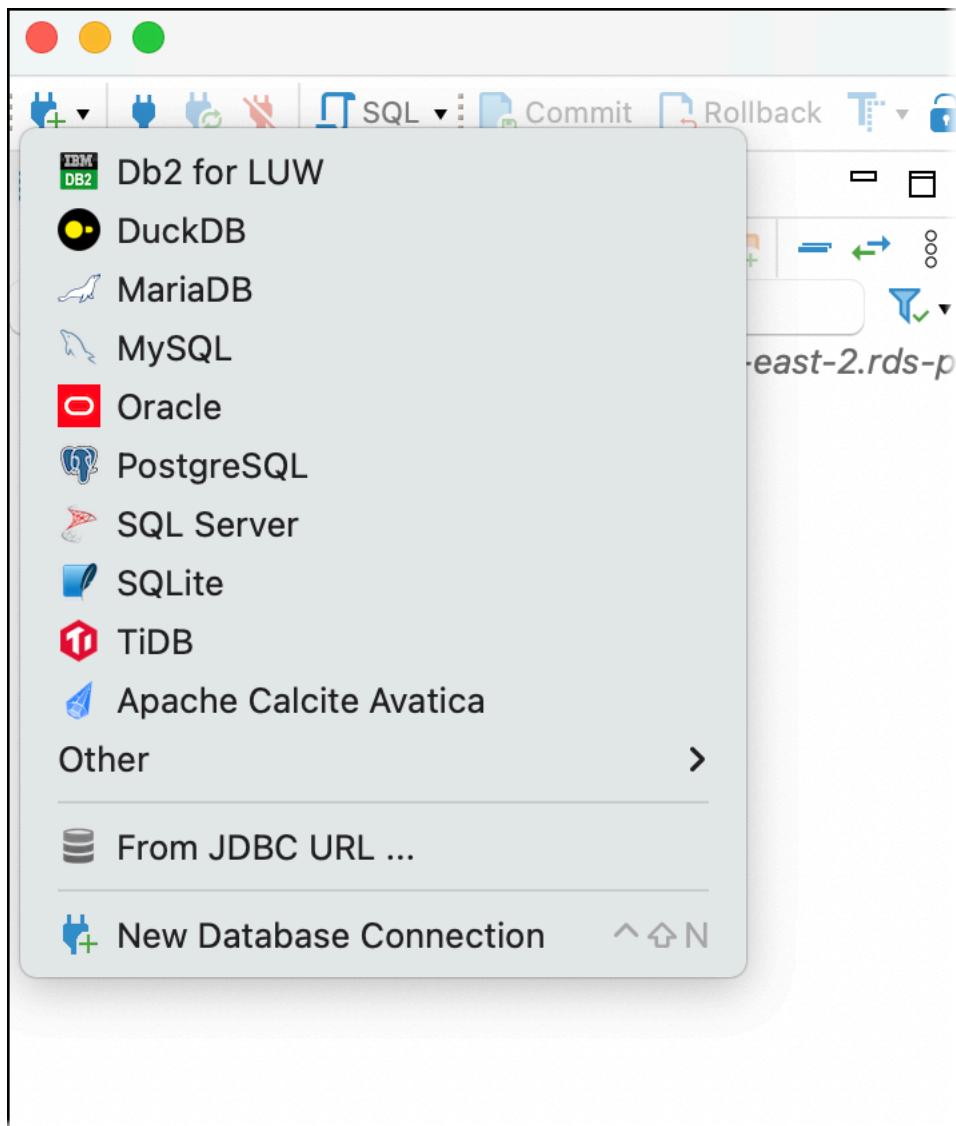
## Connecting to your Amazon RDS for Db2 DB instance with DBeaver

You can use third-party tools such as DBeaver to connect to Amazon RDS for Db2 DB instances. To download this utility, see [DBeaver Community](#).

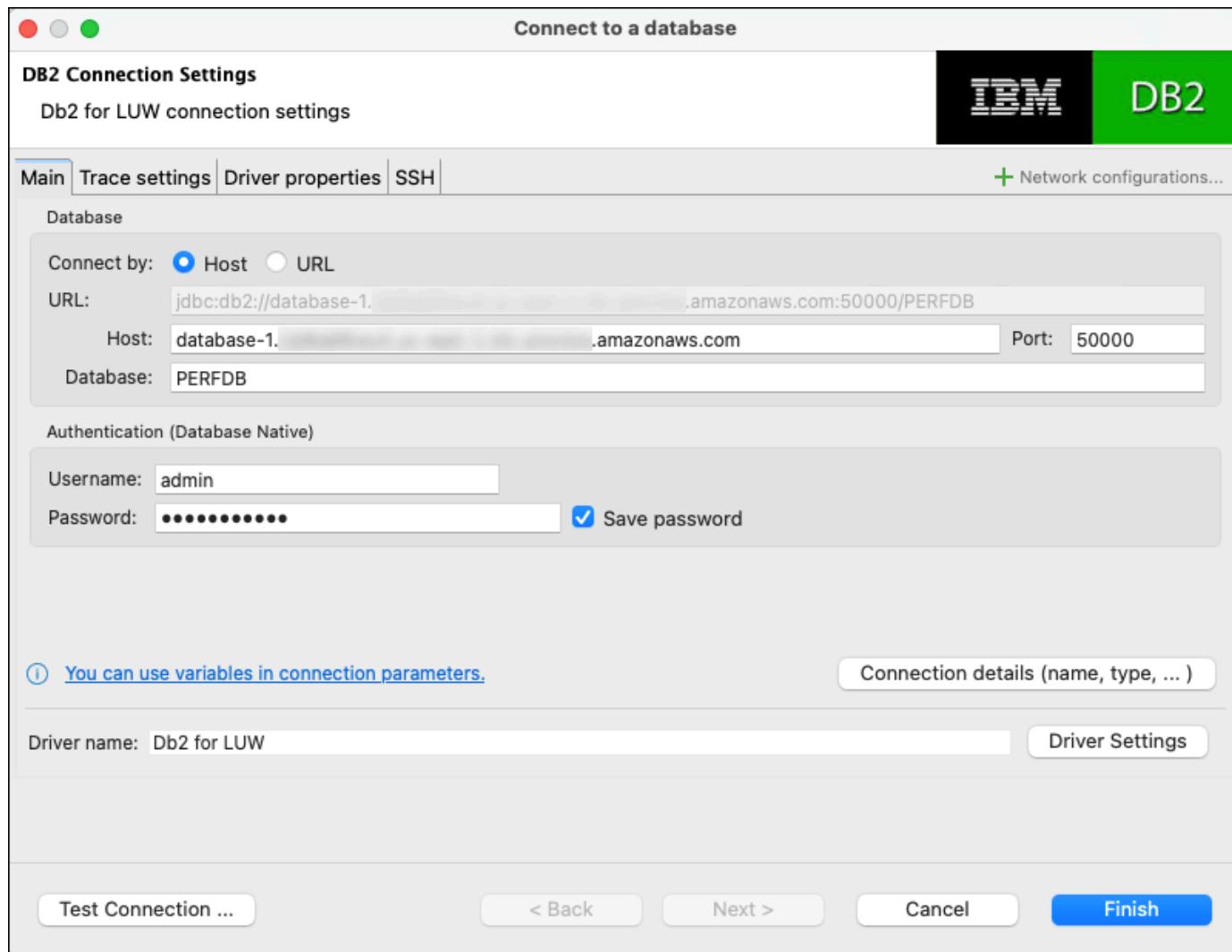
To connect to your RDS for Db2 DB instance, you need its DNS name and port number. For information about finding them, see [Finding the endpoint](#). You also need to know the database name, master username, and master password that you defined when you created your RDS for Db2 DB instance. For more information about finding them, see [Creating a DB instance](#).

### To connect to an RDS for Db2 DB instance with DBeaver

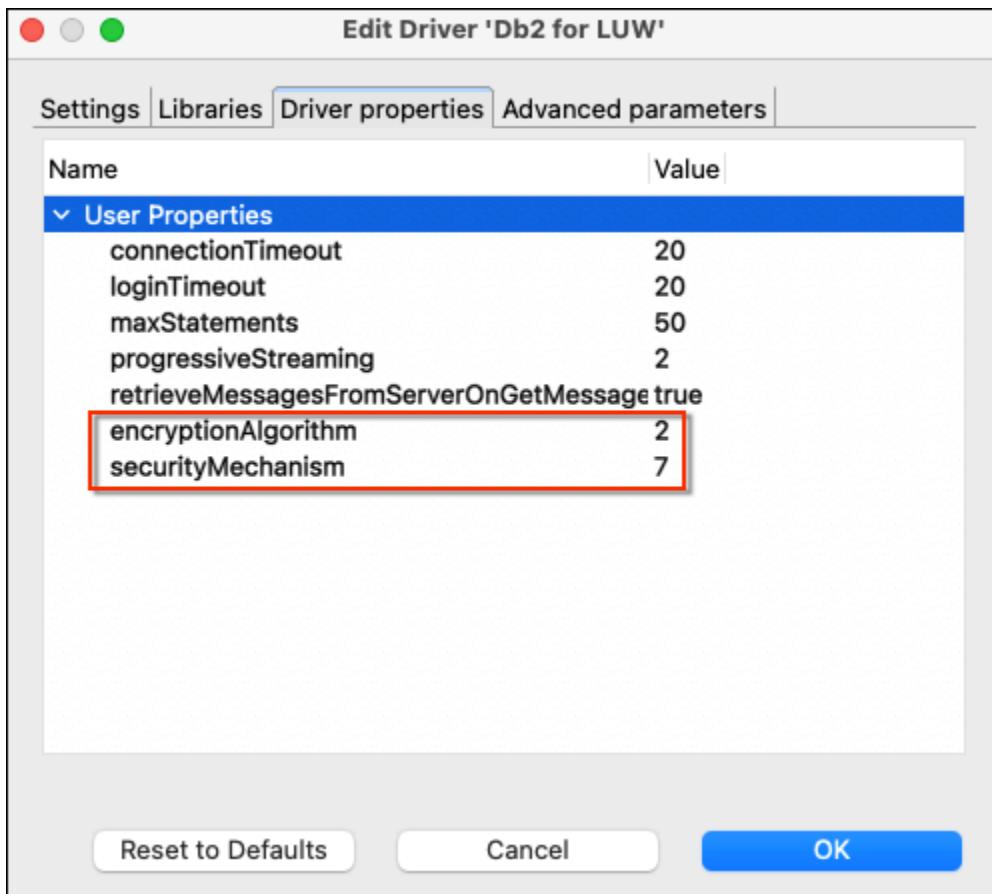
1. Start **DBeaver**.
2. Choose the **New Connection** icon in the toolbar and then choose **Db2 for LUW**.



3. In the **Connect to a database** window, provide information for your RDS for Db2 DB instance.
  - a. Enter the following information:
    - For **Host**, enter the DNS name of the DB instance.
    - For **Port**, enter the port number for the DB instance.
    - For **Database**, enter the name of the database.
    - For **Username**, enter the name of the database administrator for the DB instance.
    - For **Password**, enter the password of the database administrator for the DB instance.
  - b. Select **Save password**.
  - c. Choose **Driver Settings**.



4. In the **Edit Driver** window, specify additional security properties.
  - a. Choose the **Driver properties** tab.
  - b. Add two **User Properties**.
    - i. Open the context (right-click) menu, and then choose **Add new property**.
    - ii. For **Property Name**, add **encryptionAlgorithm**, and then choose **OK**.
    - iii. With the **encryptionAlgorithm** row selected, choose the **Value** column and add **2**.
    - iv. Open the context (right-click) menu, and then choose **Add new property**.
    - v. For **Property Name**, add **securityMechanism**, and then choose **OK**.
    - vi. With the **securityMechanism** row selected, choose the **Value** column and add **7**.
  - c. Choose **OK**.

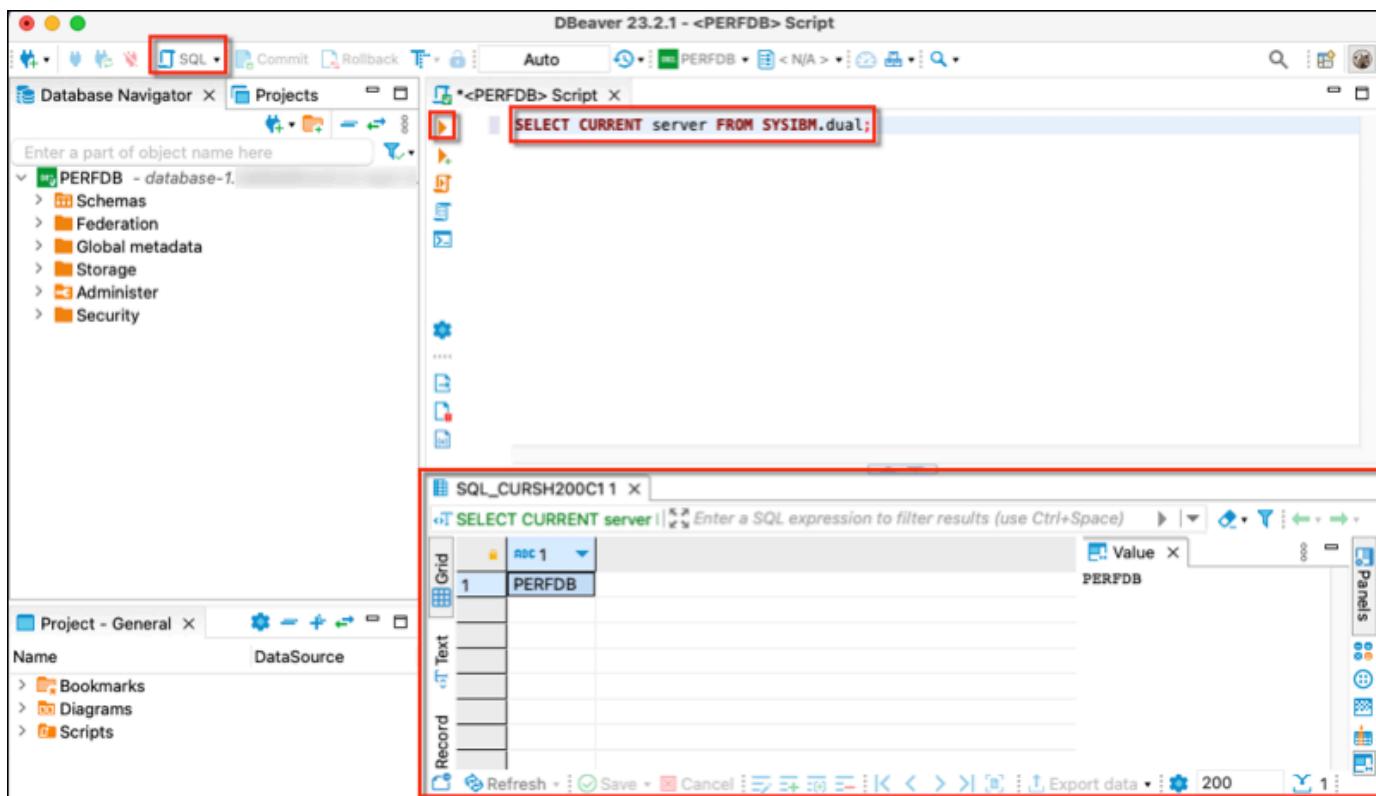


5. In the **Connect to a database** window, choose **Test Connection**. If you don't have a DB2 JDBC driver installed on your computer, then the driver automatically downloads.
6. Choose **OK**.
7. Choose **Finish**.
8. In the **Database Navigation** tab, choose the name of the database. You can now explore objects.

You are now ready to run SQL commands.

### To run SQL commands and view the results

1. In the top menu, choose **SQL**. This opens a SQL script panel.
2. In the **Script** panel, enter a SQL command.
3. To run the command, choose the **Execute SQL query** button.
4. In the SQL results panel, view the results of your SQL queries.



## Connecting to your Amazon RDS for Db2 DB instance with IBM Db2 Data Management Console

You can connect to your Amazon RDS for Db2 DB instance with IBM Db2 Data Management Console. IBM Db2 Data Management Console can administer and monitor several RDS for Db2 DB instances.

### Note

You must have an Amazon EC2 Linux or Windows machine that is on the same VPC and security group as your RDS for Db2 DB instance. The VPC and security group controls the connection to your DB instance through the internal network.

IBM Db2 Data Management Console requires a repository Db2 database to store metadata and performance metrics but can't automatically create a repository database for RDS for Db2. Instead, you must first create a repository database to monitor one or more RDS for Db2 DB instances. Then you can install IBM Db2 Data Management Console and connect to your RDS for Db2 DB instance with IBM Db2 Data Management Console.

## Topics

- [Step 1: Creating a repository database to monitor DB instances](#)
- [Step 2: Installing and setting up IBM Db2 Data Management Console](#)
- [Step 3: Configuring the repository database and connecting to RDS for Db2 DB instances](#)
- [Using IBM Db2 Data Management Console](#)

## Step 1: Creating a repository database to monitor DB instances

You can use an existing properly sized RDS for Db2 DB instance as a repository for IBM Db2 Data Management Console to monitor other RDS for Db2 DB instances. However, because the admin user doesn't have SYSCTRL authority to create buffer pools and tablespaces, using IBM Db2 Data Management Console repository creation to create a repository database fails. Instead, you must create a repository database. This repository database monitors your RDS for Db2 DB instances.

You can create a repository database in two different ways. You can create an RDS for Db2 database and then manually create a buffer pool, a user tablespace, and a system temporary tablespace. Or, you can create a separate Amazon EC2 instance to host an IBM Db2 Data Management Console repository database.

## Topics

- [Manually creating a buffer pool, a user tablespace, and a system temporary tablespace](#)
- [Creating an Amazon EC2 instance to host an IBM Db2 Data Management Console repository](#)

### Manually creating a buffer pool, a user tablespace, and a system temporary tablespace

#### To create a buffer pool, a user tablespace, and a system temporary tablespace

1. Connect to the rdsadmin database. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 connect to rdsadmin user master_username using master_password
```

2. Create a buffer pool for IBM Db2 Data Management Console. In the following example, replace *database\_name* with the name of the repository you created for IBM Db2 Data Management Console to monitor your RDS for Db2 DB instances.

```
db2 "call rdsadmin.create_bufferpool('database_name',
```

```
'BP4CONSOLE', 1000, 'Y', 'Y', 32768)"
```

3. Create a user tablespace for IBM Db2 Data Management Console. In the following example, replace *database\_name* with the name of the repository you created for IBM Db2 Data Management Console to monitor your RDS for Db2 DB instances.

```
db2 "call rdsadmin.create_tablespace('database_name',  
'TS4CONSOLE', 'BP4CONSOLE', 32768)"
```

4. Create a system temporary tablespace for IBM Db2 Data Management Console. In the following example, replace *database\_name* with the name of the repository you created for IBM Db2 Data Management Console to monitor your RDS for Db2 DB instances.

```
db2 "call rdsadmin.create_tablespace('database_name',  
'TS4CONSOLE_TEMP', 'BP4CONSOLE', 32768, 0, 0, 'S')"
```

You are now ready to install IBM Db2 Data Management Console. For more information about installation and setup, see [Step 2: Installing and setting up IBM Db2 Data Management Console](#).

### Creating an Amazon EC2 instance to host an IBM Db2 Data Management Console repository

You can create a separate Amazon Elastic Compute Cloud (Amazon EC2) instance to host an IBM Db2 Data Management Console repository. For information about creating an Amazon EC2 instance, see [Tutorial: Get started with Amazon EC2 Linux instances](#) in the *Amazon EC2 User Guide*.

## Step 2: Installing and setting up IBM Db2 Data Management Console

After you create a buffer pool, a user tablespace, and a system temporary tablespace, you are ready to install and set up IBM Db2 Data Management Console.

### Important

You must have an Amazon EC2 Linux or Windows machine that is on the same VPC and security group as your RDS for Db2 DB instance. The VPC and security group controls the connection to your DB instance through the internal network. Also, you must have already [created a repository database](#) for IBM Db2 Data Management Console.

## To install and set up IBM Db2 Data Management Console

1. Download IBM Db2 Data Management Console from [IBM Db2 Data Management Console Version 3.1x releases](#) on the IBM Support website.
2. Install IBM Db2 Data Management Console.
3. Open IBM Db2 Data Management Console and use the IP address of your Amazon EC2 machine and the port number you used for the HTTP or HTTPS connection to your Amazon EC2 instance. For example, use `http://xx.xx.xx.xx:11080` or `https://xx.xx.xx.xx:11081`. Replace `xx.xx.xx.xx` with the IP address of your Amazon EC2 machine. 11080 and 11081 are the default ports for HTTP and HTTPS connections.
4. (Optional) If you want to use port 80 or 443 on your Amazon EC2 instance, you can use either Apache httpd or a Nginx HTTP server to proxy the IBM Db2 Data Management Console port to either port 80 or 443. For more information, see [Apache HTTP Server Project](#) and [the nginx website](#).

To allow connection to IBM Db2 Data Management Console, you must edit the inbound rules in your security group. If you use a proxy, change the TCP/IP port 80 or 443 to redirect to the IBM Db2 Data Management Console ports. If you aren't using a proxy, change the TCP/IP port 80 or 443 to the default ports 11080 (HTTP) or 11081 (HTTPS).

You are now ready to log in to IBM Db2 Data Management Console to configure the repository database and to connect to your RDS for Db2 DB instances. For more information, see [Configuring the repository database and connecting to DB instances](#).

## Step 3: Configuring the repository database and connecting to RDS for Db2 DB instances

When you connect to the repository database for the first time, IBM Db2 Data Management Console automatically configures the repository. After the repository database is configured, you can add database connections to IBM Db2 Data Management Console.

To connect to your RDS for Db2 DB instance, you need its DNS name and port number. For information about finding them, see [Finding the endpoint](#). You also need to know the database name, master username, and master password that you defined when you created your RDS for Db2 DB instance. For more information about finding them, see [Creating a DB instance](#). If you are connecting over the internet, allow traffic to the database port. For more information, see [Creating a DB instance](#).

## To connect to RDS for Db2 DB instances with IBM Db2 Data Management Console

1. Log in to IBM Db2 Data Management Console with the credentials you set during installation.
2. Configure the repository.
  - a. In the **Connection and database** section, enter the following information for your RDS for Db2 DB instance:
    - For **Host**, enter the DNS name of the DB instance.
    - For **Port**, enter the port number for the DB instance.
    - For **Database**, enter the name of the database.

**Connection and database**

Set up a repository on the database to enable monitoring, run SQL statements, and explore database objects. Make sure the database for the repository exists even before you start configuring the repository. You can use your own Db2 server or use the standard edition with the restricted license for this repository database. If the database is not already created, can also use the [Db2 docker](#) image and get started.

**Important:** For a Db2 repository database, the user must have minimum of DBADM with DATAACCESS on the database and SYSCTRL on database instance privilege. To configure the repository by a normal Db2 user, refer to this [procedure](#).

Connection type	Host
IBM Db2	[REDACTED]
Port	Database
50000	SAMPLE
Repository schema <small>(1)</small>	JDBC URL attribute (optional)
IBMCORSOLE	Example: traceLevel=32;progressiveStream

- b. In the **Security and credential** section, enter the following information for your RDS for Db2 DB instance:
  - For **Security type**, choose **Encrypted user and password**.
  - For **Username**, enter the name of the database administrator for the DB instance.
  - For **Password**, enter the password of the database administrator for the DB instance.
- c. Choose **Test connection**.

**Note**

If the connection is unsuccessful, confirm that the database port is open through the inbound rules in your security group. For more information, see [Considerations for security groups with Amazon RDS for Db2](#).

If you didn't [manually create a buffer pool, a user tablespace, and a system temporary tablespace](#) in RDS for Db2, you might see the following error message:



Error:

"ADMIN" does not have the privilege to perform operation "CREATE BUFFERPOOL".. SQLCODE=-552, SQLSTATE=42502

For a Db2 repository database, the user must have minimum of DBADM with DATAACCESS on the database and SYSCTRL on database instance privilege. To configure the repository by a normal Db2 user, refer to this [procedure](#)

Make sure that you created a buffer table, a tablespace, and objects for an IBM Db2 Data Management Console repository to monitor your RDS for Db2 DB instance. Or, you can use an Amazon EC2 Db2 DB instance to host an IBM Db2 Data Management Console repository to monitor your RDS for Db2 DB instance. For more information, see [Step 1: Creating a repository database to monitor DB instances](#).

- d. After you successfully test your connection, choose **Next**.

**Security and credential**  
Specify the security and credentials to establish a connection and manage your Db2 database.

Use SSL ⓘ

Security type: Encrypted user and password

Encryption algorithm: AES

Username: rdsdb

Password: .....

If IBM Db2 Data Management Console finds the buffer pool, the user tablespace, and the system temporary tablespace in the RDS for Db2 DB instance, then IBM Db2 Data Management Console automatically configures the repository database. If you use your Db2 instance on your Amazon EC2 instance as the repository database, then IBM Db2 Data Management Console automatically creates the buffer pool and other objects.

3. In the **Set statistics event monitor opt-in** window, choose **Next**.
4. (Optional) Add new connection. If you want to use a different RDS for Db2 DB instance for administration and monitoring, then add a connection to a non-repository RDS for Db2 DB instance.
  - a. In the **Connection and database** section, enter the following information for the RDS for Db2 DB instance to use for administration and monitoring:
    - For **Connection name**, enter the Db2 database identifier.
    - For **Host**, enter the DNS name of the DB instance.
    - For **Port**, enter the port number for the DB instance.
    - For **Database**, enter the name of the database.

Connection and database  
Specify the parameters to establish a connection and manage your Db2 database.  
[Learn more](#)

Connection name	Connection type
rdsdb2	IBM Db2
Host	Port
database-2. .amazon	50000 -
Database	JDBC URL attribute (optional)
DB2DB	Example: traceLevel=32;progressiveStreaming=1'

- b. In the **Security and credential** section, select **Enable monitoring data collection**.
- c. Enter the following information for your RDS for Db2 DB instance:
  - For **Username**, enter the name of the database administrator for the DB instance.
  - For **Password**, enter the password of the database administrator for the DB instance.
- d. Choose **Test connection**.
- e. After you successfully test your connection, choose **Save**.

**Security and credential**

Specify the security and credentials to establish a connection and manage your Db2 database.

Use SSL ⓘ

Enable monitoring data collection ⓘ

Security type      Encryption algorithm

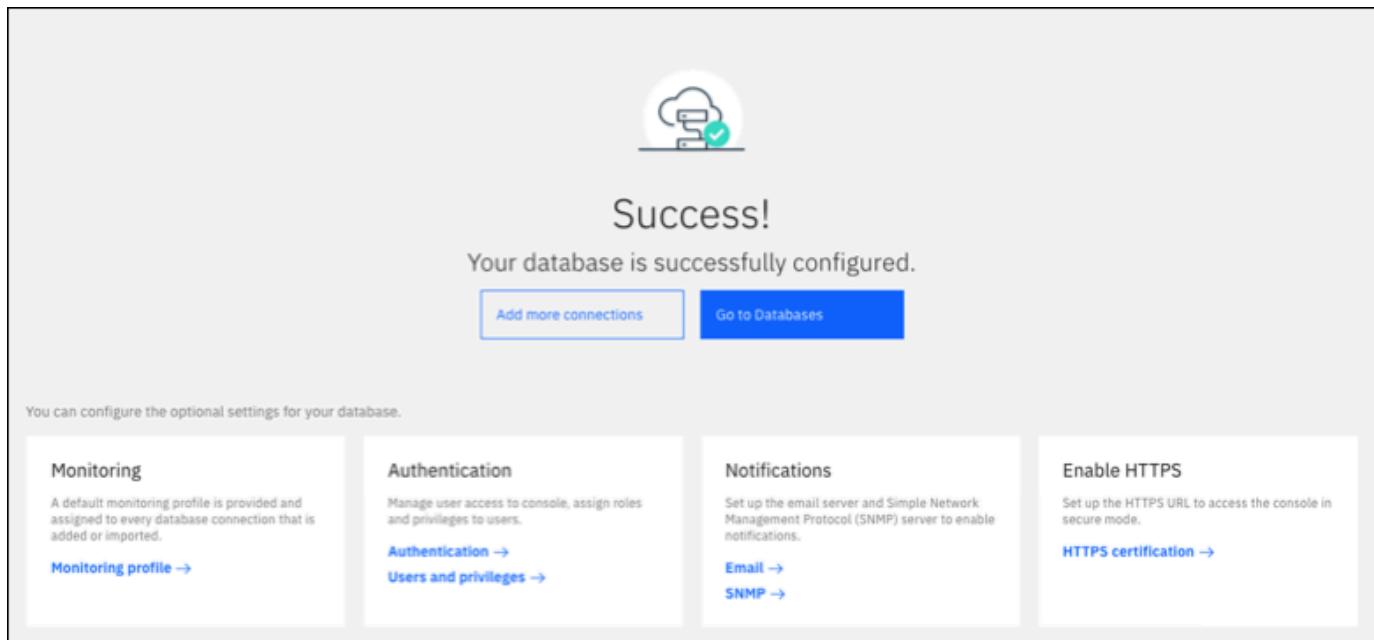
Encrypted user and password      AES

Username      Password

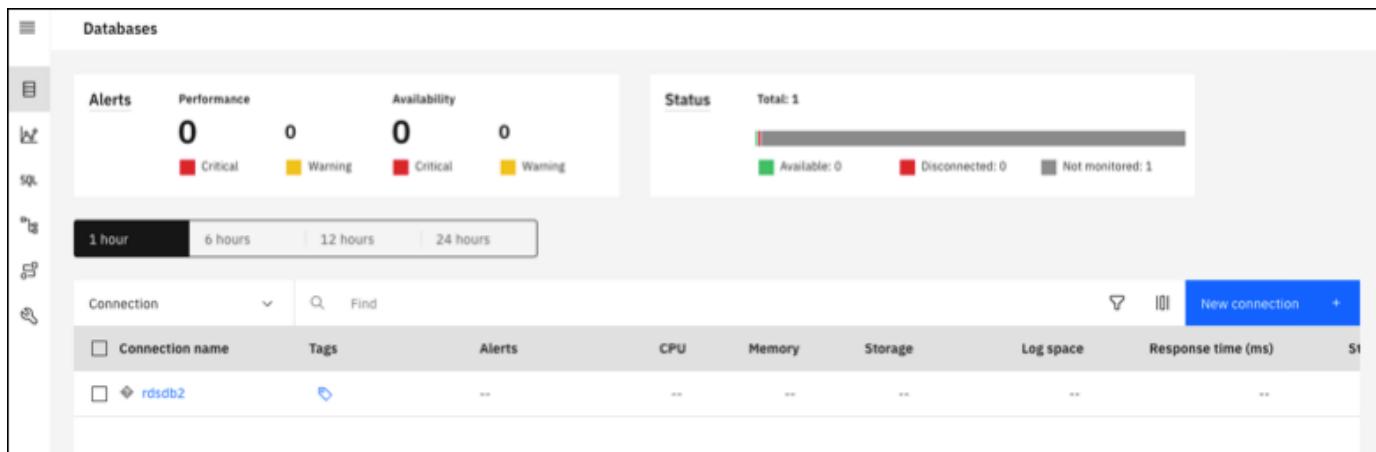
admin      \*\*\*\*\*

**Test connection**      **Skip**      **Save →**

After the connection is added, a window similar to the following appears. This window indicates that your database was successfully configured.



5. Choose **Go to Databases**. A Databases window similar to the following appears. This window is a dashboard that shows metrics, statuses, and connections.



You can now start using IBM Db2 Data Management Console.

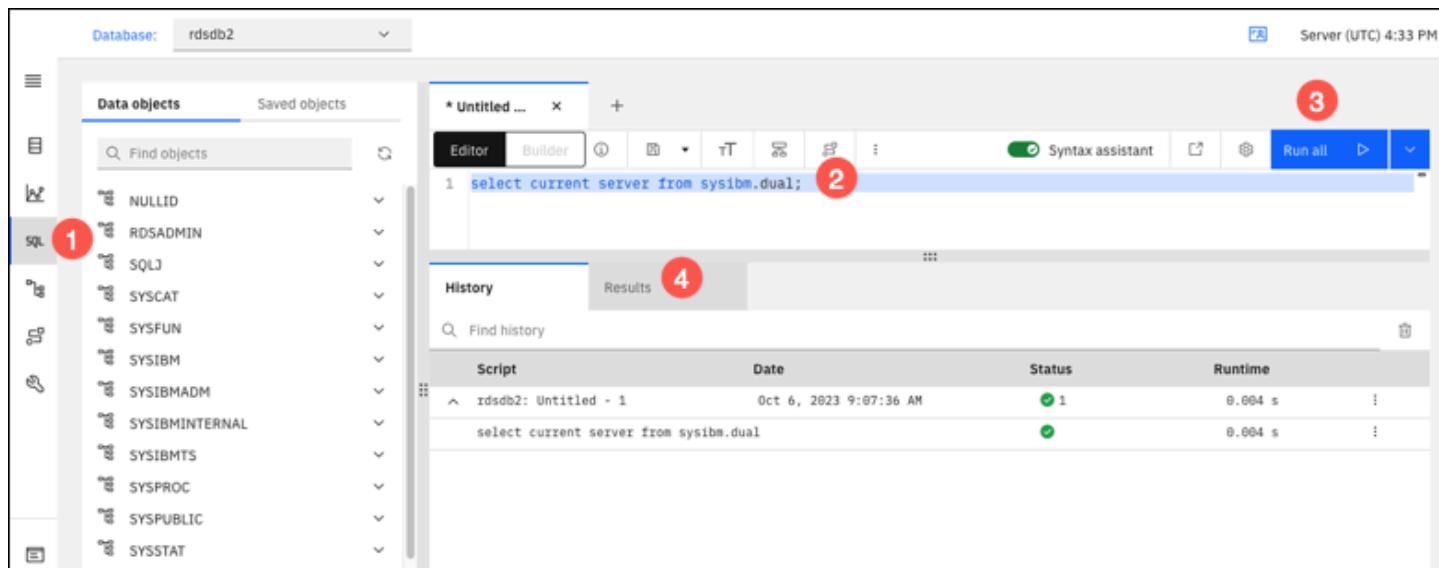
## Using IBM Db2 Data Management Console

You can use IBM Db2 Data Management Console to do the following types of tasks:

- Manage multiple RDS for Db2 DB instances.
- Run SQL commands.
- Explore, create, or change data and database objects.
- Create EXPLAIN PLAN statements in SQL.
- Tune queries.

### To run SQL commands and view the results

1. In the left navigation bar, choose **SQL**.
2. Enter a SQL command.
3. Choose **Run all**.
4. To view the results, choose the **Results** tab.



## Considerations for security groups with Amazon RDS for Db2

For you to connect to your Amazon RDS for Db2 DB instance, it must be associated with a security group that contains the necessary IP addresses and network configuration. Your RDS for Db2 DB instance might use the default security group. If you assigned a default nonconfigured security group when you created the RDS for Db2 DB instance, then the firewall prevents internet connections. For information about creating a new security group, see [Controlling access with security groups](#).

After you create the new security group, you modify your DB instance to associate it with the security group. For more information, see [Modifying an Amazon RDS DB instance](#).

You can enhance security by using SSL to encrypt connections to your DB instance. For more information, see [Using SSL/TLS with an Amazon RDS for Db2 DB instance](#).

# Securing Amazon RDS for Db2 DB instance connections

Amazon RDS for Db2 supports ways to improve security for your RDS for Db2 DB instance.

## Topics

- [Using SSL/TLS with an Amazon RDS for Db2 DB instance](#)
- [Using Kerberos authentication for Amazon RDS for Db2](#)

## Using SSL/TLS with an Amazon RDS for Db2 DB instance

SSL is an industry-standard protocol for securing network connections between client and server. After SSL version 3.0, the name was changed to TLS, but we still often refer to the protocol as SSL. Amazon RDS supports SSL encryption for Amazon RDS for Db2 DB instances. Using SSL/TLS, you can encrypt a connection between your application client and your RDS for Db2 DB instance. SSL/TLS support is available in all AWS Regions for RDS for Db2.

To enable SSL/TLS encryption for an RDS for Db2 DB instance, add the Db2 SSL option to the parameter group associated with the DB instance. Amazon RDS uses a second port, as required by Db2, for SSL/TLS connections. Doing this allows both clear text and SSL-encrypted communication to occur at the same time between a DB instance and a Db2 client. For example, you can use the port with clear text communication to communicate with other resources inside a VPC while using the port with SSL-encrypted communication to communicate with resources outside the VPC.

## Topics

- [Creating an SSL/TLS connection](#)
- [Connect to your Db2 database server](#)

## Creating an SSL/TLS connection

To create an SSL/TLS connection, choose a certificate authority (CA), download a certificate bundle for all AWS Regions, and add parameters to a custom parameter group.

### Step 1: Choose a CA and download a certificate

Choose a certificate authority (CA) and download a certificate bundle for all AWS Regions. For more information, see [Using SSL/TLS to encrypt a connection to a DB instance or cluster](#).

## Step 2: Update parameters in a custom parameter group

### **⚠ Important**

If you're using the bring your own license (BYOL) model for RDS for Db2, modify the custom parameter group that you created for your IBM Customer ID and your IBM Site ID. If you're using a different licensing model for RDS for Db2, then follow the procedure to add parameters to a custom parameter group. For more information, see [Amazon RDS for Db2 licensing options](#).

You can't modify default parameter groups for RDS for Db2 DB instances. Therefore, you must create a custom parameter group, modify it, and then attach it to your RDS for Db2 DB instances. For information about parameter groups, see [DB parameter groups for Amazon RDS DB instances](#).

Use the parameter settings in the following table.

Parameter	Value
DB2COMM	TCPIP,SSL or SSL
SSL_SVCENAME	<any port number except the number used for the non-SSL port>

### To update parameters in a custom parameter group

1. Create a custom parameter group by running the [create-db-parameter-group](#) command.

Include the following required options:

- `--db-parameter-group-name` – A name for the parameter group that you are creating.
- `--db-parameter-group-family` – The Db2 engine edition and major version. Valid values: db2-se-11-5, db2-ae-11.5.
- `--description` – A description for this parameter group.

For more information about creating a DB parameter group, see [Creating a DB parameter group in Amazon RDS](#).

2. Modify the parameters in the custom parameter group that you created by running the [modify-db-parameter-group](#) command.

Include the following required options:

- `--db-parameter-group-name` – The name of the parameter group that you created.
- `--parameters` – An array of parameter names, values, and the application methods for the parameter update.

For more information about modifying a parameter group, see [Modifying parameters in a DB parameter group in Amazon RDS](#).

3. Associate the parameter group with your RDS for Db2 DB instance. For more information, see [Associating a DB parameter group with a DB instance in Amazon RDS](#).

## Connect to your Db2 database server

Instructions for connecting to your Db2 database server are language-specific.

Java

### To connect to your Db2 database server using Java

1. Download the JDBC driver. For more information, see [DB2 JDBC Driver Versions and Downloads](#) in the IBM Support documentation.
2. Create a shell script file with the following content. This script adds all certificates from the bundle to a Java KeyStore.

 **Important**

Verify that keytool exists on the path in the script so that the script can locate it. If you use a Db2 client, you can locate the keytool under `~sqlib/java/jdk64/jre/bin`.

```
#!/bin/bash
PEM_FILE=$1
PASSWORD=$2
KEYSTORE=$3
```

```
# number of certs in the PEM file
CERTS=$(grep 'END CERTIFICATE' $PEM_FILE| wc -l)
for N in $($seq 0 $((CERTS - 1))); do
    ALIAS="${PEM_FILE%.*}-$N"
    cat $PEM_FILE |
        awk "n==$N { print }; /END CERTIFICATE/ { n++ }" |
        keytool -noprompt -import -trustcacerts -alias $ALIAS -keystore $KEYSTORE -
        storepass $PASSWORD
done
```

3. To run the shell script and import the PEM file with the certificate bundle into a Java KeyStore, run the following command. Replace *shell\_file\_name.sh* with the name of your shell script file and *password* with the password for your Java KeyStore.

```
./shell_file_name.sh global-bundle.pem password truststore.jks
```

4. To connect to your Db2 server, run the following command. Replace the following placeholders in the example with your RDS for Db2 DB instance information.
  - *ip\_address* – The IP address for your DB instance endpoint.
  - *port* – The port number for the SSL connection. This can be any port number except the number that's used for the non-SSL port.
  - *database\_name* – The name of your database in your DB instance.
  - *master\_username* – The master username for your DB instance.
  - *master\_password* – The master password for your DB instance.

```
export trustStorePassword=MyPassword
java -cp ~/dsdriver/jdbc_sqlj_driver/linuxamd64/db2jcc4.jar \
com.ibm.db2.jcc.DB2Jcc -url \
"jdbc:db2://ip_address:port/database_name:\
sslConnection=true;sslTrustStoreLocation=\
~/truststore.jks;\
sslTrustStorePassword=${trustStorePassword};\
sslVersion=TLSv1.2;\
encryptionAlgorithm=2;\
securityMechanism=7;" \
-user master_username -password master_password
```

## Node.js

### To connect to your Db2 database server using Node.js

1. Install the **node-ibm\_db** driver. For more information, see [Installing the node-ibm\\_db driver on Linux and UNIX systems](#) in the IBM Db2 documentation.
2. Create a JavaScript file based on the following content. Replace the following placeholders in the example with your RDS for Db2 DB instance information.
  - *ip\_address* – The IP address for your DB instance endpoint.
  - *master\_username* – The master username for your DB instance.
  - *master\_password* – The master password for your DB instance.
  - *database\_name* – The name of your database in your DB instance.
  - *port* – The port number for the SSL connection. This can be any port number except the number that's used for the non-SSL port.

```
var ibmdb = require("ibm_db");
const hostname = "ip_address";
const username = "master_username";
const password = "master_password";
const database = "database_name";
const port = "port";
const certPath = "/root/qa-bundle.pem";
ibmdb.open("DRIVER={DB2};DATABASE=" + database + ";HOSTNAME=" +
  hostname + ";UID=" + username + ";PWD=" + password + ";PORT=" + port +
  ";PROTOCOL=TCPIP;SECURITY=SSL;SSLServerCertificate=" + certPath + ";", function
  (err, conn){
  if (err) return console.log(err);
  conn.close(function () {
    console.log('done');
  });
});
```

3. To run the JavaScript file, run the following command.

```
node ssl-test.js
```

## Python

### To connect to your Db2 database server using Python

1. Create a Python file with the following content. Replace the following placeholders in the example with your RDS for Db2 DB instance information.

- *port* – The port number for the SSL connection. This can be any port number except the number that's used for the non-SSL port.
- *master\_username* – The master username for your DB instance.
- *master\_password* – The master password for your DB instance.
- *database\_name* – The name of your database in your DB instance.
- *ip\_address* – The IP address for your DB instance endpoint.

```
import click
import ibm_db
import sys

port = port;
master_user_id = "master_username" # Master id used to create your DB instance
master_password = "master_password" # Master password used to create your DB
instance
db_name = "database_name" # If not given "db-name"
vpc_customer_private_ip = "ip_address" # Hosts end points - Customer private IP
Addressicert_path = "/root/ssl/global-bundle.pem" # cert path

@click.command()
@click.option("--path", help="certificate path")
def db2_connect(path):

    try:
        conn =
        ibm_db.connect(f"DATABASE={db_name};HOSTNAME={vpc_customer_private_ip};PORT={port};"
PROTOCOL=TCPIP;UID={master_user_id};PWD={master_password};SECURITY=ssl;SSLServerCertifi
"", "")
        try:
            ibm_db.exec_immediate(conn, 'create table tablename (a int);')
            print("Query executed successfully")
        except Exception as e:
```

```
        print(e)
    finally:
        ibm_db.close(conn)
        sys.exit(1)
    except Exception as ex:
        print("Trying to connect...")

if __name__ == "__main__":
    db2_connect()
```

2. Create the following shell script, which runs the Python file you created. Replace *python\_file\_name.py* with the name of your Python script file.

```
#!/bin/bash
PEM_FILE=$1
# number of certs in the PEM file
CERTS=$(grep 'END CERTIFICATE' $PEM_FILE| wc -l)

for N in $(seq 0 $((CERTS - 1))); do
    ALIAS="${PEM_FILE%.*}-$N"
    cert=`cat $PEM_FILE | awk "n==$N { print }; /END CERTIFICATE/ { n++ }"`
    cat $PEM_FILE | awk "n==$N { print }; /END CERTIFICATE/ { n++ }" >
$ALIAS.pem
    python3 <python_file_name.py> --path $ALIAS.pem
    output=`echo $?`
    if [ $output == 1 ]; then
        break
    fi
done
```

3. To import the PEM file with the certificate bundle and run the shell script, run the following command. Replace *shell\_file\_name.sh* with the name of your shell script file.

```
./shell_file_name.sh global-bundle.pem
```

## Db2 CLP

### To connect to your Db2 database server using Db2 CLP

1. To connect to your Db2 instance using Db2 CLP, you need GSKit, which you can download at [IBM Fix Central](#). To use Db2 CLP, you also need the IBM Db2 client, which you can download from [Download initial Version 11.5 clients and drivers](#) in IBM Support.
2. Create a keystore.

```
gsk8capicmd_64 -keydb -create -db "directory/keystore-filename" -pw  
"changeThisPassword" -type pkcs12 -stash
```

3. Import the certificate bundles to the keystore.

```
gsk8capicmd_64 -cert -import -file global-bundle.pem -target directory/keystore-  
filename> -target_stashed
```

4. Update the Db2 instance configuration.

```
db2 update dbm cfg using SSL_CLNT_KEYDB keystore-filename  
SSL_CLNT_STASH keystore stash file immediate
```

5. Catalog the node and database.

```
db2 catalog tcpip node ssluse1 REMOTE endpoint SERVER ssl_svccname security ssl  
db2 catalog database testdb as ssltest at node ssluse1
```

6. Connect to the database.

```
db2 connect to ssltest user username using password
```

## Using Kerberos authentication for Amazon RDS for Db2

You can use Kerberos authentication to authenticate users when they connect to your Amazon RDS for Db2 DB instance. In this configuration, your DB instance works with AWS Directory Service for Microsoft Active Directory, also called AWS Managed Microsoft AD. You add the domain and other information of your AWS Managed Microsoft AD directory to your RDS for Db2 DB instance. When users authenticate with an RDS for Db2 DB instance joined to the trusting domain, authentication

requests are forwarded to the AWS Managed Microsoft AD directory that you created with AWS Directory Service.

Keeping all of your credentials in the same directory can save you time and effort. With this approach, you have a centralized place for storing and managing credentials for multiple DB instances. Using a directory can also improve your overall security profile.

In addition, you can access credentials from your own on-premises Microsoft Active Directory. To do so, create a trusting domain relationship so that the AWS Managed Microsoft AD directory trusts your on-premises Microsoft Active Directory. In this way, your users can access your RDS for Db2 DB instances with the same Windows single sign-on (SSO) experience as when they access workloads in your on-premises network.

For more information, see [What is AWS Directory Service?](#) in the *AWS Directory Service Administration Guide*.

For information about Kerberos authentication, see the following topics:

## Topics

- [Setting up Kerberos authentication for Amazon RDS for Db2 DB instances](#)
- [Connecting to Amazon RDS for Db2 with Kerberos authentication](#)

## Region and version availability

Feature availability and support varies across specific versions of each database engine, and across AWS Regions. For more information about version and Region availability of RDS for Db2 with Kerberos authentication, see [Supported Regions and DB engines for Kerberos authentication in Amazon RDS](#).

### Note

Kerberos authentication isn't supported for DB instance classes that are deprecated for RDS for Db2 DB instances. For more information, see [Amazon RDS for Db2 instance classes](#).

## Overview of Kerberos authentication for RDS for Db2 DB instances

To set up Kerberos authentication for an RDS for Db2 DB instance, complete the following general steps, which are described in more detail later:

1. Use AWS Managed Microsoft AD to create an AWS Managed Microsoft AD directory. You can use the AWS Management Console, the AWS Command Line Interface (AWS CLI), or AWS Directory Service to create the directory. For more information, see [Create your AWS Managed Microsoft AD directory](#) in the *AWS Directory Service Administration Guide*.
2. Create an AWS Identity and Access Management (IAM) role that uses the managed IAM policy `AmazonRDSDirectoryServiceAccess`. The IAM role allows Amazon RDS to make calls to your directory.

For the IAM role to allow access, the AWS Security Token Service (AWS STS) endpoint must be activated in the correct AWS Region for your AWS account. AWS STS endpoints are active by default in all AWS Regions, and you can use them without any further actions. For more information, see [Activating and deactivating AWS STS in an AWS Region](#) in the *IAM User Guide*.

3. Create or modify an RDS for Db2 DB instance by using the AWS Management Console, the AWS CLI, or the RDS API with one of the following methods:
  - Create a new RDS for Db2 DB instance using the console, the [create-db-instance](#) command, or the [CreateDBInstance](#) API operation. For instructions, see [Creating an Amazon RDS DB instance](#).
  - Modify an existing RDS for Db2 DB instance using the console, the [modify-db-instance](#) command, or the [ModifyDBInstance](#) API operation. For instructions, see [Modifying an Amazon RDS DB instance](#).
  - Restore an RDS for Db2 DB instance from a DB snapshot using the console, the [restore-db-instance-from-db-snapshot](#) command, or the [RestoreDBInstanceFromDBSnapshot](#) API operation. For instructions, see [Restoring to a DB instance](#).
  - Restore an RDS for Db2 DB instance to a point-in-time using the console, the [restore-db-instance-to-point-in-time](#) command, or the [RestoreDBInstanceToPointInTime](#) API operation. For instructions, see [Restoring a DB instance to a specified time for Amazon RDS](#).

You can locate the DB instance in the same Amazon Virtual Private Cloud (VPC) as the directory or in a different AWS account or VPC. When you create or modify the RDS for Db2 DB instance, do the following tasks:

- Provide the domain identifier (d-\* identifier) that was generated when you created your directory.
- Provide the name of the IAM role that you created.
- Verify that the DB instance security group can receive inbound traffic from the directory security group.

#### 4. Configure your Db2 client, and verify that traffic can flow between the client host and AWS Directory Service for the following ports:

- TCP/UDP port 53 – DNS
- TCP 88 – Kerberos authentication
- TCP 389 – LDAP
- TCP 464 – Kerberos authentication

## Managing a DB instance in a domain

You can use the AWS Management Console, the AWS CLI, or the RDS API to manage your DB instance and its relationship with your Microsoft Active Directory. For example, you can associate an Active Directory to enable Kerberos authentication. You can also remove the association for an Active Directory to disable Kerberos authentication. You can also move a DB instance to be externally authenticated by one Microsoft Active Directory to another.

For example, running the [modify-db-instance](#) CLI command, you can perform the following actions:

- Re-attempt enabling Kerberos authentication for a failed membership by specifying the current membership's directory ID for the --domain option.
- Disable Kerberos authentication on a DB instance by specifying none for the --domain option.
- Move a DB instance from one domain to another by specifying the domain identifier of the new domain for the --domain option.

## Understanding domain membership

After you create or modify your DB instance, it becomes a member of the domain. You can view the status of the domain membership in the console or by running the [describe-db-instances](#) command. The status of the DB instance can be one of the following:

- `kerberos-enabled` – The DB instance has Kerberos authentication enabled.
- `enabling-kerberos` – AWS is in the process of enabling Kerberos authentication on this DB instance.
- `pending-enable-kerberos` – Enabling Kerberos authentication is pending on this DB instance.

- pending-maintenance-enable-kerberos – AWS will attempt to enable Kerberos authentication on the DB instance during the next scheduled maintenance window.
- pending-disable-kerberos – Disabling Kerberos authentication is pending on this DB instance.
- pending-maintenance-disable-kerberos – AWS will attempt to disable Kerberos authentication on the DB instance during the next scheduled maintenance window.
- enable-kerberos-failed – A configuration problem prevented AWS from enabling Kerberos authentication on the DB instance. Correct the configuration problem before re-issuing the command to modify the DB instance.
- disabling-kerberos – AWS is in the process of disabling Kerberos authentication on this DB instance.

A request to enable Kerberos authentication can fail because of a network connectivity issue or an incorrect IAM role. In some cases, the attempt to enable Kerberos authentication might fail when you create or modify a DB instance. If this happens, verify that you are using the correct IAM role, and then modify the DB instance to join the domain.

## Setting up Kerberos authentication for Amazon RDS for Db2 DB instances

You use AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) to set up Kerberos authentication for an RDS for Db2 DB instance. To set up Kerberos authentication, follow these steps:

### Topics

- [Step 1: Create a directory using AWS Managed Microsoft AD](#)
- [Step 2: Create a trust](#)
- [Step 3: Create an IAM role for Amazon RDS to access AWS Directory Service](#)
- [Step 4: Create and configure users](#)
- [Step 5: Create an RDS for Db2 admin group in AWS Managed Microsoft AD](#)
- [Step 6: Modify DB parameter](#)
- [Step 7: Create or modify an RDS for Db2 DB instance](#)
- [Step 8: Retrieve the Active Directory group SID in PowerShell](#)
- [Step 9: Add SID to GroupName mappings to your RDS for Db2 DB instance](#)

- [Step 10: Configure a Db2 client](#)

## Step 1: Create a directory using AWS Managed Microsoft AD

AWS Directory Service creates a fully managed Active Directory in the AWS Cloud. When you create an AWS Managed Microsoft AD directory, AWS Directory Service creates two domain controllers and DNS servers for you. The directory servers are created in different subnets in a VPC. This redundancy helps ensure that your directory remains accessible even if a failure occurs.

When you create an AWS Managed Microsoft AD directory, AWS Directory Service performs the following tasks on your behalf:

- Sets up an Active Directory within your VPC.
- Creates a directory administrator account with the username Admin and the specified password. You use this account to manage your directory.

 **Important**

Make sure to save this password. AWS Directory Service doesn't store this password, and it can't be retrieved or reset.

- Creates a security group for the directory controllers. The security group must permit communication with the RDS for Db2 DB instance.

When you launch AWS Directory Service for Microsoft Active Directory, AWS creates an organizational unit (OU) that contains all of your directory's objects. This OU, which has the NetBIOS name that you entered when you created your directory, is located in the domain root. The domain root is owned and managed by AWS.

The Admin account that was created with your AWS Managed Microsoft AD directory has permissions for the most common administrative activities for your OU:

- Create, update, or delete users.
- Add resources to your domain such as file or print servers, and then assign permissions for those resources to users in your OU.
- Create additional OUs and containers.
- Delegate authority.

- Restore deleted objects from the Active Directory Recycle Bin.
- Run Active Directory and Domain Name Service (DNS) modules for Windows PowerShell on the AWS Directory Service.

The Admin account also has rights to perform the following domain-wide activities:

- Manage DNS configurations (add, remove, or update records, zones, and forwarders).
- View DNS event logs.
- View security event logs.

## To create a directory with AWS Managed Microsoft AD

1. Sign in to the AWS Management Console and open the AWS Directory Service console at <https://console.aws.amazon.com/directoryservicev2/>.
2. Choose **Set up directory**.
3. Choose **AWS Managed Microsoft AD**. AWS Managed Microsoft AD is the only option currently supported for use with Amazon RDS.
4. Choose **Next**.
5. On the **Enter directory information** page, provide the following information:
  - **Edition** – Choose the edition that meets your requirements.
  - **Directory DNS name** – The fully qualified name for the directory, such as corp.example.com.
  - **Directory NetBIOS name** – An optional short name for the directory, such as CORP.
  - **Directory description** – An optional description for the directory.
  - **Admin password** – The password for the directory administrator. The directory creation process creates an administrator account with the username Admin and this password.

The directory administrator password can't include the word "admin." The password is case-sensitive and must be 8–64 characters in length. It must also contain at least one character from three of the following four categories:

- Lowercase letters (a–z)
- Uppercase letters (A–Z)
- Numbers (0–9)

- Nonalphanumeric characters (~!@#\$%^&\*\_+=`|\{\}[];"'<,>,?./)
- Confirm password – Retype the administrator password.

 **Important**

Make sure that you save this password. AWS Directory Service doesn't store this password, and it can't be retrieved or reset.

6. Choose **Next**.
7. On the **Choose VPC and subnets** page, provide the following information:
  - **VPC** – Choose the VPC for the directory. You can create the RDS for Db2 DB instance in this same VPC or in a different VPC.
  - **Subnets** – Choose the subnets for the directory servers. The two subnets must be in different Availability Zones.
8. Choose **Next**.
9. Review the directory information. If changes are needed, choose **Previous** and make the changes. When the information is correct, choose **Create directory**.

## Review & create Info

### Review

Directory type	VPC
Microsoft AD	vpc-0d6c7cf411cf1e4e2 ( [REDACTED] )
Operating system version	Subnets
Windows Server 2019	RDS-Pvt-subnet-4   subnet-0d7ee6515db17b7a4 ( [REDACTED] us-west-2d)
Directory DNS name	RDS-Pvt-subnet-1   subnet-0ffff968223abe72a ( [REDACTED] us-west-2a)
corp.example.com	
Directory NetBIOS name	
CORP	
Directory description	
My directory	

### Pricing

Edition	Free trial eligible <a href="#">Learn more</a>
Standard	30-day limited trial
Domain controllers charge	
~USD [REDACTED] /mo	
* Includes two domain controllers, USD [REDACTED] /mo for each additional domain controller.	

[Cancel](#) [Previous](#) [Create directory](#)

It takes several minutes for the directory to be created. When it has been successfully created, the **Status** value changes to **Active**.

To see information about your directory, choose the directory ID under **Directory ID**. Make a note of the **Directory ID** value. You need this value when you create or modify your RDS for Db2 DB instance.

The screenshot shows the AWS Directory Service console. At the top, there's a breadcrumb navigation: Directory Service > Directories > d-92674e684f. Below it, the directory name "d-92674e684f" is displayed with an "Actions" dropdown menu. A large box labeled "Directory details" contains various configuration settings:

Directory type Microsoft AD	Directory DNS name corp.example.com	Directory ID <b>d-92674e684f</b>
Edition Standard	Directory NetBIOS name CORP	Description - <a href="#">Edit</a> My directory
Operating system version Windows Server 2019	Directory administration EC2 instance(s) -	

Below the "Directory details" box, there are tabs for Networking & security, Scale & share, Application management, and Maintenance. The Networking & security tab is currently selected.

## Step 2: Create a trust

If you plan to use AWS Managed Microsoft AD only, skip to [Step 3: Create an IAM role for Amazon RDS to access AWS Directory Service](#).

To enable Kerberos authentication using your self-managed Active Directory, you must create a forest trust relationship between your self-managed Active Directory and the . A forest trust is a trust relationship between a Microsoft AD and a self-managed Active Directory and the AWS Managed Microsoft AD created in the previous step. The trust can also be two-way, where both Active Directories trust each other. For more information about setting up forest trusts using AWS Directory Service, see [When to create a trust relationship](#) in the *AWS Directory Service Administration Guide*.

## Step 3: Create an IAM role for Amazon RDS to access AWS Directory Service

For Amazon RDS to call AWS Directory Service for you, your AWS account needs an IAM role that uses the managed IAM policy `AmazonRDSDirectoryServiceAccess`. This role allows Amazon RDS to make calls to AWS Directory Service.

When you create a DB instance using the AWS Management Console and your console user account has the `iam:CreateRole` permission, the console creates the needed IAM role automatically. In this case, the role name is `rds-directoryservice-kerberos-access-role`. Otherwise, you

must create the IAM role manually. When you create this IAM role, choose **Directory Service**, and attach the AWS managed policy **AmazonRDSDirectoryServiceAccess** to it.

For more information about creating IAM roles for a service, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.

 **Note**

The IAM role used for Windows Authentication for RDS for Microsoft SQL Server can't be used for RDS for Db2.

As an alternative to using the **AmazonRDSDirectoryServiceAccess** managed policy, you can create policies with the required permissions. In this case, the IAM role must have the following IAM trust policy:

JSON

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": [  
                    "directoryservice.rds.amazonaws.com",  
                    "rds.amazonaws.com"  
                ]  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

The role must also have the following IAM role policy:

## JSON

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "ds:DescribeDirectories",  
                "ds:AuthorizeApplication",  
                "ds:UnauthorizeApplication",  
                "ds:GetAuthorizedApplicationDetails"  
            ],  
            "Effect": "Allow",  
            "Resource": "*"  
        }  
    ]  
}
```

### Step 4: Create and configure users

You can create users by using the Active Directory Users and Computers tool. This is one of the Active Directory Domain Services and Active Directory Lightweight Directory Services tools. For more information, see [Add Users and Computers to the Active Directory domain](#) in the Microsoft documentation. In this case, users are individuals or other entities, such as their computers, that are part of the domain and whose identities are being maintained in the directory.

To create users in an AWS Directory Service directory, you must be connected to a Windows-based Amazon EC2 instance that's a member of the AWS Directory Service directory. At the same time, you must be signed in as a user that has privileges to create users. For more information, see [Create a user](#) in the *AWS Directory Service Administration Guide*.

### Step 5: Create an RDS for Db2 admin group in AWS Managed Microsoft AD

RDS for Db2 doesn't support Kerberos authentication for the master user or the two Amazon RDS reserved users rdsdb and rdsadmin. Instead, you need to create a new group called masterdba in AWS Managed Microsoft AD. For more information, see [Create a Group Account in Active Directory](#) in the Microsoft documentation. Any users that you add to this group will have master user privileges.

After you enable Kerberos authentication, the master user loses the masterdba role. As a result, the master user won't be able to access the instance local user group membership unless you disable Kerberos authentication. To continue to use the master user with password login, create a user on AWS Managed Microsoft AD with the same name as the master user. Then, add that user to the group masterdba.

## Step 6: Modify DB parameter

If you plan to use AWS Managed Microsoft AD only, skip to [Step 7: Create or modify an RDS for Db2 DB instance](#).

To enable Kerberos authentication using your self-managed Active Directory, you must set the parameter `rds.active_directory_configuration` to `AWS_MANAGED_AD_WITH_TRUST` in your parameter group. By default, this parameter is set to `AWS_MANAGED_AD` for using AWS Managed Microsoft AD only.

For information about modifying DB parameters, see [Modifying the parameters in parameter groups](#).

## Step 7: Create or modify an RDS for Db2 DB instance

Create or modify an RDS for Db2 DB instance for use with your directory. You can use the AWS Management Console, the AWS CLI, or the RDS API to associate a DB instance with a directory. You can do this in one of the following ways:

- Create a new RDS for Db2 DB instance using the console, the [create-db-instance](#) command, or the [CreateDBInstance](#) API operation. For instructions, see [Creating an Amazon RDS DB instance](#).
- Modify an existing RDS for Db2 DB instance using the console, the [modify-db-instance](#) command, or the [ModifyDBInstance](#) API operation. For instructions, see [Modifying an Amazon RDS DB instance](#).
- Restore an RDS for Db2 DB instance from a DB snapshot using the console, the [restore-db-instance-from-db-snapshot](#) command, or the [RestoreDBInstanceFromDBSnapshot](#) API operation. For instructions, see [Restoring to a DB instance](#).
- Restore an RDS for Db2 DB instance to a point-in-time using the console, the [restore-db-instance-to-point-in-time](#) command, or the [RestoreDBInstanceToPointInTime](#) API operation. For instructions, see [Restoring a DB instance to a specified time for Amazon RDS](#).

Kerberos authentication is only supported for RDS for Db2 DB instances in a VPC. The DB instance can be in the same VPC as the directory, or in a different VPC. The DB instance must use a

security group that allows ingress and egress within the directory's VPC so the DB instance can communicate with the directory.

## Console

When you use the console to create, modify, or restore a DB instance, choose **Password and Kerberos authentication** in the **Database authentication** section. Then choose **Browse Directory**. Select the directory or choose **Create directory** to use the Directory Service.

The screenshot shows the 'Database authentication' configuration page. Under 'Database authentication options', the 'Password and Kerberos authentication' option is selected. In the 'Directory' input field, the value 'corp.example.com (d-92674e684f)' is entered and highlighted with a red box. A 'Browse Directory' button is also visible.

## AWS CLI

When you use the AWS CLI, the following parameters are required for the DB instance to be able to use the directory that you created:

- For the `--domain` parameter, use the domain identifier ("d-\*" identifier) generated when you created the directory.
- For the `--domain-iam-role-name` parameter, use the role you created that uses the managed IAM policy `AmazonRDSDirectoryServiceAccess`.

The following example modifies a DB instance to use a directory. Replace the following placeholders in the example with your own values:

- `db_instance_name` – The name of your RDS for Db2 DB instance.

- *directory\_id* – The ID of the AWS Directory Service for Microsoft Active Directory directory that you created.
- *role\_name* – The name of the IAM role that you created.

```
aws rds modify-db-instance --db-instance-identifier db_instance_name --domain d-directory_id --domain-iam-role-name role_name
```

### Important

If you modify a DB instance to enable Kerberos authentication, reboot the DB instance after making the change.

## Step 8: Retrieve the Active Directory group SID in PowerShell

A security ID (SID) uniquely identifies a security principal or security group. When a security group or account is created in Active Directory, Active Directory assigns a SID to the group. To retrieve the AD security group SID from Active Directory, use the Get-ADGroup cmdlet in a Windows client machine that is part of the Active Directory domain. The Identity parameter specifies the Active Directory group name that you want the SID for.

The following example returns the SID of the Active Directory group adgroup1.

```
C:\Users\Admin> Get-ADGroup -Identity adgroup1 | select SID  
  
SID  
-----  
S-1-5-21-3168537779-1985441202-1799118680-1612
```

You must generate this mapping for all the groups that are relevant to the database.

## Step 9: Add SID to GroupName mappings to your RDS for Db2 DB instance

You need to add the SID to GroupName mappings created in the previous step to your RDS for Db2 DB instance. For each mapping, call the following stored procedure. Replace the *SID* and *group\_name* with your own information.

```
db2 connect to rdsadmin  
db2 "call rdsadmin.set_sid_group_mapping(?, 'SID', 'group_name')"
```

For more information, see [rdsadmin.set\\_sid\\_group\\_mapping](#).

For information about checking the task status, see [rdsadmin.get\\_task\\_status](#).

## Step 10: Configure a Db2 client

### To configure a Db2 client

1. Create an **/etc/krb5.conf** file (or equivalent) to point to the domain.

 **Note**

For Windows operating systems, create a **C:\windows\krb5.ini** file.

2. Verify that traffic can flow between the client host and AWS Directory Service. Use a network utility such as Netcat for the following tasks:
  - a. Verify traffic over DNS for port 53.
  - b. Verify traffic over TCP/UDP for port 53 and for Kerberos, which includes ports 88 and 464 for AWS Directory Service.
3. Verify that traffic can flow between the client host and the DB instance over the database port. You can use the command db2 to connect and access the database.

The following example is /etc/krb5.conf file content for AWS Managed Microsoft AD:

```
[libdefaults]
default_realm = EXAMPLE.COM
[realms]
EXAMPLE.COM = {
kdc = example.com
admin_server = example.com
}
[domain_realm]
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

## Connecting to Amazon RDS for Db2 with Kerberos authentication

Use the following procedure to connect to your Amazon RDS for Db2 DB instance with Kerberos authentication.

## To connect to RDS for Db2 with Kerberos authentication

1. At a command prompt, run the following command. In the following example, replace *username* with your Microsoft Active Directory username.

```
kinit username
```

2. If the RDS for Db2 DB instance is using a publicly accessible VPC, add the IP address for your DB instance endpoint to your /etc/hosts file on the Amazon EC2 client. The following example obtains the IP address and then adds it to the /etc/hosts file.

```
% dig +short Db2-endpoint.AWS-Region.rds.amazonaws.com  
;; Truncated, retrying in TCP mode.  
ec2-34-210-197-118.AWS-Region.compute.amazonaws.com.  
34.210.197.118  
  
% echo "34.210.197.118 Db2-endpoint.AWS-Region.rds.amazonaws.com" >> /etc/hosts
```

3. Use the following command to log in to an RDS for Db2 DB instance that is associated with Active Directory. Replace *database\_name* with the name of your RDS for Db2 database.

```
db2 connect to database_name
```

# Administering your Amazon RDS for Db2 DB instance

This topic covers the common management tasks that you perform with an Amazon RDS for Db2 DB instance. Some tasks are the same for all Amazon RDS DB instances. Other tasks are specific to RDS for Db2.

The following tasks are common to all RDS databases. There are also tasks specific to RDS for Db2, such as connecting to an RDS for Db2 database with a standard SQL client.

Task area	Relevant documentation
<b>Instance classes, storage, and PIOPS</b>  If you are creating a production instance, learn how instance classes, storage types, and Provisioned IOPS work in Amazon RDS.	<a href="#">DB instance classes</a>  <a href="#">Amazon RDS storage types</a>
<b>Multi-AZ deployments</b>  A production DB instance should use Multi-AZ deployments. Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances.	<a href="#">Configuring and managing a Multi-AZ deployment for Amazon RDS</a>
<b>Amazon VPC</b>  If your AWS account has a default virtual private cloud (VPC), then your DB instance is automatically created inside the default VPC. If your account doesn't have a default VPC, and you want the DB instance in a VPC, create the VPC and subnet groups before you create the DB instance.	<a href="#">Working with a DB instance in a VPC</a>
<b>Security groups</b>  By default, DB instances use a firewall that prevents access. Make sure that you create a security group with the correct IP addresses and network configuration to access the DB instance.	<a href="#">Controlling access with security groups</a>
<b>Parameter groups</b>	<a href="#">Adding IBM IDs to a parameter group for RDS for Db2 DB instances</a>

Task area	Relevant documentation
Because your RDS for Db2 DB instance requires that you add the <code>rds.ibm_customer_id</code> and <code>rds.ibm_site_id</code> parameters, create a parameter group before you create the DB instance. If your DB instance requires other specific database parameters, also add them to this parameter group before you create the DB instance.	<a href="#">Parameter groups for Amazon RDS</a>
<b>Option groups</b>  If your DB instance requires specific database options, create an option group before you create the DB instance.	<a href="#">Options for Amazon RDS for Db2 DB instances</a>
<b>Connecting to your DB instance</b>  After creating a security group and associating it to a DB instance, you can connect to the DB instance with any standard SQL client application such as IBM Db2 CLP.	<a href="#">Connecting to your Db2 DB instance</a>
<b>Backup and restore</b>  You can configure your DB instance to take automated storage backups, or take manual storage snapshots, and then restore instances from the backups or snapshots.	<a href="#">Backing up, restoring, and exporting data</a>
<b>Monitoring</b>  You can monitor an RDS for Db2 DB instance with IBM Db2 Data Management Console.  You can also monitor an RDS for Db2 DB instance by using CloudWatch Amazon RDS metrics, events, and enhanced monitoring.	<a href="#">Connecting to your Amazon RDS for Db2 DB instance with IBM Db2 Data Management Console</a>  <a href="#">Viewing metrics in the Amazon RDS console</a>  <a href="#">Viewing Amazon RDS events</a>  <a href="#">Monitoring OS metrics with Enhanced Monitoring</a>

Task area	Relevant documentation
<b>Log files</b>  You can access the log files for your RDS for Db2 DB instance.	<a href="#">Monitoring Amazon RDS log files</a>

## Topics

- [Performing common system tasks for Amazon RDS for Db2 DB instances](#)
- [Performing common database tasks for Amazon RDS for Db2 DB instances](#)

## Performing common system tasks for Amazon RDS for Db2 DB instances

You can perform certain common database administrator tasks related to the system on your Amazon RDS DB instances running Db2. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

For information about granting and revoking privileges and attaching to the remote database for RDS for Db2, see the following topics.

## Topics

- [Granting and revoking privileges for RDS for Db2](#)
- [Attaching to the remote RDS for Db2 DB instance](#)

## Creating a custom database endpoint

When you migrate to Amazon RDS for Db2, you can use custom database endpoint URLs to minimize changes to your application. For example, if you use db2.example.com as your current DNS record, you can add it to Amazon Route 53. In Route 53, you can use private hosted zones to map your current DNS database endpoint to an RDS for Db2 database endpoint. To add a custom A or CNAME record for an Amazon RDS database endpoint, see [Registering and managing domains using Amazon Route 53](#) in the *Amazon Route 53 Developer Guide*.

**Note**

If you can't transfer your domain to Route 53, you can use your DNS provider to create a CNAME record for the RDS for Db2 database endpoint URL. Consult your DNS provider documentation.

## Granting and revoking privileges for RDS for Db2

Users gain access to databases through membership in groups that are attached to databases.

Use the following procedures to grant and revoke privileges to control access to your database.

These procedures use IBM Db2 CLP running on a local machine to connect to an RDS for Db2 DB instance. Be sure to catalog the TCPIP node and the database to connect to your RDS for Db2 DB instance running on your local machine. For more information, see [Connecting to your Amazon RDS for Db2 DB instance with IBM Db2 CLP](#).

### Topics

- [Granting a user access to your database](#)
- [Changing a user's password](#)
- [Adding groups to a user](#)
- [Removing groups from a user](#)
- [Removing a user](#)
- [Listing users](#)
- [Creating a role](#)
- [Granting a role](#)
- [Revoking a role](#)
- [Dropping a role](#)
- [Granting database authorization](#)
- [Revoking database authorization](#)

## Granting a user access to your database

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 connect to rdsadmin user master_username using master_password
```

This command produces output similar to the following example:

### Database Connection Information

```
Database server      = DB2/LINUXX8664 11.5.8.0
SQL authorization ID = ADMIN
Local database alias = RDSADMIN
```

2. Add a user to your authorization list by calling `rdsadmin.add_user`. For more information, see [rdsadmin.add\\_user](#).

```
db2 "call rdsadmin.add_user(
    'username',
    'password',
    'group_name,group_name' )"
```

3. (Optional) Add additional groups to the user by calling `rdsadmin.add_groups`. For more information, see [rdsadmin.add\\_groups](#).

```
db2 "call rdsadmin.add_groups(
    'username',
    'group_name,group_name' )"
```

4. Confirm the authorities that are available to the user. In the following example, replace `rds_database_alias`, `master_user`, and `master_password` with your own information. Also, replace `username` with the user's username.

```
db2 terminate
db2 connect to rds_database_alias user master_user using master_password
db2 "SELECT SUBSTR(AUTHORITY,1,20) AUTHORITY, D_USER, D_GROUP, D_PUBLIC
```

```
FROM TABLE (SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID ('username', 'U') ) AS  
T  
ORDER BY AUTHORITY"
```

This command produces output similar to the following example:

AUTHORITY	D_USER	D_GROUP	D_PUBLIC
<hr/>			
ACCESSCTRL	N	N	N
BINDADD	N	N	N
CONNECT	N	N	N
CREATETAB	N	N	N
CREATE_EXTERNAL_ROUT	N	N	N
CREATE_NOT_FENCED_RO	N	N	N
CREATE_SECURE_OBJECT	N	N	N
DATAACCESS	N	N	N
DBADM	N	N	N
EXPLAIN	N	N	N
IMPLICIT_SCHEMA	N	N	N
LOAD	N	N	N
QUIESCE_CONNECT	N	N	N
SECADM	N	N	N
SQLADM	N	N	N
SYSADM	*	N	*
SYSCTRL	*	N	*
SYSMAINT	*	N	*
SYSMON	*	N	*
WLMADM	N	N	N

- Grant the RDS for Db2 roles ROLE\_NULLID\_PACKAGES, ROLE\_TABLESPACES, and ROLE PROCEDURES to the group that you added the user to. For more information, see [Amazon RDS for Db2 default roles](#).

 **Note**

We create RDS for Db2 DB instances in RESTRICTIVE mode. Therefore, the RDS for Db2 roles ROLE\_NULLID\_PACKAGES, ROLE\_TABLESPACES, and ROLE PROCEDURES grant execute privileges on NULLID packages for IBM Db2 CLP and Dynamic SQL. These roles also grant user privileges on tablespaces.

- a. Connect to your Db2 database. In the following example, replace *database\_name*, *master\_user*, and *master\_password* with your own information.

```
db2 connect to database_name user master_user using master_password
```

- b. Grant the role ROLE\_NULLED\_PACKAGES to a group. In the following example, replace *group\_name* with the name of the group that you want to add the role to.

```
db2 "grant role ROLE_NULLID_PACKAGES to group group_name"
```

- c. Grant the role ROLE\_TABLESPACES to the same group. In the following example, replace *group\_name* with the name of the group that you want to add the role to.

```
db2 "grant role ROLE_TABLESPACES to group group_name"
```

- d. Grant the role ROLE\_PROCEDURES to the same group. In the following example, replace *group\_name* with the name of the group that you want to add the role to.

```
db2 "grant role ROLE_PROCEDURES to group group_name"
```

6. Grant connect, bindadd, createtab, and IMPLICIT\_SCHEMA authorities to the group that you added the user to. In the following example, replace *group\_name* with the name of the second group that you added the user to.

```
db2 "grant usage on workload SYSDEFAULTUSERWORKLOAD to public"  
db2 "grant connect, bindadd, createtab, implicit_schema on database to  
group group_name"
```

7. Repeat steps 4 through 6 for each additional group that you added the user to.
8. Test the user's access by connecting as the user, creating a table, inserting values into the table, and returning data from the table. In the following example, replace *rds\_database\_alias*, *username*, and *password* with the name of the database and the user's username and password.

```
db2 connect to rds_database_alias user username using password  
db2 "create table t1(c1 int not null)"  
db2 "insert into t1 values (1),(2),(3),(4)"  
db2 "select * from t1"
```

## Changing a user's password

### To change a user's password

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 connect to rdsadmin user master_username using master_password
```

2. Change the password by calling `rdsadmin.change_password`. For more information, see [rdsadmin.change\\_password](#).

```
db2 "call rdsadmin.change_password(  
      'username',  
      'new_password')"
```

## Adding groups to a user

### To add groups to a user

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 connect to rdsadmin user master_username using master_password
```

2. Add groups to a user by calling `rdsadmin.add_groups`. For more information, see [rdsadmin.add\\_groups](#).

```
db2 "call rdsadmin.add_groups(  
      'username',  
      'group_name,group_name')"
```

## Removing groups from a user

1. Connect to the rdsadmin database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 connect to rdsadmin user master_username using master_password
```

2. Remove groups by calling rdsadmin.remove\_groups. For more information, see [rdsadmin.remove\\_groups](#).

```
db2 "call rdsadmin.remove_groups(  
      'username',  
      'group_name,group_name' )"
```

## Removing a user

### To remove a user from the authorization list

1. Connect to the rdsadmin database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 connect to rdsadmin user master_username using master_password
```

2. Remove a user from your authorization list by calling rdsadmin.remove\_user. For more information, see [rdsadmin.remove\\_user](#).

```
db2 "call rdsadmin.remove_user('username')"
```

## Listing users

To list users on an authorization list, call the rdsadmin.list\_users stored procedure. For more information, see [rdsadmin.list\\_users](#).

```
db2 "call rdsadmin.list_users()"
```

## Creating a role

You can use the [`rdsadmin.create\_role`](#) stored procedure to create a role.

### To create a role

1. Connect to the `rdsadmin` database. In the following example, replace `master_username` and `master_password` with your information.

```
db2 connect to rdsadmin user master_username using master_password
```

2. Set Db2 to output content.

```
db2 set serveroutput on
```

3. Create a role. For more information, see [the section called “rdsadmin.create\\_role”](#).

```
db2 "call rdsadmin.create_role(  
      'database_name',  
      'role_name' )"
```

4. Set Db2 to not output content.

```
db2 set serveroutput off
```

## Granting a role

You can use the [`rdsadmin.grant\_role`](#) stored procedure to assign a role to a role, user, or group.

### To assign a role

1. Connect to the `rdsadmin` database. In the following example, replace `master_username` and `master_password` with your information.

```
db2 connect to rdsadmin user master_username using master_password
```

2. Set Db2 to output content.

```
db2 set serveroutput on
```

3. Assign a role. For more information, see [the section called “rdsadmin.grant\\_role”](#).

```
db2 "call rdsadmin.grant_role(
    'database_name',
    'role_name',
    'grantee',
    'admin_option')"
```

4. Set Db2 to not output content.

```
db2 set serveroutput off
```

## Revoking a role

You can use the [rdsadmin.revoke\\_role](#) stored procedure to revoke a role from a role, user, or group.

### To revoke a role

1. Connect to the rdsadmin database. In the following example, replace *master\_username* and *master\_password* with your information.

```
db2 connect to rdsadmin user master_username using master_password
```

2. Revoke a role. For more information, see [the section called “rdsadmin.revoke\\_role”](#).

```
db2 "call rdsadmin.revoke_role(
    ?,
    'database_name',
    'role_name',
    'grantee')"
```

## Dropping a role

You can use the [rdsadmin.drop\\_role](#) stored procedure to drop a role.

### To drop a role

1. Connect to the rdsadmin database. In the following example, replace *master\_username* and *master\_password* with your information.

```
db2 connect to rdsadmin user master_username using master_password
```

2. Drop a role. For more information, see [the section called "rdsadmin.drop\\_role".](#)

```
db2 "call rdsadmin.drop_role(  
?,  
'database_name',  
'role_name')"
```

## Granting database authorization

The master user, who has DBADM authorization, can grant DBADM, ACCESSCTRL, or DATAACCESS authorization to a role, user, or group.

### To grant database authorization

1. Connect to the rdsadmin database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 connect to rdsadmin user master_username using master_password
```

2. Grant a user access by calling `rdsadmin.dbadm_grant`. For more information, see [rdsadmin.dbadm\\_grant](#).

```
db2 "call rdsadmin.dbadm_grant(  
?,  
'database_name',  
'authorization',  
'grantee')"
```

## Example use case

The following procedure walks you through creating a role, granting DBADM authorization to the role, assigning the role to a user, and granting the role to a group.

1. Connect to the rdsadmin database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 connect to rdsadmin user master_username using master_password
```

2. Create a role called PROD\_ROLE for a database called TESTDB. For more information, see [rdsadmin.create\\_role](#).

```
db2 "call rdsadmin.create_role(  
      'TESTDB',  
      'PROD_ROLE')"
```

3. Assign the role to a user called PROD\_USER. The PROD\_USER is given admin authorization to assign roles. For more information, see [rdsadmin.grant\\_role](#).

```
db2 "call rdsadmin.grant_role(  
      ?,  
      'TESTDB',  
      'PROD_ROLE',  
      'USER PROD_USER',  
      'Y')"
```

4. (Optional) Provide additional authorization or privileges. The following example grants DBADM authorization to a role named PROD\_ROLE for a database called FUNDPROD. For more information, see [rdsadmin.dbadm\\_grant](#).

```
db2 "call rdsadmin.dbadm_grant(  
      ?,  
      'FUNDPROD',  
      'DBADM',  
      'ROLE PROD_ROLE')"
```

5. Terminate your session.

```
db2 terminate
```

6. Connect to the TESTDB database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 connect to TESTDB user master_username using master_password
```

## 7. Add more authorizations to the role.

```
db2 "grant connect, implicit_schema on database to role PROD_ROLE"
```

## 8. Grant the role PROD\_ROLE to a group.

```
db2 "grant role PROD_ROLE to group PRODGRP"
```

Users who belong to the group PRODGRP can now perform actions such as connecting to the TESTDB database, creating tables, or creating schemas.

## Revoking database authorization

The master user, who has DBADM authorization, can revoke DBADM, ACCESSCTRL, or DATAACCESS authorization from a role, user, or group.

### To revoke database authorization

1. Connect to the rdsadmin database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 connect to rdsadmin user master_username using master_password
```

2. Revoke user access by calling `rdsadmin.dbadm_revoke`. For more information, see [rdsadmin.dbadm\\_revoke](#).

```
db2 "call rdsadmin.dbadm_revoke(  
    ?,  
    'database_name',  
    'authorization',  
    'grantee')"
```

## Attaching to the remote RDS for Db2 DB instance

Use the following steps to attach to your remote RDS for Db2 DB instance and run get\_snapshot operations.

### To attach to the remote RDS for Db2 DB instance

1. Run a client-side IBM Db2 CLP session. For information about cataloging your RDS for Db2 DB instance and database, see [Connecting to your Amazon RDS for Db2 DB instance with IBM Db2 CLP](#). Make a note of the master username and master password for your RDS for Db2 DB instance.
2. Attach to the RDS for Db2 DB instance. In the following example, replace *node\_name*, *master\_username*, and *master\_password* with the TCPIP node name that you catalogued and the master username and master password for your RDS for Db2 DB instance.

```
db2 attach to node_name user master_username using master_password
```

After attaching to the remote RDS for Db2 DB instance, you can run the following commands and other get\_snapshot commands. For more information, see [GET SNAPSHOT command](#) in the IBM Db2 documentation.

```
db2 list applications  
db2 get snapshot for all databases  
db2 get snapshot for database manager  
db2 get snapshot for all applications
```

## Performing common database tasks for Amazon RDS for Db2 DB instances

You can perform certain common DBA tasks related to databases on your Amazon RDS for Db2 DB instances. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances. Also, the master user can't run commands or utilities requiring SYSADM, SYSMAINT, or SYSCTRL authorities.

For information about common tasks for buffer pools, databases, and tablespaces, see the following topics.

## Topics

- [Common tasks for buffer pools](#)
- [Common tasks for databases](#)
- [Common tasks for tablespaces](#)

## Common tasks for buffer pools

You can create, alter, or drop buffer pools for an RDS for Db2 database. Creating, altering, or dropping buffer pools requires higher-level SYSADM or SYSCTRL authority, which isn't available to the master user. Instead, use Amazon RDS stored procedures.

You can also flush buffer pools.

## Topics

- [Creating a buffer pool](#)
- [Altering a buffer pool](#)
- [Dropping a buffer pool](#)
- [Flushing the buffer pools](#)

### Creating a buffer pool

To create a buffer pool for your RDS for Db2 database, call the `rdsadmin.create_bufferpool` stored procedure. For more information, see [CREATE BUFFERPOOL statement](#) in the IBM Db2 documentation.

#### To create a buffer pool

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Create a buffer pool by calling `rdsadmin.create_bufferpool`. For more information, see [rdsadmin.create\\_bufferpool](#).

```
db2 "call rdsadmin.create_bufferpool(
```

```
'database_name',
'buffer_pool_name',
buffer_pool_size,
'immediate',
'automatic',
page_size,
number_block_pages,
block_size)"
```

## Altering a buffer pool

To alter a buffer pool for your RDS for Db2 database, call the `rdsadmin.alter_bufferpool` stored procedure. For more information, see [ALTER BUFFERPOOL statement](#) in the IBM Db2 documentation.

### To alter a buffer pool

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Alter a buffer pool by calling `rdsadmin.alter_bufferpool`. For more information, see [rdsadmin.alter\\_bufferpool](#).

```
db2 "call rdsadmin.alter_bufferpool(
'database_name',
'buffer_pool_name',
buffer_pool_size,
'immediate',
'automatic',
change_number_blocks,
number_block_pages,
block_size)"
```

## Dropping a buffer pool

To drop a buffer pool for your RDS for Db2 database, call the `rdsadmin.drop_bufferpool` stored procedure. For more information, see [Dropping buffer pools](#) in the IBM Db2 documentation.

**⚠ Important**

Make sure that no tablespaces are assigned to the buffer pool that you want to drop.

## To drop a buffer pool

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Drop a buffer pool by calling `rdsadmin.drop_bufferpool`. For more information, see [rdsadmin.drop\\_bufferpool](#).

```
db2 "call rdsadmin.drop_bufferpool(  
      'database_name',  
      'buffer_pool_name' )"
```

## Flushing the buffer pools

You can flush the buffer pools to force a checkpoint so that RDS for Db2 writes pages from memory to storage.

**ⓘ Note**

You don't need to flush the buffer pools. Db2 writes logs synchronously before it commits transactions. The dirty pages might still be in a buffer pool, but Db2 writes them to storage asynchronously. Even if the system shuts down unexpectedly, when you restart the database, Db2 automatically performs crash recovery. During crash recovery, Db2 writes committed changes to the database or rolls back changes for uncommitted transactions.

## To flush the buffer pools

1. Connect to your Db2 database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *rds\_database\_alias*, *master\_username*, and *master\_password* with your own information.

```
db2 connect to rds_database_alias user master_username using master_password
```

2. Flush the buffer pools.

```
db2 flush bufferpools all
```

## Common tasks for databases

You can create, drop, or restore databases on your RDS for Db2 DB instance. Creating, dropping, or restoring databases requires higher-level SYSADM authority, which isn't available to the master user. Instead, use Amazon RDS stored procedures.

You can also perform common management tasks such as monitoring, maintenance, and the collection of information about your databases.

### Topics

- [Creating a database](#)
- [Configuring settings for a database](#)
- [Modifying database parameters](#)
- [Configuring log retention](#)
- [Listing log information](#)
- [Using fine-grained access control \(FGAC\)](#)
- [Deactivating a database](#)
- [Activating a database](#)
- [Reactivating a database](#)
- [Dropping a database](#)
- [Backing up a database](#)
- [Restoring a database](#)
- [Listing databases](#)

- [Collecting information about databases](#)
- [Forcing applications off of databases](#)
- [Generating performance reports](#)

## Creating a database

To create a database on your RDS for Db2 DB instance, call the `rdsadmin.create_database` stored procedure. For more information, see [CREATE DATABASE command](#) in the IBM Db2 documentation.

### Note

If you plan on modifying the `db2_compatibility_vector` parameter, modify the parameter before creating a database. For more information, see [Setting the db2\\_compatibility\\_vector parameter](#).

## To create a database

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Create a database by calling `rdsadmin.create_database`. For more information, see [rdsadmin.create\\_database](#).

```
db2 "call rdsadmin.create_database(  
      'database_name',  
      'database_page_size',  
      'database_code_set',  
      'database_territory',  
      'database_collation',  
      'database_autoconfigure_str',  
      'database_non-restrictive')"
```

3. (Optional) Create additional databases by calling `rdsadmin.create_database` for each database you want to create. Each Db2 DB instance can contain up to 50 databases. For more information, see [rdsadmin.create\\_database](#).

```
db2 "call rdsadmin.create_database('database_name')"
```

4. (Optional) Confirm that your database was created by using one of the following methods:
  - Call `rdsadmin.list_databases`. For more information, see [rdsadmin.list\\_databases](#).
  - Run the following SQL command:

```
db2 "select varchar(r.task_type,25) as task_type, r.database_name,  
      varchar(r.lifecycle,15) as lifecycle, r.created_at, r.database_name,  
      varchar(bson_to_json(task_input_params),256) as input_params,  
      varchar(r.task_output,1024) as task_output  
    from table(rdsadmin.get_task_status(null,null,'create_database'))  
      as r order by created_at desc"
```

## Configuring settings for a database

To configure the settings for a database on your RDS for Db2 DB instance, call the `rdsadmin.set_configuration` stored procedure. For example, you could configure the number of buffers or buffer manipulators to create during a restore operation.

### To configure settings for a database

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. (Optional) Check your current configuration settings by calling `rdsadmin.show_configuration`. For more information, see [the section called "rdsadmin.show\\_configuration"](#).

```
db2 "call rdsadmin.show_configuration('name')"
```

3. Configure the settings for the database by calling `rdsadmin.set_configuration`. For more information, see [the section called “rdsadmin.set\\_configuration”](#).

```
db2 "call rdsadmin.set_configuration(  
      'name',  
      'value')"
```

## Modifying database parameters

Amazon RDS for Db2 uses three types of parameters: database manager configuration parameters, registry variables, and database configuration parameters. You can update the first two types through parameter groups and the last type through the [rdsadmin.update\\_db\\_param](#) stored procedure.

### Note

You can only modify the values of existing parameters. You can't add new parameters that RDS for Db2 doesn't support.

For more information these parameters and how to modify their values, see [the section called “Db2 parameters”](#).

## Configuring log retention

To configure how long Amazon RDS retains log files for your RDS for Db2 database, call the `rdsadmin.set_archive_log_retention` stored procedure.

### To configure log retention for a database

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. (Optional) Check your current configuration for log retention by calling `rdsadmin.show_archive_log_retention`. For more information, see [the section called “rdsadmin.show\\_archive\\_log\\_retention”](#).

```
db2 "call rdsadmin.show_archive_log_retention(
?,  
'database_name')"
```

3. Configure log retention for the database by calling `rdsadmin.set_archive_log_retention`. For more information, see [the section called “rdsadmin.set\\_archive\\_log\\_retention”](#).

```
db2 "call rdsadmin.set_archive_log_retention(
?,  
'database_name',  
'archive_log_retention_hours')"
```

## **Listing log information**

To list details about archive log files, including such details as total storage size used, call the `rdsadmin.list_archive_log_information` stored procedure.

### **To list log information for a database**

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Return a list of log file information by calling `rdsadmin.list_archive_log_information`. For more information, see [the section called “rdsadmin.list\\_archive\\_log\\_information”](#).

```
db2 "call rdsadmin.list_archive_log_information(
?,  
'database_name')"
```

## **Using fine-grained access control (FGAC)**

To use fine-grained access control commands to control access to table data in a database on an RDS for Db2 DB instance, call the `rdsadmin.fgac_command` stored procedure. You might want to

use FGAC to limit access to data based on user roles or data attributes. For example, you could limit access to patient health care data based on the type of data or to certain medical care providers.

## To use fine-grained access control to control access to table data in a database

1. Connect to the rdsadmin database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Run various fine-grained access control commands by calling rdsadmin.fgac\_command. For more information, see [rdsadmin.fgac\\_command](#).

```
db2 "call rdsadmin.fgac_command(  
?,  
'database_name',  
'fgac_command')"
```

## Deactivating a database

To deactivate a database on your RDS for Db2 DB instance, call the rdsadmin.deactivate\_database stored procedure.

By default, Amazon RDS activates a database when you create a database on your RDS for Db2 DB instance. You can deactivate infrequently used databases to conserve memory resources.

## To deactivate a database

1. Connect to the rdsadmin database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Deactivate a database by calling rdsadmin.deactivate\_database. For more information, see [rdsadmin.deactivate\\_database](#).

```
db2 "call rdsadmin.deactivate_database(  
?,
```

```
'database_name' )"
```

## Activating a database

To activate a database on a standalone RDS for Db2 DB instance, call the `rdsadmin.activate_database` stored procedure.

By default, Amazon RDS activates a database when you create a database on your RDS for Db2 DB instance. You can deactivate infrequently used databases to conserve memory resources, and then later activate a deactivated database.

### To activate a database

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Activate a database by calling `rdsadmin.activate_database`. For more information, see [rdsadmin.activate\\_database](#).

```
db2 "call rdsadmin.activate_database(  
    ?,  
    'database_name' )"
```

## Reactivating a database

To reactivate a database on a replica source RDS for Db2 DB instance, call the `rdsadmin.reactivate_database` stored procedure. After you make changes to database configurations, you might need to reactivate a database on an RDS for Db2 DB instance. To determine if you need to reactivate a database, connect to the database and run `db get db cfg show detail`.

You can also call this stored procedure to reactivate a database on a standalone RDS for Db2 DB instance after you make changes to database configurations. Or, you could reactivate a database on a standalone RDS for Db2 DB instance by first calling the `rdsadmin.deactivate_database`

stored procedure and then the `rdsadmin.activate_database` stored procedure. For more information, see [Deactivating a database](#) and [Activating a database](#).

## To reactivate a database

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Reactivate a database by calling `rdsadmin.reactivate_database`. For more information, see [rdsadmin.reactivate\\_database](#).

```
db2 "call rdsadmin.reactivate_database(  
    ?,  
    'database_name' )"
```

## Dropping a database

To drop a database from your RDS for Db2 DB instance, call the `rdsadmin.drop_database` stored procedure. For more information, see [Dropping databases](#) in the IBM Db2 documentation.

### Note

You can drop a database by calling the stored procedure only if certain conditions are met. For more information, see [the section called “Usage notes” for rdsadmin.drop\\_database](#).

## To drop a database

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Drop a database by calling `rdsadmin.drop_database`. For more information, see [rdsadmin.drop\\_database](#).

```
db2 "call rdsadmin.drop_database('database_name')"
```

## Backing up a database

To back up a database in your RDS for Db2 DB instance to Amazon S3, call the `rdsadmin.backup_database` stored procedure. For more information, see [BACKUP DATABASE command](#) in the IBM Db2 documentation.

### Note

This stored procedure uses the integration with Amazon S3. Make sure that you have configured the integration before proceeding. For more information, see [Integrating an Amazon RDS for Db2 DB instance with Amazon S3](#).

## To back up a database

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Create a VPC gateway endpoint for S3. If you already have a VPC gateway endpoint for S3, skip to Step 4.

For an RDS for Db2 DB instance to be able to interact with Amazon S3, you must have a VPC and an Amazon S3 gateway endpoint for private subnets to use. For more information, see [Step 1: Create a VPC gateway endpoint for Amazon S3](#).

3. Confirm the VPC gateway endpoint for S3. For more information, see [Step 2: Confirm that your VPC gateway endpoint for Amazon S3 exists](#).
4. Back up a database by calling `rdsadmin.backup_database`. For more information, see [rdsadmin.backup\\_database](#).

```
db2 "call rdsadmin.backup_database(
```

```
?,
'database_name',
's3_bucket_name',
's3_prefix',
'backup_type',
'compression_option',
'util_impact_priority',
'num_files',
'parallelism',
'num_buffers')"
```

## 5. Terminate your connection.

```
terminate
```

6. (Optional) Confirm that the backup files were uploaded to your Amazon S3 bucket under `s3_prefix/dbi_resource_id/db_name`. If the files don't appear at `s3_prefix/dbi_resource_id/db_name`, check the status of backing up your database to identify any issues. For more information, see [rdsadmin.get\\_task\\_status](#). If you can't resolve any identified issues, contact [AWS Support](#).
7. (Optional) After the backup to Amazon S3 completes, you can restore the backup to an RDS for Db2 DB instance or to another location such as a local server. For information about restoring to an RDS for Db2 DB instance, see [Restoring a database](#).

## Restoring a database

To move a database from an Amazon S3 bucket to your RDS for Db2 DB instance, call the `rdsadmin.restore_database` stored procedure. For more information, see [RESTORE DATABASE command](#) in the IBM Db2 documentation.

### To restore a database

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. (Optional) Check your current configuration settings to optimize the restore operation by calling `rdsadmin.show_configuration`. For more information, see [the section called "rdsadmin.show\\_configuration"](#).

```
db2 "call rdsadmin.show_configuration('name')"
```

3. Configure the settings to optimize the restore operation by calling `rdsadmin.set_configuration`. Explicitly setting these values can improve the performance when restoring databases with large volumes of data. For more information, see [the section called "rdsadmin.set\\_configuration"](#).

```
db2 "call rdsadmin.set_configuration(
    'name',
    'value')"
```

4. Restore the database by calling `rdsadmin.restore_database`. For more information, see [the section called "rdsadmin.restore\\_database"](#).

```
db2 "call rdsadmin.restore_database(
    ?,
    'database_name',
    's3_bucket_name',
    's3_prefix',
    'restore_timestamp',
    'backup_type')"
```

5. (Optional) Confirm that your database was restored by calling `rdsadmin.list_databases` and checking that the restored database is listed. For more information, see [rdsadmin.list\\_databases](#).

6. Bring the database back online and apply additional transaction logs by calling `rdsadmin.rollforward_database`. For more information, see [the section called "rdsadmin.rollforward\\_database"](#).

```
db2 "call rdsadmin.rollforward_database(
    ?,
    'database_name',
    's3_bucket_name',
    's3_prefix',
    'rollforward_to_option',
    'complete_rollforward')"
```

7. (Optional) Check the status of the `rdsadmin.rollforward_database` stored procedure by calling the [the section called “rdsadmin.rollforward\\_status”](#) stored procedure.
8. If you set `complete_rollforward` to FALSE in the previous step, then you must finish bringing the database back online by calling `rdsadmin.complete_rollforward`. For more information, see [the section called “rdsadmin.complete\\_rollforward”](#).

```
db2 "call rdsadmin.complete_rollforward(  
?,  
'database_name' )"
```

9. (Optional) Check the status of the `rdsadmin.complete_rollforward` stored procedure by calling the [the section called “rdsadmin.rollforward\\_status”](#) stored procedure.

## **Listing databases**

You can list all of your databases running on Amazon RDS for Db2 by calling the `rdsadmin.list_databases` user-defined function.

### **To list your databases**

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. List your databases by calling `rdsadmin.list_databases`. For more information, see [rdsadmin.list\\_databases](#).

```
db2 "select * from table(rdsadmin.list_databases())"
```

## **Collecting information about databases**

To collect information about a database on a RDS for Db2 DB instance, call the `rdsadmin.db2pd_command` stored procedure. This information can help with monitoring your databases or troubleshooting issues.

## To collect information about a database

1. Connect to the rdsadmin database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Collect information about the database by calling rdsadmin.db2pd\_command. For more information, see [rdsadmin.db2pd\\_command](#).

```
db2 "call rdsadmin.db2pd_command('db2pd_cmd')"
```

## Forcing applications off of databases

To force applications off of a database on your RDS for Db2 DB instance, call the rdsadmin.force\_application stored procedure. Before you perform maintenance on your databases, force applications off of your databases.

### To force applications off of a database

1. Connect to the rdsadmin database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Force applications off of a database by calling rdsadmin.force\_application. For more information, see [rdsadmin.force\\_application](#).

```
db2 "call rdsadmin.force_application(  
    ?,  
    'applications')"
```

## Generating performance reports

You can generate performance reports with a procedure or a script. For information about using a procedure, see [DBSUMMARY procedure - Generate a summary report of system and application performance metrics](#) in the IBM Db2 documentation.

Db2 includes a db2mon.sh file in its ~sqllib/sample/perf directory. Running the script produces a low-cost, extensive SQL metrics report. To download the db2mon.sh file and related script files, see the [perf](#) directory in the IBM db2-samples GitHub repository.

### To generate performance reports with the script

1. Connect to your Db2 database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 connect to rdsadmin user master_username using master_password
```

2. Create a buffer pool named db2monbp with a page size of 4096 by calling `rdsadmin.create_bufferpool`. For more information, see [rdsadmin.create\\_bufferpool](#).

```
db2 "call rdsadmin.create_bufferpool('database_name', 'db2monbp', 4096)"
```

3. Create a temporary tablespace named db2montmptbsp that uses the db2monbp buffer pool by calling `rdsadmin.create_tablespace`. For more information, see [rdsadmin.create\\_tablespace](#).

```
db2 "call rdsadmin.create_tablespace('database_name', \  
'db2montmptbsp', 'db2monbp', 4096, 1000, 100, 'T')"
```

4. Open the db2mon.sh script, and modify the line about connecting to a database.

- a. Remove the following line.

```
db2 -v connect to $dbName
```

- b. Replace the line in the previous step with the following line. In the following example, replace *master\_username* and *master\_password* with the master username and master password for your RDS for Db2 DB instance.

```
db2 -v connect to $dbName user master_username using master_password
```

- c. Remove the following lines.

```
db2 -v create bufferpool db2monbp
```

```
db2 -v create user temporary tablespace db2montmptbsp bufferpool db2monbp
```

```
db2 -v drop tablespace db2montmptbsp
```

```
db2 -v drop bufferpool db2monbp
```

5. Run the db2mon.sh script to output a report at specified intervals. In the following example, replace *absolute\_path* with the complete path to the script file, *rds\_database\_alias* with the name of your database, and *seconds* with the number of seconds (0 to 3600) between report generation.

```
absolute_path/db2mon.sh rds_database_alias seconds | tee -a db2mon.out
```

## Examples

The following example shows that the script file is located in the perf directory under the home directory.

```
/home/db2inst1/sqllib/samples/perf/db2mon.sh rds_database_alias seconds | tee -a db2mon.out
```

6. Drop the buffer pool and the tablespace that were created for the db2mon.sh file. In the following example, replace *master\_username* and *master\_password* with the master username and master password for your RDS for Db2 DB instance. Replace *database\_name* with the name of your database. For more information, see [rdsadmin.drop\\_tablespace](#) and [rdsadmin.drop\\_bufferpool](#).

```
db2 connect to rdsadmin user master_username using master_password
```

```
db2 "call rdsadmin.drop_tablespace('database_name', 'db2montmptbsp')"
```

```
db2 "call rdsadmin.drop_bufferpool('database_name', 'db2monbp')"
```

## Managing storage

Db2 uses automatic storage to manage the physical storage for database objects such as tables, indexes, and temporary files. Instead of manually allocating storage space and keeping track of which storage paths are being used, automatic storage allows the Db2 system to create and manage storage paths as needed. This can simplify administration of Db2 databases and reduce the likelihood of errors due to human mistakes. For more information, see [Automatic storage](#) in the IBM Db2 documentation.

With RDS for Db2, you can dynamically increase the storage size with automatic expansion of the logical volumes and the file system. For more information, see [Working with storage for Amazon RDS DB instances](#).

## Common tasks for tablespaces

You can create, alter, rename, or drop tablespaces for an RDS for Db2 database. Creating, altering, renaming, or dropping tablespaces requires higher-level SYSADM authority, which isn't available to the master user. Instead, use Amazon RDS stored procedures.

### Topics

- [Creating a tablespace](#)
- [Altering a tablespace](#)
- [Renaming a tablespace](#)
- [Dropping a tablespace](#)
- [Checking the status of a tablespace](#)
- [Returning detailed information about tablespaces](#)
- [Listing the state and storage group for a tablespace](#)
- [Listing the tablespaces of a table](#)
- [Listing tablespace containers](#)

### Creating a tablespace

To create a tablespace for your RDS for Db2 database, call the `rdsadmin.create_tablespace` stored procedure. For more information, see [CREATE TABLESPACE statement](#) in the IBM Db2 documentation.

## Important

To create a tablespace, you must have a buffer pool of the same page size to associate with the tablespace. For more information, see [Common tasks for buffer pools](#).

## To create a tablespace

1. Connect to the rdsadmin database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Create a tablespace by calling `rdsadmin.create_tablespace`. For more information, see [rdsadmin.create\\_tablespace](#).

```
db2 "call rdsadmin.create_tablespace(  
      'database_name',  
      'tablespace_name',  
      'buffer_pool_name',  
      tablespace_initial_size,  
      tablespace_increase_size,  
      'tablespace_type'")"
```

## Altering a tablespace

To alter a tablespace for your RDS for Db2 database, call the `rdsadmin.alter_tablespace` stored procedure. You can use this stored procedure to change the buffer pool of a tablespace, lower the high water mark, or bring a tablespace online. For more information, see [ALTER TABLESPACE statement](#) in the IBM Db2 documentation.

## To alter a tablespace

1. Connect to the rdsadmin database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Alter a tablespace by calling `rdsadmin.alter_tablespace`. For more information, see [rdsadmin.alter\\_tablespace](#).

```
db2 "call rdsadmin.alter_tablespace(
    'database_name',
    'tablespace_name',
    'buffer_pool_name',
    buffer_pool_size,
    tablespace_increase_size,
    'max_size', 'reduce_max',
    'reduce_stop',
    'reduce_value',
    'lower_high_water',
    'lower_high_water_stop',
    'switch_online')"
```

## Renaming a tablespace

To change the name of a tablespace for your RDS for Db2 database, call the `rdsadmin.rename_tablespace` stored procedure. For more information, see [RENAME TABLESPACE statement](#) in the IBM Db2 documentation.

### To rename a tablespace

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Rename a tablespace by calling `rdsadmin.rename_tablespace`. For more information, including restrictions on what you can name a tablespace, see [rdsadmin.rename\\_tablespace](#).

```
db2 "call rdsadmin.rename_tablespace(
    'database_name',
    'source_tablespace_name',
    'target_tablespace_name')"
```

## Dropping a tablespace

To drop a tablespace for your RDS for Db2 database, call the `rdsadmin.drop_tablespace` stored procedure. Before you drop a tablespace, first drop any objects in the tablespace such as tables, indexes, or large objects (LOBs). For more information, see [Dropping table spaces](#) in the IBM Db2 documentation.

### To drop a tablespace

1. Connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. Drop a tablespace by calling `rdsadmin.drop_tablespace`. For more information, see [rdsadmin.drop\\_tablespace](#).

```
db2 "call rdsadmin.drop_tablespace(  
      'database_name',  
      'tablespace_name' )"
```

## Checking the status of a tablespace

You can check the status of a tablespace by using the `cast` function.

### To check the status of a tablespace

1. Connect to your Db2 database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `rds_database_alias`, `master_username`, and `master_password` with your own information.

```
db2 connect to rds_database_alias user master_username using master_password
```

2. Return a summary output.

For a summary output:

```
db2 "select cast(tbsp_id as smallint) as tbsp_id,  
      cast(tbsp_name as varchar(35)) as tbsp_name,
```

```
cast(tbsp_type as varchar(3)) as tbsp_type,  
cast(tbsp_state as varchar(10)) as state,  
cast(tbsp_content_type as varchar(8)) as contents from  
table(mon_get_tablespace(null,-1)) order by tbsp_id"
```

## Returning detailed information about tablespaces

You can return information about a tablespace for one member or all members by using the cast function.

### To return detailed information about tablespaces

1. Connect to your Db2 database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *rds\_database\_alias*, *master\_username*, and *master\_password* with your own information.

```
db2 connect to rds_database_alias user master_username using master_password
```

2. Return details about all tablespaces in the database for one member or for all members.

For one member:

```
db2 "select cast(member as smallint) as member,  
cast(tbsp_id as smallint) as tbsp_id,  
cast(tbsp_name as varchar(35)) as tbsp_name,  
cast(tbsp_type as varchar(3)) as tbsp_type,  
cast(tbsp_state as varchar(10)) as state,  
cast(tbsp_content_type as varchar(8)) as contents,  
cast(tbsp_total_pages as integer) as total_pages,  
cast(tbsp_used_pages as integer) as used_pages,  
cast(tbsp_free_pages as integer) as free_pages,  
cast(tbsp_page_top as integer) as page_hwm,  
cast(tbsp_page_size as integer) as page_sz,  
cast(tbsp_extent_size as smallint) as extent_sz,  
cast(tbsp_prefetch_size as smallint) as prefetch_sz,  
cast(tbsp_initial_size as integer) as initial_size,  
cast(tbsp_increase_size_percent as smallint) as increase_pct,  
cast(storage_group_name as varchar(12)) as stogroup from  
table(mon_get_tablespace(null,-1)) order by member, tbsp_id "
```

For all members:

```
db2 "select cast(member as smallint) as member
      cast(tbsp_id as smallint) as tbsp_id,
      cast(tbsp_name as varchar(35)) as tbsp_name,
      cast(tbsp_type as varchar(3)) as tbsp_type,
      cast(tbsp_state as varchar(10)) as state,
      cast(tbsp_content_type as varchar(8)) as contents,
      cast(tbsp_total_pages as integer) as total_pages,
      cast(tbsp_used_pages as integer) as used_pages,
      cast(tbsp_free_pages as integer) as free_pages,
      cast(tbsp_page_top as integer) as page_hwm,
      cast(tbsp_page_size as integer) as page_sz,
      cast(tbsp_extent_size as smallint) as extent_sz,
      cast(tbsp_prefetch_size as smallint) as prefetch_sz,
      cast(tbsp_initial_size as integer) as initial_size,
      cast(tbsp_increase_size_percent as smallint) as increase_pct,
      cast(storage_group_name as varchar(12)) as stogroup from
      table(mon_get_tablespace(null,-2)) order by member, tbsp_id "
```

## **Listing the state and storage group for a tablespace**

You can list the state and storage group for a tablespace by running a SQL statement.

To list the state and storage group for a tablespace, run the following SQL statement:

```
db2 "SELECT varchar(tbsp_name, 30) as tbsp_name,
          varchar(TBSP_STATE, 30) state,
          tbsp_type,
          varchar(storage_group_name,30) storage_group
     FROM TABLE(MON_GET_TABLESPACE('',-2)) AS t"
```

## **Listing the tablespaces of a table**

You can list the tablespaces for a table by running a SQL statement.

To list the tablespaces of a table, run the following SQL statement. In the following example, replace *SCHEMA\_NAME* and *TABLE\_NAME* with the names of your schema and table:

```
db2 "SELECT
      VARCHAR(SD.TBSpace,30) AS DATA_SPACE,
      VARCHAR(SL.TBSpace,30) AS LONG_SPACE,
      VARCHAR(SI.TBSpace,30) AS INDEX_SPACE
```

```
FROM
  SYSCAT.DATAPARTITIONS P
  JOIN SYSCAT.TABLESPACES SD ON SD.TBSPACEID = P.TBSPACEID
  LEFT JOIN SYSCAT.TABLESPACES SL ON SL.TBSPACEID = P.LONG_TBSPACEID
  LEFT JOIN SYSCAT.TABLESPACES SI ON SI.TBSPACEID = P.INDEX_TBSPACEID
WHERE
  TABSCHEMA = 'SCHEMA_NAME'
  AND TABNAME = 'TABLE_NAME'"
```

## List tablespace containers

You can list all tablespace containers or specific tablespace containers by using the `cast` command.

### To list the tablespace containers for a tablespace

1. Connect to your Db2 database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace `rds_database_alias`, `master_username`, and `master_password` with your own information:

```
db2 connect to rds_database_alias user master_username using master_password
```

2. Return a list of all tablespace containers in the database or specific tablespace containers.

For all tablespace containers:

```
db2 "select cast(member as smallint) as member,
      cast(tbsp_name as varchar(35)) as tbsp_name,
      cast(container_id as smallint) as id,
      cast(container_name as varchar(60)) as container_path, container_type as type from
      table(mon_get_container(null,-2)) order by member,tbsp_id,container_id"
```

For specific tablespace containers:

```
db2 "select cast(member as smallint) as member,
      cast(tbsp_name as varchar(35)) as tbsp_name,
      cast(container_id as smallint) as id,
      cast(container_name as varchar(60)) as container_path, container_type as type from
      table(mon_get_container('TBSP_1',-2)) order by member, tbsp_id,container_id"
```

# Integrating an Amazon RDS for Db2 DB instance with Amazon S3

You can transfer files between your Amazon RDS for Db2 DB instance and an Amazon Simple Storage Service (Amazon S3) bucket with Amazon RDS stored procedures. For more information, see [Amazon RDS for Db2 stored procedure reference](#).

## Note

Your DB instance and your Amazon S3 bucket must be in the same AWS Region.

For RDS for Db2 to integrate with Amazon S3, your DB instance must have access to an Amazon S3 bucket where your RDS for Db2 resides. If you don't currently have an S3 bucket, [create a bucket](#).

## Topics

- [Step 1: Create an IAM policy](#)
- [Step 2: Create an IAM role and attach your IAM policy](#)
- [Step 3: Add your IAM role to your RDS for Db2 DB instance](#)

## Step 1: Create an IAM policy

In this step, you create an AWS Identity and Access Management (IAM) policy with the permissions required to transfer files from your Amazon S3 bucket to your RDS DB instance. This step assumes that you have already created an S3 bucket. For more information, see [Creating a bucket](#) in the [Amazon S3 User Guide](#).

Before you create the policy, note the following pieces of information:

- The Amazon Resource Name (ARN) for your bucket
- The ARN for your AWS Key Management Service (AWS KMS) key, if your bucket uses SSE-KMS or SSE-S3 encryption.

The IAM policy that you create should contain the following information. Replace `{amzn-s3-demo-bucket}` with the name of your S3 bucket.

## JSON

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowS3BucketAccess",  
            "Effect": "Allow",  
            "Action": [  
                "kms:GenerateDataKey",  
                "kms:Decrypt",  
                "s3:PutObject",  
                "s3:GetObject",  
                "s3:AbortMultipartUpload",  
                "s3>ListBucket",  
                "s3:GetObjectVersion",  
                "s3>ListMultipartUploadParts",  
                "s3:GetBucketAcl",  
                "s3:GetBucketLocation"  
            ],  
            "Resource": [  
                "arn:aws:s3:::${amzn-s3-demo-bucket}/*",  
                "arn:aws:s3:::${amzn-s3-demo-bucket}"  
            ]  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3>ListAllMyBuckets"  
            ],  
            "Resource": [  
                "*"  
            ]  
        }  
    ]  
}
```

You can create an IAM policy by using the AWS Management Console or the AWS Command Line Interface (AWS CLI).

## Console

### To create an IAM policy to allow Amazon RDS to access your Amazon S3 bucket

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Policies**.
3. Choose **Create policy**, and then choose **JSON**.
4. Add actions by service. To transfer files from an Amazon S3 bucket to Amazon RDS, you must select bucket permissions and object permissions.
5. Expand **Resources**. You must specify your bucket and object resources.
6. Choose **Next**.
7. For **Policy name**, enter a name for this policy.
8. (Optional) For **Description**, enter a description for this policy.
9. Choose **Create policy**.

## AWS CLI

### To create an IAM policy to allow Amazon RDS to access your Amazon S3 bucket

1. Create a JSON file that contains the following JSON policy document. Replace *{amzn-s3-demo-bucket}* with the name of your S3 bucket.

JSON

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowS3BucketAccess",  
            "Effect": "Allow",  
            "Action": [  
                "kms:GenerateDataKey",  
                "kms:Decrypt",  
                "s3:PutObject",  
                "s3:GetObject",  
                "s3:AbortMultipartUpload",  
                "s3:ListBucket"  
            ]  
        }  
    ]  
}
```

```
        "s3>ListBucket",
        "s3GetObjectVersion",
        "s3ListMultipartUploadParts",
        "s3GetBucketAcl",
        "s3GetBucketLocation"
    ],
    "Resource": [
        "arn:aws:s3:::${amzn-s3-demo-bucket}/*",
        "arn:aws:s3:::${amzn-s3-demo-bucket}"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "s3>ListAllMyBuckets"
    ],
    "Resource": [
        "*"
    ]
}
]
```

- Run the [create-policy](#) command. In the following example, replace *iam\_policy\_name* and *iam\_policy\_file\_name* with a name for your IAM policy and the name of the JSON file you created in Step 1.

For Linux, macOS, or Unix:

```
aws iam create-policy \
--policy-name iam_policy_name \
--policy-document file://iam_policy_file_name.json
```

For Windows:

```
aws iam create-policy ^
--policy-name iam_policy_name ^
--policy-document file://iam_policy_file_name.json
```

- After the policy is created, note the ARN of the policy. You need the ARN for [Step 2: Create an IAM role and attach your IAM policy](#).

For information about creating an IAM policy, see [Creating IAM policies](#) in the IAM User Guide.

## Step 2: Create an IAM role and attach your IAM policy

This step assumes that you have created the IAM policy in [Step 1: Create an IAM policy](#). In this step, you create a IAM role for your RDS for Db2 DB instance and then attach your IAM policy to the role.

You can create an IAM role for your DB instance by using the AWS Management Console or the AWS CLI.

### Console

#### To create an IAM role and attach your IAM policy to it

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**.
3. Choose **Create role**.
4. For **Trusted entity type**, select **AWS service**.
5. For **Service or use case**, select **RDS**, and then select **RDS – Add Role to Database**.
6. Choose **Next**.
7. For **Permissions policies**, search for and select the name of the IAM policy that you created.
8. Choose **Next**.
9. For **Role name**, enter a role name.
10. (Optional) For **Description**, enter a description for the new role.
11. Choose **Create role**.

### AWS CLI

#### To create an IAM role and attach your IAM policy to it

1. Create a JSON file that contains the following JSON policy document:

JSON

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "rds.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

- Run the [create-role](#) command. In the following example, replace *iam\_role\_name* and *iam\_assume\_role\_policy\_file\_name* with a name for your IAM role and the name of the JSON file that you created in Step 1.

For Linux, macOS, or Unix:

```
aws iam create-role \  
    --role-name iam_role_name \  
    --assume-role-policy-document file://iam_assume_role_policy_file_name.json
```

For Windows:

```
aws iam create-role ^  
    --role-name iam_role_name ^  
    --assume-role-policy-document file://iam_assume_role_policy_file_name.json
```

- After the role is created, note the ARN of the role. You need the ARN for [Step 3: Add your IAM role to your RDS for Db2 DB instance](#).
- Run the [attach-role-policy](#) command. In the following example, replace *iam\_policy\_arn* with the ARN of the IAM policy that you created in [Step 1: Create an IAM policy](#). Replace *iam\_role\_name* with the name of the IAM role that you just created.

For Linux, macOS, or Unix:

```
aws iam attach-role-policy \  
    --policy-arn iam_policy_arn \  
    --role-name iam_role_name
```

For Windows:

```
aws iam attach-role-policy ^
--policy-arn iam_policy_arn ^
--role-name iam_role_name
```

For more information, see [Creating a role to delegate permissions to an IAM user](#) in the *IAM User Guide*.

## Step 3: Add your IAM role to your RDS for Db2 DB instance

In this step, you add your IAM role to your RDS for Db2 DB instance. Note the following requirements:

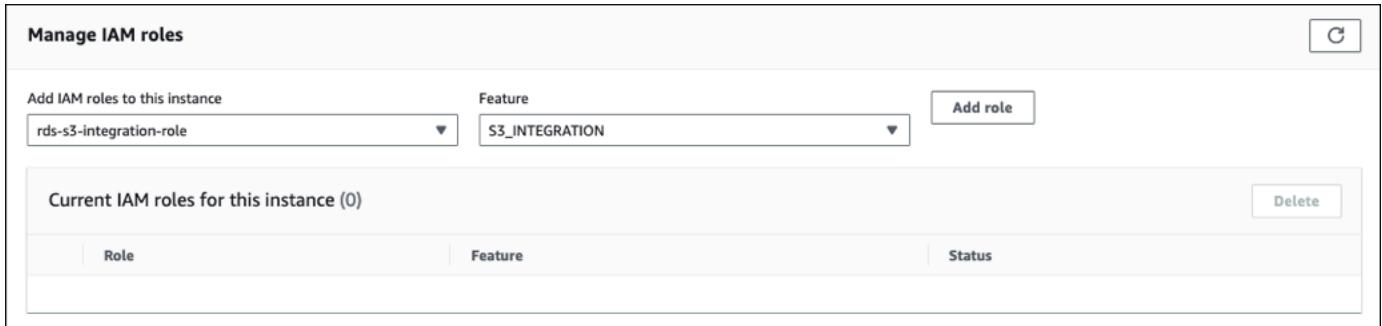
- You must have access to an IAM role with the required Amazon S3 permissions policy attached to it.
- You can only associate one IAM role with your RDS for Db2 DB instance at a time.
- Your RDS for Db2 DB instance must be in the **Available** state.

You can add an IAM role to your DB instance by using the AWS Management Console or the AWS CLI.

### Console

#### To add an IAM role to your RDS for Db2 DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose your RDS for Db2 DB instance name.
4. On the **Connectivity & security** tab, scroll down to the **Manage IAM roles** section at the bottom of the page.
5. For **Add IAM roles to this instance**, choose the role that you created in [Step 2: Create an IAM role and attach your IAM policy](#).
6. For **Feature**, choose **S3\_INTEGRATION**.
7. Choose **Add role**.



## AWS CLI

To add an IAM role to your RDS for Db2 DB instance, run the [add-role-to-db-instance](#) command. In the following example, replace *region*, *db\_instance\_name*, and *iam\_role\_arn* with the name of the AWS Region where your DB instance exists, the name of your DB instance, and the ARN of the IAM role that you created in [Step 2: Create an IAM role and attach your IAM policy](#).

For Linux, macOS, or Unix:

```
aws rds add-role-to-db-instance \
--region $region \
--db-instance-identifier $db_instance_name \
--feature-name S3_INTEGRATION \
--role-arn $iam_role_arn \
```

For Windows:

```
aws rds add-role-to-db-instance ^
--region $region \
--db-instance-identifier $db_instance_name ^
--feature-name S3_INTEGRATION ^
--role-arn $iam_role_arn ^
```

To confirm that the role was successfully added to your RDS for Db2 DB instance, run the [describe-db-instances](#) command. In the following example, replace *db\_instance\_name* with the name of your DB instance.

For Linux, macOS, or Unix:

```
aws rds describe-db-instances \
--filters "Name=db-instance-id,Values=$db_instance_name" \
```

```
--query 'DBInstances[].AssociatedRoles'
```

For Windows:

```
aws rds describe-db-instances ^
--filters "Name=db-instance-id,Values=db_instance_name" ^
--query 'DBInstances[].AssociatedRoles'
```

This command produces output similar to the following example:

```
[  
 [  
 {  
     "RoleArn": "arn:aws:iam::0123456789012:role/rds-db2-s3-role",  
     "FeatureName": "S3_INTEGRATION",  
     "Status": "ACTIVE"  
 }  
 ]  
 ]
```

# Migrating data to Amazon RDS for Db2

You can migrate self-managed Db2 databases to Amazon RDS for Db2 by using either AWS or native Db2 tools.

For information about migrating from your Db2 database to Amazon RDS for Db2 using AWS services, see [Using AWS services to migrate data from Db2 to Amazon RDS for Db2](#).

For information about migrating from your Db2 database to Amazon RDS for Db2 using native Db2 tools, see [Using native Db2 tools to migrate data from Db2 to Amazon RDS for Db2](#).

## Using AWS services to migrate data from Db2 to Amazon RDS for Db2

In Amazon RDS, there are several ways you can migrate data from a Db2 database to Amazon RDS for Db2. You can perform a one-time migration of your Db2 database from Linux, AIX, or Windows environments to Amazon RDS for Db2. To minimize downtime, you can perform a near-zero downtime migration. You can migrate your data by saving it to Amazon S3 and loading it one table at a time into your Db2 database. You can also perform a synchronous migration through replication or use AWS Database Migration Service.

For one-time migrations for Linux-based Db2 databases, Amazon RDS only supports offline and online backups. Amazon RDS doesn't support incremental and Delta backups. For near-zero downtime migrations for Linux-based Db2 databases, Amazon RDS requires online backups. We recommend that you use online backups for near-zero downtime migrations and offline backups for migrations that can handle downtime.

### Topics

- [Migrating from Linux to Linux for Amazon RDS for Db2](#)
- [Migrating from Linux to Linux with near-zero downtime for Amazon RDS for Db2](#)
- [Migrating synchronously from Linux to Linux for Amazon RDS for Db2](#)
- [Migrating from AIX or Windows to Linux for Amazon RDS for Db2](#)
- [Migrating Db2 data through Amazon S3 to Amazon RDS for Db2](#)
- [Migrating to Amazon RDS for Db2 with AWS Database Migration Service \(AWS DMS\)](#)

## Migrating from Linux to Linux for Amazon RDS for Db2

With this migration approach, you back up your self-managed Db2 database to an Amazon S3 bucket. Then, you use Amazon RDS stored procedures to restore your Db2 database to an Amazon

RDS for Db2 DB instance. For more information about using Amazon S3, see [Integrating an Amazon RDS for Db2 DB instance with Amazon S3](#).

Backup and restore for RDS for Db2 follows the IBM Db2 supported upgrade paths and restrictions. For more information, see [Supported upgrade paths for Db2 servers](#) and [Upgrade restrictions for Db2 servers](#) in the IBM Db2 documentation.

## Topics

- [Limitations and recommendations for using native restore](#)
- [Backing up your database to Amazon S3](#)
- [Creating a default automatic storage group](#)
- [Restoring your Db2 database](#)

### Limitations and recommendations for using native restore

The following limitations and recommendations apply to using native restore:

- Amazon RDS only supports migrating on-premises versions of Db2 that match supported RDS for Db2 versions. For more information about the supported versions, see [Supported Db2 minor versions on Amazon RDS](#).
- Amazon RDS only supports offline and online backups for native restore. Amazon RDS doesn't support incremental or Delta backups.
- You can't restore from an Amazon S3 bucket in an AWS Region that is different from the Region where your RDS for Db2 DB instance is located.
- Amazon S3 limits the size of files that are uploaded to an Amazon S3 bucket to 5 TB. If your database backup file exceeds 5 TB, then split the backup file into smaller files.
- Amazon RDS doesn't support non-fenced external routines, incremental restores, or Delta restores.
- You can't restore from an encrypted source database, but you can restore to an encrypted Amazon RDS DB instance.

The restoration process differs depending on your configuration.

If you set USE\_STREAMING\_RESTORE to TRUE, Amazon RDS directly streams your backup from your S3 bucket during restoration. Streaming significantly reduces storage requirements. You only

need to provision storage space equal to or greater than either the size of the backup or the size of the original database, whichever is larger.

If you set `USE_STREAMING_RESTORE` to `FALSE`, Amazon RDS first downloads the backup to your RDS for Db2 DB instance and then extracts the backup. Extraction requires additional storage space. You must provision storage space equal to or greater than the sum of the size of the backup plus the size of the original database.

The maximum size of the restored database equals the maximum supported database size minus any space required for temporary storage during the restoration process.

## Backing up your database to Amazon S3

To back up your database on Amazon S3, you need the following AWS components:

- *An Amazon S3 bucket to store your backup files:* Upload any backup files that you want to migrate to Amazon RDS. We recommend that you use offline backups for migrations that can handle downtime. If you already have an S3 bucket, you can use that bucket. If you don't have an S3 bucket, see [Creating a bucket in the Amazon S3 User Guide](#).

### Note

If your database is large and would take a long time to transfer to an S3 bucket, you can order an AWS Snow Family device and ask AWS to perform the backup. After you copy your files to the device and return it to the Snow Family team, the team transfers your backed-up images to your S3 bucket. For more information, see the [AWS Snow Family documentation](#).

- *An IAM role to access the S3 bucket:* If you already have an IAM role, you can use that role. If you don't have a role, see [Step 2: Create an IAM role and attach your IAM policy](#).
- *An IAM policy with trust relationships and permissions attached to your IAM role:* For more information, see [Step 1: Create an IAM policy](#).
- *The IAM role added to your RDS for Db2 DB instance:* For more information, see [Step 3: Add your IAM role to your RDS for Db2 DB instance](#).

## Creating a default automatic storage group

Your source database must have a default automatic storage group. If your database doesn't have a default automatic storage group, you must create one.

## To create a default automatic storage group

1. Connect to your source database. In the following example, replace *source\_database* with the name of your database.

```
db2 connect to source_database
```

2. Create an automatic storage group and set it as the default. In the following example, replace *storage\_path* with the absolute path to where the storage group is located.

```
db2 "create stogroup IBMSTOGROUP ON storage_path set as default"
```

3. Terminate backend processes.

```
db2 terminate
```

4. Deactivate the database and stop all database services. In the following example, replace *source\_database* with the name of the database that you created the storage group for.

```
db2 deactivate db source_database
```

5. Back up the database. In the following example, replace *source\_database* with the name of the database that you created the storage group for. Replace *file\_system\_path* with the absolute path to where you want to back up the database.

```
db2 backup database source_database to file_system_path
```

## Restoring your Db2 database

After you back up your database on Amazon S3 and create an automatic storage group, you are ready to restore your Db2 database to your RDS for Db2 DB instance.

### To restore your Db2 database from your Amazon S3 bucket to your RDS for Db2 DB instance

1. Connect to your RDS for Db2 DB instance. For more information, see [Connecting to your Db2 DB instance](#).
2. (Optional) To ensure that your database is configured with the optimal settings, check the values for the following parameters by calling [the section called "rdsadmin.show\\_configuration"](#):

- RESTORE\_DATABASE\_NUM\_BUFFERS
- RESTORE\_DATABASE\_PARALLELISM
- RESTORE\_DATABASE\_NUM\_MULTI\_PATHS
- USE\_STREAMING\_RESTORE

Use [the section called “rdsadmin.set\\_configuration”](#) to modify these values as needed.

Properly configuring these parameters can significantly improve performance when restoring databases with large volumes of data. For most migration scenarios, we recommend setting USE\_STREAMING\_RESTORE to TRUE because it reduces storage requirements and can improve restoration speed.

3. Restore your database by calling `rdsadmin.restore_database`. For more information, see [rdsadmin.restore\\_database](#).

## Migrating from Linux to Linux with near-zero downtime for Amazon RDS for Db2

With this migration approach, you migrate a Linux-based Db2 database from one self-managed Db2 database (source) to Amazon RDS for Db2. This approach results in minimal to no outage or downtime for the application or users. This approach backs up your database and restores it with log replay, which helps prevent disruptions to ongoing operations and provides high availability of your database.

To achieve near-zero downtime migration, RDS for Db2 implements restore with log replay. This approach takes a backup of your self-managed Linux-based Db2 database and restores it on the RDS for Db2 server. With Amazon RDS stored procedures, you then apply subsequent transaction logs to bring the database up to date.

### Topics

- [Limitations and recommendations for near-zero downtime migration](#)
- [Backing up your database to Amazon S3](#)
- [Creating a default automatic storage group](#)
- [Migrating your Db2 database](#)

### Limitations and recommendations for near-zero downtime migration

The following limitations and recommendations apply to using near-zero downtime migration:

- Amazon RDS requires an online backup for near-zero downtime migration. This is because Amazon RDS keeps your database in a rollforward pending state as you upload your archived transaction logs. For more information, see [the section called "Migrating your Db2 database".](#)
- You can't restore from an Amazon S3 bucket in an AWS Region that is different from the Region where your RDS for Db2 DB instance is located.
- Amazon S3 limits the size of files uploaded to an S3 bucket to 5 TB. If your database backup file exceeds 5 TB, then split the backup file into smaller files.
- Amazon RDS doesn't support non-fenced external routines, incremental restores, or Delta restores.
- You can't restore from an encrypted source database, but you can restore to an encrypted Amazon RDS DB instance.

The restoration process differs depending on your configuration.

If you set `USE_STREAMING_RESTORE` to `TRUE`, Amazon RDS directly streams your backup from your S3 bucket during restoration. Streaming significantly reduces storage requirements. You only need to provision storage space equal to or greater than either the size of the backup or the size of the original database, whichever is larger.

If you set `USE_STREAMING_RESTORE` to `FALSE`, Amazon RDS first downloads the backup to your RDS for Db2 DB instance and then extracts the backup. Extraction requires additional storage space. You must provision storage space equal to or greater than the sum of the size of the backup plus the size of the original database.

The maximum size of the restored database equals the maximum supported database size minus any space required for temporary storage during the restoration process.

## Backing up your database to Amazon S3

To back up your database on Amazon S3, you need the following AWS components:

- *An Amazon S3 bucket to store your backup files:* Upload any backup files that you want to migrate to Amazon RDS. Amazon RDS requires an online backup for near-zero downtime migration. If you already have an S3 bucket, you can use that bucket. If you don't have an S3 bucket, see [Creating a bucket](#) in the *Amazon S3 User Guide*.

**Note**

If your database is large and would take a long time to transfer to an S3 bucket, you can order an AWS Snow Family device and ask AWS to perform the backup. After you copy your files to the device and return it to the Snow Family team, the team transfers your backed-up images to your S3 bucket. For more information, see the [AWS Snow Family documentation](#).

- *An IAM role to access the S3 bucket:* If you already have an AWS Identity and Access Management (IAM) role, you can use that role. If you don't have a role, see [Step 2: Create an IAM role and attach your IAM policy](#).
- *An IAM policy with trust relationships and permissions attached to your IAM role:* For more information, see [Step 1: Create an IAM policy](#).
- *The IAM role added to your RDS for Db2 DB instance:* For more information, see [Step 3: Add your IAM role to your RDS for Db2 DB instance](#).

## Creating a default automatic storage group

Your source database must have a default automatic storage group. If your database doesn't have a default automatic storage group, you must create one.

### To create a default automatic storage group

1. Connect to your source database. In the following example, replace *source\_database* with the name of your database.

```
db2 connect to source_database
```

2. Create an automatic storage group and set it as the default. In the following example, replace *storage\_path* with the absolute path to where the storage group is located.

```
db2 "create stogroup IBMSTOGROUP ON storage_path set as default"
```

3. Terminate backend processes.

```
db2 terminate
```

## Migrating your Db2 database

After you set up for near-zero downtime migration, you are ready to migrate your Db2 database from your Amazon S3 bucket to your RDS for Db2 DB instance.

### To perform a near-zero downtime migration of backup files from your Amazon S3 bucket to your RDS for Db2 DB instance

1. Perform an online backup of your source database. For more information, see [BACKUP DATABASE command](#) in the IBM Db2 documentation.
2. Copy the backup of your database to an Amazon S3 bucket. For information about using Amazon S3, see the [Amazon Simple Storage Service User Guide](#).
3. Connect to the `rdsadmin` server with the `master_username` and `master_password` for your RDS for Db2 DB instance.

```
db2 connect to rdsadmin user master_username using master_password
```

4. (Optional) To ensure that your database is configured with the optimal settings, check the values for the following parameters by calling [the section called "rdsadmin.show\\_configuration"](#):
  - RESTORE\_DATABASE\_NUM\_BUFFERS
  - RESTORE\_DATABASE\_PARALLELISM
  - RESTORE\_DATABASE\_NUM\_MULTI\_PATHS
  - USE\_STREAMING\_RESTORE

Use [the section called "rdsadmin.set\\_configuration"](#) to modify these values as needed.

Properly configuring these parameters can significantly improve performance when restoring databases with large volumes of data. For most migration scenarios, we recommend setting `USE_STREAMING_RESTORE` to `TRUE` because it reduces storage requirements and can improve restoration speed.

5. Restore the backup on the RDS for Db2 server by calling `rdsadmin.restore_database`. Set `backup_type` to `ONLINE`. For more information, see [rdsadmin.restore\\_database](#).
6. Copy your archive logs from your source server to your S3 bucket. For more information, see [Archive logging](#) in the IBM Db2 documentation.

7. Apply archive logs as many times as needed by calling `rdsadmin.rollforward_database`. Set `complete_rollforward` to FALSE to keep the database in a ROLL-FORWARD PENDING state. For more information, see [rdsadmin.rollforward\\_database](#).
8. After you apply all of the archive logs, bring the database online by calling `rdsadmin.complete_rollforward`. For more information, see [rdsadmin.complete\\_rollforward](#).
9. Switch application connections to the RDS for Db2 server by either updating your application endpoints for the database or by updating the DNS endpoints to redirect traffic to the RDS for Db2 server. You can also use the Db2 automatic client reroute feature on your self-managed Db2 database with the RDS for Db2 database endpoint. For more information, see [Automatic client reroute description and setup](#) in the IBM Db2 documentation.
10. (Optional) Shut down your source database.

## Migrating synchronously from Linux to Linux for Amazon RDS for Db2

With this migration approach, you set up replication between your self-managed Db2 database and your Amazon RDS for Db2 DB instance. Changes made to the self-managed database replicates to the RDS for Db2 DB instance in near real-time. This approach can provide continuous availability and minimize downtime during the migration process.

## Migrating from AIX or Windows to Linux for Amazon RDS for Db2

With this migration approach, you use native Db2 tools to back up your self-managed Db2 database to an Amazon S3 bucket. Native Db2 tools include the `export` utility, the `db2move` system command, or the `db2look` system command. Your Db2 database can either be self-managed or in Amazon Elastic Compute Cloud (Amazon EC2). You can move data from your AIX or Windows system to your Amazon S3 bucket. Then, use a Db2 client to load data directly from the S3 bucket to your Amazon RDS for Db2 database. Downtime depends on the size of your database. For more information about using Amazon S3, see [Integrating an Amazon RDS for Db2 DB instance with Amazon S3](#).

### To migrate your Db2 database to RDS for Db2

1. Prepare to back up your database. Configure sufficient storage amount to hold the backup on your self-managed Db2 system.
2. Back up your database.

- a. Run the [db2look system command](#) to extract the data definition language (DDL) file for all objects.
  - b. Run either the [Db2 export utility](#), the [db2move system command](#), or a [CREATE EXTERNAL TABLE statement](#) to unload the Db2 table data to storage on your Db2 system.
3. Move your backup to an Amazon S3 bucket. For more information, see [Integrating an Amazon RDS for Db2 DB instance with Amazon S3](#).

 **Note**

If your database is large and would take a long time to transfer to an S3 bucket, you can order an AWS Snow Family device and ask AWS to perform the backup. After you copy your files to the device and return it to the Snow Family team, the team transfers your backed-up images to your S3 bucket. For more information, see the [AWS Snow Family documentation](#).

4. Use a Db2 client to load data directly from your S3 bucket to your RDS for Db2 database. For more information, see [Migrating with Amazon S3](#).

## Migrating Db2 data through Amazon S3 to Amazon RDS for Db2

With this migration approach, you first save data from a single table into a data file that you place in an Amazon S3 bucket. Then, you use the [LOAD command](#) to load the data from that data file into a table in your Amazon RDS for Db2 database. For more information about using Amazon S3, see [Integrating an Amazon RDS for Db2 DB instance with Amazon S3](#).

### Topics

- [Saving your data to Amazon S3](#)
- [Loading your data into RDS for Db2 tables](#)

### Saving your data to Amazon S3

To save data from a single table to Amazon S3, use a database utility to extract the data from your database management system (DBMS) into a CSV file. Then, upload the data file to Amazon S3.

For storing data files on Amazon S3, you need the following AWS components:

- *An Amazon S3 bucket to store your backup files:* If you already have an S3 bucket, you can use that bucket. If you don't have an S3 bucket, see [Creating a bucket](#) in the *Amazon S3 User Guide*.
- *An IAM role to access the S3 bucket:* If you already have an IAM role, you can use that role. If you don't have a role, see [Step 2: Create an IAM role and attach your IAM policy](#).
- *An IAM policy with trust relationships and permissions attached to your IAM role:* For more information, see [Step 1: Create an IAM policy](#).
- *The IAM role added to your RDS for Db2 DB instance:* For more information, see [Step 3: Add your IAM role to your RDS for Db2 DB instance](#).

## Loading your data into RDS for Db2 tables

After you save your data files to Amazon S3, you can load the data from these files into individual tables on your RDS for Db2 DB instance.

### To load your Db2 table data into your RDS for Db2 DB database table

1. Connect to the rdsadmin database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 connect to rdsadmin user master_username using master_password
```

2. Catalog a storage access alias that points to the Amazon S3 bucket where your saved files are stored. Take note of the name of this alias for use in the next step. You only need to perform this step once if you plan to load multiple tables from data files stored in the same Amazon S3 bucket.

The following example catalogs an alias named *my\_s3\_alias* that grants a user named *jorge\_souza* access to a bucket named *amzn-s3-demo-bucket*.

```
db2 "call rdsadmin.catalog_storage_access(?, 'my_s3_alias', 'amzn-s3-demo-bucket',  
'USER', 'jorge_souza')"
```

For more information about this stored procedure, See [the section called "rdsadmin.catalog\\_storage\\_access"](#).

3. Run the LOAD command using the storage access alias that points to your Amazon S3 bucket.

**Note**

If the LOAD command returns an error, then you might need to create a VPC gateway endpoint for Amazon S3 and add outbound rules to the security group. For more information, see [the section called “File I/O error”](#).

The following example loads data from a data file named *my\_s3\_datafile.csv* into a table named *my\_db2\_table*. The example assumes that the data file is in the Amazon S3 bucket that the alias named *my\_s3\_alias* points to.

```
db2 "load from db2remote://my_s3_alias//my_s3_datafile.csv of DEL insert  
into my_db2_table";
```

The following example loads LOBs from a data file named *my\_table1\_export.ixf* into a table named *my\_db2\_table*. The example assumes that the data file is in the Amazon S3 bucket that the alias named *my\_s3\_alias* points to.

```
db2 "call sysproc.admin_cmd('load from  
"db2remote://my_s3_alias//my_table1_export.ixf" of ixr  
lobs from "db2remote://my_s3_alias://" xml from "db2remote://my_s3_alias://"  
modified by lobsinfile implicitlyhiddeninclude identityoverride  
generatedoverride periodoverride transactionidoverride  
messages on server  
replace into "my_schema".my_db2_table"  
nonrecoverable  
indexing mode incremental allow no access')"
```

Repeat this step for each data file in the Amazon S3 bucket that you want to load into a table in your RDS for Db2 DB instance.

For more information about the LOAD command, see [LOAD command](#).

## Migrating to Amazon RDS for Db2 with AWS Database Migration Service (AWS DMS)

You can use AWS DMS for one-time migrations and then synchronize from Db2 on Linux, Unix (such as AIX), and Windows to Amazon RDS for Db2. For more information, see [What is AWS Database Migration Service?](#).

## Using native Db2 tools to migrate data from Db2 to Amazon RDS for Db2

You can use several native Db2 tools, utilities, and commands to move data directly from a Db2 database to an Amazon RDS for Db2 database. To use these native Db2 tools, you must be able to connect your client machine to an RDS for Db2 DB instance. For more information, see [Connecting a client machine to an Amazon RDS for Db2 DB instance](#).

### Note

Another way to move your data is to first save it to an Amazon S3 bucket, and then use the LOAD command to transfer that data into a table in your RDS for Db2 database. This method provides the best performance when migrating a large amount of data because of good network connectivity between RDS for Db2 and S3. For more information, see [the section called “Migrating with Amazon S3”](#).

Tool name	Use case	Limitations
<a href="#">db2look</a>	Copying metadata from a self-managed Db2 database to an RDS for Db2 database.	<ul style="list-style-type: none"><li>You must modify the syntax for creating buffer pools, creating tablespaces, and creating roles to match the syntax used by the <a href="#">RDS for Db2 stored procedures</a>.</li></ul>
<a href="#">IMPORT command</a>	Migrating small tables and tables with large objects (LOBs) from a client machine	<ul style="list-style-type: none"><li>Slower than the LOAD utility because of INSERT and DELETE logging operations.</li></ul>

Tool name	Use case	Limitations
	to the RDS for Db2 DB instance.	<ul style="list-style-type: none"> <li>Poor performance with limited network bandwidth.</li> </ul>
<a href="#">INGEST utility</a>	Continually streaming data from files and pipes <i>without</i> large objects (LOBs) on the client machine to the RDS for Db2 DB instance. Supports INSERT and MERGE operations.	<ul style="list-style-type: none"> <li>Can't stream data files that contain LOBs. Use the IMPORT command instead.</li> <li>Connectivity required between self-managed Db2 database and RDS for Db2 database.</li> </ul>
<a href="#">INSERT command</a>	Copying data in small tables from a self-managed Db2 database to an RDS for Db2 database.	<ul style="list-style-type: none"> <li>Connectivity required between self-managed Db2 database and RDS for Db2 database.</li> <li>Poor performance with limited network bandwidth.</li> </ul>
<a href="#">LOAD CLIENT command</a>	Migrating small tables <i>without</i> large objects (LOBs) from a client machine to the RDS for Db2 DB instance.	<ul style="list-style-type: none"> <li>Can't migrate data files that contain LOBs. Use the IMPORT command instead.</li> <li>Poor performance with limited network bandwidth.</li> </ul>

## Connecting a client machine to an Amazon RDS for Db2 DB instance

To use any of the native Db2 tools to move data from a Db2 database to an Amazon RDS for Db2 database, you must first connect your client machine to an RDS for Db2 DB instance.

The client machine can be any of the following:

- An Amazon Elastic Compute Cloud (Amazon EC2) instance on Linux, Windows, or macOS. This instance should be in the same virtual private cloud (VPC) as your RDS for Db2 DB instance, AWS Cloud9, or AWS CloudShell.

- A self-managed Db2 instance in an Amazon EC2 instance. The instances should be in the same VPC.
- A self-managed Db2 instance in an Amazon EC2 instance. The instances can be in different VPCs if you enabled VPC peering. For more information, see [Create a VPC peering connection](#) in the *Amazon Virtual Private Cloud VPC Peering Guide*.
- A local machine running Linux, Windows, or macOS in a self-managed environment. You must either have public connectivity to RDS for Db2 or enable VPN connectivity between self-managed Db2 instances and AWS.

To connect your client machine to your RDS for Db2 DB instance, log in to your client machine with IBM Db2 Data Management Console. For more information, see [Creating an Amazon RDS DB instance](#) and [IBM Db2 Data Management Console](#).

You can use AWS Database Migration Service (AWS DMS) to run queries against the database, run an SQL execution plan, and monitor the database. For more information, see [What is AWS Database Migration Service?](#) in the *AWS Database Migration Service User Guide*.

After you successfully connect your client machine to your RDS for Db2 DB instance, you are ready to use any native Db2 tool to copy data. For more information, see [Using native Db2 tools to migrate data from Db2 to Amazon RDS for Db2](#).

## Copying database metadata from Db2 to Amazon RDS for Db2 with db2look

db2look is a native Db2 tool that extracts data definition language (DDL) files, objects, authorizations, configurations, WLM, and database layouts. You can use db2look to copy database metadata from a self-managed Db2 database to an Amazon RDS for Db2 database. For more information, see [Mimicking databases using db2look](#) in the IBM Db2 documentation.

### To copy the database metadata

1. Run the db2look tool on your self-managed Db2 system to extract the DDL file. In the following example, replace *database\_name* with the name of your Db2 database.

```
db2look -d database_name -e -l -a -f -wlm -cor -createdb -printdbcfg -o db2look.sql
```

2. If your client machine has access to the source (self-managed Db2) database and the RDS for Db2 DB instance, you can create the db2look.sql file on the client machine by directly attaching to the remote instance. Then catalog the remote self-managed Db2 instance.

- a. Catalog the node. In the following example, replace *dns\_ip\_address* and *port* with the DNS name or the IP address and the port number of the self-managed Db2 database.

```
db2 catalog tcpip node srcnode REMOTE dns_ip_address server port
```

- b. Catalog the database. In the following example, replace *source\_database\_name* and *source\_database\_alias* with the name of the self-managed Db2 database and the alias that you want to use for this database.

```
db2 catalog database source_database_name as source_database_alias at node  
srcnode \  
authentication server_encrypt
```

- c. Attach to the source database. In the following example, replace *source\_database\_alias*, *user\_id*, and *user\_password* with the alias that you created in the previous step and the user ID and password for the self-managed Db2 database.

```
db2look -d source_database_alias -i user_id -w user_password -e -l -a -f -wlm \  
-cor -createdb -printdbcfg -o db2look.sql
```

3. If you can't access the remote self-managed Db2 database from the client machine, copy the db2look.sql file to the client machine. Then catalog the RDS for Db2 DB instance.

- a. Catalog the node. In the following example, replace *dns\_ip\_address* and *port* with the DNS name or the IP address and the port number of the RDS for Db2 DB instance.

```
db2 catalog tcpip node remnode REMOTE dns_ip_address server port
```

- b. Catalog the database. In the following example, replace *rds\_database\_name* and *rds\_database\_alias* with the name of the RDS for Db2 database and the alias that you want to use for this database.

```
db2 catalog database rds_database_name as rds_database_alias at node remnode \  
authentication server_encrypt
```

- c. Catalog the admin database that manages RDS for Db2. You can't use this database to store any data.

```
db2 catalog database rdsadmin as rdsadmin at node remnode authentication  
server_encrypt
```

4. Create buffer pools and tablespaces. The administrator doesn't have privileges to create buffer pools or tablespaces. However, you can use Amazon RDS stored procedures to create them.

- a. Find the names and definitions of the buffer pools and tablespaces in the db2look.sql file.
- b. Connect to Amazon RDS using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *master\_username* and *master\_password* with your own information.

```
db2 connect to rdsadmin user master_username using master_password
```

- c. Create a buffer pool by calling `rdsadmin.create_bufferpool`. For more information, see [rdsadmin.create\\_bufferpool](#).

```
db2 "call rdsadmin.create_bufferpool(  
      'database_name',  
      'buffer_pool_name',  
      buffer_pool_size,  
      'immediate',  
      'automatic',  
      page_size,  
      number_block_pages,  
      block_size)"
```

- d. Create a tablespace by calling `rdsadmin.create_tablespace`. For more information, see [rdsadmin.create\\_tablespace](#).

```
db2 "call rdsadmin.create_tablespace(  
      'database_name',  
      'tablespace_name',  
      'buffer_pool_name',  
      tablespace_initial_size,  
      tablespace_increase_size,  
      'tablespace_type')"
```

- e. Repeat steps c or d for each additional buffer pool or tablespace that you want to add.
- f. Terminate your connection.

```
db2 terminate
```

## 5. Create tables and objects.

- a. Connect to your RDS for Db2 database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *rds\_database\_name*, *master\_username*, and *master\_password* with your own information.

```
db2 connect to rds_database_name user master_username using master_password
```

- b. Run the db2look.sql file.

```
db2 -tvf db2look.sql
```

- c. Terminate your connection.

```
db2 terminate
```

## Importing data from a client machine to Amazon RDS for Db2 with the IMPORT command

You can use the IMPORT command from a client machine to import your data into the Amazon RDS for Db2 server.

### Important

The IMPORT command method is useful for migrating small tables and tables that include large objects (LOBs). The IMPORT command is slower than the LOAD utility because of the INSERT and DELETE logging operations. If your network bandwidth between the client machine and RDS for Db2 is limited, we recommend that you use a different migration approach. For more information, see [Using native Db2 tools to migrate data from Db2 to Amazon RDS for Db2](#).

## To import data into the RDS for Db2 server

1. Log in to your client machine with IBM Db2 Data Management Console. For more information, see [Connecting to your Amazon RDS for Db2 DB instance with IBM Db2 Data Management Console](#).
2. Catalog the RDS for Db2 database on the client machine.
  - a. Catalog the node. In the following example, replace *dns\_ip\_address* and *port* with the DNS name or the IP address and the port number of the self-managed Db2 database.

```
db2 catalog tcpip node srcnode REMOTE dns_ip_address server port
```

- b. Catalog the database. In the following example, replace *source\_database\_name* and *source\_database\_alias* with the name of the self-managed Db2 database and the alias that you want to use for this database.

```
db2 catalog database source_database_name as source_database_alias at node  
srcnode \  
authentication server_encrypt
```

3. Attach to the source database. In the following example, replace *source\_database\_alias*, *user\_id*, and *user\_password* with the alias you created in the previous step and the user ID and password for the self-managed Db2 database.

```
db2look -d source_database_alias -i user_id -w user_password -e -l -a -f -wlm \  
-cor -createdb -printdbcfg -o db2look.sql
```

4. Generate the data file by using the EXPORT command on your self-managed Db2 system. In the following example, replace *directory* with the directory on your client machine where your data file exists. Replace *file\_name* and *table\_name* with the name of the data file and the name of the table.

```
db2 "export to /directory/file_name.txt of del lobs to /directory/lobs/ \  
modified by coldel\| select * from table_name"
```

5. Connect to your RDS for Db2 database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *rds\_database\_alias*, *master\_username*, and *master\_password* with your own information.

```
db2 connect to rds_database_alias user master_username using master_password
```

6. Use the IMPORT command to import data from a file on the client machine into the remote RDS for Db2 database. For more information, see [IMPORT command](#) in the IBM Db2 documentation. In the following example, replace *directory* and *file\_name* with the directory on your client machine where your data file exists and the name of the data file. Replace *SCHEMA\_NAME* and *TABLE\_NAME* with the name of your schema and table.

```
db2 "IMPORT from /directory/file_name.tbl OF DEL LOBS FROM /directory/lobs/ \  
modified by coldel\| replace into SCHEMA_NAME.TABLE_NAME"
```

7. Terminate your connection.

```
db2 terminate
```

## Importing data from a client machine to Amazon RDS for Db2 with the LOAD command

You can use the LOAD CLIENT command to load data from a file on a client machine to the RDS for Db2 server. Because no SSH connectivity exists to the RDS for Db2 server, you can use the LOAD CLIENT command on either your self-managed Db2 server or your Db2 client machine.

### Important

The LOAD CLIENT command method is useful for migrating small tables. If your network bandwidth between the client and RDS for Db2 is limited, we recommend that you use a different migration approach. For more information, see the [Using native Db2 tools to migrate data from Db2 to Amazon RDS for Db2](#).

If your data file includes references to large object file names, then the LOAD command won't work because large objects (LOBs) need to reside on the Db2 server. If you try to load LOBs from the client machine to the RDS for Db2 server, you will receive an SQL3025N error. Use the [IMPORT command](#) instead.

## To load data to the RDS for Db2 server

1. Log in to your client machine with IBM Db2 Data Management Console. For more information, see [Connecting to your Amazon RDS for Db2 DB instance with IBM Db2 Data Management Console](#).
2. Catalog the RDS for Db2 database on the client machine.
  - a. Catalog the node. In the following example, replace *dns\_ip\_address* and *port* with the DNS name or the IP address and the port number of the self-managed Db2 database.

```
db2 catalog tcpip node srcnode REMOTE dns_ip_address server port
```

- b. Catalog the database. In the following example, replace *source\_database\_name* and *source\_database\_alias* with the name of the self-managed Db2 database and the alias that you want to use for this database.

```
db2 catalog database source_database_name as source_database_alias at node  
srcnode \  
authentication server_encrypt
```

3. Attach to the source database. In the following example, replace *source\_database\_alias*, *user\_id*, and *user\_password* with the alias you created in the previous step and the user ID and password for the self-managed Db2 database.

```
db2look -d source_database_alias -i user_id -w user_password -e -l -a -f -wlm \  
-cor -createdb -printdbcfg -o db2look.sql
```

4. Generate the data file by using the EXPORT command on your self-managed Db2 system. In the following example, replace *directory* with the directory on your client machine where your data file exists. Replace *file\_name* and *TABLE\_NAME* with the name of the data file and the name of the table.

```
db2 "export to /directory/file_name.txt of del modified by coldel\| \  
select * from TPCH.TABLE_NAME"
```

5. Connect to your RDS for Db2 database using the master username and master password for your RDS for Db2 DB instance. In the following example, replace *rds\_database\_alias*, *master\_username*, and *master\_password* with your own information.

```
db2 connect to rds_database_alias user master_username using master_password
```

6. Use the LOAD command to load data from a file on the client machine to the remote RDS for Db2 database. For more information, see [LOAD command](#) in the IBM Db2 documentation. In the following example, replace *directory* with the directory on your client machine where your data file exists. Replace *file\_name* and *TABLE\_NAME* with the name of the data file and the name of the table.

```
db2 "LOAD CLIENT from /directory/file_name.txt \  
modified by coldel\| replace into TPCH.TABLE_NAME \  
nonrecoverable without prompting"
```

7. Terminate your connection.

```
db2 terminate
```

## Importing data from Db2 to Amazon RDS for Db2 with the INSERT command

You can use the INSERT command from a self-managed Db2 server to insert your data into an Amazon RDS for Db2 database. With this migration approach, you use a nickname for the remote RDS for Db2 DB instance. Your self-managed Db2 database (source) must be able to connect to the RDS for Db2 database (target).

### **Important**

The INSERT command method is useful for migrating small tables. If your network bandwidth between your self-managed Db2 database and RDS for Db2 database is limited, we recommend that you use a different migration approach. For more information, see [Using native Db2 tools to migrate data from Db2 to Amazon RDS for Db2](#).

## To copy data from a self-managed Db2 database to an RDS for Db2 database

1. Catalog the RDS for Db2 DB instance on the self-managed Db2 instance.
  - a. Catalog the node. In the following example, replace *dns\_ip\_address* and *port* with the DNS name or the IP address and the port number of the self-managed Db2 database.

```
db2 catalog tcpip node remnode REMOTE dns_ip_address SERVER port
```

- b. Catalog the database. In the following example, replace *rds\_database\_name* with the name of the database on your RDS for Db2 DB instance.

```
db2 catalog database rds_database_name as remdb at node remnode \  
authentication server_encrypt
```

2. Enable federation on the self-managed Db2 instance. In the following example, replace *source\_database\_name* with the name of your database on the self-managed Db2 instance.

```
db2 update dbm cfg using FEDERATED YES source_database_name
```

3. Create tables on the RDS for Db2 DB instance.

- a. Catalog the node. In the following example, replace *dns\_ip\_address* and *port* with the DNS name or the IP address and the port number of the self-managed Db2 database.

```
db2 catalog tcpip node srcnode REMOTE dns_ip_address server port
```

- b. Catalog the database. In the following example, replace *source\_database\_name* and *source\_database\_alias* with the name of the self-managed Db2 database and the alias that you want to use for this database.

```
db2 catalog database source_database_name as source_database_alias at node  
srcnode \  
authentication server_encrypt
```

4. Attach to the source database. In the following example, replace *source\_database\_alias*, *user\_id*, and *user\_password* with the alias that you created in the previous step and the user ID and password for the self-managed Db2 database.

```
db2look -d source_database_alias -i user_id -w user_password -e -l -a -f -wlm \  
-cor -createdb -printdbcfg -o db2look.sql
```

5. Set up federation, and create a nickname for the RDS for Db2 database table on the self-managed Db2 instance.

- a. Connect to your local database. In the following example, replace *source\_database\_name* with the name of the database on your self-managed Db2 instance.

```
db2 connect to source_database_name
```

- b. Create a wrapper to access Db2 data sources.

```
db2 create wrapper drda
```

- c. Define a data source on a federated database. In the following example, replace *admin* and *admin\_password* with your credentials for your self-managed Db2 instance. Replace *rds\_database\_name* with the name of the database on your RDS for Db2 DB instance.

```
db2 "create server rdsdb2 type DB2/LUW version '11.5.9.0' \
      wrapper drda authorization "admin" password "admin_password" \
      options( dbname 'rds_database_name', node 'remnode')"
```

- d. Map the users on the two databases. In the following example, replace *master\_username* and *master\_password* with your credentials for your RDS for Db2 DB instance.

```
db2 "create user mapping for user server rdsdb2 \
      options (REMOTE_AUTHID 'master_username', REMOTE_PASSWORD
      'master_password')"
```

- e. Verify the connection to the RDS for Db2 server.

```
db2 set passthru rdsdb2
```

- f. Create a nickname for the table in the remote RDS for Db2 database. In the following example, replace *NICKNAME* and *TABLE\_NAME* with a nickname for the table and the name of the table.

```
db2 create nickname REMOTE.NICKNAME for RDSDB2.TABLE_NAME.NICKNAME
```

6. Insert data into the table in the remote RDS for Db2 database. Use the nickname in a select statement on the local table in the self-managed Db2 instance. In the following example, replace *NICKNAME* and *TABLE\_NAME* with a nickname for the table and the name of the table.

```
db2 "INSERT into REMOTE.NICKNAME select * from RDS2DB2.TABLE_NAME.NICKNAME"
```

## Importing data from Db2 to Amazon RDS for Db2 with the INGEST utility

You can use the INGEST utility to continually stream data from files and pipes on a client machine to a target Amazon RDS for Db2 DB instance. The INGEST utility supports INSERT and MERGE operations. For more information, see [Ingest utility](#) in the IBM Db2 documentation.

Because the INGEST utility supports nicknames, you can use the utility to transfer data from your self-managed Db2 database to an RDS for Db2 database. This approach works as long as network connectivity exists between the two databases.

 **Important**

The INGEST utility doesn't support large objects (LOBs). Use the [IMPORT command](#) instead.

To use the RESTARTABLE feature of the INGEST utility, run the following command on the RDS for Db2 database.

```
db2 "call sysproc.sysinstallobjects('INGEST','C',NULL,NULL)"
```

# Amazon RDS for Db2 federation

You can use your Amazon RDS for Db2 database as a federated database. After setting up federation for RDS for Db2, you will be able to access and query data across multiple databases from your RDS for Db2 database. Federation saves you from needing to migrate data to your RDS for Db2 database or consolidate data into a single database.

By using your RDS for Db2 database as a federated database, you can continue to access to all RDS for Db2 features and can take advantage of various AWS services, all while keeping your data in different databases. You can set up both homogeneous federation which connects different databases of the same type, or heterogeneous federation which connects different databases of different types.

You first connect your Db2 database in RDS for Db2 to remote databases. Then you can run queries against all your connected databases. For example, you can run a SQL JOIN statement that joins tables in your RDS for Db2 database with tables in a remote Db2 on z/OS database.

## Topics

- [Homogeneous federation](#)
- [Heterogeneous federation](#)

## Homogeneous federation

You can set up homogeneous federation between your RDS for Db2 database and the following Db2 family of products:

- Db2 for Linux, UNIX, Windows (LUW)
- Db2 iSeries
- Db2 for z/OS

RDS for Db2 homogeneous federation doesn't support the following actions:

- Running CATALOG commands to set up a node directory and a remote database on an RDS for Db2 host database
- Setting up Workload Balancing (WLB) when federating to Db2 on z/OS
- Configuring the IBM data server driver configuration file (`db2dsdriver.cfg`)

RDS for Db2 homogeneous federation has the following requirements:

- You must create the DRDA wrapper in UNFENCED mode. If you don't, then federation won't work in RDS for Db2.
- You must allow incoming and outgoing traffic from your RDS for Db2 host database to your remote host databases. For more information, see [Provide access to your DB instance in your VPC by creating a security group](#).

## Topics

- [Step 1: Create a DRDA wrapper and a federated server](#)
- [Step 2: Create a user mapping](#)
- [Step 3: Check the connection](#)

## Step 1: Create a DRDA wrapper and a federated server

For homogeneous federation, create a DRDA wrapper and a federated server. The connection to the remote host uses HOST, PORT, and DBNAME.

Choose one of the following methods based on the type of your remote Db2 database:

- **Db2 for Linux, UNIX, and Windows (LUX) database** – Run the following SQL commands. In the following example, replace *server\_name* with the name of the server that you will use for federation. Replace *db2\_version* with the version of your remote Db2 database. Replace *username* and *password* with your credentials for the remote Db2 database you want to connect to. Replace *db\_name*, *dns\_name*, and *port* with the appropriate values for the remote Db2 database you want to connect to.

```
create wrapper drda options(DB2_FENCED 'N');
create server server_name type DB2/LUW wrapper drda version 'db2_version'
    authorization "master_username" password "master_password" options (add DBNAME
    'db_name', add HOST 'dns_name', add PORT 'port');
```

## Example

```
create wrapper drda options(DB2_FENCED 'N');
```

```
create server SERVER1 type DB2/LUW wrapper drda version '11.5' authorization  
"sysuser" password "*****" options (add DBNAME 'TESTDB2', add HOST  
'ip-123-45-67-899.us-west-1.compute.internal', add PORT '25010');
```

- **Db2 iSeries** – Run the following SQL commands. In the following example, replace *wrapper\_name* and *library\_name* with a name for your DRDA wrapper and the [wrapper library file](#). Replace *server\_name* with the name of the server that you will use for federation. Replace *db2\_version* with the version of your remote Db2 database. Replace *username* and *password* with your credentials for the remote Db2 database you want to connect to. Replace *dns\_name*, *port*, and *db\_name* with the appropriate values for the remote Db2 database you want to connect to.

```
create wrapper wrapper_name library 'library_name' options(DB2_FENCED 'N');  
create server server_name type db2/mvs version db2_version wrapper wrapper_name  
authorization "sername" password "password" options (HOST 'dns_name', PORT 'port',  
DBNAME 'db_name');
```

## Example

```
create wrapper WRAPPER1 library 'libdb2drda.so' options(DB2_FENCED 'N');  
create server SERVER1 type db2/mvs version 11 wrapper WRAPPER1 authorization  
"sysuser" password "*****" options (HOST 'test1.123.com', PORT '446', DBNAME  
'STLEC1');
```

- **Db2 for z/OS** – Run the following SQL commands. In the following example, replace *wrapper\_name* and *library\_name* with a name for your DRDA wrapper and the [wrapper library file](#). Replace *server\_name* with the name of the server that you will use for federation. Replace *db2\_version* with the version of your remote Db2 database. Replace *username* and *password* with your credentials for the remote Db2 database you want to connect to. Replace *dns\_name*, *port*, and *db\_name* with the appropriate values for the remote Db2 database you want to connect to.

```
create wrapper wrapper_name library 'library_name' options(DB2_FENCED 'N');  
create server server_name type db2/mvs version db2_version wrapper wrapper_name  
authorization "username" password "password" options (HOST 'dns_name', PORT 'port',  
DBNAME 'db_name');
```

## Example

```
create wrapper WRAPPER1 library 'libdb2drda.so' OPTIONS(DB2_FENCED 'N');
create server SERVER1 type db2/mvs version 11 wrapper WRAPPER1 authorization
"sysuser" password "*****" options (HOST 'test1.123.com', PORT '446', DBNAME
'STLEC1');
```

## Step 2: Create a user mapping

Create a user mapping to associate your federated server with your data source server by running the following SQL command. In the following example, replace *server\_name* with the name of the remote server than you want to perform operations on. This is the server that you created in [step 1](#). Replace *username* and *password* with your credentials for this remote server.

```
create user mapping for user server server_name options (REMOTE_AUTHID 'username',
REMOTE_PASSWORD 'password');
```

For more information, see [User mappings](#) in the IBM Db2 documentation.

## Step 3: Check the connection

Confirm that setting up your federation was successful by checking the connection. Open a session to send native SQL commands to your remote data source using the SET PASSTHRU command, and then create a table on the remote data server.

1. Open and close a session to submit SQL to a data source. In the following example, replace *server\_name* with the name of the server that you created for federation in step 1.

```
set passthru server_name;
```

2. Create a new table. In the following example, replace *column\_name*, *data\_type*, and *value* with the appropriate items for your table.

```
create table table_name
( column_name data_type(value), column_name data_type(value);
```

For more information, see [CREATE TABLE statement](#) in the IBM Db2 documentation.

3. Create an index, insert values for rows into the table, and reset the connection. Resetting the connection drops the connection but retains the back-end processes. In the following

example, replace *index\_name*, *table\_name*, *column\_name*, and *columnx\_value* with your information.

```
create index index_name on table_name(column_name);
insert into table_name values(column1_value,column2_value,column3_value);
insert into table_name values(column1_value,column2_value,column3_value);
set passthru reset;

connect reset;
```

4. Connect to your remote Db2 database, create a nickname for your remote server, and perform operations. When you are done accessing data in the remote Db2 database, reset and then terminate the connection. In the following example, replace *database\_name* with the name of your remote Db2 database. Replace *nickname* with a name. Replace *server\_name* and *table\_name* with the name of the remote server and table on that server that you want to perform operations on. Replace *username* with the information for your remote server. Replace *sql\_command* with the operation to perform on the remote server.

```
connect to database_name;
create nickname nickname for server_name."username".table_name";
select sql_command from nickname;
connect reset;
terminate;
```

## Example

The following example creates a pass-through session to allow operations on the federated server testdb10.

Next, it creates the table t1 with three columns with different data types.

Then, the example creates the index i1\_t1 on three columns in table t1. Afterwards, it inserts two rows with values for these three columns, and then disconnects.

Last, the example connects to the remote Db2 database testdb2 and creates a nickname for the table t1 in the federated server testdb10. It creates the nickname with the username TESTUSER for that data source. An SQL command outputs all data from the table t1. The example disconnects and ends the session.

```
set passthru testdb10;
```

```
create table t1 ( c1 decimal(13,0), c2 char(200), c3 int);

create index i1_t1 on t1(c3);
insert into t1 values(1,'Test',1);
insert into t1 values(2,'Test 2',2);
connect reset;

connect to testdb2;
create nickname remote_t1 for testdb10."TESTUSER"."T1";
select * from remote_t1;
connect reset;
terminate;
```

## Heterogeneous federation

You can set up heterogeneous federation between your RDS for Db2 database and other data sources such as Oracle and Microsoft SQL Server. For a complete list of data sources that Db2 LUW supports, see [Data Source Support Matrix of Federation Bundled in Db2 LUW V11.5](#) on the IBM Support site.

RDS for Db2 heterogeneous federation doesn't support the following items:

- Native wrappers for the other data sources
- JDBC wrappers for the other data sources
- Federation to Sybase, Informix, and Teradata data sources because these data sources require client software installation on RDS for Db2

RDS for Db2 heterogeneous federation has the following requirements:

- RDS for Db2 only supports the ODBC wrapper method.
- If you create an explicit definition of a wrapper, then you must set the option DB2\_FENCED to 'N'. For a list of valid wrapper options for ODBC, see [ODBC options](#) in the IBM Db2 documentation.
- You must allow incoming and outgoing traffic from your RDS for Db2 host database to your remote host database. For more information, see [Provide access to your DB instance in your VPC by creating a security group](#).

For information about federation to Oracle, see [How to query Oracle by using Db2 Federation and the ODBC driver?](#) on the IBM Support site.

For more information about data sources that support federation, see [Data Source Support Matrix of Federation Bundled in Db2 LUW V11.5](#) on the IBM Support site.

## Topics

- [Step 1: Create an ODBC wrapper](#)
- [Step 2: Create a federated server](#)
- [Step 3: Create a user mapping](#)
- [Step 4: Check the connection](#)

### Step 1: Create an ODBC wrapper

Create a wrapper by running the following command:

```
db2 "create wrapper odbc options( module '/home/rdsdb/sqllib/federation/odbc/lib/libodbc.so')"
```

### Step 2: Create a federated server

Create a federated server by running the following command. In the following example, replace *server\_name* with the name of the server that you will use for federation. Replace *wrapper\_type* with the appropriate wrapper. Replace *db\_version* with the version of your remote database. Replace *dns\_name*, *port*, and *service\_name* with the appropriate values for the remote database that you want to connect to.

```
db2 "create server server_name type wrapper_type version db_version options (HOST 'dns_name', PORT 'port', SERVICE_NAME 'service_name')"
```

For information about wrapper types, see [Data Source Support Matrix of Federation Bundled in Db2 LUW V11.5](#) on the IBM Support site.

## Example

The following example creates a federated server for a remote Oracle database.

```
db2 "create server server1 type oracle_odbc version 12.1 options (HOST  
'test1.amazon.com', PORT '1521', SERVICE_NAME 'pdbc01.amazon.com')"
```

## Step 3: Create a user mapping

Create a user mapping to associate your federated server with your data source server by running the following SQL command. In the following example, replace *server\_name* with the name of the remote server than you want to perform operations on. This is the server that you created in [step 2](#). Replace *username* and *password* with your credentials for this remote server.

```
create user mapping for user server server_name options (REMOTE_AUTHID 'username',  
REMOTE_PASSWORD 'password');
```

For more information, see [User mappings](#) in the IBM Db2 documentation.

## Step 4: Check the connection

Confirm that setting up your federation was successful by checking the connection. Open a session to send native SQL commands to your remote data source using the SET PASSTHRU command, and then create a table on the remote data server.

1. Open and close a session to submit SQL to a data source. In the following example, replace *server\_name* with the name of the server that you created for federation in [step 2](#).

```
set passthru server_name;
```

2. Create a new table. In the following example, replace *column\_name*, *data\_type*, and *value* with the appropriate items for your table.

```
create table table_name  
( column_name data_type(value), column_name data_type(value));
```

For more information, see [CREATE TABLE statement](#) in the IBM Db2 documentation.

3. Create an index, insert values for rows into the table, and reset the connection. Resetting the connection drops the connection but retains the back-end processes. In the following example, replace *index\_name*, *table\_name*, *column\_name*, and *columnx\_value* with your information.

```
create index index_name on table_name(column_name);
```

```
insert into table_name values(column1_value,column2_value,column3_value);  
insert into table_name values(column1_value,column2_value,column3_value);  
set passthru reset;  
  
connect reset;
```

4. Connect to your remote Db2 database, create a nickname for your remote server, and perform operations. When you are done accessing data in the remote Db2 database, reset and then terminate the connection. In the following example, replace *database\_name* with the name of your remote Db2 database. Replace *nickname* with a name. Replace *server\_name* and *table\_name* with the name of the remote server and table on that server that you want to perform operations on. Replace *username* with the information for your remote server. Replace *sql\_command* with the operation to perform on the remote server.

```
connect to database_name;  
create nickname nickname for server_name."username".table_name";  
select sql_command from nickname;  
connect reset;  
terminate;
```

## Example

The following example creates a pass-through session to allow operations on the federated server testdb10.

Next, it creates the table t1 with three columns with different data types.

Then, the example creates the index i1\_t1 on three columns in table t1. Afterwards, it inserts two rows with values for these three columns, and then disconnects.

Last, the example connects to the remote Db2 database testdb2 and creates a nickname for the table t1 in the federated server testdb10. It creates the nickname with the username TESTUSER for that data source. An SQL command outputs all data from the table t1. The example disconnects and ends the session.

```
set passthru testdb10;  
  
create table t1 ( c1 decimal(13,0), c2 char(200), c3 int);  
  
create index i1_t1 on t1(c3);
```

```
insert into t1 values(1,'Test',1);
insert into t1 values(2,'Test 2',2);
connect reset;

connect to testdb2;
create nickname remote_t1 for testdb10."TESTUSER"."T1";
select * from remote_t1;
connect reset;
terminate;
```

# Working with replicas for Amazon RDS for Db2

RDS for Db2 supports creating replica databases to provide read scaling and disaster recovery capabilities. You can create replicas in two modes: read-only replicas for offloading read workloads, and standby replicas for cross-region disaster recovery. RDS for Db2 uses IBM Db2 High Availability Disaster Recovery (HADR) technology for replication. For more information, see [High availability disaster recovery \(HADR\)](#) in the IBM Db2 documentation.

A *Db2 replica* database is a physical copy of your primary database. A Db2 replica in read-only mode is called a *read replica*. A Db2 replica in standby mode is called a *standby replica*. Db2 doesn't permit writes in a replica, but you can promote a replica to make it writable. The promoted replica has the replicated data to the point when the request was made to promote it. For more information, see [Promoting a read replica to be a standalone DB instance](#).

For a summary of the features and behaviors of RDS for Db2 replicas, see [Differences between read replicas for DB engines](#).

## Read-only and standby replicas

When creating or modifying a Db2 replica, you can place it in either of the following modes:

### Read-only

This is the default. HADR transmits and applies changes from the source database to all read replica databases. For read-only replicas, the Db2 environment variable DB2\_HADR\_ROS is set to ON. The isolation level for read queries on the replica database is Uncommitted Read. For more information, see [Isolation level on the active standby database](#) in the IBM Db2 documentation.

For general information about read replicas that applies to all DB engines, see [Working with DB instance read replicas](#). For more information about Db2 HADR, see [High availability disaster recovery \(HADR\)](#) in the IBM Db2 documentation.

### Standby

For standby replicas, the Db2 environment variable DB2\_HADR\_ROS is set to OFF so that the replica databases don't accept user connections. The primary use for standby replicas is cross-Region disaster recovery.

A standby replica can't serve a read-only workload. The standby replica doesn't have any archive logs.

You can create up to three replicas from one source DB instance. You can create a combination of read-only and standby DB replicas for the same source DB instance. After you create a replica, you can change the replica mode. for more information, see [Modifying the RDS for Db2 replica mode](#).

Before creating replicas, make sure that you meet all requirements. For more information, see [Requirements and considerations for RDS for Db2 replicas](#).

## Database activations

Db2 HADR is configured at the database level. After you create replicas, HADR is set for all Db2 databases, including `rdsadmin`, which RDS fully manages. Before you create Db2 replicas, you must explicitly activate all databases. Otherwise, creation of replicas fails and Amazon RDS emits an event. After a DB instance has one or more replicas, you can't activate or deactivate any databases on the DB instance by using the `rdsadmin.activate_database` or `rdsadmin.deactivate_database` stored procedures. For more information, see [Stored procedures for databases for RDS for Db2](#).

## HADR configurations

You can see all HADR configurations for a database by connecting to the database and then running `db2 get db cfg`.

## Archive log retention

Amazon RDS purges logs from a primary DB instance after the following conditions have been met:

- The logs are at least two hours old.
- The setting for archive log retention hours has passed.
- The archive logs were successfully replicated to all replica DB instances. This condition applies both to DB instances in the same AWS Region and to cross-Region DB instances.

For information about setting archive log retention hours, see [rdsadmin.set\\_archive\\_log\\_retention](#).

Amazon RDS checks and cleans up each database individually. If a database loses the HADR connection or if information about the connection isn't available, then Amazon RDS skips the database and doesn't purge the archive logs.

## Outages during Db2 replication

When you create a replica, Amazon RDS takes a DB snapshot of your source DB instance and begins replication. When the DB snapshot operation begins, the source DB instance experiences a very brief I/O suspension. The I/O suspension typically lasts about one second. However, if the source DB instance is a Multi-AZ deployment, then the source DB instance doesn't experience any I/O suspension. This is because with Multi-AZ deployments, the snapshot is taken from the secondary DB instance.

The DB snapshot becomes the Db2 replica. Amazon RDS sets the necessary parameters and permissions for the source database and replica without any service interruption. Similarly, if you delete a replica, no outage occurs.

## Requirements and considerations for RDS for Db2 replicas

Db2 replica requirements fall into several categories: licensing and versioning, backup and restore considerations, replication behavior, and general operational considerations. Before creating a Db2 replica, familiarize yourself with the following requirements and considerations.

### Version and licensing requirements for RDS for Db2 replicas

Before you create an RDS for Db2 replica, review the following information about versions and licensing models:

- **Supported versions** – All Db2 11.5 versions support replica DB instances.

Source and replica DB instances must use the same major version. Db2 replicas support minor version upgrades but not major version upgrades. For information about upgrading DB instances, see [Upgrading a DB instance engine version](#).

 **Note**

When upgrading a source DB instance, all replicas are automatically upgraded to maintain version compatibility.

- **Valid licensing models and replica modes** – Both Db2 Advanced Edition (AE) and Standard Edition (SE) can create replicas in read-only or standby mode for both the Bring Your Own License (BYOL) model and the Db2 license through AWS Marketplace model.
- **Custom parameter group** – You must specify a custom parameter group for the replica.

For replicas that use the BYOL model, this custom parameter group must include your IBM Site ID and IBM Customer ID. For more information, see [IBM IDs for bring your own license \(BYOL\) for Db2](#). You can specify this custom parameter group for the replica by using the AWS Management Console, the AWS CLI , or the RDS API.

- **vCPU count** varies by replica mode and licensing model:
  - **Standby replicas** always use two vCPUs regardless of DB instance size.
    - **BYOL model** – AWS License Manager configurations show that RDS for Db2 DB instances use two vCPUs.
    - **Db2 license through AWS Marketplace model** – Bills reflect license costs for two vCPUs.
  - **Read-only replicas** use the same vCPU count as the DB instance size.
    - **BYOL model** – AWS License Manager configurations show that RDS for Db2 DB instances use the same number of vCPUs that match the DB instance size.
    - **Db2 license through AWS Marketplace model** – Bills reflect license costs for the same number of vCPUs that match the DB instance size.

## Backup and restore considerations for RDS for Db2 replicas

Replica backups have different behavior than primary database backups. Consider the following backup and restore requirements:

- To create snapshots of RDS for Db2 replicas or turn on automatic backups, make sure to set the backup retention period manually. Automatic backups aren't turned on by default.
- When you restore a replica backup, you restore to the database time, not the time that the backup was taken. The *database time* refers to the latest applied transaction time of the data in the backup. The difference is significant because a replica can lag behind the primary database for minutes or hours. When there are multiple databases, RDS for Db2 uses the earliest database time.

To find the difference, run the AWS CLI [describe-db-snapshots](#) command or call the RDS API [DescribeDBSnapshots](#) operation. Compare the SnapshotDatabaseTime value to the OriginalSnapshotCreateTime value. The SnapshotDatabaseTime value is the database time of the replica backup. The OriginalSnapshotCreateTime value is the latest applied transaction on the primary database.

For more information about backups and restoring backups, see [Working with RDS for Db2 replica backups](#).

## Replication considerations for RDS for Db2 replicas

Db2 replicas use HADR technology with specific limitations and behaviors. Review the following replication considerations:

- Replication uses Db2 HADR for all databases on the RDS for Db2 DB instance.
- Replication doesn't support the LOAD command. If you run the LOAD command from the source DB instance, you will receive inconsistent data.
- RDS for Db2 doesn't replicate the following items:
  - Storage access. Be aware of data, such as external tables, that rely on storage access.
  - Non-inline LOBs.
  - Binaries of external stored procedures (in C or Java).
- For standby replicas, RDS for Db2 replicates the following items:
  - Local users, except master users
  - Database configuration parameters
- For read-only replicas, RDS for Db2 replicates the following items:
  - Local users, except master users
  - SID group mappings

## Miscellaneous considerations for RDS for Db2 replicas

Additional operational considerations apply to Db2 replicas. Review the following items:

- RDS for Db2 replicates database configurations to the replicas. When RDS for Db2 promotes a replica, it deactivates and activates each database.
- RDS for Db2 replicates the local users, but not the master user, and SID group mappings to the replicas. You can modify the master user on the replica. For more information, see [Modifying an Amazon RDS DB instance](#).
- All databases must be in an active state. For information about activating databases, see [Stored procedures for databases for RDS for Db2](#).

- All stored procedures for creating, dropping, restoring, or rolling forward databases must be completed before creating a replica. For information about these stored procedures, see [Stored procedures for databases for RDS for Db2](#).
- When the replica is created, Amazon RDS sets the database-level parameter `blocknonlogged` for all databases on the source DB instance to YES. When the source replica becomes a standalone instance again, Amazon RDS sets the value back to NO. For more information, see [blocknonlogged - Block creation of tables that allow non-logged activity configuration parameter](#) in the IBM Db2 documentation.
- When the replica is created, Amazon RDS sets the database-level parameter `logindexbuild` for all databases on the source DB instance to YES. When the source replica becomes a standalone instance again, Amazon RDS sets the value back to NO. For more information, see [logindexbuild - Log index pages created configuration parameter](#) in the IBM Db2 documentation.

## Preparing to create an RDS for Db2 replica

Before creating an RDS for Db2 replica, you must complete the following tasks for successful replication. These tasks help prevent common issues and ensure optimal performance.

### Task 1: Enable automatic backups

Before a DB instance can serve as a source DB instance, you must enable automatic backups on the source DB instance. This is a prerequisite for all replica creation operations. To learn how to perform this procedure, see [Enabling automated backups](#).

For information about backups specific to Db2 replicas, see [Working with RDS for Db2 replica backups](#).

### Task 2: Plan compute and storage resources

Ensure that the source DB instance and its replicas are sized properly, in terms of compute and storage, to suit their operational load. If a replica reaches compute, network, or storage resource capacity, the replica stops receiving or applying changes from its source. For information about monitoring replica performance and resource utilization, see [Monitoring read replication](#).

RDS for Db2 doesn't intervene to mitigate high replica lag between a source DB instance and its replicas. If you experience high replica lag, see [Monitoring Db2 replication lag](#) for troubleshooting guidance.

You can modify the storage and CPU resources of a replica independently from its source and other replicas. For more information, see [Modifying an Amazon RDS DB instance](#).

## Task 3: Prepare databases

Before creating a replica, confirm that your databases are ready based on the following points:

- The DB instance contains all databases that you want present on the DB instance. After replica creation, you can't create, drop, or native restore a database on the DB instance. Any calls to the [rdsadmin.create\\_database](#), [rdsadmin.drop\\_database](#), or [rdsadmin.restore\\_database](#) stored procedures fail.
- All databases on the DB instance are in an active state. If any database is in an inactive state, replica creation will fail. For information about activating databases, see [Stored procedures for databases for RDS for Db2](#).

## Next steps

After completing all the preparation tasks, you are ready to create a Db2 replica.

- To create a read-only replica, see [Creating a read replica](#).
- To create a standby replica, see [Creating a standby Db2 replica](#).

## Creating an RDS for Db2 replica in standby mode

By default, Db2 replicas are created in read-only mode. You can create a replica in standby mode for disaster recovery purposes. Standby replicas don't accept user connections but provide faster failover capabilities for cross-Region scenarios.

Before creating a standby replica, make sure that you have completed the preparation tasks. For more information, see [Preparing to create an RDS for Db2 replica](#). After creating a standby replica, you can change the replica mode. For more information, see [Modifying the RDS for Db2 replica mode](#).

You can create a standby replica using the AWS Management Console, the AWS CLI, or the RDS API. For information about creating a read-only replica, see [Creating a read replica](#).

## Console

### To create a standby replica from a source RDS for Db2 DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the RDS for Db2 DB instance that you want to use as the source for a standby replica.
4. For **Actions**, choose **Create replica**.
5. For **Replica mode**, choose **Standby**.
6. Choose the settings that you want to use. For **DB instance identifier**, enter a name for the standby replica. Adjust other settings as needed.
7. For **Regions**, choose the AWS Region where the standby replica will be launched.
8. Choose your instance size and storage type. We recommend that you use the same DB instance class and storage type as the source DB instance for the standby replica.
9. For **Multi-AZ deployment**, choose **Create a standby instance** to create a standby of your replica in another Availability Zone for failover support for the standby replica.
10. Choose the other settings that you want to use.
11. Choose **Create replica**.

In the **Databases** page, the standby replica has the role **Replica**.

## AWS CLI

To create a Db2 replica in standby mode, use the AWS CLI command [create-db-instance-read-replica](#) with `--replica-mode` set to `mounted`.

### Example

For Linux, macOS, or Unix:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier my_standby_replica \
    --source-db-instance-identifier my_db_instance \
    --replica-mode mounted
```

For Windows:

```
aws rds create-db-instance-read-replica ^
--db-instance-identifier my_standby_replica ^
--source-db-instance-identifier my_db_instance ^
--replica-mode mounted
```

## RDS API

To create a Db2 replica in standby mode, specify ReplicaMode=mounted in the RDS API operation [CreateDBInstanceReadReplica](#).

## Modifying the RDS for Db2 replica mode

You can change the replica mode of an existing Db2 replica between read-only and standby modes. This flexibility allows you to adapt your replica configuration based on changing requirements for read workloads or disaster recovery needs.

You might want to change replica modes in the following scenarios:

- **Read-only to standby** – When you no longer need read capacity but want to maintain disaster recovery capabilities
- **Standby to read-only** – When you need to add read capacity for reporting or analytics workloads

Before changing replica modes, make sure the following conditions are met:

- The replica is in an available state.
- No active maintenance operations are running on the replica.
- You have the necessary permissions to modify the DB instance.

The change operation can take a few minutes. During the operation, the DB instance status changes to **modifying**. For more information about status changes, see [Viewing Amazon RDS DB instance status](#). When you change from read-only to standby mode, the replica disconnects all active connections.

## Important

Because changing replica modes temporarily interrupts service, plan the change during a maintenance window to minimize impact on your applications.

You can modify the replica mode using the AWS Management Console, the AWS CLI, or the RDS API.

## Console

### To change the replica mode of a Db2 replica

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the replica database that you want to modify.
4. Choose **Modify**.
5. For **Replica mode**, choose the desired mode:
  - **Read-only** – For read workloads
  - **Standby** – For disaster recovery
6. Choose the other settings that you want to change.
7. Choose **Continue**.
8. For **Scheduling of modifications**, choose **Apply immediately**.
9. Choose **Modify DB instance**.
10. After the modification completes, verify the replica mode change in the **Databases** page. The replica status should show as **Available** when the change is complete.

## AWS CLI

To change a Db2 replica from read-only mode to standby mode, set `--replica-mode` to `mounted` in the AWS CLI command [modify-db-instance](#). To change a Db2 replica from standby mode to read-only mode, set `--replica-mode` to `open-read-only`.

The following example changes a replica from read-only to standby mode:

## Example

For Linux, macOS, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier my_db2_replica \
--replica-mode mounted
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier my_db2_replica ^
--replica-mode mounted
```

The following example changes a replica from standby to read-only mode:

## Example

For Linux, macOS, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier my_db2_replica \
--replica-mode open-read-only
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier my_db2_replica ^
--replica-mode open-read-only
```

## RDS API

To change a Db2 replica from read-only mode to standby mode, set ReplicaMode=mounted in [ModifyDBInstance](#). To change a Db2 replica from standby mode to read-only mode, set ReplicaMode=open-read-only.

The following is an example API call to change the replica mode from read-only to standby:

```
{
    "DBInstanceIdentifier": "my_db2_replica",
    "ReplicaMode": "mounted",
```

```
"ApplyImmediately": true  
}
```

The following is an example API call to change the replica mode from standby to read-only:

```
{  
    "DBInstanceIdentifier": "my_db2_replica",  
    "ReplicaMode": "open-read-only",  
    "ApplyImmediately": true  
}
```

For information about the differences between replica modes, see [Working with replicas for Amazon RDS for Db2](#). For troubleshooting replica issues, see [Troubleshooting RDS for Db2 replication issues](#).

## Working with RDS for Db2 replica backups

You can create and restore backups of an RDS for Db2 replica just like a primary database. However, there are important differences in how replica backups work, particularly regarding restore timing and backup retention settings.

RDS for Db2 supports both automatic backups and manual snapshots for replicas. RDS for Db2 doesn't support point-in-time restore. For information about RDS backups, see [Backing up, restoring, and exporting data](#).

### Key differences for replica backups

Replica backups differ from primary database backups in several important ways:

- Automatic backups aren't enabled by default for replicas.
- Restore operations use database time rather than backup creation time.
- Replica lag can affect the actual data restored. For information about monitoring replica lag, see [Monitoring Db2 replication lag](#).

### Enabling automatic backups for RDS for Db2 replicas

Unlike primary databases, RDS for Db2 replicas don't have automated backups enabled by default. You must manually configure the backup retention period to enable automatic backups. Enable automated backups by setting the backup retention period to a positive nonzero value.

## Console

### To enable automatic backups immediately

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to modify.
3. Choose **Modify**.
4. For **Backup retention period**, choose a positive nonzero value, for example three days.
5. Choose **Continue**.
6. Choose **Apply immediately**.
7. Choose **Modify DB instance** to save your changes and enable automated backups.

## AWS CLI

To enable automated backups, use the AWS CLI [modify-db-instance](#) command.

Include the following parameters:

- `--db-instance-identifier`
- `--backup-retention-period`
- `--apply-immediately` or `--no-apply-immediately`

The following example enables automated backups by setting the backup retention period to three days. The changes are applied immediately.

### Example

For Linux, macOS, or Unix:

```
aws rds modify-db-instance \
  --db-instance-identifier my_db_instance \
  --backup-retention-period 3 \
  --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
  --db-instance-identifier my_db_instance ^
  --backup-retention-period 3 ^
  --apply-immediately
```

## RDS API

To enable automated backups, use the RDS API [ModifyDBInstance](#) operation with the following required parameters:

- `DBInstanceIdentifier`
- `BackupRetentionPeriod`

## Restoring an RDS for Db2 replica backup

You can restore an RDS for Db2 replica backup the same way that you can restore a backup of the primary database. For more information, see [Restoring to a DB instance](#).

The most important consideration when restoring replica backups is understanding the difference between database time and backup creation time, especially when replica lag is present.

You can monitor replication lag and ensure that your backups contain the expected data. For information about the `ReplicaLag` metric, see [Amazon CloudWatch metrics for Amazon RDS](#).

## Understanding timing differences

When you restore a replica backup, you must determine the point in time to which you are restoring. The *database time* refers to the latest applied transaction time of the data in the backup. When you restore a replica backup, you restore to the database time, not the time when the backup completed. The difference is significant because a replica can lag behind the primary database by minutes or hours. Thus, the database time of a replica backup might be much earlier than the snapshot creation time.

To find the difference between database time and creation time, run the AWS CLI [describe-db-snapshots](#) command or call the RDS API [DescribeDBSnapshots](#) operation. Compare the `SnapshotDatabaseTime` value and the `OriginalSnapshotCreateTime` value. The `SnapshotDatabaseTime` value is the earliest database time among all the databases of the replica backup. The `OriginalSnapshotCreateTime` value is the latest applied transaction on

the primary database. Note that replication lags could be different for multiple databases, and the database time could be in between these two times.

The following AWS CLI example shows the difference between the two times:

For Linux, macOS, or Unix:

```
aws rds describe-db-snapshots \
--db-instance-identifier my_db2_replica \
--db-snapshot-identifier my_replica_snapshot
```

For Windows:

```
aws rds describe-db-snapshots ^
--db-instance-identifier my_db2_replica ^
--db-snapshot-identifier my_replica_snapshot
```

This command produces output similar to the following example.

```
{
    "DBSnapshots": [
        {
            "DBSnapshotIdentifier": "my_replica_snapshot",
            "DBInstanceIdentifier": "my_db2_replica",
            "SnapshotDatabaseTime": "2022-07-26T17:49:44Z",
            ...
            "OriginalSnapshotCreateTime": "2021-07-26T19:49:44Z"
        }
    ]
}
```

## Troubleshooting RDS for Db2 replication issues

This topic describes common RDS for Db2 replication issues and provides troubleshooting guidance for both read-only and standby replicas. In addition to reviewing the following troubleshooting information, make sure that you followed the [requirements and considerations](#), and completed the [preparation steps](#) before creating Db2 replicas.

### Replica creation failures

Replica creation can fail for several reasons:

- **Inactive databases** – All databases on the source DB instance must be active before creating replicas.

For information about activating databases, see [Stored procedures for databases for RDS for Db2](#).

- **Missing automatic backups** – The source DB instance must have automatic backups enabled.

For information about enabling backups, see [Enabling automatic backups for RDS for Db2 replicas](#).

- **Parameter group issues** – Custom parameter groups are required for replicas. For BYOL licensing, the parameter group must include the IBM Site ID and IBM Customer ID.

For more information, see [IBM IDs for bring your own license \(BYOL\) for Db2](#).

## Monitoring Db2 replication lag

To monitor replication lag in Amazon CloudWatch, view the Amazon RDS ReplicaLag metric.

For more information about replication lag time, see [Monitoring read replication](#) and [Amazon CloudWatch metrics for Amazon RDS](#). For information about setting up CloudWatch alarms for replica lag, see [Monitoring Amazon RDS metrics with Amazon CloudWatch](#).

For a read-only replica, if the lag time is too long, query the MON\_GET\_HADR table for the status of the replica DB instance.

For a standby replica, if the lag time is too long, query the MON\_GET\_HADR table for the status of the source DB instance. Don't query the replica DB instance because replica DB instances don't accept user connections.

Common causes of high replication lag include the following reasons:

- Insufficient compute resources on the replica
- Network connectivity issues between the source and the replica
- High write activity on the source database
- Storage performance limitations on the replica

If high replication lag persists, consider scaling your replica resources. For more information, see [Modifying an Amazon RDS DB instance](#).

## Db2 replication errors

Db2 replication can be in an error state for a number of reasons. Perform the following actions:

- Monitor events and the DB instance state to make sure that the DB instance is replicating.

For more information, see [Working with Amazon RDS event notification](#).

- Check the diagnostic logs for the Db2 replica in the Amazon RDS console. In the logs, look for errors in HADR messages. Compare the log sequence number to the primary sequence number.

For information about accessing and interpreting Db2 diagnostic logs, see [Amazon RDS for Db2 database log files](#). For information about Db2 HADR configuration and troubleshooting, see [Working with replicas for Amazon RDS for Db2](#).

If replication errors persist, you might need to recreate the replica.

## Connection issues

If you can't connect to your replica, review the following information about the replica modes:

- **Standby replicas** – They don't accept user connections by design. Use read-only replicas for read workloads.
- **Read-only replicas** – Check your security group settings, network ACLs, and parameter group configurations.

For more information, see [Control traffic to your AWS resources using security groups](#) in the [Amazon VPC User Guide](#), [Control subnet traffic with network access control lists](#) in the [Amazon VPC User Guide](#), and [Parameter groups for Amazon RDS](#).

## Performance issues

If replica performance is poor, review the following suggestions:

- Ensure the replica has adequate compute and storage resources.
- Monitor the ReplicaLag metric in Amazon CloudWatch.
- Consider scaling up the replica DB instance class.

For information about modifying resources or instance classes, see [Modifying an Amazon RDS DB instance](#).

For information monitoring replication lag, see [Monitoring replication lag](#) and [Amazon CloudWatch metrics for Amazon RDS](#). For information about setting up CloudWatch alarms for replica lag, see [Monitoring Amazon RDS metrics with Amazon CloudWatch](#).

## Options for Amazon RDS for Db2 DB instances

The following shows the options, or additional features, that are available for Amazon RDS instances running the Db2 DB engine. To enable these options, you can add them to a custom option group, and then associate the option group with your DB instance. For more information about working with option groups, see [Working with option groups](#).

Amazon RDS supports the following options for Db2:

Option	Option ID
<a href="#">Db2 audit logging</a>	DB2_AUDIT

## Db2 audit logging

With Db2 audit logging, Amazon RDS records database activity, including users logging on to the database and queries run against the database. RDS uploads the completed audit logs to your Amazon S3 bucket, using the AWS Identity and Access Management (IAM) role that you provide.

### Topics

- [Setting up Db2 audit logging](#)
- [Managing Db2 audit logging](#)
- [Viewing audit logs](#)
- [Troubleshooting Db2 audit logging](#)

### Setting up Db2 audit logging

To enable audit logging for an Amazon RDS for Db2 database, you enable the DB2\_AUDIT option on the RDS for Db2 DB instance. Then, configure an audit policy to enable the feature for the specific database. To enable the option on the RDS for Db2 DB instance, you configure the option settings for the DB2\_AUDIT option. You do so by providing the Amazon Resource Names (ARNs) for your Amazon S3 bucket and the IAM role with permissions to access your bucket.

To set up Db2 audit logging for an RDS for Db2 database, complete the following steps.

### Topics

- [Step 1: Create an Amazon S3 bucket](#)
- [Step 2: Create an IAM policy](#)
- [Step 3: Create an IAM role and attach your IAM policy](#)
- [Step 4: Configure an option group for Db2 audit logging](#)
- [Step 5: Configure the audit policy](#)
- [Step 6: Check the audit configuration](#)

#### Step 1: Create an Amazon S3 bucket

If you haven't already done so, create an Amazon S3 bucket where Amazon RDS can upload your RDS for Db2 database's audit log files. The following restrictions apply to the S3 bucket that you use as a target for audit files:

- It must be in the same AWS Region as your RDS for Db2 DB instance.
- It must not be open to the public.
- The bucket owner must also be the IAM role owner.

To learn how to create an Amazon S3 bucket, see [Creating a bucket](#) in the *Amazon S3 User Guide*.

After you enable audit logging, Amazon RDS automatically sends the logs from your DB instance to the following locations:

- DB instance level logs – *bucket\_name/db2-audit-logs/dbi\_resource\_id/date\_time\_utc/*
- Database level logs – *bucket\_name/db2-audit-logs/dbi\_resource\_id/date\_time\_utc/db\_name/*

Take note of the Amazon Resource Name (ARN) for your bucket. This information is needed to complete subsequent steps.

## Step 2: Create an IAM policy

Create an IAM policy with the permissions required to transfer audit log files from your DB instance to your Amazon S3 bucket. This step assumes that you have an S3 bucket.

Before you create the policy, gather the following information:

- The ARN for your bucket.
- The ARN for your AWS Key Management Service (AWS KMS) key, if your bucket uses SSE-KMS encryption.

Create an IAM policy that includes the following permissions:

```
"s3>ListBucket",
"s3:GetBucketAcl",
"s3:GetBucketLocation",
"s3:PutObject",
"s3>ListMultipartUploadParts",
"s3:AbortMultipartUpload",
"s3>ListAllMyBuckets"
```

**Note**

Amazon RDS needs the `s3>ListAllMyBuckets` action internally to verify that the same AWS account owns both the S3 bucket and the RDS for Db2 DB instance.

If your bucket uses SSE-KMS encryption, also include the following permissions for your IAM role and AWS KMS key.

Include the following permissions to the policy for your IAM role.

```
"kms:GenerateDataKey",
"kms:Decrypt"
```

Include the following permissions to the key policy for your AWS KMS key. Replace `111122223333` with your account number and `AROA123456789EXAMPLE` with your IAM role name.

```
{
  "Sid": "Allow RDS role to use the key",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:sts::111122223333:assumed-role/AROA123456789EXAMPLE/RDS-Db2Audit",
      "arn:aws:iam::111122223333:role/AROA123456789EXAMPLE"
    ]
  },
  "Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

You can create an IAM policy by using the AWS Management Console or the AWS Command Line Interface (AWS CLI).

## Console

### To create an IAM policy to allow Amazon RDS to access your Amazon S3 bucket

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Policies**.
3. Choose **Create policy**, and then choose **JSON**.
4. In **Add actions**, filter by **S3**. Add access **ListBucket**, **GetBucketAcl**, and **GetBucketLocation**.
5. For **Add a resource**, choose **Add**. For **Resource type**, choose **bucket**, and enter the name of your bucket. Then, choose **Add resource**.
6. Choose **Add new statement**.
7. In **Add actions**, filter by **S3**. Add access **PutObject**, **ListMultipartUploadParts**, and **AbortMultipartUpload**.
8. For **Add a resource**, choose **Add**. For **Resource type**, choose **object**, and enter *your bucket name/\**. Then, choose **Add resource**.
9. Choose **Add new statement**.
10. In **Add actions**, filter by **S3**. Add access **ListAllMyBuckets**.
11. For **Add a resource**, choose **Add**. For **Resource type**, choose **All Resources**. Then, choose **Add resource**.
12. If you're using your own KMS keys to encrypt the data:
  1. Choose **Add new statement**.
  2. In **Add actions**, filter by KMS. Add access **GenerateDataKey** and **Decrypt**.
  3. For **Add a resource**, choose **Add**. For **Resource type**, choose **All Resources**. Then, choose **Add resource**.
13. Choose **Next**.
14. For **Policy name**, enter a name for this policy.
15. (Optional) For **Description**, enter a description for this policy.
16. Choose **Create policy**.

## AWS CLI

### To create an IAM policy to allow Amazon RDS to access your Amazon S3 bucket

1. Run the [create-policy](#) command. In the following example, replace *iam\_policy\_name* and *amzn-s3-demo-bucket* with a name for your IAM policy and the name of your target Amazon S3 bucket.

For Linux, macOS, or Unix:

```
aws iam create-policy \
--policy-name iam_policy_name \
--policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Action": [
                "s3>ListBucket",
                "s3>GetBucketAcl",
                "s3>GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        },
        {
            "Sid": "Statement2",
            "Effect": "Allow",
            "Action": [
                "s3>PutObject",
                "s3>ListMultipartUploadParts",
                "s3>AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Sid": "Statement3",
            "Effect": "Allow",
            "Action": [
                "s3>GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}'
```

```
        "s3>ListAllMyBuckets"
    ],
    "Resource": [
        "*"
    ],
},
{
    "Sid": "Statement4",
    "Effect": "Allow",
    "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
    ],
    "Resource": [
        "*"
    ]
}
]
}'
```

For Windows:

```
aws iam create-policy ^
--policy-name iam_policy_name ^
--policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Action": [
                "s3>ListBucket",
                "s3>GetBucketAcl",
                "s3>GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        },
        {
            "Sid": "Statement2",
            "Effect": "Allow",
            "Action": [
```

```
        "s3:PutObject",
        "s3>ListMultipartUploadParts",
        "s3:AbortMultipartUpload"
    ],
    "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
    ]
},
{
    "Sid": "Statement3",
    "Effect": "Allow",
    "Action": [
        "s3>ListAllMyBuckets"
    ],
    "Resource": [
        "*"
    ]
},
{
    "Sid": "Statement4",
    "Effect": "Allow",
    "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
    ],
    "Resource": [
        "*"
    ]
}
]
}'
```

2. After the policy is created, note the ARN of the policy. You need the ARN for [Step 3: Create an IAM role and attach your IAM policy](#).

For information about creating an IAM policy, see [Creating IAM policies](#) in the IAM User Guide.

### Step 3: Create an IAM role and attach your IAM policy

This step assumes that you created the IAM policy in [Step 2: Create an IAM policy](#). In this step, you create an IAM role for your RDS for Db2 DB instance and then attach your IAM policy to the role.

You can create an IAM role for your DB instance by using the console or the AWS CLI.

## Console

### To create an IAM role and attach your IAM policy to it

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**.
3. Choose **Create role**.
4. For **Trusted entity type**, select **AWS service**.
5. For **Service or use case**, select **RDS**, and then select **RDS – Add Role to Database**.
6. Choose **Next**.
7. For **Permissions policies**, search for and select the name of the IAM policy that you created.
8. Choose **Next**.
9. For **Role name**, enter a role name.
10. (Optional) For **Description**, enter a description for the new role.
11. Choose **Create role**.

## AWS CLI

### To create an IAM role and attach your IAM policy to it

1. Run the [create-role](#) command. In the following example, replace *iam\_role\_name* with a name for your IAM role.

For Linux, macOS, or Unix:

```
aws iam create-role \
--role-name iam_role_name \
--assume-role-policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rds.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'
```

```
]  
}'
```

For Windows:

```
aws iam create-role ^  
--role-name iam_role_name ^  
--assume-role-policy-document '{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "rds.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}'
```

2. After the role is created, note the ARN of this role. You need this ARN for the next step, [Step 4: Configure an option group for Db2 audit logging](#).
3. Run the [attach-role-policy](#) command. In the following example, replace *iam\_policy\_arn* with the ARN of the IAM policy that you created in [Step 2: Create an IAM policy](#). Replace *iam\_role\_name* with the name of the IAM role that you just created.

For Linux, macOS, or Unix:

```
aws iam attach-role-policy \  
--policy-arn iam_policy_arn \  
--role-name iam_role_name
```

For Windows:

```
aws iam attach-role-policy ^  
--policy-arn iam_policy_arn ^  
--role-name iam_role_name
```

For more information, see [Creating a role to delegate permissions to an IAM user](#) in the *IAM User Guide*.

## Step 4: Configure an option group for Db2 audit logging

The process for adding the Db2 audit logging option to an RDS for Db2 DB instance is as follows:

1. Create a new option group, or copy or modify an existing option group.
2. Add and configure all required options.
3. Associate the option group with the DB instance.

After you add the Db2 audit logging option, you don't need to restart your DB instance. As soon as the option group is active, you can create audits and store audit logs in your S3 bucket.

### To add and configure Db2 audit logging on a DB instance's option group

1. Choose one of the following:
  - Use an existing option group.
  - Create a custom DB option group, and use that option group. For more information, see [Creating an option group](#).
2. Add the **DB2\_AUDIT** option to the option group, and configure the option settings. For more information about adding options, see [Adding an option to an option group](#).
  - For **IAM\_ROLE\_ARN**, enter the ARN of the IAM role that you created in [the section called "Create an IAM role and attach your IAM policy"](#).
  - For **S3\_BUCKET\_ARN**, enter the ARN of the S3 bucket to use for your Db2 audit logs. The bucket must be in the same Region as your RDS for Db2 DB instance. The policy associated with the IAM role you entered must allow the required operations on this resource.
3. Apply the option group to a new or existing DB instance. Choose one of the following:
  - If you are creating a new DB instance, apply the option group when you launch the instance.
  - On an existing DB instance, apply the option group by modifying the instance and then attaching the new option group. For more information, see [Modifying an Amazon RDS DB instance](#).

## Step 5: Configure the audit policy

To configure the audit policy for your RDS for Db2 database, connect to the `rdsadmin` database using the master username and master password for your RDS for Db2 DB instance. Then, call the

rdsadmin.configure\_db\_audit stored procedure with the DB name of your database and the applicable parameter values.

The following example connects to the database and configures an audit policy for testdb with the categories AUDIT, CHECKING, OBJMAINT, SECMAINT, SYSADMIN, and VALIDATE. The status value BOTH logs success and failures, and the ERROR\_TYPE is NORMAL by default. For more information about how to use this stored procedure, see [the section called "rdsadmin.configure\\_db\\_audit".](#)

```
db2 "connect to rdsadmin user master_user using master_password"  
db2 "call rdsadmin.configure_db_audit('testdb', 'ALL', 'BOTH', ?)"
```

## Step 6: Check the audit configuration

To make sure that your audit policy is set up correctly, check the status of your audit configuration.

To check the configuration, connect to the rdsadmin database using the master username and master password for your RDS for Db2 DB instance. Then, run the following SQL statement with the DB name of your database. In the following example, the DB name is *testdb*.

```
db2 "select task_id, task_type, database_name, lifecycle,  
      varchar(bson_to_json(task_input_params), 500) as task_params,  
      cast(task_output as varchar(500)) as task_output  
    from table(rdsadmin.get_task_status(null, 'testdb', 'CONFIGURE_DB_AUDIT'))"
```

Sample Output

TASK_ID	TASK_TYPE	DATABASE_NAME	LIFECYCLE
2	CONFIGURE_DB_AUDIT	DB2DB	SUCCESS

... continued ...

TASK_PARAMS
{ "AUDIT_CATEGORY" : "ALL", "CATEGORY_SETTING" : "BOTH" }

... continued ...

TASK_OUTPUT
2023-12-22T20:27:03.029Z Task execution has started.

2023-12-22T20:27:04.285Z Task execution has completed successfully.

## Managing Db2 audit logging

After you set up Db2 audit logging, you can modify the audit policy for a specific database, or disable audit logging at the database level or for the entire DB instance. You can also change the Amazon S3 bucket where your log files are uploaded to.

### Topics

- [Modifying a Db2 audit policy](#)
- [Modifying the location of your log files](#)
- [Disabling Db2 audit logging](#)

### Modifying a Db2 audit policy

To modify the audit policy for a specific RDS for Db2 database, run the `rdsadmin.configure_db_audit` stored procedure. With this stored procedure, you can change the categories, category settings, and error type configuration of the audit policy. For more information, see [the section called “rdsadmin.configure\\_db\\_audit”](#).

### Modifying the location of your log files

To change the Amazon S3 bucket where your log files are uploaded to, do one of the following:

- Modify the current option group attached to your RDS for Db2 DB instance – Update the `S3_BUCKET_ARN` setting for the `DB2_AUDIT` option to point to the new bucket. Also, make sure to update the IAM policy attached to the IAM role specified by the `IAM_ROLE_ARN` setting in the attached option group. This IAM policy must provide your new bucket with the required access permissions. For information about the permissions required in the IAM policy, see [Create an IAM policy](#).
- Attach your RDS for Db2 DB instance to a different option group – Modify your DB instance to change the option group that's attached to it. Make sure that the new option group is configured with the correct `S3_BUCKET_ARN` and `IAM_ROLE_ARN` settings. For information about how to configure these settings for the `DB2_AUDIT` option, see [Configure an option group](#).

When you modify the option group, make sure that you apply the changes immediately. For more information, see [the section called “Modifying a DB instance”](#).

## Disabling Db2 audit logging

To disable Db2 audit logging, do one of the following:

- Disable audit logging for the RDS for Db2 DB instance – Modify your DB instance and remove the option group with the DB2\_AUDIT option from it. For more information, see [the section called “Modifying a DB instance”](#).
- Disable audit logging for a specific database – Stop audit logging and remove the audit policy by calling `rdsadmin.disable_db_audit` with the DB name of your database. For more information, see [the section called “rdsadmin.disable\\_db\\_audit”](#).

```
db2 "call rdsadmin.disable_db_audit(
    'db_name',
    ?)"
```

## Viewing audit logs

After you enable Db2 audit logging, wait for at least one hour before viewing the audit data in your Amazon S3 bucket. Amazon RDS automatically sends the logs from your RDS for Db2 DB instance to the following locations:

- DB instance level logs – `bucket_name/db2-audit-logs/dbi_resource_id/date_time_utc/`
- Database level logs – `bucket_name/db2-audit-logs/dbi_resource_id/date_time_utc/db_name/`

The following example screenshot of the Amazon S3 console shows a list of folders for RDS for Db2 DB instance level log files.

Amazon S3 > Buckets > db2-audit-logs-devo > db2-audit-logs/ > db-5N7FXOY4GDP7RG2NSH2ZTAI2W4/ > 2024-01-15\_22:50:00\_UTC/

### 2024-01-15\_22:50:00\_UTC/

Copy S3 URI

**Objects** Properties

**Objects (10) Info** Copy S3 URI Copy URL Download Open Delete Actions ▾ Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	audit.del	del	January 15, 2024, 14:50:01 (UTC-08:00)	9.4 KB	Standard
<input type="checkbox"/>	auditlobs	-	January 15, 2024, 14:50:01 (UTC-08:00)	0 B	Standard
<input type="checkbox"/>	checking.del	del	January 15, 2024, 14:50:01 (UTC-08:00)	127.5 KB	Standard
<input type="checkbox"/>	context.del	del	January 15, 2024, 14:50:01 (UTC-08:00)	0 B	Standard
<input type="checkbox"/>	execute.del	del	January 15, 2024, 14:50:01 (UTC-08:00)	0 B	Standard
<input type="checkbox"/>	objmaint.del	del	January 15, 2024, 14:50:02 (UTC-08:00)	0 B	Standard
<input type="checkbox"/>	SAMPLE/	Folder	-	-	-
<input type="checkbox"/>	secmaint.del	del	January 15, 2024, 14:50:02 (UTC-08:00)	0 B	Standard
<input type="checkbox"/>	sysadmin.del	del	January 15, 2024, 14:50:02 (UTC-08:00)	28.5 KB	Standard
<input type="checkbox"/>	validate.del	del	January 15, 2024, 14:50:01 (UTC-08:00)	72.6 KB	Standard

The following example screenshot of the Amazon S3 console shows database level log files for the RDS for Db2 DB instance.

Amazon S3 > Buckets > db2-audit-logs-devo > db2-audit-logs/ > db-5N7FXOY4GDP7RG2NSH2ZTAI2W4/ > 2024-01-15\_22:50:00\_UTC/ > SAMPLE/

### SAMPLE/

Copy S3 URI

**Objects** Properties

**Objects (9) Info** Copy S3 URI Copy URL Download Open Delete Actions ▾ Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	audit.del	del	January 15, 2024, 14:50:01 (UTC-08:00)	9.4 KB	Standard
<input type="checkbox"/>	auditlobs	-	January 15, 2024, 14:50:01 (UTC-08:00)	0 B	Standard
<input type="checkbox"/>	checking.del	del	January 15, 2024, 14:50:01 (UTC-08:00)	127.5 KB	Standard
<input type="checkbox"/>	context.del	del	January 15, 2024, 14:50:01 (UTC-08:00)	0 B	Standard
<input type="checkbox"/>	execute.del	del	January 15, 2024, 14:50:01 (UTC-08:00)	0 B	Standard
<input type="checkbox"/>	objmaint.del	del	January 15, 2024, 14:50:01 (UTC-08:00)	0 B	Standard
<input type="checkbox"/>	secmaint.del	del	January 15, 2024, 14:50:01 (UTC-08:00)	0 B	Standard
<input type="checkbox"/>	sysadmin.del	del	January 15, 2024, 14:50:01 (UTC-08:00)	28.5 KB	Standard
<input type="checkbox"/>	validate.del	del	January 15, 2024, 14:50:01 (UTC-08:00)	72.6 KB	Standard

## Troubleshooting Db2 audit logging

Use the following information to troubleshoot common issues with Db2 audit logging.

### Can't configure the audit policy

If calling the stored procedure `rdsadmin.configure_db_audit` returns an error, it could be that the option group with the `DB2_AUDIT` option isn't associated with the RDS for Db2 DB instance. Modify the DB instance to add the option group, and then try calling the stored procedure again. For more information, see [Modifying an Amazon RDS DB instance](#).

### No data in the Amazon S3 bucket

If logging data is missing from the Amazon S3 bucket, check the following:

- The Amazon S3 bucket is in the same Region as your RDS for Db2 DB instance.
- The role you specified in the `IAM_ROLE_ARN` option setting is configured with the required permissions to upload logs to your Amazon S3 bucket. For more information, see [Create an IAM policy](#).
- The ARNs for the `IAM_ROLE_ARN` and `S3_BUCKET_ARN` option settings are correct in the option group associated with your RDS for Db2 DB instance. For more information, see [Configure an option group](#).

You can check the task status of your audit logging configuration by connecting to the database and running a SQL statement. For more information, see [Check the audit configuration](#).

You can also check events to find out more about why logs might be missing. For information about how to view events, see [the section called "Viewing logs, events, and streams in the Amazon RDS console"](#).

# External stored procedures for Amazon RDS for Db2

You can create external routines and register them with your Amazon RDS for Db2 databases as external stored procedures. Currently, RDS for Db2 only supports Java-based routines for external stored procedures.

## Java-based external stored procedures

Java-based external stored procedures are external Java routines that you register with your RDS for Db2 database as external stored procedures.

### Topics

- [Limitations for Java-based external stored procedures](#)
- [Configuring Java-based external stored procedures](#)

## Limitations for Java-based external stored procedures

Before you develop your external routine, consider the following limitations and restrictions.

To create your external routine, make sure to use the Java Development Kit (JDK) provided by Db2. For more information, see [Java software support for Db2 database products](#).

Your Java program can create files only in the /tmp directory, and Amazon RDS doesn't support enabling executable or Set User ID (SUID) permissions on these files. Your Java program also can't use socket system calls or the following system calls:

- \_sysctl
- acct
- afs\_syscall
- bpf
- capset
- chown
- chroot
- create\_module
- delete\_module

- fanotify\_init
- fanotify\_mark
- finit\_module
- fsconfig
- fsopen
- fspick
- get\_kernel\_syms
- getpmsg
- init\_module
- mount
- move\_mount
- nfsservctl
- open\_by\_handle\_at
- open\_tree
- pivot\_root
- putpmsg
- query\_module
- quotactl
- reboot
- security
- setdomainname
- setfsuid
- sethostname
- sysfs
- tuxcall
- umount2
- uselib
- ustat
- vhangup
- vserver

For additional restrictions on external routines for Db2, see [Restrictions on external routines](#) in the IBM Db2 documentation.

## Configuring Java-based external stored procedures

To configure an external stored procedure, create a .jar file with your external routine, install it on your RDS for Db2 database, and then register it as an external stored procedure.

### Topics

- [Step 1: Enable external stored procedures](#)
- [Step 2: Install the .jar file with your external routine](#)
- [Step 3: Register the external stored procedure](#)
- [Step 4: Validate the external stored procedure](#)

### Step 1: Enable external stored procedures

To enable external stored procedures, in a custom parameter group associated with your DB instance, set the parameter `db2_alternate_authz_behaviour` to one of the following values:

- EXTERNAL\_ROUTINE\_DBADM – Implicitly grants any user, group, or role with DBADM authority the CREATE\_EXTERNAL\_ROUTINE permission.
- EXTERNAL\_ROUTINE\_DBAUTH – Allows a user with DBADM authority to grant CREATE\_EXTERNAL\_ROUTINE permission to any user, group, or role. In this case, no user, group, or role is implicitly granted this permission, not even a user with DBADM authority.

For more information about this setting, see [GRANT \(database authorities\) statement](#) in the IBM Db2 documentation.

You can create and modify a custom parameter group by using the AWS Management Console, the AWS CLI, or the Amazon RDS API.

### Console

#### To configure the `db2_alternate_authz_behaviour` parameter in a custom parameter group

1. If you want to use a different custom DB parameter group than the one your DB instance is using, create a new DB parameter group. If you're using the bring your own license (BYOL) model, make sure that the new custom parameter group includes the IBM IDs. For information

about these IDs, see [the section called “IBM IDs for bring your own license \(BYOL\) for Db2”](#). For more information about creating a DB parameter group, see [Creating a DB parameter group in Amazon RDS](#).

2. Set the value for the db2\_alternate\_authz\_behaviour parameter in your custom parameter group. For more information about modifying a parameter group, see [Modifying parameters in a DB parameter group in Amazon RDS](#).

## AWS CLI

### To configure the db2\_alternate\_authz\_behaviour parameter in a custom parameter group

1. If you want to use a different custom DB parameter group than the one your DB instance is using, create a custom parameter group by running the [create-db-parameter-group](#) command. If you're using the bring your own license (BYOL) model, make sure that the new custom parameter group includes the IBM IDs. For information about these IDs, see [the section called “IBM IDs for bring your own license \(BYOL\) for Db2”](#).

Include the following required options:

- --db-parameter-group-name – A name for the parameter group that you are creating.
- --db-parameter-group-family – The Db2 engine edition and major version. Valid values are db2-se-11.5 and db2-ae-11.5.
- --description – A description for this parameter group.

For more information about creating a DB parameter group, see [Creating a DB parameter group in Amazon RDS](#).

The following example shows you how to create a custom parameter group named MY\_EXT\_SP\_PARAM\_GROUP for the parameter group family db2-se-11.5.

For Linux, macOS, or Unix:

```
aws rds create-db-parameter-group \
--region us-east-1 \
--db-parameter-group-name MY_EXT_SP_PARAM_GROUP \
--db-parameter-group-family db2-se-11.5 \
--description "test db2 external routines"
```

## For Windows:

```
aws rds create-db-parameter-group ^
--region us-east-1 ^
--db-parameter-group-name MY_EXT_SP_PARAM_GROUP ^
--db-parameter-group-family db2-se-11.5 ^
--description "test db2 external routines"
```

2. Modify the `db2_alternate_authz_behaviour` parameter in your custom parameter group by running the [modify-db-parameter-group](#) command.

Include the following required options:

- `--db-parameter-group-name` – The name of the parameter group that you created.
- `--parameters` – An array of parameter names, values, and the application methods for the parameter update.

For more information about modifying a parameter group, see [Modifying parameters in a DB parameter group in Amazon RDS](#).

The following example shows you how to modify the parameter group `MY_EXT_SP_PARAM_GROUP` by setting the value of `db2_alternate_authz_behaviour` to `EXTERNAL_ROUTINE_DBADM`.

For Linux, macOS, or Unix:

```
aws rds modify-db-parameter-group \
--db-parameter-group-name MY_EXT_SP_PARAM_GROUP \
--parameters
"ParameterName='db2_alternate_authz_behaviour',ParameterValue='EXTERNAL_ROUTINE_DBADM',Appl
```

## For Windows:

```
aws rds modify-db-parameter-group ^
--db-parameter-group-name MY_EXT_SP_PARAM_GROUP ^
--parameters
"ParameterName='db2_alternate_authz_behaviour',ParameterValue='EXTERNAL_ROUTINE_DBADM',Appl
```

## RDS API

### To configure the db2\_alternate\_authz\_behaviour parameter in a custom parameter group

1. If you want to use a different custom DB parameter group than the one your DB instance is using, create a new DB parameter group by using the Amazon RDS API [CreateDBParameterGroup](#) operation. If you're using the bring your own license (BYOL) model, make sure that the new custom parameter group includes the IBM Db2 IDs. For information about these IDs, see [the section called "IBM IDs for bring your own license \(BYOL\) for Db2"](#).

Include the following required parameters:

- DBParameterGroupName
- DBParameterGroupFamily
- Description

For more information about creating a DB parameter group, see [Creating a DB parameter group in Amazon RDS](#).

2. Modify the db2\_alternate\_authz\_behaviour parameter in your custom parameter group that you created by using the RDS API [ModifyDBParameterGroup](#) operation.

Include the following required parameters:

- DBParameterGroupName
- Parameters

For more information about modifying a parameter group, see [Modifying parameters in a DB parameter group in Amazon RDS](#).

### Step 2: Install the .jar file with your external routine

After you create your Java routine, create the .jar file and then run db2 "call sqlj.install\_jar('file:*file\_path*', *jar\_ID*)" to install it on your RDS for Db2 database.

The following example shows you how to create a Java routine and install it on an RDS for Db2 database. The example includes sample code for a simple routine that you can use to test the process. This example makes the following assumptions:

- The Java code is compiled on a server where Db2 is installed. This is a best practice because not compiling with the IBM-provided JDK can result in unexplained errors.
- The server has the RDS for Db2 database cataloged locally.

If you'd like to try out the process with the following sample code, copy it and then save it to a file named MYJAVASP.java.

```
import java.sql.*;  
public class MYJAVASP  
{  
    public static void my_JAVASP (String inparam) throws SQLException, Exception  
    {  
        try  
        {  
            // Obtain the calling context's connection details.  
            Connection myConn = DriverManager.getConnection("jdbc:default:connection");  
            String myQuery = "INSERT INTO TEST.TEST_TABLE VALUES (?, CURRENT DATE)";  
            PreparedStatement myStmt = myConn.prepareStatement(myQuery);  
            myStmt.setString(1, inparam);  
            myStmt.executeUpdate();  
        }  
        catch (SQLException sql_ex)  
        {  
            throw sql_ex;  
        }  
        catch (Exception ex)  
        {  
            throw ex;  
        }  
    }  
}
```

The following command compiles the Java routine.

```
~/sqllib/java/jdk64/bin/javac MYJAVASP.java
```

The following command creates the .jar file.

```
~/sqllib/java/jdk64/bin/jar cvf MYJAVASP.jar MYJAVASP.class
```

The following commands connect to the database named MY\_DB2\_DATABASE and install the .jar file.

```
db2 "connect to MY_DB2_DATABASE user master_username using master_password"
```

```
db2 "call sqlj.install_jar('file:/tmp/MYJAVASP.jar', 'MYJAVASP')"
```

```
db2 "call sqlj.refresh_classes()"
```

### Step 3: Register the external stored procedure

After you install the .jar file on your RDS for Db2 database, register it as a stored procedure by running the db2 CREATE PROCEDURE or db2 REPLACE PROCEDURE command.

The following example shows you how to connect to the database and register the Java routine created in the previous step as a stored procedure.

```
db2 "connect to MY_DB2_DATABASE user master_username using master_password"
```

```
create procedure TESTSP.MYJAVASP (in input char(6))
specific myjavasp
dynamic result sets 0
deterministic
language java
parameter style java
no dbinfo
fenced
threadsafe
modifies sql data
program type sub
external name 'MYJAVASP!my_JAVASP';
```

### Step 4: Validate the external stored procedure

Use the following steps to test the sample external stored procedure that was registered in the previous step.

#### To validate the external stored procedure

1. Create a table like TEST.TEST\_TABLE in the following example.

```
db2 "create table TEST.TEST_TABLE(C1 char(6), C2 date)"
```

2. Call the new external stored procedure. The call returns a status of 0.

```
db2 "call TESTSP.MYJAVASP('test')"  
Return Status = 0
```

3. Query the table you created in step 1 to verify the results of the stored procedure call.

```
db2 "SELECT * from TEST.TEST_TABLE"
```

The query produces output similar to the following example:

C1	C2
-----	-----
test	02/05/2024

# Known issues and limitations for Amazon RDS for Db2

The following items are known issues and limitations for working with Amazon RDS for Db2:

## Topics

- [Authentication limitation](#)
- [Non-fenced routines](#)
- [Non-automatic storage tablespaces during migration](#)
- [Setting the db2\\_compatibility\\_vector parameter](#)
- [Migrating databases that contain INVALID packages](#)

## Authentication limitation

Amazon RDS sets db2auth to JCC\_ENFORCE\_SECMEC by default. However, if you don't want to enforce userid and password encryption over the wire, you can override this setting by changing the db2auth parameter to CLEAR\_TEXT in the parameter group. For more information, see [Modifying parameters in a DB parameter group in Amazon RDS](#).

## Non-fenced routines

RDS for Db2 doesn't support the creation of non-fenced routines and the migration of these routines by backing up and restoring data. To check if your database contains any non-fenced routines, run the following SQL command:

```
SELECT 'COUNT:' || count(*) FROM SYSCAT.ROUTINES where fenced='N' and routineschema not in ('SQLJ','SYSCAT','SYSFUN','SYSIBM','SYSIBMADM','SYSPROC','SYSTOOLS')
```

## Non-automatic storage tablespaces during migration

RDS for Db2 doesn't support the creation of new non-automatic storage tablespaces. When you use native restore for a one-time migration of your database, RDS for Db2 automatically converts your non-automatic storage tablespaces to automatic ones, and then restores your database to RDS for Db2. For information about one-time migrations, see [Migrating from Linux to Linux for Amazon RDS for Db2](#) and [Migrating from AIX or Windows to Linux for Amazon RDS for Db2](#).

## Setting the db2\_compatibility\_vector parameter

With Amazon RDS, you can create an initial database when you create the DB instance and then modify parameters in an associated parameter group. However, for Db2, if you want to set the db2\_compatibility\_vector parameter in a parameter group, you must first modify the parameter in a custom parameter group, create the DB instance without a database, and then create a database using the rdsadmin.create\_database stored procedure.

### To set the db2\_compatibility\_vector parameter

1. [Create a custom parameter group](#). (You can't modify parameters in default parameter groups.)
2. [Modify the parameter](#).
3. [Create a DB instance](#).
4. [Create a database](#) using the rdsadmin.create\_database stored procedure.
5. [Associate the parameter group](#) with the DB instance that contains the database.

## Migrating databases that contain INVALID packages

If you migrate Db2 databases that contain INVALID packages to RDS for Db2 by using the RESTORE command, you could encounter issues when you start to use the databases. INVALID packages can cause issues because of the authorization setup for the DB instance user rdsdb and the removal of authorization from PUBLIC. INVALID packages cause the following commands to fail:

- db2updv115
- db2 "call SYSPROC.ADMIN\_REVALIDATE\_DB\_OBJECTS()"

Before migrating your database with the RESTORE command, ensure that your database doesn't contain INVALID packages by running the following command:

```
db2 "SELECT 'COUNT:' || count(*) FROM SYSCAT.INVALIDOBJECTS"
```

If the command returns a count greater than zero, then call the following command:

```
db2 "call SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS()"
```

Afterwards, call the previous command to confirm that your database no longer contains INVALID packages.

```
db2 "SELECT 'COUNT:' || count(*) FROM SYSCAT.INVALIDOBJECTS"
```

Now you are ready to make a backup of your database and restore it to your RDS for Db2 DB instance.

# Amazon RDS for Db2 stored procedure reference

You can manage your Amazon RDS for Db2 DB instances running the Db2 engine by calling built-in stored procedures.

Stored procedure	Category	Description
<a href="#"><u>the section called "rdsadmin.activate_database"</u></a>	Databases	Use the <code>rdsadmin.activate_database</code> stored procedure to activate a database on a standalone RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.add_groups"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.add_groups</code> stored procedure to add one or more groups to a user for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.add_user"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.add_user</code> stored procedure to add a user to an authorization list for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.alter_bufferpool"</u></a>	Buffer pools	Use the <code>rdsadmin.alter_bufferpool</code> stored procedure to modify a buffer pool for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.alter_tablespace"</u></a>	Tablespaces	Use the <code>rdsadmin.alter_tablespace</code> stored procedure to modify a tablespace for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.backup_database"</u></a>	Database	Use the <code>rdsadmin.backup_database</code> stored procedure to back up a database on an RDS for Db2 DB instance to an Amazon S3 bucket. Then you can restore the backup from Amazon S3 to an RDS for Db2 DB instance or to another location such as a local server.
<a href="#"><u>the section called "rdsadmin.catalog_storage_access"</u></a>	Storage access	Use the <code>rdsadmin.catalog_storage_access</code> stored procedure to catalog a storage alias for accessing an Amazon S3 bucket with

Stored procedure	Category	Description
		Db2 data files for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.change_password"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.change_password</code> stored procedure to change a user's password for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.complete_rollforward"</u></a>	Databases	Use the <code>rdsadmin.complete_rollforward</code> stored procedure to bring a database on an RDS for Db2 DB instance online from a ROLL-FORWARD PENDING state. A ROLL-FORWARD PENDING state occurs when you called <a href="#"><u>the section called "rdsadmin.rollforward_database"</u></a> but set the <code>complete_rollforward</code> parameter to FALSE.
<a href="#"><u>the section called "rdsadmin.configure_db_audit"</u></a>	Audit policies	Use the <code>rdsadmin.configure_db_audit</code> stored procedure to modify an audit policy for a database on an RDS for Db2 DB instance. If no audit policy exists, running this stored procedure creates an audit policy.
<a href="#"><u>the section called "rdsadmin.create_bufferpool"</u></a>	Buffer pools	Use the <code>rdsadmin.create_bufferpool</code> stored procedure to create a buffer pool for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.create_database"</u></a>	Databases	Use the <code>rdsadmin.create_database</code> stored procedure to create a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.create_role"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.create_role</code> stored procedure to create a role to attach to a database on an RDS for Db2 DB instance.

Stored procedure	Category	Description
<a href="#"><u>the section called "rdsadmin.create_tablespace"</u></a>	Tablespaces	Use the <code>rdsadmin.create_tablespace</code> stored procedure to create a tablespace for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.db2pd_command"</u></a>	Databases	Use the <code>rdsadmin.db2pd_command</code> stored procedure collect information about a database on an RDS for Db2 DB instance. This information can help with monitoring and troubleshooting databases in RDS for Db2.
<a href="#"><u>the section called "rdsadmin.bbadm_grant"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.bbadm_grant</code> stored procedure to grant one or more authorization types ( <code>DBADM</code> , <code>ACCESSCTRL</code> , or <code>DATAACCESS</code> ) to one or more roles, users, or groups for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.bbadm_revoke"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.bbadm_revoke</code> stored procedure to revoke one or more authorization types ( <code>DBADM</code> , <code>ACCESSCTRL</code> , or <code>DATAACCESS</code> ) from one or more roles, users, or groups for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.deactivate_database"</u></a>	Databases	Use the <code>rdsadmin.deactivate_database</code> stored procedure to deactivate a database on an RDS for Db2 DB instance. You can deactivate databases to conserve memory resources.
<a href="#"><u>the section called "rdsadmin.disable_db_audit"</u></a>	Audit policies	Use the <code>rdsadmin.disable_db_audit</code> stored procedure to stop audit logging and remove an audit policy from a database on an RDS for Db2 DB instance.

Stored procedure	Category	Description
<a href="#"><u>the section called "rdsadmin.drop_bufferpool"</u></a>	Buffer pools	Use the <code>rdsadmin.drop_bufferpool</code> stored procedure to drop a buffer pool from a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.drop_database"</u></a>	Databases	Use the <code>rdsadmin.drop_database</code> stored procedure to drop a database from an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.drop_role"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.drop_role</code> stored procedure to delete a role from a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.drop_tablespace"</u></a>	Tablespaces	Use the <code>rdsadmin.drop_tablespace</code> stored procedure to drop a tablespace from a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.fgac_command"</u></a>	Databases	Use the <code>rdsadmin.fgac_command</code> stored procedure to control access at the row or column level to table data in your database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.force_application"</u></a>	Databases	Use the <code>rdsadmin.force_application</code> stored procedure to force applications off of a database on an RDS for Db2 DB instance to perform maintenance.
<a href="#"><u>the section called "rdsadmin.grant_role"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.grant_role</code> stored procedure to assign a role to a grantee role, user, or group in a database on an RDS for Db2 DB instance. You can also use this stored procedure to give the grantee role DBADM authorization to assign roles.

Stored procedure	Category	Description
<a href="#"><u>the section called "rdsadmin.list_archive_log_information"</u></a>	Databases	Use the <code>rdsadmin.list_archive_log_i</code> nformation stored procedure to return information about archive logs for a database on an RDS for Db2 DB instance. This information includes details such as size and creation date of individual log files, and the total storage used by the archive log files.
<a href="#"><u>the section called "rdsadmin.list_sid_group_mapping"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.list_sid_group_map</code> ping stored procedure to return a list of all security ID (SID) and Active Directory group mappings configured on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.list_users"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.list_users</code> stored procedure to return a list of users on an authorization list for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.reactivate_database"</u></a>	Databases	Use the <code>rdsadmin/reactivate_database</code> stored procedure to reactivate a database on an RDS for Db2 DB instance after you make database configuration changes. For a database on a standalone DB instance, you can use either this stored procedure or the <a href="#"><u>rdsadmin.activate_database</u></a> stored procedure. For a database on a replica source DB instance, you must use the <code>rdsadmin/reactivate_database</code> stored procedure.
<a href="#"><u>the section called "rdsadmin.remove_groups"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.remove_groups</code> stored procedure to remove one or more groups from a user for a database on an RDS for Db2 DB instance.

Stored procedure	Category	Description
<a href="#"><u>the section called "rdsadmin.remove_sid_group_mapping"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.remove_sid_group_mapping</code> stored procedure to remove a security ID (SID) and its corresponding Active Directory group mapping from an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.remove_user"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.remove_user</code> stored procedure to remove a user from an authorization list for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.rename_tablespace"</u></a>	Tablespaces	Use the <code>rdsadmin.rename_tablespace</code> stored procedure to rename a tablespace for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.restore_database"</u></a>	Databases	Use the <code>rdsadmin.restore_database</code> stored procedure to restore a database on an RDS for Db2 DB instance from an Amazon S3 bucket.
<a href="#"><u>the section called "rdsadmin.revoke_role"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.revoke_role</code> stored procedure to revoke a role from a grantee role, user, or group in a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.rollforward_database"</u></a>	Databases	Use the <code>rdsadmin.rollforward_database</code> stored procedure to bring a database on an RDS for Db2 DB instance online, and to apply transaction logs after you restored a database on an RDS for Db2 DB instance by calling <a href="#"><u>the section called "rdsadmin.restore_database"</u></a> .

Stored procedure	Category	Description
<a href="#"><u>the section called "rdsadmin.rollforward_status"</u></a>	Databases	Use the <code>rdsadmin.rollforward_status</code> stored procedure to query the rollforward status of calling the <a href="#"><u>the section called "rdsadmin.rollforward_database"</u></a> or <a href="#"><u>the section called "rdsadmin.complete_rollforward"</u></a> stored procedure on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.set_archive_log_retention"</u></a>	Databases	Use the <code>rdsadmin.set_archive_log_retention</code> stored procedure to configure how long to retain archive log files for a database on an RDS for Db2 DB instance. You can also use this stored procedure to disable archive log retention.
<a href="#"><u>the section called "rdsadmin.set_configuration"</u></a>	Databases	Use the <code>rdsadmin.set_configuration</code> stored procedure to configure certain settings for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.set_sid_group_mapping"</u></a>	Granting and revoking privileges	Use the <code>rdsadmin.set_sid_group_mapping</code> stored procedure to create a mapping between a security ID (SID) and the corresponding Active Directory group on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.show_archive_log_retention"</u></a>	Databases	Use the <code>rdsadmin.show_archive_log_retention</code> stored procedure to return the current archive log retention setting for a database on an RDS for Db2 DB instance.
<a href="#"><u>the section called "rdsadmin.show_configuration"</u></a>	Databases	Use the <code>rdsadmin.show_configuration</code> stored procedure to return one or more settings that are modifiable for a database on an RDS for Db2 DB instance.

Stored procedure	Category	Description
<a href="#"><u>the section called "rdsadmin.uncatalog_storage_access"</u></a>	Storage access	Use the <code>rdsadmin.uncatalog_storage_access</code> stored procedure to remove a storage alias for accessing an Amazon S3 bucket with Db2 data files.
<a href="#"><u>the section called "rdsadmin.update_db_param"</u></a>	Databases	Use the <code>rdsadmin.update_db_param</code> stored procedure to update database parameters for a database on an RDS for Db2 DB instance.

## Topics

- [Considerations for Amazon RDS for Db2 stored procedures](#)
- [Stored procedures for granting and revoking privileges for RDS for Db2](#)
- [Stored procedures for audit policies for RDS for Db2](#)
- [Stored procedures for buffer pools for RDS for Db2](#)
- [Stored procedures for databases for RDS for Db2](#)
- [Stored procedures for storage access for RDS for Db2](#)
- [Stored procedures for tablespaces for RDS for Db2](#)

## Considerations for Amazon RDS for Db2 stored procedures

Before using the Amazon RDS system stored procedures for RDS for Db2 DB instances running the Db2 engine, review the following information:

- Before running the stored procedures, you must first connect to the `rdsadmin` database as the master user for your RDS for Db2 DB instance. In the following example, replace `master_username` and `master_password` with your own information.

```
db2 "connect to rdsadmin user master_username using master_password"
```

- The stored procedures return the `ERR_MESSAGE` parameter, which indicates whether the stored procedure ran successfully or not and why it didn't run successfully.

### Examples

The following example indicates that the stored procedure ran successfully.

```
Parameter Name : ERR_MESSAGE  
Parameter Value : -  
Return Status = 0
```

The following example indicates that the stored procedure didn't run successfully because the Amazon S3 bucket name used in the stored procedure wasn't valid.

```
Parameter Name : ERR_MESSAGE  
Parameter Value : Invalid S3 bucket name  
Return Status = -1006
```

For error messages returned when calling stored procedures, see [the section called "Stored procedure errors".](#)

For information about checking the status of a stored procedure, see [rdsadmin.get\\_task\\_status](#).

# Stored procedures for granting and revoking privileges for RDS for Db2

The built-in stored procedures described in this topic manage users, roles, groups, and authorization for Amazon RDS for Db2 databases. To run these procedures, the master user must first connect to the `rdsadmin` database.

For tasks that use these stored procedures, see [Granting and revoking privileges](#) and [Setting up Kerberos authentication](#).

Refer to the following built-in stored procedures for information about their syntax, parameters, usage notes, and examples.

## Stored procedures

- [`rdsadmin.create\_role`](#)
- [`rdsadmin.grant\_role`](#)
- [`rdsadmin.revoke\_role`](#)
- [`rdsadmin.drop\_role`](#)
- [`rdsadmin.add\_user`](#)
- [`rdsadmin.change\_password`](#)
- [`rdsadmin.list\_users`](#)
- [`rdsadmin.remove\_user`](#)
- [`rdsadmin.add\_groups`](#)
- [`rdsadmin.remove\_groups`](#)
- [`rdsadmin.dbadm\_grant`](#)
- [`rdsadmin.dbadm\_revoke`](#)
- [`rdsadmin.set\_sid\_group\_mapping`](#)
- [`rdsadmin.list\_sid\_group\_mapping`](#)
- [`rdsadmin.remove\_sid\_group\_mapping`](#)

## `rdsadmin.create_role`

Creates a role.

### Syntax

```
db2 "call rdsadmin.create_role(
```

```
'database_name' ,  
'role_name' )"
```

## Parameters

The following parameters are required:

*database\_name*

The name of the database the command will run on. The data type is varchar.

*role\_name*

The name of the role that you want to create. The data type is varchar.

## Usage notes

For information about checking the status of creating a role, see [rdsadmin.get\\_task\\_status](#).

## Examples

The following example creates a role called MY\_ROLE for database DB2DB.

```
db2 "call rdsadmin.create_role(  
      'DB2DB',  
      'MY_ROLE')"
```

## rdsadmin.grant\_role

Assigns a role to a role, user, or group.

## Syntax

```
db2 "call rdsadmin.grant_role(  
      ?,  
      'database_name',  
      'role_name',  
      'grantee',  
      'admin_option')"
```

## Parameters

The following output parameter is required:

?

A parameter marker that outputs the unique identifier for the task. This parameter only accepts ?.

The following input parameters are required:

*database\_name*

The name of the database the command will run on. The data type is varchar.

*role\_name*

The name of the role that you want to create. The data type is varchar.

*grantee*

The role, user, or group to receive authorization. The data type is varchar. Valid values: ROLE, USER, GROUP, PUBLIC.

Format must be value followed by name. Separate multiple values and names with commas.

Example: 'USER *user1*, *user2*, GROUP *group1*, *group2*'. Replace the names with your own information.

The following input parameter is optional:

*admin\_option*

Specifies whether the grantee ROLE has DBADM authorization to assign roles. The data type is char. The default is N.

## Usage notes

For information about checking the status of assigning a role, see [rdsadmin.get\\_task\\_status](#).

## Examples

### Example 1: Assigning role to role, user, and group, and granting authorization

The following example assigns a role called ROLE\_TEST for database TESTDB to the role called role1, the user called user1, and the group called group1. ROLE\_TEST is given admin authorization to assign roles.

```
db2 "call rdsadmin.grant_role(
?, 
'TESTDB',
'ROLE_TEST',
'ROLE role1, USER user1, GROUP group1',
'Y')"
```

## Example 2: Assigning role to PUBLIC and not granting authorization

The following example assigns a role called ROLE\_TEST for database TESTDB to PUBLIC. ROLE\_TEST isn't given admin authorization to assign roles.

```
db2 "call rdsadmin.grant_role(
?, 
'TESTDB',
'ROLE_TEST',
'PUBLIC')"
```

## rdsadmin.revoke\_role

Revokes a role from a role, user, or group.

### Syntax

```
db2 "call rdsadmin.revoke_role(
?, 
'database_name',
'role_name',
'grantee')"
```

### Parameters

The following output parameter is required:

?

A parameter marker that outputs the unique identifier for the task. This parameter only accepts ?.

The following input parameters are required:

## *database\_name*

The name of the database the command will run on. The data type is varchar.

## *role\_name*

The name of the role that you want to revoke. The data type is varchar.

## *grantee*

The role, user, or group to lose authorization. The data type is varchar. Valid values: ROLE, USER, GROUP, PUBLIC.

Format must be value followed by name. Separate multiple values and names with commas.

Example: 'USER *user1*, *user2*, GROUP *group1*, *group2*'. Replace the names with your own information.

## Usage notes

For information about checking the status of revoking a role, see [rdsadmin.get\\_task\\_status](#).

## Examples

### Example 1: Revoking role from role, user, and group

The following example revokes a role called ROLE\_TEST for database TESTDB from the role called role1, the user called user1, and the group called group1.

```
db2 "call rdsadmin.revoke_role(
?, 
'TESTDB',
'ROLE_TEST',
'ROLE role1, USER user1, GROUP group1')"
```

### Example 2: Revoking role from PUBLIC

The following example revokes a role called ROLE\_TEST for database TESTDB from PUBLIC.

```
db2 "call rdsadmin.revoke_role(
?, 
'TESTDB',
'ROLE_TEST',
'PUBLIC')"
```

## rdsadmin.drop\_role

Drops a role.

### Syntax

```
db2 "call rdsadmin.drop_role(
?, 
'database_name',
'role_name')"
```

### Parameters

The following output parameter is required:

?

A parameter marker that outputs the unique identifier for the task. This parameter only accepts ?.

The following input parameters are required:

*database\_name*

The name of the database the command will run on. The data type is varchar.

*role\_name*

The name of the role that you want to drop. The data type is varchar.

### Usage notes

For information about checking the status of dropping a role, see [rdsadmin.get\\_task\\_status](#).

### Examples

The following example drops a role called ROLE\_TEST for database TESTDB.

```
db2 "call rdsadmin.drop_role(
?, 
'TESTDB',
'ROLE_TEST')"
```

## rdsadmin.add\_user

Adds a user to an authorization list.

### Syntax

```
db2 "call rdsadmin.add_user(  
      'username',  
      'password',  
      'group_name,group_name')"
```

### Parameters

The following parameters are required:

*username*

A user's username. The data type is varchar.

*password*

A user's password. The data type is varchar.

The following parameter is optional:

*group\_name*

The name of a group that you want to add the user to. The data type is varchar. The default is an empty string or null.

### Usage notes

You can add a user to one or more groups by separating the group names with commas.

You can create a group when you create a new user, or when you [add a group to an existing user](#). You can't create a group by itself.

#### Note

The maximum number of users that you can add by calling `rdsadmin.add_user` is 5,000.

For information about checking the status of adding a user, see [rdsadmin.get\\_task\\_status](#).

## Examples

The following example creates a user called `jorge_souza` and assigns the user to the groups called `sales` and `inside_sales`.

```
db2 "call rdsadmin.add_user(
    'jorge_souza',
    '*****',
    'sales,inside_sales')"
```

## rdsadmin.change\_password

Changes a user's password.

### Syntax

```
db2 "call rdsadmin.change_password(
    'username',
    'new_password')"
```

### Parameters

The following parameters are required:

*username*

A user's username. The data type is varchar.

*new\_password*

A new password for the user. The data type is varchar.

### Usage notes

For information about checking the status of changing a password, see [rdsadmin.get\\_task\\_status](#).

## Examples

The following example changes the password for `jorge_souza`.

```
db2 "call rdsadmin.change_password(  
    'jorge_souza',  
    '*****')"
```

## rdsadmin.list\_users

Lists users on an authorization list.

### Syntax

```
db2 "call rdsadmin.list_users()"
```

### Usage notes

For information about checking the status of listing users, see [rdsadmin.get\\_task\\_status](#).

## rdsadmin.remove\_user

Removes user from authorization list.

### Syntax

```
db2 "call rdsadmin.remove_user('username')"
```

### Parameters

The following parameter is required:

*username*

A user's username. The data type is varchar.

### Usage notes

For information about checking the status of removing a user, see [rdsadmin.get\\_task\\_status](#).

### Examples

The following example removes jorge\_souza from being able to access databases in RDS for Db2 DB instances.

```
db2 "call rdsadmin.remove_user('jorge_souza')"
```

## rdsadmin.add\_groups

Adds groups to a user.

### Syntax

```
db2 "call rdsadmin.add_groups(  
      'username',  
      'group_name,group_name' )"
```

### Parameters

The following parameters are required:

*username*

A user's username. The data type is varchar.

*group\_name*

The name of a group that you want to add the user to. The data type is varchar. The default is an empty string.

### Usage notes

You can add one or more groups to a user by separating the group names with commas. For information about checking the status of adding groups, see [rdsadmin.get\\_task\\_status](#).

### Examples

The following example adds the `direct_sales` and `b2b_sales` groups to user `jorge_souza`.

```
db2 "call rdsadmin.add_groups(  
      'jorge_souza',  
      'direct_sales,b2b_sales' )"
```

## rdsadmin.remove\_groups

Removes groups from a user.

## Syntax

```
db2 "call rdsadmin.remove_groups(  
      'username',  
      'group_name,group_name')"
```

## Parameters

The following parameters are required:

*username*

A user's username. The data type is varchar.

*group\_name*

The name of a group that you want to remove the user from. The data type is varchar.

## Usage notes

You can remove one or more groups from a user by separating the group names with commas.

For information about checking the status of removing groups, see [rdsadmin.get\\_task\\_status](#).

## Examples

The following example removes the `direct_sales` and `b2b_sales` groups from user `jorge_souza`.

```
db2 "call rdsadmin.remove_groups(  
      'jorge_souza',  
      'direct_sales,b2b_sales')"
```

## rdsadmin.dbadm\_grant

Grants DBADM, ACCESSCTRL, or DATAACCESS authorization to a role, user, or group.

## Syntax

```
db2 "call rdsadmin.dbadm_grant(  
      ?,
```

```
'database_name',  
'authorization',  
'grantee')"
```

## Parameters

The following output parameter is required:

?

A parameter marker that outputs the unique identifier for the task. This parameter only accepts ?.

The following input parameters are required:

*database\_name*

The name of the database the command will run on. The data type is varchar.

*authorization*

The type of authorization to grant. The data type is varchar. Valid values: DBADM, ACCESSCTRL, DATAACCESS.

Separate multiple types with commas.

*grantee*

The role, user, or group to receive authorization. The data type is varchar. Valid values: ROLE, USER, GROUP.

Format must be value followed by name. Separate multiple values and names with commas. Example: 'USER *user1*, *user2*, GROUP *group1*, *group2*'. Replace the names with your own information.

## Usage notes

The role to receive access must exist.

For information about checking the status of granting database admin access, see [rdsadmin.get\\_task\\_status](#).

## Examples

### Example 1: Granting database admin access to role

The following example grants database admin access to the database named TESTDB for the role ROLE\_DBA.

```
db2 "call rdsadmin.dbadm_grant(
?, 
'TESTDB',
'DBADM',
'ROLE ROLE_DBA')"
```

### Example 2: Granting database admin access to user and group

The following example grants database admin access to the database named TESTDB for user1 and group1.

```
db2 "call rdsadmin.dbadm_grant(
?, 
'TESTDB',
'DBADM',
'USER user1, GROUP group1')"
```

### Example 3: Granting database admin access to multiple users and groups

The following example grants database admin access to the database named TESTDB for user1, user2, group1, and group2.

```
db2 "call rdsadmin.dbadm_grant(
?, 
'TESTDB',
'DBADM',
'USER user1, user2, GROUP group1, group2')"
```

## rdsadmin.dbadm\_revoke

Revokes DBADM, ACCESSCTRL, or DATAACCESS authorization from a role, user, or group.

### Syntax

```
db2 "call rdsadmin.dbadm_revoke(
```

```
?,
'database_name',
'authorization',
'grantee')
```

## Parameters

The following output parameter is required:

?

The unique identifier for the task. This parameter only accepts ?.

The following input parameters are required:

*database\_name*

The name of the database the command will run on. The data type is varchar.

*authorization*

The type of authorization to revoke. The data type is varchar. Valid values: DBADM, ACCESSCTRL, DATAACCESS.

Separate multiple types with commas.

*grantee*

The role, user, or group to revoke authorization from. The data type is varchar. Valid values: ROLE, USER, GROUP.

Format must be value followed by name. Separate multiple values and names with commas.

Example: 'USER *user1*, *user2*, GROUP *group1*, *group2*'. Replace the names with your own information.

## Usage notes

For information about checking the status of revoking database admin access, see [rdsadmin.get\\_task\\_status](#).

## Examples

### Example 1: Revoking database admin access from role

The following example revokes database admin access to the database named TESTDB for the role ROLE\_DBA.

```
db2 "call rdsadmin.dbadm_revoke(
?, 
'TESTDB',
'DBADM',
'ROLE ROLE_DBA')"
```

### Example 2: Revoking database admin access from user and group

The following example revokes database admin access to the database named TESTDB for user1 and group1.

```
db2 "call rdsadmin.dbadm_revoke(
?, 
'TESTDB',
'DBADM',
'USER user1, GROUP group1')"
```

### Example 3: Revoking database admin access from multiple users and groups

The following example revokes database admin access to the database named TESTDB for user1, user2, group1, and group2.

```
db2 "call rdsadmin.dbadm_revoke(
?, 
'TESTDB',
'DBADM',
'USER user1, user2, GROUP group1, group2')"
```

## rdsadmin.set\_sid\_group\_mapping

Creates a mapping between a security ID (SID) and the corresponding Active Directory group.

### Syntax

```
db2 "call rdsadmin.set_sid_group_mapping(
?, 
'SID',
```

```
'group_name')"
```

## Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameters are required:

*SID*

The security ID (SID). The data type is varchar.

*group\_name*

The name of the Active Directory group to map to the SID. The data type is varchar.

## Usage notes

Use this stored procedure to enable Kerberos authentication with Active Directory groups. If the SID or group\_name already exists in the mapping, this stored procedure fails.

For information about how to find the SID for a group, see [Step 8: Retrieve the Active Directory group SID in PowerShell](#).

For information about checking the status of creating a mapping, see [rdsadmin.get\\_task\\_status](#).

## Examples

The following example maps a SID to a group called my\_group.

```
db2 "call rdsadmin.set_sid_group_mapping(
    ?,
    'S-1-5-21-9146495592-531070549-834388463-513',
    'my_group')"
```

## rdsadmin.list\_sid\_group\_mapping

Lists all security ID (SID) and Active Directory group mappings configured on the DB instance.

## Syntax

```
db2 "call rdsadmin.list_sid_group_mapping()"
```

## Usage notes

For information about checking the status of listing mappings, see [rdsadmin.get\\_task\\_status](#).

## rdsadmin.remove\_sid\_group\_mapping

Removes a security ID (SID) and its corresponding Active Directory group mapping from a DB instance.

## Syntax

```
db2 "call rdsadmin.remove_sid_group_mapping(
    ?,
    'SID')"
```

## Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameter is required:

*SID*

The security ID (SID). The data type is varchar.

## Usage notes

For information about how to find the SID for a group, see [Step 8: Retrieve the Active Directory group SID in PowerShell](#).

For information about checking the status of removing mappings, see [rdsadmin.get\\_task\\_status](#).

## Examples

The following example removes a SID mapping from the group it was mapped to.

```
db2 "call rdsadmin.remove_sid_group_mapping(  
?,  
'S-1-5-21-9146495592-531070549-834388463-513')"
```

## Stored procedures for audit policies for RDS for Db2

The built-in stored procedures described in this topic manage audit policies for Amazon RDS for Db2 databases that use audit logging. For more information, see [the section called “Db2 audit logging”](#). To run these procedures, the master user must first connect to the `rdsadmin` database.

Refer to the following built-in stored procedures for information about their syntax, parameters, usage notes, and examples.

### Stored procedures

- [`rdsadmin.configure\_db\_audit`](#)
- [`rdsadmin.disable\_db\_audit`](#)

#### `rdsadmin.configure_db_audit`

Configures the audit policy for the RDS for Db2 database specified by `db_name`. If the policy you're configuring doesn't exist, calling this stored procedure creates it. If this policy does exist, calling this stored procedure modifies it with the parameter values that you provide.

#### Syntax

```
db2 "call rdsadmin.configure_db_audit(  
      'db_name',  
      'category',  
      'category_setting',  
      '?')"
```

#### Parameters

The following parameters are required.

##### `db_name`

The DB name of the RDS for Db2 database to configure the audit policy for. The data type is `varchar`.

##### `category`

The name of the category to configure this audit policy for. The data type is `varchar`. The following are valid values for this parameter:

- ALL – With ALL, Amazon RDS doesn't include the CONTEXT, EXECUTE, or ERROR categories.
- AUDIT
- CHECKING
- CONTEXT
- ERROR
- EXECUTE – You can configure this category with data or without data. With data means to also log input data values provided for any host variables and parameter markers. The default is without data. For more information, see the description of the *category\_setting* parameter and the [the section called “Examples”](#).
- OBJMAINT
- SECMAINT
- SYSADMIN
- VALIDATE

For more information about these categories, see the [IBM Db2 documentation](#).

### *category\_setting*

The setting for the specified audit category. The data type is varchar.

The following table shows the valid category setting values for each category.

Category	Valid category settings
ALL	BOTH FAILURE SUCCESS NONE
AUDIT	
CHECKING	
CONTEXT	
OBJMAINT	
SECMAINT	
SYSADMIN	

Category	Valid category settings
VALIDATE	
ERROR	AUDIT   NORMAL . The default is NORMAL.
EXECUTE	BOTH, WITH BOTH, WITHOUT FAILURE, WITH FAILURE, WITHOUT SUCCESS, WITH SUCCESS, WITHOUT NONE

## Usage notes

Before you call `rdsadmin.configure_db_audit`, make sure the RDS for Db2 DB instance with the database you're configuring the audit policy for is associated with an option group that has the DB2\_AUDIT option. For more information, see [the section called "Setting up Db2 audit logging"](#).

After you configure the audit policy, you can check the status of the audit configuration for the database by following the steps in [Check the audit configuration](#).

Specifying ALL for the category parameter doesn't include the CONTEXT, EXECUTE, or ERROR categories. To add these categories to your audit policy, call `rdsadmin.configure_db_audit` separately with each category that you want to add. For more information, see [the section called "Examples"](#).

## Examples

The following examples create or modify the audit policy for a database named TESTDB. In examples 1 through 5, if the ERROR category wasn't previously configured, this category is set to NORMAL (the default). To change that setting to AUDIT, follow [Example 6: Specifying the ERROR category](#).

### Example 1: Specifying the ALL category

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'ALL', 'BOTH', ?)"
```

In the example, the call configures the AUDIT, CHECKING, OBJMAINT, SECMAINT, SYSADMIN, and VALIDATE categories in the audit policy. Specifying BOTH means that both successful and failing events will be audited for each of these categories.

## Example 2: Specifying the EXECUTE category with data

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'EXECUTE', 'SUCCESS,WITH', ?)"
```

In the example, the call configures the EXECUTE category in the audit policy. Specifying SUCCESS,WITH means that logs for this category will include only successful events, and will include input data values provided for host variables and parameter markers.

## Example 3: Specifying the EXECUTE category without data

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'EXECUTE', 'FAILURE,WITHOUT', ?)"
```

In the example, the call configures the EXECUTE category in the audit policy. Specifying FAILURE,WITHOUT means that logs for this category will include only failing events, and won't include input data values provided for host variables and parameter markers.

## Example 4: Specifying the EXECUTE category without status events

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'EXECUTE', 'NONE', ?)"
```

In the example, the call configures the EXECUTE category in the audit policy. Specifying NONE means that no events in this category will be audited.

## Example 5: Specifying the OBJMAINT category

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'OBJMAINT', 'NONE', ?)"
```

In the example, the call configures the OBJMAINT category in the audit policy. Specifying NONE means that no events in this category will be audited.

## Example 6: Specifying the ERROR category

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'ERROR', 'AUDIT', ?)"
```

In the example, the call configures the ERROR category in the audit policy. Specifying AUDIT means that all errors, including errors occurring within audit logging itself, are captured in the logs. The default error type is NORMAL. With NORMAL, errors generated by the audit are ignored and only the SQLCODEs for errors associated with the operation being performed are captured.

## rdsadmin.disable\_db\_audit

Stops audit logging for the RDS for Db2 database specified by *db\_name* and removes the audit policy configured for it.

### Note

This stored procedure only removes audit policies that were configured by calling [the section called "rdsadmin.configure\\_db\\_audit".](#)

## Syntax

```
db2 "call rdsadmin.disable_db_audit('db_name', ?)"
```

## Parameters

The following parameters are required.

### *db\_name*

The DB name of the RDS for Db2 database to disable audit logging for. The data type is varchar.

## Usage notes

Calling `rdsadmin.disable_db_audit` doesn't disable audit logging for the RDS for Db2 DB instance. To disable audit logging at the DB instance level, remove the option group from the DB instance. For more information, see [Disabling Db2 audit logging](#).

## Examples

The following example disables audit logging for a database named TESTDB.

```
db2 "call rdsadmin.disable_db_audit('TESTDB', ?)"
```

## Stored procedures for buffer pools for RDS for Db2

The built-in stored procedures described in this topic manage buffer pools for Amazon RDS for Db2 databases. To run these procedures, the master user must first connect to the `rdsadmin` database.

These stored procedures are used in a variety of tasks. This list isn't exhaustive.

- [Common tasks for buffer pools](#)
- [Generating performance reports](#)
- [Copying database metadata with db2look](#)
- [Creating a repository database for IBM Db2 Data Management Console](#)

Refer to the following built-in stored procedures for information about their syntax, parameters, usage notes, and examples.

### Stored procedures

- [rdsadmin.create\\_bufferpool](#)
- [rdsadmin.alter\\_bufferpool](#)
- [rdsadmin.drop\\_bufferpool](#)

### **rdsadmin.create\_bufferpool**

Creates a buffer pool.

#### Syntax

```
db2 "call rdsadmin.create_bufferpool(  
      'database_name',  
      'buffer_pool_name',  
      buffer_pool_size,  
      'immediate',  
      'automatic',  
      page_size,  
      number_block_pages,  
      block_size)"
```

#### Parameters

The following parameters are required:

## *database\_name*

The name of the database to run the command on. The data type is `varchar`.

## *buffer\_pool\_name*

The name of the buffer pool to create. The data type is `varchar`.

The following parameters are optional:

## *buffer\_pool\_size*

The size of the buffer pool in number of pages. The data type is `integer`. The default is -1.

## *immediate*

Specifies whether the command runs immediately. The data type is `char`. The default is Y.

## *automatic*

Specifies whether to set the buffer pool to automatic. The data type is `char`. The default is Y.

## *page\_size*

The page size of the buffer pool. The data type is `integer`. Valid values: 4096, 8192, 16384, 32768. The default is 8192.

## *number\_block\_pages*

The number of block pages in the buffer pools. The data type is `integer`. The default is 0.

## *block\_size*

The block size for the block pages. The data type is `integer`. Valid values: 2 to 256. The default is 32.

## Usage notes

For information about checking the status of creating a buffer pool, see [rdsadmin.get\\_task\\_status](#).

## Examples

### Example 1: Creating buffer pool with default parameters

The following example creates a buffer pool called BP8 for a database called TESTDB with default parameters, so the buffer pool uses an 8 KB page size.

```
db2 "call rdsadmin.create_bufferpool(  
    'TESTDB',  
    'BP8')"
```

### Example 2: Creating buffer pool to run immediately with automatic allocation

The following example creates a buffer pool called BP16 for a database called TESTDB that uses a 16 KB page size with an initial page count of 1,000 and is set to automatic. Db2 runs the command immediately. If you use an initial page count of -1, then Db2 will use automatic allocation of pages.

```
db2 "call rdsadmin.create_bufferpool(  
    'TESTDB',  
    'BP16',  
    1000,  
    'Y',  
    'Y',  
    16384)"
```

### Example 3: Creating buffer pool to run immediately using block pages

The following example creates a buffer pool called BP16 for a database called TESTDB. This buffer pool has a 16 KB page size with an initial page count of 10,000. Db2 runs the command immediately using 500 block pages with a block size of 512.

```
db2 "call rdsadmin.create_bufferpool(  
    'TESTDB',  
    'BP16',  
    10000,  
    'Y',  
    'Y',  
    16384,  
    500,  
    512)"
```

## rdsadmin.alter\_bufferpool

Alters a buffer pool.

## Syntax

```
db2 "call rdsadmin.alter_bufferpool(  
    'database_name',  
    'buffer_pool_name',  
    buffer_pool_size,  
    'immediate',  
    'automatic',  
    change_number_blocks,  
    number_block_pages,  
    block_size)"
```

## Parameters

The following parameters are required:

*database\_name*

The name of the database to run the command on. The data type is varchar.

*buffer\_pool\_name*

The name of the buffer pool to alter. The data type is varchar.

*buffer\_pool\_size*

The size of the buffer pool in number of pages. The data type is integer.

The following parameters are optional:

*immediate*

Specifies whether the command runs immediately. The data type is char. The default is Y.

*automatic*

Specifies whether to set the buffer pool to automatic. The data type is char. The default is N.

*change\_number\_blocks*

Specifies whether there is a change to the number of block pages in the buffer pool. The data type is char. The default is N.

*number\_block\_pages*

The number of block pages in the buffer pools. The data type is integer. The default is 0.

## *block\_size*

The block size for the block pages. The data type is `integer`. Valid values: 2 to 256. The default is 32.

## Usage notes

For information about checking the status of altering a buffer pool, see [rdsadmin.get\\_task\\_status](#).

## Examples

The following example alters a buffer pool called BP16 for a database called TESTDB to non-automatic, and changes the size to 10,000 pages. Db2 runs this command immediately.

```
db2 "call rdsadmin.alter_bufferpool(
    'TESTDB',
    'BP16',
    10000,
    'Y',
    'N')"
```

## rdsadmin.drop\_bufferpool

Drops a buffer pool.

## Syntax

```
db2 "call rdsadmin.drop_bufferpool(
    'database_name',
    'buffer_pool_name'")"
```

## Parameters

The following parameters are required:

### *database\_name*

The name of the database that the buffer pool belongs to. The data type is `varchar`.

### *buffer\_pool\_name*

The name of the buffer pool to drop. The data type is `varchar`.

## Usage notes

For information about checking the status of dropping a buffer pool, see [rdsadmin.get\\_task\\_status](#).

## Examples

The following example drops a buffer pool called BP16 for a database called TESTDB.

```
db2 "call rdsadmin.drop_bufferpool(  
      'TESTDB',  
      'BP16')"
```

## Stored procedures for databases for RDS for Db2

The built-in stored procedures described in this topic manage databases for Amazon RDS for Db2. To run these procedures, the master user must first connect to the `rdsadmin` database.

These stored procedures are used in a variety of tasks. This list isn't exhaustive.

- [Common tasks for databases](#)
- [Creating databases with EBCDIC collation](#)
- [Collecting information about databases](#)
- [Modifying database configuration parameters](#)
- [Migrating from Linux to Linux](#)
- [Migrating from Linux to Linux with near-zero downtime](#)

Refer to the following built-in stored procedures for information about their syntax, parameters, usage notes, and examples.

### Stored procedures

- [rdsadmin.create\\_database](#)
- [rdsadmin.deactivate\\_database](#)
- [rdsadmin.activate\\_database](#)
- [rdsadmin.reactivate\\_database](#)
- [rdsadmin.drop\\_database](#)
- [rdsadmin.update\\_db\\_param](#)
- [rdsadmin.set\\_configuration](#)
- [rdsadmin.show\\_configuration](#)
- [rdsadmin.backup\\_database](#)
- [rdsadmin.restore\\_database](#)
- [rdsadmin.rollforward\\_database](#)
- [rdsadmin.rollforward\\_status](#)
- [rdsadmin.complete\\_rollforward](#)
- [rdsadmin.db2pd\\_command](#)
- [rdsadmin.force\\_application](#)

- [rdsadmin.set\\_archive\\_log\\_retention](#)
- [rdsadmin.show\\_archive\\_log\\_retention](#)
- [rdsadmin.list\\_archive\\_log\\_information](#)
- [rdsadmin.fgac\\_command](#)

## rdsadmin.create\_database

Creates a database.

### Syntax

```
db2 "call rdsadmin.create_database(  
      'database_name',  
      'database_page_size',  
      'database_code_set',  
      'database_territory',  
      'database_collation',  
      'database_autoconfigure_str',  
      'database_non-restrictive')"
```

### Parameters

#### Note

This stored procedure doesn't validate the combination of required parameters. When you call [rdsadmin.get\\_task\\_status](#), the user-defined function could return an error because of a combination of `database_codeset`, `database_territory`, and `database_collation` that is not valid. For more information, see [Choosing the code page, territory, and collation for your database](#) in the IBM Db2 documentation.

The following parameter is required:

#### *database\_name*

The name of the database to create. The data type is `varchar`.

The following parameters are optional:

## database\_page\_size

The default page size of the database. Valid values: 4096, 8192, 16384, 32768. The data type is integer. The default is 8192.

### **⚠ Important**

Amazon RDS supports write atomicity for 4 KiB, 8 KiB, and 16 KiB pages. In contrast, 32 KiB pages risk *torn writes*, or partial data being written to the disk. If you use 32 KiB pages, we recommend that you enable point-in-time recovery and automated backups. Otherwise, you run the risk of being unable to recover from torn pages. For more information, see [the section called “Introduction to backups”](#) and [the section called “Point-in-time recovery”](#).

## database\_code\_set

The code set for the database. The data type is varchar. The default is UTF-8.

## database\_territory

The two-letter country code for the database. The data type is varchar. The default is US.

## database\_collation

The collation sequence that determines how character strings stored in the database are sorted and compared. The data type is varchar.

Valid values:

- COMPATIBILITY – An IBM Db2 Version 2 collation sequence.
- EBCDIC\_819\_037 – ISO Latin code page, collation; CCSID 037 (EBCDIC US English).
- EBCDIC\_819\_500 – ISO Latin code page, collation; CCSID 500 (EBCDIC International).
- EBCDIC\_850\_037 – ASCII Latin code page, collation; CCSID 037 (EBCDIC US English).
- EBCDIC\_850\_500 – ASCII Latin code page, collation; CCSID 500 (EBCDIC International).
- EBCDIC\_932\_5026 – ASCII Japanese code page, collation; CCSID 5026 (EBCDIC US English).
- EBCDIC\_932\_5035 – ASCII Japanese code page, collation; CCSID 5035 (EBCDIC International).
- EBCDIC\_1252\_037 – Windows Latin code page, collation; CCSID 037 (EBCDIC US English).

- EBCDIC\_1252\_500 – Windows Latin code page, collation; CCSID 500 (EBCDIC International).
- IDENTITY – Default collation. Strings are compared byte for byte.
- IDENTITY\_16BIT – The Compatibility Encoding Scheme for UTF-16: 8-bit (CESU-8) collation sequence. For more information, see [Unicode Technical Report #26](#) on the Unicode Consortium website.
- NLSCHAR – Only for use with the Thai code page (CP874).
- SYSTEM – If you use SYSTEM, the database uses the collation sequence automatically for database\_codeset and database\_territory.

The default is IDENTITY.

Additionally, RDS for Db2 supports the following groups of collations: language-aware-collation and locale-sensitive-collation. For more information, see [Choosing a collation for a Unicode database](#) in the IBM Db2 documentation.

### *database\_autoconfigure\_str*

The AUTOCONFIGURE command syntax, for example, 'AUTOCONFIGURE APPLY DB'. The data type is varchar. The default is an empty string or null.

For more information, see [AUTOCONFIGURE command](#) in the IBM Db2 documentation.

### *database\_non-restrictive*

The granting of default authorities and privileges within the database. The data type is varchar. The default is N.

Valid values:

- N – The created database is restrictive and doesn't grant authorities or privileges.
- Y – The created database is non-restrictive and grants a set of permissions to the special group PUBLIC. For more information, see [Default privileges granted on creating a database](#) in the IBM Db2 documentation.

## Usage notes

If you plan on modifying the db2\_compatibility\_vector parameter, modify the parameter before creating a database. For more information, see [Setting the db2\\_compatibility\\_vector parameter](#).

## Special considerations:

- The CREATE DATABASE command sent to the Db2 instance uses the RESTRICTIVE option.
- RDS for Db2 only uses AUTOMATIC STORAGE tablespaces.
- RDS for Db2 uses the default values for NUMSEGS and DFT\_EXTENT\_SZ.
- RDS for Db2 uses storage encryption and doesn't support database encryption.

For more information about these considerations, see [CREATE DATABASE command](#) in the IBM Db2 documentation.

Before calling `rdsadmin.create_database`, you must connect to the `rdsadmin` database. In the following example, replace `master_username` and `master_password` with your RDS for Db2 DB instance information:

```
db2 connect to rdsadmin user master_username using master_password
```

For information about checking the status of creating a database, see [rdsadmin.get\\_task\\_status](#).

For error messages returned when calling `rdsadmin.create_database`, see [the section called "Stored procedure errors"](#).

## Examples

The following example creates a database called TESTJP with a correct combination of the `database_code_set`, `database_territory`, and `database_collation` parameters for Japan:

```
db2 "call rdsadmin.create_database('TESTJP', 4096, 'IBM-437', 'JP', 'SYSTEM')"
```

## rdsadmin.deactivate\_database

Deactivates a database.

## Syntax

```
db2 "call rdsadmin.deactivate_database(  
?,  
'database_name')"
```

## Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameter is required:

*database\_name*

The name of the database to deactivate. The data type is varchar.

## Usage notes

You can deactivate databases to conserve memory resources or to make other database configuration changes. To bring deactivated databases back online, call the [rdsadmin.activate\\_database](#) stored procedure.

You can't deactivate a database on a source DB instance during replication by calling the [rdsadmin.deactivate\\_database](#) stored procedure.

For information about checking the status of deactivating a database, see [rdsadmin.get\\_task\\_status](#).

For error messages returned when calling [rdsadmin.deactivate\\_database](#), see [the section called "Stored procedure errors"](#).

## Examples

The following example deactivates a database called TESTDB.

```
db2 "call rdsadmin.deactivate_database(?, 'TESTDB')"
```

## **rdsadmin.activate\_database**

Activates a database.

For information about the differences between [rdsadmin.reactivate\\_database](#) and [rdsadmin.activate\\_database](#), see [Usage notes](#).

## Syntax

```
db2 "call rdsadmin.activate_database(  
?,  
'database_name' )"
```

### Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameter is required:

*database\_name*

The name of the database to activate. The data type is varchar.

### Usage notes

All databases are activated by default when they are created. If you [deactivate](#) a database on a standalone DB instance to conserve memory resources or to make other database configuration changes, call the `rdsadmin.activate_database` stored procedure to activate the database again.

This stored procedure only activates a database that is on a standalone DB instance and that was deactivated by calling the [rdsadmin.deactivate\\_database](#) stored procedure. To activate a database on a replica source DB instance, you must call the [rdsadmin.reactivate\\_database](#) stored procedure.

For information about checking the status of activating a database, see [rdsadmin.get\\_task\\_status](#).

For error messages returned when calling `rdsadmin.activate_database`, see [the section called "Stored procedure errors"](#).

### Examples

The following example activates a database called TESTDB.

```
db2 "call rdsadmin.activate_database(?, 'TESTDB')"
```

## rdsadmin.reactivate\_database

Reactivates a database.

For information about differences between [rdsadmin.activate\\_database](#) and [rdsadmin.reactivate\\_database](#), see [Usage notes](#).

### Syntax

```
db2 "call rdsadmin.reactivate_database(  
?,  
'database_name' )"
```

### Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameter is required:

*database\_name*

The name of the database to reactivate. The data type is varchar.

### Usage notes

When you call the `rdsadmin.reactivate_database` stored procedure, the stored procedure first deactivates the database by calling the [rdsadmin.deactivate\\_database](#) stored procedure, and then activates the database by calling the [rdsadmin.activate\\_database](#) stored procedure.

After you make changes to database configurations, you might need to reactivate a database on an RDS for Db2 DB instance. To determine if you need to reactivate a database, connect to the database and run `db2 get db cfg show detail`.

For a database on a standalone DB instance, you can use the `rdsadmin.reactivate_database` store procedure to reactivate the database. Or, if you already called the [rdsadmin.deactivate\\_database](#) stored procedure, you can call the [rdsadmin.activate\\_database](#) stored procedure instead.

For a database on a replica source DB instance, you must use the `rdsadmin.reactivate_database` stored procedure to reactivate the database.

For information about checking the status of reactivating a database, see [rdsadmin.get\\_task\\_status](#).

For error messages returned when calling `rdsadmin.reactivate_database`, see [the section called "Stored procedure errors"](#).

## Examples

The following example reactivates a database called TESTDB.

```
db2 "call rdsadmin.reactivate_database(?, 'TESTDB')"
```

## rdsadmin.drop\_database

Drops a database.

### Syntax

```
db2 "call rdsadmin.drop_database('database_name')"
```

### Parameters

The following parameter is required:

*database\_name*

The name of the database to drop. The data type is `varchar`.

### Usage notes

You can drop a database by calling `rdsadmin.drop_database` only if the following conditions are met:

- You didn't specify the name of the database when you created your RDS for Db2 DB instance by using either the Amazon RDS console or the AWS CLI. For more information, see [Creating a DB instance](#).
- You created the database by calling the [the section called "rdsadmin.create\\_database" stored procedure](#).

- You restored the database from an offline or backed-up image by calling the [the section called "rdsadmin.restore\\_database" stored procedure.](#)

Before calling `rdsadmin.drop_database`, you must connect to the `rdsadmin` database. In the following example, replace `master_username` and `master_password` with your RDS for Db2 DB instance information:

```
db2 connect to rdsadmin user master_username using master_password
```

For information about checking the status of dropping a database, see [rdsadmin.get\\_task\\_status](#).

For error messages returned when calling `rdsadmin.drop_database`, see [the section called "Stored procedure errors"](#).

## Examples

The following example drops a database called TESTDB:

```
db2 "call rdsadmin.drop_database('TESTDB')"
```

## rdsadmin.update\_db\_param

Updates database parameters.

### Syntax

```
db2 "call rdsadmin.update_db_param(  
      'database_name',  
      'parameter_to_modify',  
      'changed_value',  
      'restart_database')"
```

### Parameters

The following parameters are required:

#### *database\_name*

The name of the database to run the task for. The data type is `varchar`.

### *parameter\_to\_modify*

The name of the parameter to modify. The data type is varchar. For more information, see [Amazon RDS for Db2 parameters](#).

### *changed\_value*

The value to change the parameter value to. The data type is varchar.

The following parameter is optional:

### *restart\_database*

Specifies whether RDS restarts the database if a restart is necessary. The data type is varchar. To modify logprimary and logfilsiz, set this parameter to 'YES'.

## Usage notes

For information about checking the status of updating database parameters, see [rdsadmin.get\\_task\\_status](#).

For error messages returned when calling `rdsadmin.update_db_param`, see [the section called "Stored procedure errors"](#).

## Examples

### Example 1: Updating a parameter

The following example updates the archretrydelay parameter to 100 for a database called TESTDB:

```
db2 "call rdsadmin.update_db_param(
    'TESTDB',
    'archretrydelay',
    '100')"
```

### Example 2: Deferring validation of objects

The following example defers the validation of created objects on a database called TESTDB to avoid dependency checking:

```
db2 "call rdsadmin.update_db_param(  
    'TESTDB',  
    'auto_reval',  
    'deferred_force')"
```

## rdsadmin.set\_configuration

Configures specific settings for the database.

### Syntax

```
db2 "call rdsadmin.set_configuration(  
    'name',  
    'value')"
```

### Parameters

The following parameters are required:

*name*

The name of the configuration setting. The data type is varchar.

*value*

The value for the configuration setting. The data type is varchar.

### Usage notes

The following table shows the configuration settings that you can control with `rdsadmin.set_configuration`.

Name	Description
RESTORE_DATABASE_N	The number of buffers to create during a restore operation.
UM_BUFFERS	This value must be less than the total memory size of the DB instance class. If this setting isn't configured, Db2 determines the value to use during the restore operation. For more information, see <a href="#">RESTORE DATABASE command</a> in the IBM Db2 documentation.

Name	Description
RESTORE_DATABASE_PARALLELISM	The number of buffer manipulators to create during a restore operation. This value must be less than double the number of vCPUs for the DB instance. If this setting isn't configured, Db2 determines the value to use during the restore operation. For more information, see <a href="#">RESTORE DATABASE command</a> in the IBM Db2 documentation.
RESTORE_DATABASE_NUM_MULTI_PATHS	The number of paths (or I/O streams) to use during a restore from Amazon S3 operation. To use this configuration setting, you must have multiple backup files. This value can improve performance when restoring databases with large volumes of data because it restores multiple database backup files in parallel. We recommend that you set this value to match the number of your database backup files. For more information, see <a href="#">BACKUP DATABASE command</a> in the IBM Db2 documentation.
USE_STREAMING_RESTORE	Specifies whether to stream backup data directly during restoration rather than first downloading the entire backup to your RDS for Db2 DB instance and then extracting it. Setting USE_STREAMING_RESTORE to TRUE significantly reduces storage requirements and can improve restore performance. This setting requires IBM Db2 engine version 11.5.9.0. sb00063198.r1 or higher, and Amazon S3 connectivity through your database's elastic network interface (ENI). For more information, see <a href="#">Remote storage</a> in the IBM Db2 documentation.

## Examples

### Example 1: Specifying number of buffers to create

The following example sets the RESTORE\_DATABASE\_NUM\_BUFFERS configuration to 150.

```
db2 "call rdsadmin.set_configuration(
    'RESTORE_DATABASE_NUM_BUFFERS',
```

```
'150' )"
```

## Example 2: Specifying number of buffer manipulators to create

The following example sets the RESTORE\_DATABASE\_PARALLELISM configuration to 8.

```
db2 "call rdsadmin.set_configuration(  
      'RESTORE_DATABASE_PARALLELISM',  
      '8')"
```

## Example 3: Specifying number of paths or I/O streams to use during restore

The following example sets the RESTORE\_DATABASE\_NUM\_MULTI\_PATHS configuration to 5.

```
db2 "call rdsadmin.set_configuration(  
      'RESTORE_DATABASE_NUM_MULTI_PATHS',  
      '5')"
```

## Example 4: Setting restoration to stream backup data

The following example sets the USE\_STREAMING\_RESTORE configuration to TRUE.

```
db2 "call rdsadmin.set_configuration(  
      'USE_STREAMING_RESTORE',  
      'TRUE')"
```

## rdsadmin.show\_configuration

Returns the current settings that you can set by using the stored procedure `rdsadmin.set_configuration`.

### Syntax

```
db2 "call rdsadmin.show_configuration(  
      'name')"
```

### Parameters

The following parameter is optional:

## *name*

The name of the configuration setting to return information about. The data type is varchar.

The following configuration names are valid:

- RESTORE\_DATABASE\_NUM\_BUFFERS – The number of buffers to create during a restore operation.
- RESTORE\_DATABASE\_PARALLELISM – The number of buffer manipulators to create during a restore operation.
- RESTORE\_DATABASE\_NUM\_MULTI\_PATHS – The number of paths (or I/O streams) to use during a restore from Amazon S3 operation.
- USE\_STREAMING\_RESTORE – Specifies whether to stream backup data directly during restoration rather than first downloading the entire backup data to your RDS for Db2 DB instance and then extracting it.

## Usage notes

If you don't specify the name of a configuration setting, `rdsadmin.show_configuration` returns information for all configuration settings that you can set by using the stored procedure `rdsadmin.set_configuration`.

## Examples

The following example returns information about the current RESTORE\_DATABASE\_PARALLELISM configuration.

```
db2 "call rdsadmin.show_configuration(
    'RESTORE_DATABASE_PARALLELISM')"
```

## **rdsadmin.backup\_database**

Backs up a database from an RDS for Db2 DB instance to an Amazon S3 bucket.

## Syntax

```
db2 "call rdsadmin.backup_database(
    ?,
    'database_name',
```

```
's3_bucket_name',
's3_prefix',
'backup_type',
'compression_option',
'util_impact_priority',
'num_files',
'parallelism',
'num_buffers')"
```

## Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameters are required:

*database\_name*

The name of the target database on an RDS for Db2 DB instance to back up. The data type is varchar.

The database must exist in the DB instance. You can't back up the rdsadmin database.

*s3\_bucket\_name*

The name of the Amazon S3 bucket where you want your backup to reside. The data type is varchar.

The S3 bucket must exist before calling `rdsadmin.backup_database`, be in the same AWS Region as the target database in the RDS for Db2 DB instance that you want to back up, and be accessible through the IAM role attached to the DB instance.

*s3\_prefix*

The prefix of the path to Amazon S3 where RDS for Db2 uploads the backup files. The data type is varchar.

The prefix is limited to 1024 characters. It must not include a leading or trailing slash (/). Because of a limitation with IBM streaming to Amazon S3, we recommend that the prefix includes subdirectories.

For better file management, RDS for Db2 creates extra directories after *s3\_prefix*. RDS for Db2 uploads all backup files to *s3\_prefix/dbi\_resource\_id/db\_name*. If you set *num\_files* higher than 1, the *db\_name* directory will contain more than one backup file.

The following is an example Amazon S3 location for backup files. In the example, *backups/daily* is the value set for the *s3\_prefix* parameter.

```
backups/daily/db-5N7FX0Y4GDP7RG2NSH2ZTAI2W4/SAMPLEDB
```

### *backup\_type*

The type of backup that determines if the database remains available during backup. The data type is `varchar`.

Valid values:

- `OFFLINE` – The database is unavailable during backup. This type is faster, but it causes downtime.
- `ONLINE` – The database remains available during backup. By default, `ONLINE` is set to `INCLUDE LOGS`.

The following parameters are optional:

### *compression\_option*

The type of compression algorithm used that impacts backup time, CPU usage, and storage costs. The data type is `varchar`. The default is `NONE`.

Valid values:

- `NONE` – The largest file size, the least CPU usage, and cheapest storage costs.
- `STANDARD` – Standard Db2 compression. Uses `libdb2compr.so`.
- `ZLIB` – Enhanced Db2 compression. Uses `libdb2zcompr.so`, but is more CPU-intensive and most expensive storage cost.

### *util\_impact\_priority*

The setting that controls the impact of the backup on the system resources. The data type is `integer`. Valid values: 1–100 (from low to high). The default is 50.

Lower values reduce the impact of the backup on the system resources, but might increase the time it takes to back up the database. Higher values might complete the backup of

the database faster, but could affect other operations. The actual impact depends on the overall system utilization and the `util_impact_lim` setting. You can view and modify the `util_impact_lim` setting in parameter groups. For more information, see [Amazon RDS for Db2 parameters](#).

### *num\_files*

The number of parallel upload streams to Amazon S3. The data type is `integer`. Valid values: 1–256.

We recommend that you only set this parameter after observing the backup performance at the default that Amazon RDS automatically calculates. Higher values could improve performance for large backups, especially with high-bandwidth connections, but at a certain point, higher values degrade performance. Also, make sure to take into account your available system resources and network capacity.

### *parallelism*

The number of tablespaces that the backup utility can read in parallel. The data type is `integer`. Valid values: 1–256.

We recommend that you only set this parameter after observing the backup performance at the default that the Db2 engine automatically calculates as the optimal value. If you set this parameter, Amazon RDS validates against the available processors and won't execute the backup request if processing power is insufficient.

### *num\_buffers*

The number of buffers to use. The data type is `integer`. Valid values: 1–268435456.

We recommend that you only set this parameter after observing the backup performance at the default that Amazon RDS automatically calculates based on memory. If you set this parameter, Amazon RDS validates against the available memory and won't execute the backup request if available memory is insufficient. If you are backing up to multiple locations (`num_files` is set to more than 1), then a higher number of buffers could improve performance. If you don't set `compression_option` to `NONE`, then you can improve performance by increasing `num_buffers` and `parallelism`.

## Usage notes

This stored procedure creates asynchronous backup tasks that stream the backup of your database directly to your Amazon S3 bucket by using the Amazon S3 integration. You can make backups

both from your local server or from an RDS for Db2 DB instance, stream them to Amazon S3, and then restore them wherever you want. For information about restoring a database to an RDS for Db2 DB instance, see [rdsadmin.restore\\_database](#).

Before calling the stored procedure, review the following considerations:

- You can only back up one database at a time.
- You can't perform a backup and restore together on a DB instance.
- Amazon S3 server-side encryption with AWS KMS (SSE-KMS) isn't supported. Even if the S3 bucket is set to SSE-KMS, the files uploaded to the S3 bucket won't use SSE-KMS encryption.
- To stream the backup files to Amazon S3, you must have already configured the integration. For more information, see [Integrating an Amazon RDS for Db2 DB instance with Amazon S3](#).
- For an RDS for Db2 DB instance to be able to interact with Amazon S3, you must have a VPC and an Amazon S3 gateway endpoint for private subnets to use. For more information, see [Step 1: Create a VPC gateway endpoint for Amazon S3](#) and [Step 2: Confirm that your VPC gateway endpoint for Amazon S3 exists](#).

Before calling `rdsadmin.backup_database`, you must connect to the `rdsadmin` database. In the following example, replace `master_username` and `master_password` with your RDS for Db2 DB instance information:

```
db2 connect to rdsadmin user master_username using master_password
```

After you back up your database, be sure to terminate the connection.

```
terminate
```

For information about checking the status of backing up a database, see [rdsadmin.get\\_task\\_status](#).

For error messages returned when calling `rdsadmin.backup_database`, see [the section called "Stored procedure errors"](#).

## Examples

All the examples back up a database called MYDB to the Amazon S3 bucket called amzn-s3-demo-bucket and set the `s3_prefix` to backups/daily.

## Example #1: Specifying database offline and unavailable with median utilization and a single upload stream

In the following example, the database is offline, which is faster but means that the database is unavailable during backup. The example performs no compression of the files, has median impact on system resources, and uses a single upload stream to Amazon S3.

```
db2 call "rdsadmin.backup_database(
?, 
'MYDB',
'amzn-s3-demo-bucket',
'backups/daily',
'OFFLINE',
'NONE',
50,
1)"
```

## Example #2: Specifying database online and available with enhanced compression, median utilization, and few parallel upload streams

In the following example, the database is online and available during backup. The example performs enhanced compression, which results in a small file size, but is CPU-intensive. It has a slightly higher than median impact on system resources and uses five upload streams to Amazon S3.

```
db2 call "rdsadmin.backup_database(
?, 
'MYDB',
'amzn-s3-demo-bucket',
'backups/daily',
'ONLINE',
'ZLIB',
60,
5)"
```

## Example #3: Specifying database offline and unavailable with defaults and system calculations

In the following example, the database is offline, which is faster but means that the database is unavailable during backup. The example uses the default compression of the files and impact on system resources. It also allows RDS for Db2 to calculate the number of parallel upload streams to Amazon S3, tablespaces to read in parallel, and buffers to use.

```
db2 "call rdsadmin.backup_database(  
?,  
'MYDB',  
'amzn-s3-demo-bucket',  
'backups/daily',  
'OFFLINE')"
```

#### Example #4: Specifying database offline and unavailable with no compression, high utilization, and custom calculations

In the following example, the database is offline, which is faster but means that the database is unavailable during backup. The example performs no compression of the files, has a high impact on system resources, and uses 20 upload streams to Amazon S3. It sets the maximum number of tablespaces to read in parallel, which can cause the backup request to fail if processing power is insufficient. It also sets the maximum number of buffers to use, which can cause the backup request to fail if memory is insufficient.

```
db2 "call rdsadmin.backup_database(  
?,  
'MYDB',  
'amzn-s3-demo-bucket',  
'backups/daily',  
'OFFLINE',  
'NONE',  
90,  
20,  
256,  
268435456)"
```

### rdsadmin.restore\_database

Restores a database from an Amazon S3 bucket to your RDS for Db2 DB instance.

#### Syntax

```
db2 "call rdsadmin.restore_database(  
?,  
'database_name',  
's3_bucket_name',  
's3_prefix',  
restore_timestamp,
```

```
'backup_type' )"
```

## Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameters are required:

*database\_name*

The name of the target database to restore in RDS for Db2. The data type is varchar.

For example, if the source database name was TESTDB and you set *database\_name* to NEWDB, then Amazon RDS restores NEWDB as the source database.

*s3\_bucket\_name*

The name of the Amazon S3 bucket where your backup resides. The data type is varchar.

*s3\_prefix*

The prefix to use for file matching during download. The data type is varchar.

If this parameter is empty, then all files in the Amazon S3 bucket will be processed. The following is an example prefix:

```
backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101
```

*restore\_timestamp*

The timestamp of the database backup image. The data type is varchar.

The timestamp is included in the backup file name. For example, 20230615010101 is the timestamp for the file name SAMPLE.0.rdsdb.DBPART000.20230615010101.001.

*backup\_type*

The type of backup. The data type is varchar. Valid values: OFFLINE, ONLINE.

Use `ONLINE` for near-zero downtime migrations. For more information, see [Migrating from Linux to Linux with near-zero downtime for Amazon RDS for Db2](#).

## Usage notes

You can use this stored procedure to migrate a Db2 database to an RDS for Db2 DB instance. For more information, see [Using AWS services to migrate data from Db2 to Amazon RDS for Db2](#). You can also use this stored procedure to create multiple copies of the same database with different database names that use the same restore image.

Before calling the stored procedure, review the following considerations:

- Before restoring a database, you must provision storage space for your RDS for Db2 DB instance that is greater than the original Db2 database on disk. If you enabled `USE_STREAMING_RESTORE`, then when you restore your backup, Amazon RDS streams the backup files directly from your S3 bucket to your RDS for Db2 DB instance. If you don't enable `USE_STREAMING_RESTORE`, you must provision storage space for your RDS for Db2 DB instance that is equal to or greater than the sum of the backup size plus the original Db2 database on disk. For more information, see [Insufficient disk space](#).
- When you restore the backup, Amazon RDS extracts the backup file on your RDS for Db2 DB instance. Each backup file must be 5 TB or smaller. If a backup file exceeds 5 TB, then you must split the backup file into smaller files.
- To restore all files using the `rdsadmin.restore_database` stored procedure, don't include the file number suffix after the timestamp in the file names. For example, the `s3_prefix` `backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101` restores the following files:

```
SAMPLE.0.rdsdb.DBPART000.20230615010101.001  
SAMPLE.0.rdsdb.DBPART000.20230615010101.002  
SAMPLE.0.rdsdb.DBPART000.20230615010101.003  
SAMPLE.0.rdsdb.DBPART000.20230615010101.004  
SAMPLE.0.rdsdb.DBPART000.20230615010101.005
```

- RDS for Db2 doesn't support non-automatic storage. For more information, see [Tablespaces not restored](#).
- RDS for Db2 doesn't support non-fenced routines. For more information, see [Non-fenced routines not allowed](#).
- To improve the performance of database restore operations, you can configure the number of buffers, buffer manipulators, and the number of multiple backup paths for RDS to use.

To optimize storage usage and to potentially improve performance, you can also directly stream a backup from Amazon S3. To check the current configuration, use [the section called "rdsadmin.show\\_configuration"](#). To change the configuration, use [the section called "rdsadmin.set\\_configuration"](#).

To bring the database online and apply additional transaction logs after restoring the database, see [rdsadmin.rollforward\\_database](#).

For information about checking the status of restoring your database, see [rdsadmin.get\\_task\\_status](#).

For error messages returned when calling `rdsadmin.restore_database`, see [the section called "Stored procedure errors"](#).

## Examples

The following example restores an offline backup with a single file or multiple files that have the `s3_prefix` `backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101`:

```
db2 "call rdsadmin.restore_database(
?, 
'SAMPLE',
'amzn-s3-demo-bucket',
'backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101',
20230615010101,
'OFFLINE')"
```

## rdsadmin.rollforward\_database

Brings the database online and applies additional transaction logs after restoring a database by calling [rdsadmin.restore\\_database](#).

## Syntax

```
db2 "call rdsadmin.rollforward_database(
?,
'database_name',
's3_bucket_name',
's3_prefix',
'rollforward_to_option',
```

```
'complete_rollback' )"
```

## Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameters are required:

*database\_name*

The name of the database to perform the operation on. The data type is varchar.

*s3\_bucket\_name*

The name of the Amazon S3 bucket where your backup resides. The data type is varchar.

*s3\_prefix*

The prefix to use for file matching during download. The data type is varchar.

If this parameter is empty, then all files in the S3 bucket will be downloaded. The following example is an example prefix:

```
backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101
```

The following input parameters are optional:

*rollforward\_to\_option*

The point to which you want to roll forward. The data type is varchar. Valid values:

END\_OF\_LOGS, END\_OF\_BACKUP with the timestamp in the format YYYY-MM-DD-HH.MM.SS.

The default is END\_OF\_LOGS.

*complete\_rollback*

Specifies whether to complete the roll-forward process. The data type is varchar. The default is TRUE.

If TRUE, then after completion, the database is online and accessible. If FALSE, then the database remains in a ROLL-FORWARD PENDING state.

## Usage notes

You can use `rds.rollforward_database` for an online backup with include logs that are produced on-premises in many different scenarios.

### Scenario 1: Restoring the database, rolling forward the included logs, and bringing the database online

After `rdsadmin.restore_database()` completes, use the syntax in [Example 1](#) to bring the database with transaction logs online.

### Scenario 2: Bringing the database online but not rolling forward the included logs.

After `rdsadmin.restore_database()` completes, use the syntax in [Example 2](#) to bring the database without the transaction logs online.

### Scenario 3: Rolling forward the included logs in the backup, and applying additional transaction logs as they are produced on-premises

After `rdsadmin.restore_database()` completes, use the syntax in [Example 3 or Example 4](#) to rollforward logs without bringing the database online.

If you set `complete_rollforward` to FALSE, then your database is in a ROLL-FORWARD PENDING state and offline. To bring the database online, you must call [`rdsadmin.complete\_rollforward`](#).

For information about checking the status of rolling forward the database, see [`rdsadmin.rollforward\_status`](#).

## Examples

### Example 1: Bringing database with transaction logs online

The following example rolls forward to an online backup of the database with transaction logs and then brings the database online:

```
db2 "call rdsadmin.rollforward_database(  
    ?,
```

```
null,  
null,  
'END_OF_LOGS',  
'TRUE')"
```

## Example 2: Bringing database without transaction logs online

The following example rolls forward to an online backup of the database without transaction logs, and then brings the database online:

```
db2 "call rdsadmin.rollforward_database(  
?,  
'TESTDB',  
'amzn-s3-demo-bucket',  
'logsfolder/',  
'END_OF_BACKUP',  
'TRUE')"
```

## Example 3: Not bringing database with transaction logs online

The following example rolls forward to an online backup of the database with transaction logs, and then doesn't bring the database online:

```
db2 "call rdsadmin.rollforward_database(  
?,  
'TESTDB',  
null,  
'onlinebackup/TESTDB',  
'END_OF_LOGS',  
'FALSE')"
```

## Example 4: Not bringing database with additional transaction logs online

The following example rolls forward to an online backup of the database with additional transaction logs, and then doesn't bring the database online:

```
db2 "call rdsadmin.rollforward_database(  
?,  
'TESTDB',  
'amzn-s3-demo-bucket',  
'logsfolder/S0000155.LOG',  
'END_OF_LOGS',
```

```
'FALSE' )"
```

## rdsadmin.rollforward\_status

Returns the output of ROLLFORWARD DATABASE *database\_name* QUERY STATUS.

### Syntax

```
db2 "call rdsadmin.rollforward_status(  
?,  
'database_name' )"
```

### Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameter is required:

*database\_name*

The name of the database to perform the operation on. The data type is varchar.

### Usage notes

After you call [rdsadmin.rollforward\\_database](#), you can call `rdsadmin.rollforward_status` to check on the status of the rollforward in the database.

For information about checking the status of this stored procedure, see [rdsadmin.get\\_task\\_status](#).

## rdsadmin.complete\_rollforward

Brings database online from a ROLL-FORWARD PENDING state.

### Syntax

```
db2 "call rdsadmin.complete_rollforward(  
?,
```

```
'database_name' )"
```

## Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameter is required:

*database\_name*

The name of the database that you want to bring online. The data type is varchar.

## Usage notes

If you called [rdsadmin.rollforward\\_database](#) with complete\_rollforward set to FALSE, then your database is in a ROLL-FORWARD PENDING state and offline. To complete the roll-forward process and bring the database online, call [rdsadmin.complete\\_rollforward](#).

For information about checking the status of completing the rollforward process, see [rdsadmin.rollforward\\_status](#).

## Examples

The following example brings the TESTDB database online:

```
db2 "call rdsadmin.complete_rollforward(  
?,  
'TESTDB' )"
```

## rdsadmin.db2pd\_command

Collects information about an RDS for Db2 database.

## Syntax

```
db2 "call rdsadmin.db2pd_command( 'db2pd_cmd' )"
```

## Parameters

The following input parameter is required:

### *db2pd\_cmd*

The name of the db2pd command that you want to run. The data type is varchar.

The parameter must start with a hyphen. For a list of parameters, see [db2pd - Monitor and troubleshoot Db2 database command](#) in the IBM Db2 documentation.

The following options aren't supported:

- -addnode
- -alldatabases
- -alldbp
- -alldbs
- -allmembers
- -alm\_in\_memory
- -cfinfo
- -cfpool
- -command
- -dbpartitionnum
- -debug
- -dump
- -everything
- -file | -o
- -ha
- -interactive
- -member
- -pages

 **Note**

-pages summary is supported.

- -pdcollection
- -repeat
- -stack
- -totalmem

The file suboption isn't supported, for example, db2pd -db testdb -tcbstats file=tcbstat.out.

The use of the stacks option isn't supported, for example, db2pd -edus interval=5 top=10 stacks.

## Usage notes

This stored procedure gathers information that can help with monitoring and troubleshooting RDS for Db2 databases.

The stored procedure uses the IBM db2pd utility to run various commands. The db2pd utility requires SYSADM authorization, which the RDS for Db2 master user doesn't have. However, with the Amazon RDS stored procedure, the master user is able to use the utility to run various commands. For more information about the utility, see [db2pd - Monitor and troubleshoot Db2 database command](#) in the IBM Db2 documentation.

The output is restricted to a maximum of 2 GB.

For information about checking the status of collecting information about the database, see [rdsadmin.get\\_task\\_status](#).

## Examples

### Example 1: Returning uptime of DB instance

The following example returns the uptime of an RDS for Db2 DB instance:

```
db2 "call rdsadmin.db2pd_command( '-' )"
```

### Example 2: Returning uptime of database

The following example returns the uptime of a database called TESTDB:

```
db2 "call rdsadmin.db2pd_command( '-db TESTDB -' )"
```

### Example 3: Returning memory usage of DB instance

The following example returns the memory usage of an RDS for Db2 DB instance:

```
db2 "call rdsadmin.db2pd_command( '-dbptnmem' )"
```

### Example 4: Returning memory sets of DB instance and database

The following example returns the memory sets of an RDS for Db2 DB instance and a database called TESTDB:

```
db2 "call rdsadmin.db2pd_command( '-inst -db TESTDB -memsets' )"
```

## rdsadmin.force\_application

Forces applications off of an RDS for Db2 database.

### Syntax

```
db2 "call rdsadmin.force_application(  
?,  
'applications'")"
```

### Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameter is required:

*applications*

The applications that you want to force off of an RDS for Db2 database. The data type is varchar. Valid values: ALL or *application\_handle*.

Separate the names of multiple applications with commas. Example:  
*'application\_handle\_1, application\_handle\_2'*.

## Usage notes

This stored procedure forces all applications off of a database so you can perform maintenance.

The stored procedure uses the IBM FORCE APPLICATION command. The FORCE APPLICATION command requires SYSADM, SYSMAINT, or SYSCTRL authorization, which the RDS for Db2 master user doesn't have. However, with the Amazon RDS stored procedure, the master user is able to use the command. For more information, see [FORCE APPLICATION command](#) in the IBM Db2 documentation.

For information about checking the status of forcing applications off of a database, see [rdsadmin.get\\_task\\_status](#).

## Examples

### Example 1: Specifying all applications

The following example forces all applications off of an RDS for Db2 database:

```
db2 "call rdsadmin.force_application(  
    ?,  
    'ALL')"
```

### Example 2: Specifying multiple applications

The following example forces application handles 9991, 8891, and 1192 off of an RDS for Db2 database:

```
db2 "call rdsadmin.force_application(  
    ?,  
    '9991, 8891, 1192')"
```

## rdsadmin.set\_archive\_log\_retention

Configures the amount of time (in hours) to retain archive log files for the specified RDS for Db2 database.

## Syntax

```
db2 "call rdsadmin.set_archive_log_retention(
    ?,
    'database_name',
    'archive_log_retention_hours')"
```

## Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameters are required:

*database\_name*

The name of the database to configure archive log retention for. The data type is varchar.

*archive\_log\_retention\_hours*

The number of hours to retain the archive log files. The data type is smallint. The default is 0, and the maximum is 168 (7 days).

If the value is 0, Amazon RDS doesn't retain the archive log files.

## Usage notes

By default, RDS for Db2 retains logs for 5 minutes. We recommend that if you use replication tools such as AWS DMS for change data capture (CDC) or IBM Q Replication, you set log retention in those tools for longer than 5 minutes.

You can view the current archive log retention setting by calling [the section called "rdsadmin.show\\_archive\\_log\\_retention".](#)

You can't configure the archive log retention setting on the `rdsadmin` database.

## Examples

### Example 1: Setting retention time

The following example sets the archive log retention time for a database called TESTDB to 24 hours.

```
db2 "call rdsadmin.set_archive_log_retention(
?, 
'TESTDB',
'24')"
```

## Example 2: Disabling retention time

The following example disables archive log retention for a database called TESTDB.

```
db2 "call rdsadmin.set_archive_log_retention(
?, 
'TESTDB',
'0')"
```

## rdsadmin.show\_archive\_log\_retention

Returns the current archive log retention setting for the specified database.

### Syntax

```
db2 "call rdsadmin.show_archive_log_retention(
?, 
'database_name')"
```

### Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameter is required:

*database\_name*

The name of the database to show the archive log retention setting for. The data type is varchar.

## Examples

The following example shows the archive log retention setting for a database called TESTDB.

```
db2 "call rdsadmin.show_archive_log_retention(  
?,  
'TESTDB' )"
```

## rdsadmin.list\_archive\_log\_information

Returns details about the archive log files, such as the size, the creation date and time, and the name of individual log files for the specified database. It also returns the total storage amount used by the log files in the database.

### Syntax

```
db2 "call rdsadmin.list_archive_log_information(  
?,  
'database_name' )"
```

### Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameter is required:

*database\_name*

The name of the database to list archive log information for. The data type is varchar.

## Examples

The following example returns archive log information for a database called TESTDB.

```
db2 "call rdsadmin.list_archive_log_information(  
?,
```

```
'TESTDB' )"
```

## rdsadmin.fgac\_command

Runs fine-grained access control (FGAC) commands.

### Syntax

```
db2 "call rdsadmin.fgac_command(  
?,  
'database_name',  
'fgac_cmd')"
```

### Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. This parameter only accepts ?.

The following input parameters are required:

*database\_name*

The name of the database that you want to run FGAC commands on. The data type is varchar.

*fgac\_cmd*

The fine-grained access control command that you want to run. The data type is varchar.

The following commands are valid:

- ALTER MASK – Alters an existing column mask in row and column access control (RCAC).
- ALTER PERMISSION – Alters properties of an existing row permission in RCAC.
- ALTER SECURITY POLICY – Alters an existing security policy for RCAC.
- ALTER SECURITY LABEL – Alters properties of an existing security label in label-based access control (LBAC).
- ALTER TABLE – Alters table structure, including adding RCAC or LBAC controls.
- CREATE MASK – Creates a new column mask for RCAC.

- CREATE PERMISSION – Creates a new row permission for RCAC.
- CREATE SECURITY LABEL – Creates a new security label for LBAC.
- CREATE SECURITY POLICY – Creates a new security policy for RCAC.
- DROP MASK – Drops an existing column mask.
- DROP PERMISSION – Drops an existing row permission.
- DROP SECURITY LABEL – Drops a security label from LBAC.
- DROP SECURITY POLICY – Drops an existing RCAC security policy.
- GRANT EXEMPTION ON RULE – Allows a user to bypass specific LBAC rules.
- GRANT SECURITY LABEL – Assigns an LBAC security label to a user.
- REVOKE EXEMPTION ON RULE – Removes a user's exemption from LBAC rules.
- REVOKE SECURITY LABEL – Removes an LBAC security label from a user.

## Usage notes

This stored procedure controls access at the row or column level to table data in your database on an RDS for Db2 DB instance. RDS for Db2 supports two types of FGAC on the database:

- Label-based access control (LBAC)
- Row and column access control (RCAC)

Before calling the stored procedure, review the following considerations:

- To escape a single quote ('), use an additional single quote. The following examples show how to escape 'apple', 'banana', and 'fruit'.

```
db2 "call rdsadmin.fgac_command(
?, 
'testdb',
'CREATE SECURITY LABEL COMPONENT FRUITSET SET{''apple'', ''banana''}')
```

```
db2 "call rdsadmin.fgac_command(
?,
'testdb',
'CREATE SECURITY LABEL COMPONENT FRUITTREE TREE(''fruit'' ROOT, ''apple'' UNDER
''fruit'', ''banana'' UNDER ''fruit''))"
```

- To escape brackets ([ ]), use a backslash (\). The following example shows how to escape [' 'apple'', ''banana''].

```
db2 "call rdsadmin.fgac_command(
?, '
testdb',
'CREATE SECURITY LABEL COMPONENT FRUITARRAY ARRAY\[''apple'', ''banana''\]'")"
```

## Examples

The following examples all run FGAC commands on a database called testdb.

### Example 1: Creating a new security label called FRUITSET

```
db2 "call rdsadmin.fgac_command(
?,
'testdb',
'CREATE SECURITY LABEL COMPONENT FRUITSET SET{''apple'', ''banana''}'")"
```

### Example 2: Creating a new mask for the EMP\_ID column that is enabled when EMP\_ID is set to less than three

```
db2 "call rdsadmin.fgac_command(
?,
'testdb',
'CREATE MASK id_MASK ON EMPLOYEE FOR COLUMN EMP_ID RETURN CASE WHEN (EMP_ID < 3)
THEN EMP_ID ELSE NULL END ENABLE')"
```

### Example 3: Creating a new mask for the DEPARTMENT column that is enabled when SESSION\_USER is set to security\_user

```
db2 "call rdsadmin.fgac_command(
?,
'testdb',
'CREATE MASK DEPARTMENT_MASK ON EMPLOYEE FOR COLUMN DEPARTMENT RETURN CASE WHEN
SESSION_USER = ''security_user'' THEN DEPARTMENT ELSE NULL END ENABLE')"
```

### Example 4: Creating a new security label called treelabel

```
db2 "call rdsadmin.fgac_command(
```

```
?,
'testdb',
'CREATE SECURITY LABEL COMPONENT treelabel TREE(''COMPANY'' ROOT, ''HR'' UNDER
''COMPANY'', ''FINANCE'' UNDER ''COMPANY'', ''IT'' UNDER ''COMPANY'')'"
```

## Stored procedures for storage access for RDS for Db2

The built-in stored procedures described in this topic manage storage access for RDS for Db2 databases that use Amazon S3 for migrating data. For more information, see [the section called "Migrating with Amazon S3".](#)

Refer to the following built-in stored procedures for information about their syntax, parameters, usage notes, and examples.

### Stored procedures

- [rdsadmin.catalog\\_storage\\_access](#)
- [rdsadmin.uncatalog\\_storage\\_access](#)

#### rdsadmin.catalog\_storage\_access

Catalogs a storage alias for accessing an Amazon S3 bucket with Db2 data files.

### Syntax

```
db2 "call rdsadmin.catalog_storage_access(
    ?,
    'alias',
    's3_bucket_name',
    'grantee_type',
    'grantee
)"
```

### Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. The datatype is varchar.

The following input parameters are required:

*alias*

The alias name for accessing remote storage in an Amazon S3 bucket. The datatype is varchar.

## *s3\_bucket\_name*

The name of the Amazon S3 bucket where your data resides. The data type is `varchar`.

## *grantee\_type*

The type of grantee to receive authorization. The data type is `varchar`. Valid values: `USER`, `GROUP`.

## *grantee*

The user or group to receive authorization. The data type is `varchar`.

## Usage notes

Amazon RDS includes the cataloged alias in the IAM role that you added to your RDS for Db2 DB instance. If you remove the IAM role from your DB instance, then Amazon RDS deletes the alias. For more information, see [the section called “Migrating with Amazon S3”](#).

For information about checking the status of cataloging your alias, see [rdsadmin.get\\_task\\_status](#).

## Examples

The following example registers an alias called SAMPLE. The user `jorge_souza` is granted access to the Amazon S3 bucket called `amzn-s3-demo-bucket`.

```
db2 "call rdsadmin.catalog_storage_access(
?, 
'SAMPLE',
'amzn-s3-demo-bucket',
'USER',
'jorge_souza')"
```

## **rdsadmin.uncatalog\_storage\_access**

Removes a storage access alias.

## Syntax

```
db2 "call rdsadmin.uncatalog_storage_access(
?, 
'alias')"
```

## Parameters

The following output parameter is required:

?

A parameter marker that outputs an error message. The datatype is varchar.

The following input parameter is required:

*alias*

The name of the storage alias to remove. The datatype is varchar.

## Usage notes

For information about checking the status of removing your alias, see [rdsadmin.get\\_task\\_status](#).

## Examples

The following example removes an alias called SAMPLE. This alias no longer provides access to the Amazon S3 bucket it was associated with.

```
db2 "call rdsadmin.uncatalog_storage_access(
?, 
'SAMPLE ')"
```

## Stored procedures for tablespaces for RDS for Db2

The built-in stored procedures described in this topic manage tablespaces for Amazon RDS for Db2 databases. To run these procedures, the master user must first connect to the `rdsadmin` database.

These stored procedures are used in a variety of tasks. This list isn't exhaustive.

- [Common tasks for tablespaces](#)
- [Generating performance reports](#)
- [Copying database metadata with db2look](#)
- [Creating a repository database for IBM Db2 Data Management Console](#)

Refer to the following built-in stored procedures for information about their syntax, parameters, usage notes, and examples.

### Stored procedures

- [`rdsadmin.create\_tablespace`](#)
- [`rdsadmin.alter\_tablespace`](#)
- [`rdsadmin.rename\_tablespace`](#)
- [`rdsadmin.drop\_tablespace`](#)

### **`rdsadmin.create_tablespace`**

Creates a tablespace.

#### Syntax

```
db2 "call rdsadmin.create_tablespace(  
      'database_name',  
      'tablespace_name',  
      'buffer_pool_name',  
      tablespace_page_size,  
      tablespace_initial_size,  
      tablespace_increase_size,  
      'tablespace_type',  
      'tablespace_prefetch_size')"
```

## Parameters

The following parameters are required:

### *database\_name*

The name of the database to create the tablespace in. The data type is varchar.

### *tablespace\_name*

The name of the tablespace to create. The data type is varchar.

The tablespace name has the following restrictions:

- It can't be the same as the name of an existing tablespace in this database.
- It can only contain the characters \_\$#@a-zA-Z0-9.
- It can't start with \_ or \$.
- It can't start with SYS.

The following parameters are optional:

### *buffer\_pool\_name*

The name of the buffer pool to assign the tablespace. The data type is varchar. The default is an empty string.

#### **⚠ Important**

You must already have a buffer pool of the same page size to associate with the tablespace.

### *tablespace\_page\_size*

The page size of the tablespace in bytes. The data type is integer. Valid values: 4096, 8192, 16384, 32768. The default is the page size used when you created the database by calling [rdsadmin.create\\_database](#).

#### **⚠ Important**

Amazon RDS supports write atomicity for 4 KiB, 8 KiB, and 16 KiB pages. In contrast, 32 KiB pages risk torn writes, or partial data being written to the disk. If you use 32

KiB pages, we recommend that you enable point-in-time recovery and automated backups. Otherwise, you run the risk of being unable to recover from torn pages. For more information, see [the section called “Introduction to backups”](#) and [the section called “Point-in-time recovery”](#).

### *tablespace\_initial\_size*

The initial size of the tablespace in kilobytes (KB). The data type is `integer`. Valid values: 48 or higher. The default is null.

If you don't set a value, Db2 sets an appropriate value for you.

#### Note

This parameter isn't applicable for temporary tablespaces because the system manages temporary tablespaces.

### *tablespace\_increase\_size*

The percentage by which to increase the tablespace when it becomes full. The data type is `integer`. Valid values: 1–100. The default is null.

If you don't set a value, Db2 sets an appropriate value for you.

#### Note

This parameter isn't applicable for temporary tablespaces because the system manages temporary tablespaces.

### *tablespace\_type*

The type of the tablespace. The data type is `char`. Valid values: U (for user data), T (for user temporary data), or S (for system temporary data). The default is U.

### *tablespace\_prefetch\_size*

The prefetch page size of the tablespace. The data type is `char`. Valid values: AUTOMATIC (case insensitive), or non-zero positive integers that are less than or equal to 32767.

## Usage notes

RDS for Db2 always creates a large database for data.

For information about checking the status of creating a tablespace, see [rdsadmin.get\\_task\\_status](#).

## Examples

### Example 1: Creating a tablespace and assigning a buffer pool

The following example creates a tablespace called SP8 and assigns a buffer pool called BP8 for a database called TESTDB. The tablespace has an initial tablespace page size of 4,096 bytes, an initial tablespace of 1,000 KB, and a table size increase set to 50%.

```
db2 "call rdsadmin.create_tablespace(
    'TESTDB',
    'SP8',
    'BP8',
    4096,
    1000,
    50)"
```

### Example 2: Creating a temporary tablespace and assigning a buffer pool

The following example creates a temporary tablespace called SP8. It assigns a buffer pool called BP8 that is 8 KiB in size for a database called TESTDB.

```
db2 "call rdsadmin.create_tablespace(
    'TESTDB',
    'SP8',
    'BP8',
    8192,
    NULL,
    NULL,
    'T')"
```

### Example 3: Creating a tablespace and assigning a prefetch page size

The following example creates a tablespace called SP8 for a database called TESTDB. The tablespace has an initial tablespace increase size of 50 and a prefetch page size of 800.

```
db2 "call rdsadmin.create_tablespace(
```

```
'TESTDB',
'SP8',
NULL,
NULL,
NULL,
50,
NULL,
'800')"
```

## rdsadmin.alter\_tablespace

Alters a tablespace.

### Syntax

```
db2 "call rdsadmin.alter_tablespace(
  'database_name',
  'tablespace_name',
  'buffer_pool_name',
  'tablespace_increase_size',
  'max_size',
  'reduce_max',
  'reduce_stop',
  'reduce_value',
  'lower_high_water',
  'lower_high_water_stop',
  'switch_online',
  'tablespace_prefetch_size')"
```

### Parameters

The following parameters are required:

*database\_name*

The name of the database that uses the tablespace. The data type is varchar.

*tablespace\_name*

The name of the tablespace to alter. The data type is varchar.

The following parameters are optional:

## *buffer\_pool\_name*

The name of the buffer pool to assign the tablespace. The data type is `varchar`. The default is an empty string.

### **⚠ Important**

You must already have a buffer pool of the same page size to associate with the tablespace.

## *tablespace\_increase\_size*

The percentage by which to increase the tablespace when it becomes full. The data type is `integer`. Valid values: 1–100. The default is 0.

## *max\_size*

The maximum size for the tablespace. The data type is `varchar`. Valid values: `integer` K | M | G, or NONE. The default is NONE.

## *reduce\_max*

Specifies whether to reduce the high water mark to its maximum limit. The data type is `char`. The default is N.

## *reduce\_stop*

Specifies whether to interrupt a previous `reduce_max` or `reduce_value` command. The data type is `char`. The default is N.

## *reduce\_value*

The number or percentage to reduce the tablespace high water mark by. The data type is `varchar`. Valid values: `integer` K| M | G, or 1–100. The default is N.

## *lower\_high\_water*

Specifies whether to run the `ALTER TABLESPACE LOWER HIGH WATER MARK` command. The data type is `char`. The default is N.

## *lower\_high\_water\_stop*

Specifies whether to run the `ALTER TABLESPACE LOWER HIGH WATER MARK STOP` command. The data type is `char`. The default is N.

## *switch\_online*

Specifies whether to run the ALTER TABLESPACE SWITCH ONLINE command. The data type is char. The default is N.

## *tablespace\_prefetch\_size*

The prefetch page size of the tablespace. The data type is char. Valid values: AUTOMATIC (case insensitive), or non-zero positive integers that are less than or equal to 32767.

### Note

This parameter only works with buffer\_pool\_name, table\_increase\_size, max\_size, and switch\_online. It doesn't work with reduce\_max, reduce\_stop, reduce\_value, lower\_high\_water, and lower\_high\_water\_stop.

## Usage notes

Before calling the stored procedure, review the following considerations:

- The rdsadmin.alter\_tablespace stored procedure won't work on a tablespace with the tablespace\_type set to T for user temporary data.
- The optional parameters reduce\_max, reduce\_stop, reduce\_value, lower\_high\_water, lower\_high\_water\_stop, and switch\_online are mutually exclusive. You can't combine them with any other optional parameter, such as buffer\_pool\_name, in the rdsadmin.alter\_tablespace command. For more information, see [Statement not valid](#).

For information about checking the status of altering a tablespace, see [rdsadmin.get\\_task\\_status](#).

For error messages returned when calling stored procedures, see [the section called "Stored procedure errors"](#).

## Examples

### Example 1: Lowering the high water mark

The following example alters a tablespace called SP8 and assigns a buffer pool called BP8 for a database called TESTDB to lower the high water mark.

```
db2 "call rdsadmin.alter_tablespace(
```

```
'TESTDB',
'SP8',
'BP8',
NULL,
NULL,
'Y')"
```

## Example 2: Reducing the high water mark

The following example runs the REDUCE MAX command on a tablespace called TBSP\_TEST in the database TESTDB.

```
db2 "call rdsadmin.alter_tablespace(
  'TESTDB',
  'TBSP_TEST',
  NULL,
  NULL,
  NULL,
  'Y')"
```

## Example 3: Interrupting commands to reduce high water mark

The following example runs the REDUCE STOP command on a tablespace called TBSP\_TEST in the database TESTDB.

```
db2 "call rdsadmin.alter_tablespace(
  'TESTDB',
  'TBSP_TEST',
  NULL,
  NULL,
  NULL,
  NULL,
  'Y')"
```

## Example 4: Changing existing prefetch page size

The following example runs the ALTER TABLESPACE SWITCH ONLINE command on a tablespace called TSBP\_TEST and changes the existing prefetch page size to 64.

```
db2 "call rdsadmin.alter_tablespace(
  'TESTDB',
```

```
'TBSP_TEST',
NULL,
NULL,
NULL,
NULL,
NULL,
NULL,
NULL,
NULL,
'Y',
'64')"
```

## rdsadmin.rename\_tablespace

Renames a tablespace.

### Syntax

```
db2 "call rdsadmin.rename_tablespace(
?,  

'database_name',  

'source_tablespace_name',  

'target_tablespace_name')"
```

### Parameters

The following parameters are required:

**?**

A parameter marker that outputs an error message. This parameter only accepts ?.

***database\_name***

The name of the database that the tablespace belongs to. The data type is varchar.

***source\_tablespace\_name***

The name of the tablespace to rename. The data type is varchar.

***target\_tablespace\_name***

The new name of the tablespace. The data type is varchar.

The new name has the following restrictions:

- It can't be the same as the name of an existing tablespace.
- It can only contain the characters `$_#@a-zA-Z0-9`.
- It can't start with `_` or `$`.
- It can't start with `SYS`.

## Usage notes

For information about checking the status of renaming a tablespace, see [rdsadmin.get\\_task\\_status](#).

You can't rename tablespaces that belong to the `rdsadmin` database.

## Examples

The following example renames a tablespace called `SP8` to `SP9` in a database called `TESTDB`.

```
db2 "call rdsadmin.rename_tablespace(
?, 
'TESTDB',
'SP8',
'SP9')"
```

## rdsadmin.drop\_tablespace

Drops a tablespace.

## Syntax

```
db2 "call rdsadmin.drop_tablespace(
'database_name',
'tablespace_name')"
```

## Parameters

The following parameters are required:

### *database\_name*

The name of the database that the tablespace belongs to. The data type is `varchar`.

## *tablespace\_name*

The name of the tablespace to drop. The data type is varchar.

### Usage notes

For information about checking the status of dropping a tablespace, see [rdsadmin.get\\_task\\_status](#).

### Examples

The following example drops a tablespace called SP8 from a database called TESTDB.

```
db2 "call rdsadmin.drop_tablespace(
    'TESTDB',
    'SP8')"
```

# Amazon RDS for Db2 user-defined function reference

The following user-defined functions are available for Amazon RDS DB instances running the Db2 engine.

## Topics

- [rdsadmin.get\\_task\\_status](#)
- [rdsadmin.list\\_databases](#)

## **rdsadmin.get\_task\_status**

Returns the status of a task.

## Syntax

```
db2 "select task_id, task_type, database_name, lifecycle,
      varchar(bson_to_json(task_input_params), 500) as task_params,
      cast(task_output as varchar(500)) as task_output
      from table(rdsadmin.get_task_status(task_id,'database_name','task_type'))"
```

## Parameters

The following parameters are optional. If you do not provide any parameters, the user-defined function returns the status of all tasks for all databases. Amazon RDS retains task history for 35 days.

### *task\_id*

The ID of the task being run. This ID is returned when you run a task. Default: 0.

### *database\_name*

The name of the database for which the task is being run.

### *task\_type*

The type of the task to query. Valid values: ADD\_GROUPS, ADD\_USER, ALTER\_BUFFERPOOL, ALTER\_TABLESPACE, CHANGE\_PASSWORD, COMPLETE\_ROLLFORWARD, CREATE\_BUFFERPOOL, CREATE\_DATABASE, CREATE\_ROLE, CREATE\_TABLESPACE, DROP\_BUFFERPOOL,

DROP\_DATABASE, DROP\_TABLESPACE, LIST\_USERS, REMOVE\_GROUPS, REMOVE\_USER, RESTORE\_DB, ROLLFORWARD\_DB\_LOG, ROLLFORWARD\_STATUS, UPDATE\_DB\_PARAM.

## Usage notes

You can use the `rdsadmin.get_task_status` user-defined function to check the status of the following tasks for Amazon RDS for Db2. This list is not exhaustive.

- Creating, altering, or dropping a buffer pool
- Creating, altering, or dropping a tablespace
- Creating or dropping a database
- Restoring a database backup from Amazon S3
- Rolling forward database logs from Amazon S3

## Examples

The following example displays the columns returned when `rdsadmin.get_task_status` is called.

```
db2 "describe select * from table(rdsadmin.get_task_status())"
```

The following example lists the status of all tasks.

```
db2 "select task_id, task_type, database_name, lifecycle,  
      varchar(bson_to_json(task_input_params), 500) as task_params,  
      cast(task_output as varchar(500)) as task_output  
      from table(rdsadmin.get_task_status(null,null,null))"
```

The following example lists the status of a specific task.

```
db2 "select task_id, task_type, database_name,  
      varchar(bson_to_json(task_input_params), 500) as task_params  
      from table(rdsadmin.get_task_status(1,null,null))"
```

The following example lists the status of a specific task and database.

```
db2 "select task_id, task_type, database_name,
```

```
varchar(bson_to_json(task_input_params), 500) as task_params  
from table(rdsadmin.get_task_status(2, 'SAMPLE', null))"
```

The following example lists the status of all ADD\_GROUPS tasks.

```
db2 "select task_id, task_type, database_name,  
      varchar(bson_to_json(task_input_params), 500) as task_params  
     from table(rdsadmin.get_task_status(null,null,'add_groups'))"
```

The following example lists the status of all tasks for a specific database.

```
db2 "select task_id, task_type, database_name,  
      varchar(bson_to_json(task_input_params), 500) as task_params  
     from table(rdsadmin.get_task_status(null,'testdb', null))"
```

The following example outputs the JSON values as columns.

```
db2 "select varchar(r.task_type,25) as task_type, varchar(r.lifecycle,10) as lifecycle,  
      r.created_at, u.* from  
      table(rdsadmin.get_task_status(null,null,'restore_db')) as r,  
      json_table(r.task_input_params, 'strict $' columns(s3_prefix varchar(500)  
           null on empty, s3_bucket_name varchar(500) null on empty) error on error ) as U"
```

## Response

The `rdsadmin.get_task_status` user-defined function returns the following columns:

### TASK\_ID

The ID of the task.

### TASK\_TYPE

Depends on the input parameters.

- ADD\_GROUPS – Adds groups.
- ADD\_USER – Adds a user.
- ALTER\_BUFFERPOOL – Alters a buffer pool.
- ALTER\_TABLESPACE – Alters a tablespace.
- CHANGE\_PASSWORD – Changes a user's password.

- COMPLETE\_ROLLFORWARD – Completes an `rdsadmin.rollforward_database` task and activates a database.
- CREATE\_BUFFERPOOL – Creates a buffer pool.
- CREATE\_DATABASE – Creates a database.
- CREATE\_ROLE – Creates a Db2 role for a user.
- CREATE\_TABLESPACE – Creates a tablespace.
- DROP\_BUFFERPOOL – Drops a buffer pool.
- DROP\_DATABASE – Drops a database.
- DROP\_TABLESPACE – Drops a tablespace.
- LIST\_USERS – Lists all users.
- REMOVE\_GROUPS – Removes groups.
- REMOVE\_USER – Removes a user.
- RESTORE\_DB – Restores a full database.
- ROLLFORWARD\_DB\_LOG – Performs an `rdsadmin.rollforward_database` task on database logs.
- ROLLFORWARD\_STATUS – Returns the status of an `rdsadmin.rollforward_database` task.
- UPDATE\_DB\_PARAM – Updates the data parameters.

#### DATABASE\_NAME

The name of the database with which the task is associated.

#### COMPLETED\_WORK\_BYTES

The number of bytes restored by the task.

#### DURATION\_MINS

The time taken to complete the task.

#### LIFECYCLE

The status of the task. Possible statuses:

- CREATED – After a task is submitted to Amazon RDS, Amazon RDS sets the status to CREATED.
- IN\_PROGRESS – After a task starts, Amazon RDS sets the status to IN\_PROGRESS. It can take up to 5 minutes for a status to change from CREATED to IN\_PROGRESS.

- **SUCCESS** – After a task completes, Amazon RDS sets the status to **SUCCESS**.
- **ERROR** – If a restore task fails, Amazon RDS sets the status to **ERROR**. For more information about the error, see [TASK\\_OUTPUT](#).

#### CREATED\_BY

The authid that created the command.

#### CREATED\_AT

The date and time when the task was created.

#### LAST\_UPDATED\_AT

The data and time when the task was last updated.

#### TASK\_INPUT\_PARAMS

The parameters differ based on the task type. All of the input parameters are represented as a JSON object. For example, the JSON keys for the RESTORE\_DB task are the following:

- DBNAME
- RESTORE\_TIMESTAMP
- S3\_BUCKET\_NAME
- S3\_PREFIX

#### TASK\_OUTPUT

Additional information about the task. If an error occurs during native restore, this column includes information about the error.

## Response examples

The following response example shows that a database called TESTJP was successfully created. For more information, see the [the section called “rdsadmin.create\\_database” stored procedure](#).

```
'1 SUCCESS CREATE_DATABASE RDSDB 2023-10-24-18.32.44.962689 2023-10-24-18.34.50.038523
1 TESTJP { "CODESET" : "IBM-437", "TERRITORY" : "JP", "COLLATION" : "SYSTEM",
"AUTOCONFIGURE_CMD" : "", "PAGESIZE" : 4096 }
2023-10-24-18.33.30.079048 Task execution has started.'
```

```
2023-10-24-18.34.50.038523 Task execution has completed successfully`.
```

The following response example explains why dropping a database failed. For more information, see the [the section called "rdsadmin.drop\\_database" stored procedure](#).

```
1 ERROR DROP_DATABASE RDSDB 2023-10-10-16.33.03.744122 2023-10-10-16.33.30.143797 -  
2023-10-10-16.33.30.098857 Task execution has started.  
2023-10-10-16.33.30.143797 Caught exception during executing task id 1, Aborting task.  
Reason Dropping database created via rds CreateDBInstance api is not allowed.  
Only database created using rdsadmin.create_database can be dropped
```

The following response example shows the successful restoration of a database. For more information, see the [the section called "rdsadmin.restore\\_database" stored procedure](#).

```
1 RESTORE_DB SAMPLE SUCCESS
```

```
{ "S3_BUCKET_NAME" : "amzn-s3-demo-bucket", "S3_PREFIX" :  
"SAMPLE.0.rdsdb3.DBPART000.20230413183211.001", "RESTORE_TIMESTAMP" :  
"20230413183211", "BACKUP_TYPE" : "offline" }
```

```
2023-11-06-18.31.03.115795 Task execution has started.  
2023-11-06-18.31.04.300231 Preparing to download  
2023-11-06-18.31.08.368827 Download complete. Starting Restore  
2023-11-06-18.33.13.891356 Task Completed Successfully
```

## rdsadmin.list\_databases

Returns a list of all databases running on an RDS for Db2 DB instance.

### Syntax

```
db2 "select * from table(rdsadmin.list_databases())"
```

### Usage notes

This user-defined function doesn't specify whether databases are in an activated or deactivated state.

If you don't see your databases in the list, call the [the section called "rdsadmin.get\\_task\\_status"](#) user-defined function and look for error messages.

## Response

The `rdsadmin.list_databases` user-defined function returns the following columns:

### DATABASE\_NAME

The name of a database.

### CREATE\_TIME

The date and time when the database was created.

## Response examples

The following response example shows a list of databases and the times when they were created. `rdsadmin` is a database that Amazon RDS manages and always appears in the output.

DATABASE_NAME	CREATE_TIME
rdsadmin	2024-10-22-03.37.48.535671
TEST	2024-10-22-03.39.36.818679
TEST1	2024-10-22-03.57.15.218009
TEST2	2024-10-22-03.59.28.029556

# Troubleshooting for Amazon RDS for Db2

The following content can help you troubleshoot issues that you encounter with RDS for Db2.

For more information about general Amazon RDS troubleshooting issues, see [Troubleshooting for Amazon RDS](#).

## Topics

- [Database connection error](#)
- [File I/O error](#)
- [Stored procedure errors](#)

## Database connection error

The following error message indicates that a database failed to connect because the server doesn't have sufficient memory.

```
SQL1643C The database manager failed to allocate shared memory because the  
database manager instance memory limit has been reached.
```

Increase the memory for your DB instance and then try to connect to your database again. For information about memory usage and recommendations for databases, see [the section called "Multiple Db2 databases"](#). For information about how to update the memory for an RDS for Db2 database, see [the section called "rdsadmin.update\\_db\\_param"](#).

## File I/O error

You might encounter a file I/O error for different reasons, such as when you use the LOAD command or call the `rdsadmin.restore_database` stored procedure.

In this example, you run the following LOAD command.

```
db2 "call sysproc.admin_cmd('load from "DB2REMOTE://s3test//public/datapump/t6.del" of  
del lobs from "DB2REMOTE://s3test/public/datapump/" modified by lobsinfile MESSAGES ON  
SERVER insert INTO RDSDB.t6 nonrecoverable ')"
```

The LOAD command returns the following message:

## Result set 1

ROWS\_READ ROWS\_SKIPPED ROWS\_LOADED ROWS\_REJECTED  
ROWS\_DELETED ROWS\_COMMITTED ROWS\_PARTITIONED NUM\_AGENTINFO\_ENTRIES  
MSG RETRIEVAL

## MSG\_REMOVE

```
SELECT SQLCODE, MSG FROM TABLE(SYSPROC.ADMIN_GET_MSGS('1594987316_285548770')) AS MSG
```

## CALL

```
SYSPROC.ADMIN_REMOVE_MSGS('1594987316_285548770')
```

1 record(s) selected.

Return Status = 0

SQL20397W Routine "SYSPROC.ADMIN\_CMD" execution has completed, but at least one error, "SQL1652", was encountered during the execution. More information is available. SQLSTATE=01H52

To view the error message, you run the SQL command as suggested in the previous response.

SELECT SQLCODE, MSG FROM

TABLE(SYSPROC.ADMIN\_GET\_MSGS('1594987316\_285548770')) AS MSG returns the following message:

## SQLCODE      MSG

```
SQL2025N An I/O error occurred. Error code "438". Media on which this error occurred:  
"DB2REMOTE://s3test//public/datapump/t6.del"  
  
SQL3500W The utility is beginning the LOAD phase at time "07/05/2024 21:21:48.082954"  
  
SQL1652N File I/O error occurred
```

The Db2 diagnostic logs contain a log file similar to the following one:

```
2024-07-05-21.20.09.440609+000 I1191321E864 LEVEL: Error  
PID      : 2710          TID : 139619509200640 PROC : db2sysc 0  
INSTANCE: rdsdb          NODE : 000          DB    : NTP  
APPHDL   : 0-12180        APPID: xxx.xx.x.xxx.xxxxxx.xxxxxxxxxxxxxx  
UOWID    : 5             ACTID: 1  
AUTHID   : ADMIN         HOSTNAME: ip-xx-xx-x-xx  
EDUID    : 147           EDUNAME: db2lmr 0  
FUNCTION: DB2 UDB, oper system services, sqloS3Client_GetObjectInfo, probe:219  
MESSAGE  : ZRC=0x870F01B6=-2029059658=SQL0_FAILED  
           "An unexpected error is encountered"  
DATA #1 : String, 29 bytes  
S3:HeadObject request failed.  
DATA #2 : signed integer, 4 bytes  
99  
DATA #3 : String, 0 bytes  
Object not dumped: Address: 0x00007EFC08A9AE38 Size: 0 Reason: Zero-length data  
DATA #4 : String, 33 bytes  
curlCode: 28, Timeout was reached
```

This file I/O error could result from a number of different scenarios. For example, the VPC associated with the security group used to create your RDS for Db2 DB instance might lack an Amazon S3 gateway endpoint. This endpoint is essential for enabling RDS for Db2 to access Amazon S3. If your RDS for Db2 DB instance is in private subnets, then an Amazon S3 gateway endpoint is required. You can specify whether your DB instance uses private or public subnets by configuring Amazon RDS subnet groups. For more information, see [Working with DB subnet groups](#).

## Topics

- [Step 1: Create a VPC gateway endpoint for Amazon S3](#)
- [Step 2: Confirm that your VPC gateway endpoint for Amazon S3 exists](#)

## Step 1: Create a VPC gateway endpoint for Amazon S3

For your RDS for Db2 DB instance to interact with Amazon S3, create a VPC and then an Amazon S3 gateway endpoint for private subnets to use.

### To create a VPC gateway endpoint for S3

1. Create a VPC. For more information see [Create a VPC](#) in the *Amazon Virtual Private Cloud User Guide*.
2. Create an Amazon S3 gateway endpoint for private subnets to use. For more information, see [Gateway endpoints](#) in the *AWS PrivateLink Guide*.

## Step 2: Confirm that your VPC gateway endpoint for Amazon S3 exists

Confirm that you successfully created an Amazon S3 gateway endpoint by using the AWS Management Console or the AWS CLI.

### Console

#### To confirm an Amazon S3 gateway endpoint

1. Sign in to the AWS Management Console and open the Amazon VPC Console at <https://console.aws.amazon.com/vpc>.
2. In the upper-right corner of the console, choose the AWS Region of your VPC.
3. Select the VPC that you created.
4. On the **Resource map** tab, under **Network connections**, confirm that an Amazon S3 gateway endpoint is listed.

### AWS CLI

To confirm an Amazon S3 gateway endpoint, run the [describe-vpc-endpoints](#) command. In the following example, replace `vpc_id` with the VPC ID, `region` with your AWS Region, and `profile` with your profile name.

For Linux, macOS, or Unix:

```
aws ec2 describe-vpc-endpoints \
--filters "Name=vpc-id,Values=$vpc_id" \
```

```
"Name=service-name,\nValues=com.amazonaws.${region}.s3" \\\n--region $region --profile=$profile \\n--query "VpcEndpoints[*].VpcEndpointId" --output text
```

For Windows:

```
aws ec2 describe-vpc-endpoints ^\n--filters "Name=vpc-id,Values=$vpc_id" ^\n"Name=service-name,^\nValues=com.amazonaws.${region}.s3" ^\n--region $region --profile=$profile ^\n--query "VpcEndpoints[*].VpcEndpointId" --output text
```

This command produces output similar to the following example if an Amazon S3 gateway endpoint exists.

```
[\n    "vpce-0ea810434ff0b97e4"\n]
```

This command produces output similar to the following example if an Amazon S3 gateway endpoint doesn't exist.

```
[]
```

If you don't see an Amazon S3 gateway endpoint listed, then [Step 1: Create a VPC gateway endpoint for Amazon S3](#).

## Stored procedure errors

This section describes various errors returned when calling stored procedures and how to resolve them.

Category	Stored procedure errors
Databases	<a href="#">rdsadmin.activate_database errors</a>
Databases	<a href="#">rdsadmin.backup_database errors</a>

Category	Stored procedure errors
Databases	<a href="#">rdsadmin.create_database errors</a>
Databases	<a href="#">rdsadmin.deactivate_database errors</a>
Databases	<a href="#">rdsadmin.drop_database errors</a>
Databases	<a href="#">rdsadmin.reactivate_database errors</a>
Databases	<a href="#">rdsadmin.restore_database errors</a>
Databases	<a href="#">rdsadmin.update_db_param errors</a>
Tablespaces	<a href="#">rdsadmin.alter_tablespace errors</a>

## rdsadmin.activate\_database errors

The following errors can occur when you call the [the section called “rdsadmin.activate\\_database”](#) stored procedure.

Error	Error message
<a href="#">Failed to allocate shared memory</a>	SQL1643C The database manager failed to allocate shared memory because the database manager instance memory limit has been reached.
<a href="#">Unable to activate because of running processes</a>	The database can't be activated because it's in the process of being created or restored.

### Failed to allocate shared memory

The following error message indicates that the stored procedure failed to activate a database because the DB instance doesn't have sufficient memory.

SQL1643C The database manager failed to allocate shared memory because the database manager instance memory limit has been reached.

Increase the memory for your DB instance and then call the `rdsadmin.activate_database` stored procedure again. For information about memory usage and recommendations for databases, see [the section called “Multiple Db2 databases”](#).

### Unable to activate because of running processes

The following error message indicates that the stored procedure couldn't activate a database because the `rdsadmin.create_database` or `rdsadmin.restore_database` stored procedure is running.

The database can't be activated because it's in the process of being created or restored.

Wait a few minutes, and then call the `rdsadmin.activate_database` stored procedure again.

### `rdsadmin.alter_tablespace` errors

The following errors can occur when you call the [the section called “rdsadmin.alter\\_tablespace”](#) stored procedure.

Error	Error message
<a href="#">Statement not valid</a>	DB21034E The command was processed as an SQL statement because it was not a valid Command Line Processor command. During SQL processing it returned:  SQL1763N Invalid ALTER TABLESPACE statement for table space "TBSP_TEST" due to reason "12"
<a href="#">tablespace_prefetch_size value not valid</a>	Invalid tablespace_prefetch_size. Set value to AUTOMATIC or to a non-zero positive numerical value.
<a href="#">tablespace_prefetch_size numerical value not valid</a>	Invalid tablespace_prefetch_size. The number of pages can't be greater than 32767.
<a href="#">Parameter can't be used with tablespace_prefetch_size</a>	You can't use tablespace_prefetch_size with <code>{parameter}</code> .

Error	Error message
<a href="#">Tablespace change failed</a>	The change to tablespace { <i>tablespace_name</i> } failed because you can only alter LARGE or REGULAR tablespaces.

## Statement not valid

The following error message indicates that the stored procedure combined mutually exclusive optional parameters with other optional parameters. The optional parameters `reduce_max`, `reduce_stop`, `reduce_value`, `lower_high_water`, `lower_high_water_stop`, and `switch_online` for the `rdsadmin.alter_tablespace` stored procedure are mutually exclusive. You can't combine them with any other optional parameter, such as `buffer_pool_name`, in the `rdsadmin.alter_tablespace` stored procedure. If you combine them, then when you call the `rdsadmin.get_task_status` user-defined function, Db2 will return this error message.

DB21034E The command was processed as an SQL statement because it was not a valid Command Line Processor command. During SQL processing it returned:  
SQL1763N Invalid ALTER TABLESPACE statement for table space "TBSP\_TEST" due to reason "12"

Call the `rdsadmin.alter_tablespace` stored procedure again without combining mutually exclusive optional parameters with other optional parameters. Then call the `rdsadmin.get_task_status` user-defined function. For more information, see [rdsadmin.alter\\_tablespace](#) and [rdsadmin.get\\_task\\_status](#).

## tablespace\_prefetch\_size value not valid

The following error message indicates that you didn't set `tablespace_prefetch_size` to `AUTOMATIC` or a non-positive numerical value. For example, you tried to set it to `testinput`.

Invalid `tablespace_prefetch_size`. Set value to `AUTOMATIC` or to a non-zero positive numerical value.

Call the `rdsadmin.alter_tablespace` stored procedure again and set `tablespace_prefetch_size` to `AUTOMATIC` or a non-positive numerical value.

## tablespace\_prefetch\_size numerical value not valid

The following error message indicates that you set `tablespace_prefetch_size` to a numerical value larger than 32767.

Invalid `tablespace_prefetch_size`. The number of pages can't be greater than 32767.

Call the `rdsadmin.alter_tablespace` stored procedure again and set `tablespace_prefetch_size` to a non-zero positive numerical value less than or equal to 32767.

### Parameter can't be used with `tablespace_prefetch_size`

The following error message indicates that you tried to use `tablespace_prefetch_size` with an incompatible parameter.

You can't use `tablespace_prefetch_size` with `{parameter}`.

Call the `rdsadmin.alter_tablespace` stored procedure again and only use `tablespace_prefetch_size` with compatible parameters. For information about parameters you can use with `tablespace_prefetch_size`, see [rdsadmin.alter\\_tablespace](#).

### Tablespace change failed

The following error message indicates that you tried to alter a tablespace.

The change to tablespace `{tablespace_name}` failed because you can only alter LARGE or REGULAR tablespaces.

### rdsadmin.backup\_database errors

The following errors can occur when you call the [the section called "rdsadmin.backup\\_database"](#) stored procedure.

Error	Error message
<a href="#">Insufficient disk space</a>	Aborting task. Reason Backing up your database failed because of insufficient disk space. Increase the storage for your DB instance and rerun the <code>rdsadmin.backup_database</code> stored procedure.
<a href="#">Internal error</a>	Caught exception during executing task id 104, Aborting task. Reason Internal Error

## Insufficient disk space

The following error message indicates that your DB instance has insufficient disk space to back up your database:

Abort task. Reason Backing up your database failed because of insufficient disk space. Increase the storage for your DB instance and rerun the `rdsadmin.backup_database` stored procedure.

When backing up a database to remote storage, make sure that you have sufficient free disk space for the backup session and the working files. Each backup session processes up to 5 GB of data, but additional space is needed for transaction logs, temporary files, and ongoing database operations.

We recommend that you have the following free disk space for backups based on the database size:

- For databases under 5 GB – The database size + 3 GB buffer
- For databases 5 GB and larger – At least 10 GB of free space

This amount of free disk space accounts for backup session processing, transaction log accumulation during backup, temporary working files, and parallel backup streams if configured. For more information, see [Increasing DB instance storage capacity](#).

Increase your disk space and then call the `rdsadmin.backup_database` stored procedure again. To confirm that the database was backed up correctly, check the task status by using `rdsadmin.get_task_status`. You can also verify that the backup files exist in your Amazon S3 bucket under `s3_prefix/dbi_resource_id/db_name`.

## Internal error

The following error message indicates that the stored procedure encountered an internal error:

Caught exception during executing task id 104, Aborting task. Reason Internal Error

Contact [AWS Support](#).

## `rdsadmin.create_database` errors

The following error can occur when you call the [the section called “`rdsadmin.create\_database`”](#) stored procedure.

Error	Error message
<a href="#">Failed to allocate shared memory</a>	SQL1643C The database manager failed to allocate shared memory because the database manager instance memory limit has been reached.

## Failed to allocate shared memory

The following error message indicates that the stored procedure failed to create a database because the DB instance doesn't have sufficient memory.

SQL1643C The database manager failed to allocate shared memory because the database manager instance memory limit has been reached.

Increase the memory for your DB instance and then call the `rdsadmin.create_database` stored procedure again. For information about memory usage and recommendations for databases, see [the section called "Multiple Db2 databases"](#).

To confirm that the database was created, call the [the section called "rdsadmin.list\\_databases"](#) user-defined function and check that the new database is listed.

## rdsadmin.deactivate\_database errors

The following error can occur when you call the [the section called "rdsadmin.deactivate\\_database"](#) stored procedure.

Error	Error message
<a href="#">Unable to deactivate because of running processes</a>	The database can't be deactivated because it's in the process of being created or restored.

## Unable to deactivate because of running processes

The following error message indicates that the stored procedure couldn't deactivate a database because the `rdsadmin.create_database` or `rdsadmin.restore_database` stored procedure is running.

The database can't be deactivated because it's in the process of being created or restored.

Wait a few minutes, and then call the `rdsadmin.deactivate_database` stored procedure again.

## rdsadmin.drop\_database errors

The following errors can occur when you call the [the section called "rdsadmin.drop\\_database"](#) stored procedure.

Error	Error message
<a href="#">Database name doesn't exist</a>	SQL0438N Application raised error or warning with diagnostic text: "Cannot drop database. Database with provided name does not exist". SQLSTATE=99993
<a href="#">Return status = 0</a>	Return Status = 0
<a href="#">Dropping database not allowed</a>	1 ERROR DROP_DATABASE RDSDB 2023-10-10-16.33.0 3.744122 2023-10-10-16.33.30.143797 - 2023-10-1 0-16.33.30.098857 Task execution has started. 2023-10-10-16.33.30.143797 Caught exception during executing task id 1, Aborting task. Reason Dropping database created via rds CreateDBInstance api is not allowed. Only database created using rdsadmin.create_database can be dropped

### Database name doesn't exist

The following error message indicates that you passed an incorrect database name in the `rdsadmin.drop_database` stored procedure.

SQL0438N Application raised error or warning with diagnostic text: "Cannot drop database. Database with provided name does not exist". SQLSTATE=99993

Call the `rdsadmin.drop_database` stored procedure again with a correct database name. To confirm that the database was dropped, call the [the section called "rdsadmin.list\\_databases"](#) user-defined function and check that the dropped database isn't listed.

## Return status = 0

The following error message indicates that the stored procedure couldn't be completed.

```
Return Status = 0
```

After you receive Return Status = 0, call the [rdsadmin.get\\_task\\_status](#) user-defined function.

## Dropping database not allowed

The following error message indicates that you created the database by using either the Amazon RDS console or the AWS CLI. You can only use the rdsadmin.drop\_database stored procedure if you created the database by calling the [the section called "rdsadmin.create\\_database" stored procedure](#).

```
1 ERROR DROP_DATABASE RDSDB 2023-10-10-16.33.03.744122 2023-10-10-16.33.30.143797 -  
2023-10-10-16.33.30.098857 Task execution has started.  
2023-10-10-16.33.30.143797 Caught exception during executing task id 1, Aborting task.  
Reason Dropping database created via rds CreateDBInstance api is not allowed.  
Only database created using rdsadmin.create_database can be dropped
```

To drop a database that you created by using either the Amazon RDS console or the AWS CLI, use a client to connect to the database and then run the appropriate command.

## rdsadmin.reactivate\_database errors

The following error can occur when you call the [the section called "rdsadmin.reactivate\\_database" stored procedure](#).

Error	Error message
<a href="#">Failed to allocate shared memory</a>	SQL1643C The database manager failed to allocate shared memory because the database manager instance memory limit has been reached.
<a href="#">Unable to reactivate because of running processes</a>	The database can't be reactivated because it's in the process of being created or restored.

## Failed to allocate shared memory

The following error message indicates that the stored procedure failed to activate a database because the DB instance doesn't have sufficient memory.

SQL1643C The database manager failed to allocate shared memory because the database manager instance memory limit has been reached.

Increase the memory for your DB instance and then call the `rdsadmin.activate_database` stored procedure again. For information about memory usage and recommendations for databases, see [the section called "Multiple Db2 databases"](#).

### Unable to reactivate because of running processes

The following error message indicates that the stored procedure couldn't reactivate a database because the `rdsadmin.create_database` or `rdsadmin.restore_database` stored procedure is running.

The database can't be reactivated because it's in the process of being created or restored.

Wait a few minutes, and then call the `rdsadmin.reactivate_database` stored procedure again.

### **rdsadmin.restore\_database errors**

The following errors can occur when you call the [the section called "rdsadmin.restore\\_database"](#) stored procedure:

Error	Error message
<a href="#">Insufficient disk space</a>	Aborting task. Reason Restoring your database failed because of insufficient disk space. Increase the storage for your DB instance and rerun the <code>rdsadmin.restore_database</code> stored procedure.
<a href="#">Internal error</a>	Caught exception during executing task id 104, Aborting task. Reason Internal Error
<a href="#">Non-fenced routines not allowed</a>	Caught exception during executing task id 2, Aborting task. Reason Non fenced routines are not allowed. Please delete the routines and retry the restore.

Error	Error message
<a href="#">Tablespaces not restored</a>	Reason SQL0970N The system attempted to write to a read-only file. Reason SQL2563W The Restore process has completed successfully. However one or more table spaces from the backup were not restored.

## Insufficient disk space

The following error message indicates that your DB instance has insufficient disk space to restore your database:

Aborting task. Reason Restoring your database failed because of insufficient disk space. Increase the storage for your DB instance and rerun the `rdsadmin.restore_database` stored procedure.

The free space on your DB instance must be more than double the size of your backup image. If your backup image is compressed, the free space on your DB instance must be more than triple the size of your backup image. For more information, see [the section called “Increasing DB instance storage capacity”](#).

Increase your disk space and then call the `rdsadmin.restore_database` stored procedure again. To confirm that the database was restored, call the [the section called “rdsadmin.list\\_databases”](#) user-defined function and check that the restored database is listed.

## Internal error

The following error message indicates that the stored procedure encountered an internal error:

Caught exception during executing task id 104, Aborting task. Reason Internal Error

Contact [AWS Support](#).

## Non-fenced routines not allowed

The following error message indicates that your database contains non-fenced routines:

Caught exception during executing task id 2, Aborting task. Reason Non fenced routines are not allowed. Please delete the routines and retry the restore.

RDS for Db2 doesn't support non-fenced routines. Remove the non-fenced routines from the source database, and then call `rdsadmin.restore_database` again. To confirm that the database was restored, call the [the section called "rdsadmin.list\\_databases"](#) user-defined function and check that the restored database is listed. For more information, see [the section called "Non-fenced routines"](#).

## Tablespaces not restored

The following error message indicates that RDS for Db2 successfully restored your database, but couldn't restore one or more tablespaces:

Reason SQL0970N The system attempted to write to a read-only file.  
Reason SQL2563W The Restore process has completed successfully. However one or more table spaces from the backup were not restored.

RDS for Db2 doesn't support non-automatic storage. Convert non-automatic storage to automatic storage and then call `rdsadmin.restore_database` again. For more information, see [Converting a nonautomatic storage database to use automatic storage](#) in the IBM Db2 documentation.

Databases with non-automatic SMS storage require manual restoration. If your database has non-automatic SMS storage, contact [AWS Support](#).

For information about non-automatic storage and one-time migrations, see [the section called "Non-automatic storage tablespaces during migration"](#).

## rdsadmin.update\_db\_param errors

The following error can occur when you call the [the section called "rdsadmin.update\\_db\\_param"](#) stored procedure.

Error	Error message
<a href="#">Parameter not supported or modifiable</a>	SQL0438N Application raised error or warning with diagnostic text: "Parameter is either not supported or not modifiable to customers". SQLSTATE=99993

### Parameter not supported or modifiable

The following error message indicates that you tried to modify a database configuration parameter that either isn't supported or isn't modifiable.

```
SQL0438N Application raised error or warning with diagnostic text: "Parameter  
is either not supported or not modifiable to customers". SQLSTATE=99993
```

You can see which parameters are modifiable by viewing your parameter groups. For more information, see [the section called “View parameter values for a DB parameter group”](#).