PUBLIC

Document Version: 2026.18 – 2026-08-25

Accessing SAP Datasphere via the Command Line

Content

1 Accessing SAP Datasphere via the Command Line

Many of the features available to SAP Datasphere users can also be accessed via the command line.

> **Note**
>
> The SAP Datasphere command line interface module has been renamed from `dwc` to `datasphere`. The command `dwc` has been decommissioned at the end of 2023: please use the new `datasphere` command instead. For more information, see https://www.npmjs.com/package/@sap/datasphere-cli .

The primary commands available are:

Command	Description
`datasphere configuration`	List, upload, and delete TLS server certificates (see Managing TLS Server Certificates via the Command Line [page 161]).
`datasphere dbusers`	List, create, modify, and delete database users (see Managing Database Users via the Command Line [page 42]).
`datasphere global-roles`	List, create, modify, and delete global roles (see Managing Global Roles via the Command Line [page 19]).
`datasphere marketplace`	List, create, modify, and delete marketplace objects (see Managing the Data Marketplace via the Command Line [page 82]).
`datasphere objects`	List, read, and delete modeling objects (see Manage Modeling Objects and Tasks via the Command Line [page 60]).
`datasphere scoped-roles`	Manage scoped roles (see Managing Scoped Roles via the Command Line [page 22]).
`datasphere spaces`	List, create, modify, and delete spaces and connections (see Manage Spaces and Space Access via the Command Line [page 33]) and (see Managing Connections via the Command Line [page 163]).
`datasphere tasks`	Run certain tasks and obtain task logs (see Running and Managing Tasks via the Command Line [page 74]).
`datasphere users`	List, create, modify, and delete SAP Datasphere users (see Managing Users via the Command Line [page 29]).
`datasphere workload`	Set space priorities and statement limits (see Managing Priorities and Statement Limits for Spaces or Groups via the Command Line [page 55]).

> **Note**
>
> See the blog @sap/datasphere-cli: Command-Line Interface for SAP Datasphere: Overview (updated September 2022) for a summary of blogs about working with the command line interface.

2 Installing and Configuring the Command Line Interface

The command line interface, `datasphere` must be installed and configured before you can use it to access your SAP Datasphere tenant.

The following pages help you get started:

- To use `datasphere`, you must install it (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- We recommend that you log in via an OAuth client (see Log into the Command Line Interface via an OAuth Client [page 8]).
- To simplify issuing commands, you can set the host value to identify the SAP Datasphere tenant you are currently working with (see Set a Host Value to Identify Your SAP Datasphere Tenant [page 8]).
- In addition to the command-specific options listed, there are general options that can be used with any command (see Miscellaneous Options and Commands [page 15]).

API Rate Limiting

Authenticated requests are associated either with the authenticated username, the OAuth client ID, or the tenant ID. Unauthenticated requests are associated with the originating IP address, and not the user.

Requests are limited to approximately 300 per user per minute (25 per user per minute for the *Connections* and *Certificates* APIs). If you exceed the limit, you will receive the `HTTP 429 Too Many Requests` response status code and can review the following request response headers for further information:

- `X-Ratelimit-Limit` - Rate limit per user per minute.
- `X-Ratelimit-Remaining` - Remaining number of requests for the current timeframe for the current user.
- `X-Ratelimit-Reset` - Time in seconds until the rate limit is reset to the defined limit.
- `Retry-After` - Time in seconds the user agent should wait before making a follow-up request.

2.1 Install or Update the SAP Datasphere Command Line Interface

The SAP Datasphere command line interface (`datasphere`) is a `Node.js` package that you download using the Node Package Manager (`npm`).

Context

`datasphere` is listed on https://www.npmjs.com/ at https://www.npmjs.com/package/@sap/datasphere-cli .

> **Note**
>
> The SAP Datasphere command line interface module has been renamed from `dwc` to `datasphere`. The command `dwc` has been decommissioned at the end of 2023: please use the new `datasphere` command instead. For more information, see https://www.npmjs.com/package/@sap/datasphere-cli .

Prerequisites

You have installed the following on your system:

- `Node.js` version >= 20 and <=24
- `npm` version >= 8 and <=11

`npm` is distributed with `Node.js`. Therefore, when you download `Node.js`, `npm` is automatically installed. To download the `Node.js` installer, see nodejs.org .

You can test if `Node.js` and `npm` are installed on your system by executing the following commands:

- `node -v`
- `npm -v`

If `Node.js` and `npm` are already installed, then their current versions will appear. If you receive an error, you have not installed them yet.

Procedure

1. Run the following command:

```
npm install -g @sap/datasphere-cli
```

> **Note**
>
> To update `datasphere` to the latest version at any time, you just need to run `npm install -g @sap/datasphere-cli` again.

2. Test if the installation was successful by running the following command:

```
datasphere --version
```

3. Log into the Command Line Interface via an OAuth Client [page 8].

2.2 Set a Host Value to Identify Your SAP Datasphere Tenant

If you specify the url to your SAP Datasphere tenant with the `host set` command, you can issue any number of other commands without the need to include the `--host` option.

> **Note**
>
> The SAP Datasphere command line interface module has been renamed from `dwc` to `datasphere`. The command `dwc` has been decommissioned at the end of 2023: please use the new `datasphere` command instead. For more information, see https://www.npmjs.com/package/@sap/datasphere-cli .

To specify the url to your SAP Datasphere tenant, enter:

```
datasphere config host set <url>
```

To display the stored url, enter:

```
datasphere config host show
```

To delete the stored url, enter:

```
datasphere config host clean
```

2.3 Log into the Command Line Interface via an OAuth Client

If an administrator has created an OAuth client (with *Purpose* set to *Interactive Usage* or *Technical User*) for `datasphere` command line interface users to log into, there are several methods for accessing it.

This topic contains the following sections:

- Prerequisites [page 9]
- Log in by Passing OAuth Client Information as Options [page 10]
- Log in by Passing OAuth Client Information in an Options File [page 11]
- Run Commands Without a Login by Passing Tokens in a Secrets File [page 12]
- Log Into and Switch Between Multiple SAP Datasphere Tenants [page 13]
- Log Out from the Command Line Interface via an Oauth Client [page 14]

- Troubleshoot Login Issues [page 14]

Prerequisites

To connect to SAP Datasphere via the `datasphere` command line interface, you must have a global or scoped role granting necessary privileges for the commands you want to use and access to the appropriate spaces. You must, in addition obtain access to an appropriate OAuth client:

OAuth Purpose	Interactive Usage	Technical User
Authorization Grant	Authorization Code Three- Legged (User-Client-Server): Manually propagate authorization to another service.	Client Credentials Two-Legged (Client-Server): Service to service communication via APIs. Supported for the following commands only: • `datasphere configuration certificates` • `datasphere marketplace` • `datasphere objects` • `datasphere spaces connections` • `datasphere tasks` **Note** A technical user OAuth client cannot start task chains containing SAP BW bridge process chains.
Authentication	Manually authenticate to generate the authorization code.	None required. The OAuth client contains the necessary credentials and privileges.
Parameters	• Client ID • Secret • Authorization URL • Token URL	• Client ID • Secret • Token URL

For more information, see Create OAuth2.0 Clients to Authenticate Against SAP Datasphere.

Note

See the following blogs for more information about working with the command line interface and OAuth:

- @sap/datasphere-cli: Getting Rid of Passcodes Thanks to OAuth Client Support (published September 2022)
- FAQ and Troubleshooting Guide for @sap/datasphere-cli (published June 2023)

Log in by Passing OAuth Client Information as Options

You can log in by passing the OAuth client information as options on the command line.

> **Note**
>
> You must have write access to the CLI installation folder.

To log in, enter the following command and press `Return`:

```
datasphere login
    --client-id "<id>"
    --client-secret "<secret>"
    --authorization-url "<url>"
    --token-url "<url>"
    --host "<url>"
    --browser "<browser-name>"
    --authorization-flow "<flow>"
```

> **Note**
>
> If you've logged in with an OAuth client with *Purpose* set to *Interactive Usage*, you will be directed to log in with your SAP Datasphere username and password in a browser window once at the beginning of your OAuth session to determine your space permissions.

Complete the parameters as follows:

Parameter	Description
`--client-id "<id>"`	Enter the *OAuth Client ID* provided by your administrator. **Note** In certain environments, including macOS, you must encode the client ID as a URI when passing it as a parameter.
`--client-secret "<secret>"`	Enter the *Secret* provided by your administrator. **Note** In certain environments, including macOS, you must encode the secrets as a URI when passing it as a parameter.
`--authorization-url "<url>"`	[optional] Enter the *Authorization URL* provided by your administrator.
`--token-url "<url>"`	[optional] Enter the *Token URL* provided by your administrator.
`--host "<url>"`	Enter the URL of your SAP Datasphere tenant. You can copy the URL of any page in your tenant. Alternatively, set a host value (see Set a Host Value to Identify Your SAP Datasphere Tenant [page 8]).

Parameter	Description
`--browser "<browser-name>"`	[optional] Enter your `<browser-name>`, if you are not using Google Chrome, from among the following, to specify which browser to open to obtain privileges and space permissions: • `chrome` (default) • `edge` • `firefox` • `brave` • `safari`
`--authorization-flow "<flow>"`	[optional] Enter the appropriate authorization flow for your OAuth client: • `authorization_code` (default) - Interactive Usage purpose (see Create an OAuth2.0 Client with an Interactive Usage Purpose) • `client_credentials` - Technical User purpose (see Create an OAuth2.0 Client with a Technical User Purpose)

Log in by Passing OAuth Client Information in an Options File

You can log in more securely by passing the OAuth client information in an options file instead of on the command line.

> **Note**
>
> You must have write access to the CLI installation folder.

To log in, enter the following command and press `Return`:

```
datasphere login
    --options-file <file>.json
```

> **Note**
>
> If you've logged in with an OAuth client with *Purpose* set to *Interactive Usage*, you will be directed to log in with your SAP Datasphere username and password in a browser window once at the beginning of your OAuth session to determine your space permissions.

Complete the parameters as follows:

Parameter	Description
`--options-file <file>.json`	Enter a path to a `.json` file containing the basic OAuth and other login options: `{ "client-id": "<client-id>", "client-secret": "<client-secret>", "authorization-url": "<authorization-url>", "token-url": "<token-url>", "host": "<url>", "browser": "<browser-name>", "authorization-flow": "<flow>" }`

Run Commands Without a Login by Passing Tokens in a Secrets File

You can avoid running a `login` command (and entering your SAP Datasphere username and password) at the beginning of each OAuth session by extracting the personal access and refresh tokens and passing them either as options or in a secrets file.

To extract your tokens, log into `datasphere` as usual, and then enter the following command and press `Return`:

```
datasphere config secrets show
```

Example output:

```
$ datasphere config secrets show
[
{
"id": 0,
"client_id": "sb-0d85e619...",
"client_secret": "1dcc0522-...",
"tenantUrl": "https://somehost.eu10.cloud.sap",
...
},
{
"id": 1,
"client_id": "sb-0d85e619...",
"client_secret": "1dcc0522-...",
"tenantUrl": "https://somehost.us10.cloud.sap",
...
}
]
```

Then copy the values for `access_token` and `refresh_token`. You can pass these values either as options on the command line or in a secrets file.

> **Note**
>
> Your access and refresh tokens are valid for 720 hours (30 days).

Having extracted your tokens, you no longer need to log in at the beginning of your session and can pass your tokens in a secrets file with any command.

For example, the following command, to read a space, can be run without having logged in beforehand:

```
datasphere spaces read --space My_SPACE --secrets-file <secrets-file>.json
```

Where the `<secrets-file>.json` contains the following options:

```
{
    "client_id": "<client-id>",
    "client_secret": "<client-secret>",
    "authorization_url": "<authorization-url>",
    "token_url": "<token-url>",
    "access_token": "<access-token>",
    "refresh_token": "<refresh-token>"},
    "host": "<url>",
    "browser": "<browser-name>",
    "authorization_flow": "<flow>"
                                          }
```

> **Note**
>
> Secrets files use versions of the options with underscores instead of hyphens.

Log Into and Switch Between Multiple SAP Datasphere Tenants

You can log into multiple SAP Datasphere tenants one after the other and then switch between them by changing the `--host` option as necessary.

For example, you can log into your development SAP Datasphere tenant by entering:

```
datasphere login --secrets-file dev-secrets.json --host https://my-datasphere-dev
```

Then log into your production SAP Datasphere tenant by entering:

```
datasphere login --secrets-file prod-secrets.json --host https://my-datasphere-
prod
```

You are now logged into both the dev and prod tenants, but currently connected to the `https://my-datasphere-prod` and any commands you enter will be applied to this tenant.

To connect to the development tenant, enter:

```
datasphere config host set https://my-datasphere-dev
```

You are now connected to the `https://my-datasphere-dev` tenant, and any commands you enter will be applied to this tenant.

If at any point you are unsure which tenant you are currently connected to, you can enter:

```
datasphere config host show
```

Log Out from the Command Line Interface via an Oauth Client

To log out from an account, use the `datasphere logout` command: this command allows you to optionally specify the ID of the login/secrets to remove using the option `--login-id <id>`.

To log out, enter the following command and press `Return`:

```
datasphere logout
--login-id <id>
```

By default, when you omit the option `--login-id <id>` the login/secrets with ID 0 are removed.

Parameter	Description
`--login-id <id>`	Optional: specifies the login ID (choices: "0", default: "0").

Troubleshoot Login Issues

You might encounter login issues when accessing SAP Datasphere via the command line, even though you are able to log into your tenant via your Web browser.

This is commonly due to missing, expired or otherwise invalid certificates in your Node.js trust store.

To debug certificate issues, you can use the following OpenSSL or curl commands:

- `openssl s_client -connect <tenant_url>:443 -showcerts`
- `curl -vI <tenant_url>`
- `curl --verbose --cacert /etc/ssl/certs/ca-certificates.crt <tenant_url>`

To temporarily disable certificate validation use:

- `js process.env.NODE_TLS_REJECT_UNAUTHORIZED = "0";`

> **Note**
>
> This setting should only be used for debugging and should not be used in productive environments.

Common fixes for these issues include:

- Add your company's CA certificate (or other missing or expired certificates) to your Node.js trust store.
- Contact your administrator to see if other proxy methods are available in your environment.

For further information, see the following blog: FAQ and Troubleshooting Guide for @sap/datasphere-cli.

2.4 Miscellaneous Options and Commands

Only command-specific options are displayed in the SAP Datasphere command line interface. The remaining general options are listed here.

This topic contains the following sections:

General Options

Option	Description
`--host "<url>"`	Enter the URL of your SAP Datasphere tenant. You can copy the URL of any page in your tenant. Alternatively, set a host value (see Set a Host Value to Identify Your SAP Datasphere Tenant [page 8]).
`--options-file <file>.json`	[optional] Enter a path to a `.json` file containing all of some of your `datasphere` options, listed using the full option names.
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.
`--no-pretty`	[optional] Don't pretty-print the output.
`--verbose`	[optional] Print detailed log information to the console.
`--tls-version <version>`	[optional] Set the TLS (Transport Layer Security) protocol to version TLS 1.2 instead of the default value TLS 1.3. Possible values: `TLSv1.3` or `TLSv1.2`. If the request from the command line fails with the version TLS 1.3 because of a wrong protocol error, the command line has a fallback mechanism to automatically retry the same call with the version TLS 1.2.

OAuth Options

These options allow you to log in and work with an OAuth client:

Option	Description
`--client-id "<id>"`	Enter the *OAuth Client ID* provided by your administrator. **Note** In certain environments, including macOS, you must encode the client ID as a URI when passing it as a parameter.
`--options-file <file>.json`	[optional] Enter a path to a `.json` file containing the basic OAuth options: `{ "client-id": "<client-id>", "client-secret": "<client-secret>", "authorization-url": "<authorization-url>", "token-url": "<token-url>", "host": "<url>", "browser": "<browser-name>", "authorization-flow": "<flow>" }`
`--secrets-file <file>.json`	[optional] Enter a path to a `.json` file containing the extended OAuth options, including the extracted tokens: `{ "client_id": "<client-id>", "client_secret": "<client-secret>", "authorization_url": "<authorization-url>", "token_url": "<token-url>", "access_token": "<access-token>", "refresh_token": "<refresh-token>"}, "host": "<url>", "browser": "<browser-name>", "authorization_flow": "<flow>" }` **Note** Secrets files use versions of the options with underscores instead of hyphens. Passing this file with the tokens allows you to run any command without having first to log in.
`--access-token "<token>"`	[optional] Enter the access token for interactive oauth session authentication.
`--refresh-token "<token>"`	[optional] Refresh the access token for interactive oauth session authentication.
`--code "<code>"`	[optional] Enter the code for oauth token retrieval.
`--token-url "<url>"`	[optional] Enter the *Token URL* provided by your administrator.
`--authorization-url "<url>"`	[optional] Enter the *Authorization URL* provided by your administrator.

Option	Description
`--browser "<browser-name>"`	[optional] Enter your `<browser-name>`, if you are not using Google Chrome, from among the following, to specify which browser to open to obtain privileges and space permissions: • `chrome` (default) • `edge` • `firefox` • `brave` • `safari`
`--authorization-flow "<flow>"`	[optional] Enter the appropriate authorization flow for your OAuth client: • `authorization_code` (default) - Interactive Usage purpose (see Create an OAuth2.0 Client with an Interactive Usage Purpose) • `client_credentials` - Technical User purpose (see Create an OAuth2.0 Client with a Technical User Purpose)
`--expires-in "<expires>"`	[optional] Enter the date when the interactive oauth session authentication expires.

See also Log into the Command Line Interface via an OAuth Client [page 8].

Passcode Option

This option lets you request a passcode for running a command:

Option	Description
`--passcode <code>`	[optional] If you are not logged into an OAuth client, you must provide a passcode that you have obtained from your SAP Datasphere tenant. You can include a passcode with your command using the `--passcode` parameter. If you do not include the `--passcode` parameter, `datasphere` will prompt you to obtain a passcode: 1. Enter y and `datasphere` will open the passcode page for your tenant. 2. If you are not already logged in, you must enter your username and password. 3. When you arrive at the passcode page, copy the temporary authentication code and paste it into the command line. **Note** If you are not logged into an OAuth client, you must enter a new passcode for each command that you issue with `datasphere`.

Configuration Options

The `config` category contains commands for configuring `datasphere`:

Command	Description
`cache`	Work with the local CLI cache.
`host`	Configure host properties.
`passcode-url` [option]	Display the passcode URL.
`secrets`	Work with the locally stored secrets.
`help` [command]	Display help for command.

The following sub-commands are available:

Command	Description
`datasphere config cache init --host "<url>"`	Download the file of available `datasphere` commands from the SAP Datasphere instance.
`datasphere config cache clean`	Delete the local file of available `datasphere` commands.
`datasphere --version`	Display the version of `datasphere`.
`datasphere <command> --help --host "<url>"`	Display help for the specified `datasphere` command.
`datasphere config secrets show`	Display the parameters for the OAuth client you are logged into.
`datasphere config secrets check --secrets-file <file.json>`	Check the validity of the contents of the secrets file.
`datasphere config passcode-url --host "<url>"`	Display the passcode url for the SAP Datasphere server.

3 Manage Users and Roles via the Command Line

Users with an administrator role can use the `datasphere` command line interface to manage users and roles.

Objects	Command
Global roles	`datasphere global-roles` See Managing Global Roles via the Command Line [page 19].
Scoped Roles	`datasphere scoped-roles` See Managing Scoped Roles via the Command Line [page 22].
Users	`datasphere users` See Managing Users via the Command Line [page 29].

3.1 Managing Global Roles via the Command Line

You can use the `datasphere` command line interface to list and read global roles and add users to and remove users from them.

This topic contains the following sections:

- Prerequisites [page 19]
- List Global Roles [page 20]
- List Users in a Global Role [page 20]
- Add Users to Global Roles [page 21]
- Remove Users from Global Roles [page 21]

Prerequisites

To manage global roles and privileges via the command line, you must have a global role that grants you the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *Role* (`CRUD----`) - To access the (*Roles*) and (*Authorization Overview*) areas in the (*Security*) tool and to create, update, and delete roles.
- *User* (`-------M`) - To add users to roles.

The *DW Administrator* global role, for example, grants these privileges. For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

To browse the available commands, enter the following and press `Return`:

```
datasphere global-roles
```

> **Note**
>
> You cannot create or delete global roles via the command line.

For general information about working with global roles in SAP Datasphere , see Assign Users to a Role.

List Global Roles

You can list the global roles in your tenant, and optionally write them to a file.

Enter the following command and press `Return`:

```
datasphere global-roles list
     [--output <file>.json]
```

Complete the parameters as follows:

Parameter	Description
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

List Users in a Global Role

To read the list of users added to a global role and optionally write it to a file, enter the following command and press `Return`:

```
datasphere global-roles users list
    --role <ID>
    --output <file>.json
```

Complete the parameters as follows:

Parameter	Description
`--role <id>`	Enter the ID of a global role.
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

For example, to write the list of users added to the global role `Custom_Administrator` to the file `admins.json` , enter the following command and press `Return`:

```
datasphere global-roles users list --role Custom_Administrator --output
admins.json
```

Add Users to Global Roles

You can add users to a global role.

Enter the following command and press `Return`:

```
datasphere global-roles users add
    --role <ID>
    --users <userID[,userID...]>
```

Complete the parameters as follows:

Parameter	Description
`--role <id>`	Enter the ID of a global role.
`--users <userID[,userID...]>`	Enter a comma-separated list of user IDs.

For example, to add the users `AADAMS` and `SSMITH` to the global role `Custom_Administrator`, enter the following command and press `Return`:

```
datasphere global-roles users add --role Custom_Administrator --users
AADAMS,SSMITH
```

Remove Users from Global Roles

You can remove users from a global role.

Enter the following command and press `Return`:

```
datasphere global-roles users remove
    --role <ID>
    --users <userID[,userID...]>
```

Complete the parameters as follows:

Parameter	Description
`--role <id>`	Enter the ID of a global role.
`--users <userID[,userID...]>`	Enter a comma-separated list of user IDs.

For example, to remove the users `AADAMS` and `SSMITH` from the global role `Custom_Administrator`, enter the following command and press `Return`:

```
datasphere global-roles users remove --role Custom_Administrator --users
AADAMS,SSMITH
```

3.2 Managing Scoped Roles via the Command Line

You can use the `datasphere` command line interface to create, read, update, and delete scoped roles.

This topic contains the following sections:

Prerequisites

To manage scoped roles and privileges via the command line, you must have a global role that grants you the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *Role* (`CRUD----`) - To access the (*Roles*) and (*Authorization Overview*) areas in the (*Security*) tool and to create, update, and delete roles.
- *User* (`-------M`) - To add users to roles.
- *Spaces* (`-------M`) - To add spaces to scoped roles.

The *DW Administrator* global role, for example, grants these privileges. For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

To browse the available commands, enter the following and press `Return`:

```
datasphere scoped-roles
```

For general information about working with scoped roles in SAP Datasphere , see Create a Scoped Role to Assign Privileges to Users in Spaces .

List Scoped Roles

You can list the scoped roles in your tenant, and optionally write them to a file.

Enter the following command and press `Return`:

```
datasphere scoped-roles list
     [--output <file>.json]
```

Complete the parameters as follows:

Parameter	Description
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

Read Scoped Roles

You can read the CSN/JSON definition of a scoped role and optionally write it to a file.

Enter the following command and press `Return`:

```
datasphere scoped-roles read
    --role <id>
    [--output <file>.json]
```

To read the list of spaces that are assigned to a scoped role, and optionally write it to a file, enter the following command and press `Return`:

```
datasphere scoped-roles scopes read
    --role <ID>
    --output <file>.json
```

To read the list of users assigned to a scoped role and optionally write it to a file, enter the following command and press `Return`:

```
datasphere scoped-roles users read
    --role <ID>
```

```
    --output <file>.json
```

Complete the parameters as follows:

Parameter	Description
`--role <id>`	Enter the ID of a scoped role.
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

Create Scoped Roles

You can create scoped roles by providing a definition in a JSON file or an input string.

Enter the following command and press `Return`:

```
datasphere scoped-roles create
    --file-path <file>.json|--input '<stringified-json>'
```

Complete the parameters as follows:

Parameter	Description

`--file-path <file>.json`	[optional] Enter a path to a file with a `.json` extension containing your scoped role definition.
	`{`
	`  "name": "<Name>",`
	`  "description": "<Description>",`
	`  "inheritance": "<Role_Template_Name>"`
	`}`

`--input '<stringified-json>'`	[optional] Provide your input in stringified json format instead of via the `--file-path` option.

For example, to create the scoped role `Sales_Modeler` based on the role template `Custom_Modeler`, enter the following command and press `Return`:

```
datasphere scoped-roles create --file-path sales_modeler.json
```

Where the file `sales_modeler.json` contains the following:

```
{
    "name": "Sales_Modeler",
    "description": "Modeler for Sales spaces",
    "inheritance": "Custom_Modeler"
}
```

Update Scoped Roles

You can update the description or template role of a scoped role by providing a new definition in a JSON file or an input string.

Enter the following command and press `Return`:

```
datasphere scoped-roles update
    --role <ID>
    --file-path <file>.json|--input '<stringified-json>'
```

Complete the parameters as follows:

Parameter	Description
`--role <id>`	Enter the ID of a scoped role.
`--file-path <file>.json`	[optional] Enter a path to a file with a `.json` extension containing your scoped role definition. `{ "description": "<Description>", "inheritance": "<Role_Template_Name>" }`
`--input '<stringified-json>'`	[optional] Provide your input in stringified json format instead of via the `--file-path` option.

Add Spaces to Scoped Roles

You can add spaces to a scoped role as a comma-separated list of space IDs.

> **Note**
>
> You must add one or more spaces to a scoped role before you can assign users to it.

Enter the following command and press `Return`:

```
datasphere scoped-roles scopes add
    --role <ID>
    --scopes <ID[,ID...]>
```

Complete the parameters as follows:

Parameter	Description
`--role <id>`	Enter the ID of a scoped role.

Parameter	Description
`--scopes <ID[,ID...]>`	Enter a comma-separated list of space IDs.

For example, to add the two spaces SALES_EU and SALES_US to the scoped role `Sales_Modeler`, enter the following command and press `Return`:

```
datasphere scoped-roles scopes add --role Sales_Modeler --scopes
SALES_EU,SALES_US
```

Add Users to Scoped Roles

You can add users to a scoped role (and specify the spaces they will be given access to) by providing a definition in a JSON file or an input string.

> **Note**
>
> You must add your spaces first, before you can add users to them (see Add Spaces to Scoped Roles [page 25]).

Enter the following command and press `Return`:

```
datasphere scoped-roles users add
    --role <ID>
    --file-path <file>.json|--input '<stringified-json>'
```

Complete the parameters as follows:

Parameter	Description
`--role <id>`	Enter the ID of a scoped role.
`--file-path <file>.json`	Enter a path to a file with a `.json` extension containing the list of users to add and the scopes to add them to: `[ { "id": "<userID>", "scopes": ["<spaceID>","<spaceID>"] }, { "id": "<userID>", "scopes": ["<spaceID>","<spaceID>"] } ]`
`--input '<stringified-json>'`	[optional] Provide your input in stringified json format instead of via the `--file-path` option.

For example, to add the users BBAXTER and JJONES to the spaces SALES_EU and SALES_US via the scoped role Sales_Modeler, enter the following command and press Return:

```
datasphere scoped-roles users add --role Sales_Modeler --file-path add.json
```

Where the file add.json contains the following:

```
[
    {
          "id": "BBAXTER",
          "scopes": ["SALES_EU", "SALES_US"]
    },
    {
          "id": "JJONES",
          "scopes": ["SALES_EU", "SALES_US"]
    }
]
```

> **Note**
>
> Users with a *DW Space Administrator* role (or equivalent permissions) can add users to their space using the `datasphere spaces users add` command (see Add Users to a Space [page 39]).

Remove Users from Scoped Roles

Enter the following command and press Return:

```
datasphere scoped-roles users remove
    --role <ID>
    --file-path <file>.json|--input '<stringified-json>'
```

Complete the parameters as follows:

Parameter	Description
`--role <id>`	Enter the ID of a scoped role.
`--file-path <file>.json`	[optional] Enter a path to a file with a `.json` extension containing the list of users to remove and the spaces to remove them from: `[ { "id": "<userID>", "scopes": ["<spaceID>","<spaceID>"] }, { "id": "<userID>", "scopes": ["<spaceID>","<spaceID>"] } ]`
`--input '<stringified-json>'`	[optional] Provide your input in stringified json format instead of via the `--file-path` option.

For example, to remove `BBAXTER` and `JJONES` from `SALES_EU` via the scoped role `Sales_Modeler`, enter the following command and press `Return`:

```
datasphere scoped-roles users remove --role Sales_Modeler --file-path remove.json
```

Where the file `remove.json` contains the following:

```
[
    {
        "id": "BBAXTER",
        "scopes": ["SALES_EU"]
    },
    {
        "id": "JJONES",
        "scopes": ["SALES_EU"]
    }
]
```

> **Note**
>
> Users with a *DW Space Administrator* role (or equivalent permissions) can remove users from their space using the `datasphere spaces users remove` command (see Remove Users from a Space [page 41]).

Remove Scopes from Scoped Roles

You can remove scopes from a scoped role.

Enter the following command and press `Return`:

```
datasphere scoped-roles scopes remove
    --role <ID>
    --scopes <ID[,ID...]>
```

Complete the parameters as follows:

Parameter	Description
`--role <id>`	Enter the ID of a scoped role.
`--scopes <ID[,ID...]>`	Enter a comma-separated list of space IDs.

For example, to remove the space `SALES_EU` from the scoped role `Sales_Modeler`, enter the following command and press `Return`:

```
datasphere scoped-roles scopes remove --role Sales_Modeler --scopes SALES_EU
```

Delete Scoped Roles

You can delete scoped roles.

Enter the following command and press `Return`:

```
datasphere scoped-roles delete
    --role <ID>
    [--force]
```

Complete the parameters as follows:

Parameter	Description
`--role <id>`	Enter the ID of a scoped role.
`--force`	[optional] Suppress the *Confirm Deletion* step and delete the scoped role without confirmation.

For example, to delete the scoped role `Sales_Modeler` and suppress the confirmation, enter the following command and press `Enter`:

```
datasphere scoped-roles delete --role Sales_Modeler --force
```

3.3 Managing Users via the Command Line

You can use the `datasphere` command line interface to list, create, update, and delete users via the command line.

This topic contains the following sections:

Prerequisites

To manage users via the command line, you must have a global role that grants you the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *User* (`CRUD----`) - To access the (*Users*)area in the (*Security*) tool and to create, update, and delete users.
- *User* (`-------M`) - To assign users to roles.

The *DW Administrator* global role, for example, grants these privileges. For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

To browse the available commands, enter the following and press `Return`:

```
datasphere users
```

For general information about working with users in SAP Datasphere, see Managing SAP Datasphere Users

List Users

You can list the users in your tenant, and optionally write them to a file.

Enter the following command and press `Return`:

```
datasphere users list
    [--accept <format>]
    [--output file.json]
```

Complete the parameters as follows:

Parameter	Description
`--accept <format>`	[optional] Specify the format to return the user definition in. You can choose between: • `application/vnd.sap.datasphere.space.users.list+json` - [default] Standard list including id, first name, last name, display name, and email. • `application/vnd.sap.datasphere.space.users.details+json` - In addition, includes manager and roles.
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

Create Users

You can create users by providing definitions in a JSON file or an input string.

Enter the following command and press `Return`:

```
datasphere users create
    --file-path <file>.json|--input '<stringified-json>'
```

Complete the parameters as follows:

Parameter	Description
`--file-path <file>.json`	[optional] Enter a path to a file with a `.json` extension containing your user definitions. `[ { "id": "<userID>", "firstName": "<firstName>", "lastName": "<lastName>", "email": "<email>" }, { "id": "<userID>", "firstName": "<firstName>", "lastName": "<lastName>", "email": "<email>" } ]`
`--input '<stringified-json>'`	[optional] Provide your input in stringified json format instead of via the `--file-path` option.

For example, to create users from a file, enter the following command and press `Return`:

```
datasphere users create --file-path users.json
```

Where the file `users.json` contains the following:

```
[
    {
      "id": "BBaxter",
      "firstName": "Bob",
      "lastName": "Baxter",
      "email": "b.baxter@acme.com"
    },
    {
      "id": "JJones",
      "firstName": "Jennifer",
      "lastName": "Jones",
      "email": "j.jones@acme.com"
    }
]
```

Update Users

You can update the email or manager of a user by providing a new definition in a JSON file or an input string.

Enter the following command and press `Return`:

```
datasphere users update
    --file-path <file>.json|--input '<stringified-json>'
```

Complete the parameters as follows:

Parameter	Description
`--file-path <file>.json`	[optional] Enter a path to a file with a `.json` extension containing your user definitions. `[ { "id": "<userID>", "email": "<email>" }, { "id": "<userID>", "manager": "<manager userID>" }, { "id": "<userID>", "manager": "<manager userID>", "email": "<email>" } ]`
`--input '<stringified-json>'`	[optional] Provide your input in stringified json format instead of via the `--file-path` option.

Delete Users

```
datasphere users delete
    --users <ID[,ID...]>
    [--force]
```

Complete the parameters as follows:

Parameter	Description
`--users <ID[,ID...]>`	Enter a comma-separated list of user IDs.
`--force`	[optional] Suppress the *Confirm Deletion* step and delete the scoped role without confirmation.

4 Manage Spaces and Space Access via the Command Line

Users with an administrator role can use the `datasphere` command line interface to create, read, update, and delete spaces. Users with a space administrator role can update some space properties, add (or remove) users, database users and HDI containers, and delete spaces.

Objects	Command
Spaces	`datasphere spaces` See Managing Spaces via the Command Line [page 33]. **Note** The commands `datasphere spaces create` and `datasphere spaces delete` are not supported for file spaces (spaces with a storage type of *SAP HANA Data Lake Files*).
Space Users	`datasphere spaces users` See Managing Space Users via the Command Line [page 38].
Database Users	`datasphere dbusers` See Managing Database Users via the Command Line [page 42]. **Note** The `datasphere dbusers <command>` commands are not supported for file spaces (spaces with a storage type of *SAP HANA Data Lake Files*).
Priorities and Statement Limits	`datasphere workload` See Managing Priorities and Statement Limits for Spaces or Groups via the Command Line [page 55].

4.1 Managing Spaces via the Command Line

You can use the `datasphere` command line interface to list, read, create, update, and delete spaces.

The following sections are available in this topic:

Prerequisites

To manage spaces via the command line, you must have either:

A global role that grants you the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *Spaces* (`C------M`) - To create a space, modify its properties (except for the sections *HDI Containers* and *Time Data*), and delete a space.
- *User* (`-R------`) - To allow the creation of spaces.

The *DW Administrator* role template, for example, grants these privileges.

A scoped role that grants you access to a space with the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *Spaces* (`-RU-----`) - To modify all space properties (except for the sections *Space Storage*, *Data Lake Access*, and *Workload Management*).
- *Spaces* (`---D----`) - To delete a space.

  > ⚠ Caution
  >
  > When you delete a space, the space (along with its content and data) is permanently deleted from the database and cannot be recovered.

The *DW Space Administrator* role template, for example, grants these privileges.

For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

To browse the available commands, enter the following and press `Return`:

```
datasphere spaces
```

For general information about working with spaces, see Creating Spaces and Allocating Resources and Managing Your Space.

List Spaces

To list the spaces available to you on the tenant, enter the following command and press `Return`:

```
datasphere spaces list
    [--host "<url>"]
    [--output <file>.json]
```

Complete the parameters as follows:

Parameter	Description
`--host "<url>"`	Enter the URL of your SAP Datasphere tenant. You can copy the URL of any page in your tenant. Alternatively, set a host value (see Set a Host Value to Identify Your SAP Datasphere Tenant [page 8]).
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

Read a Space

To read a space definition to the console or to a file, enter the following command and press `Return`:

```
datasphere spaces read
    --space <id>
    [--definitions [<obj1>,<obj2>]]
    [--no-space-definition]
    [--output <file>.json]
```

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.

Parameter	Description
`--definitions [<obj1>,<obj2>]`	[Optional] Read the object definitions contained in the space. You can use the `--definitions` parameter by itself to read all the objects, or specify a comma-separated list of object technical names. Object definitions are read using the standard CSN syntax (see Core Data Services Schema Notation (CSN)). The following objects can be read (exported): • Local Tables - The definition of a local table without delta capture contains the structure of the table only, and does not have dependencies on any other objects. The definition of a local table with delta capture enabled contains the definition of an additional internal table with the suffix `_Delta`. When you export a table with delta capture enabled, this table is exported too. • Remote Tables - The definition of a remote table contains information about its connection. **Note** Remote tables exported from one space can be imported into another only if they were originally imported from a connection created in v2021.19 or later. • Views - The definition of a view contains the definitions of all its sources and any used data access controls. When you export a view, these objects are exported too. • Data Access Controls - The definition of a data access control contains the definition of its permissions entity. When you export a data access control, the permissions entity is exported too. **Note** You can also export content from and import content to your space via: • The `objects` commands, which support a wider selection of object types (see Manage Modeling Objects and Tasks via the Command Line [page 60]). • The (*Transport*) app (see Transporting Content Between Tenants). • *Export to CSN/JSON File* buttons in selected *Data Builder* editors (see Importing and Exporting Objects in CSN/JSON Files).
`--no-space-definition`	[Optional] Suppress the display of the `spaceDefinition` property. When used with the `--definitions` option, this allows you to read object definitions without seeing the other properties of the space.
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

The space definition is written to the console or to the specified file.

Create or Update a Space

To create or update a space, you must first prepare a space definition file (see The Space Definition File Format [page 50]).

> **Note**
>
> You need only complete the parameters that you want to set. All other space properties are either set to default values or keep their current values. If your file contains valid CSN object definitions, then these entities will be created or updated in the space.

When your file is ready, enter the following command and press `Return`:

```
datasphere spaces create
    --file-path <file>
```

> **Note**
>
> This feature is not supported for file spaces (spaces with a storage type of *SAP HANA Data Lake Files*).

Complete the parameters as follows:

Parameter	Description
`--file-path <file>.json`	Enter a path to a file with a .json extension containing your space definition.
`--force-definition-deployment`	[Optional] Deploy changes to objects even if they will generate validation messages warning of impacts to objects that depend on them. Using this option is equivalent to clicking the *Deploy Anyway* button in the *Validation Messages* dialog (see Modifying Objects That Have Dependent Objects).
`--enforce-database-user-deletion`	[Optional] Allow the deletion of database users and their associated Open SQL schemas, when the `dbusers` section is present in the space definition file. If any existing database user is not included in the `dbusers` section and this option is omitted, then they will not be deleted in order to protect against unintended data loss.
`--input '<stringified-json>'`	[Optional] Provide your space definition in stringified json format instead of via the `--file-path` option. For example: `--input '{"MY_SPACE":{"spaceDefinition":{"version":"1.0.4"}}}'`

The space is created or updated as you have specified.

> **Note**
>
> If any parameters are set incorrectly, the creation or update is canceled and an error message is written to the console.

Delete a Space

To delete a space, enter the following command and press `Return`:

```
datasphere spaces delete
    --space <id>
```

```
    [--force]
```

> **Note**
>
> This feature is not supported for file spaces (spaces with a storage type of *SAP HANA Data Lake Files*).

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--force`	Delete the space without confirmation. If you do not include this option you will be prompted to confirm the deletion.

If prompted, confirm that you want to delete the space. The space is deleted and a confirmation message is written to the console.

> **Caution**
>
> When you delete a space, the space (along with its content and data) is permanently deleted from the database and cannot be recovered.

4.2 Managing Space Users via the Command Line

You can use the `datasphere` command line interface to add and remove users from spaces and to update user roles in spaces.

This topic contains the following sections:

Prerequisites

To manage user access to spaces via the command line, you must have a scoped role that grants you access to your space with the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *Spaces* (`-RU-----`) - To open and update your space in the *Space Management* tool.
- *Space Files* (`-R------`) - To view objects in your space.
- *Scoped Role User Assignment* (`-------M`) - To manage the users who can access your space.

The *DW Space Administrator* role template, for example, grants these privileges. For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

To browse the available commands, enter the following and press `Return`:

```
datasphere spaces users
```

For general information about working with space users, see Control User Access to Your Space.

List Space Users

You can read a list of users in your space and optionally write it to a file.

Enter the following command and press `Return`:

```
datasphere spaces users read
    --space <ID>
    [--accept <format>]
    [--output <file>.json]
```

Complete the parameters as follows:

Parameter	Description
`--space <ID>`	Enter the ID of your space.
`--accept <format>`	[optional] Specify the format to return the list in. You can choose between: • `application/vnd.sap.datasphere.space.users.list+json` - [default] User IDs only. • `application/vnd.sap.datasphere.space.users.details+json` - User IDs and scoped roles.
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

Add Users to a Space

To add a user to your space, you must add them to a scoped role that includes your space as a scope and gives the appropriate privileges.

Enter the following command and press `Return`:

```
datasphere spaces users add
```

```
    --space <ID>
    --file-path <file>.json|--input '<stringified-json>'
```

Complete the parameters as follows:

Parameter	Description
`--space <ID>`	Enter the ID of your space.
`--file-path <file>.json`	Enter a path to a file with a `.json` extension containing the list of users to add and the scoped roles you want to add them to: `[ { "id": "<UserID>", "roles": "<RoleID[,RoleID...]>" }, { "id": "<UserID>", "roles": "<RoleID[,RoleID...]>" } ]` Each user must exist on your tenant and each scoped role that you assign them to must include your space as a scope.

For example, to add the users `BBAXTER` and `JJONES` to the space `SALES_US` via the scoped role `Sales_Modeler`, enter the following command and press `Return`:

```
datasphere spaces users add --space SALES_US --file-path add.json
```

Where the file `add.json` contains the following (where `<package>` and `<id>` for the tenant are `t` and `W` respectively):

```
[
            {
                "id": "BBAXTER",
                "roles": ["Sales_Admin","Sales_Modeler"]
            },
            {
                "id": "JJONES",
                "roles": ["Sales_Modeler"]
            }
        ]
```

Update Space User Roles

You can update the roles that are assigned to users for your space. For example you may want to remove a user's modeling privileges and leave here only with viewing privileges

Enter the following command and press `Return`:

```
datasphere spaces users update
    --space <ID>
    --file-path <file>.json|--input '<stringified-json>'
```

Complete the parameters as follows:

Parameter	Description
`--space <ID>`	Enter the ID of your space.
`--file-path <file>.json`	Enter a path to a file with a `.json` extension containing containing the list of users to update and the new scoped roles you want them to belong to `[ { "id": "<UserID>", "roles": ["<RoleID>","<RoleID>"] }, { "id": "<UserID>", "roles": ["<RoleID>","<RoleID>"] } ]`

For example, to give `BBAXTER` the roles of both `Sales_Modeler` and `Sales_Integrator`, and to remove `JJONES` from the role `Sales_Modeler` and to add her to `Sales_Space_Admin`, enter the following command and press `Return`:

```
datasphere spaces users update --space SALES_US --file-path update.json
```

Where the file `update.json` contains the following:

```
[
            {
                "id": "BBAXTER",
                "roles": ["Sales_Modeler","Sales_Integrator"]
            },
            {
                "id": "JJONES",
                "roles": ["Sales_Space_Admin"]
            }
        ]
```

Remove Users from a Space

You can remove users from your space.

Enter the following command and press `Return`:

```
datasphere spaces users remove
    --space <ID>
    --file-path <file>.json|--input '<stringified-json>'
```

Complete the parameters as follows:

Parameter	Description
`--space <ID>`	Enter the ID of your space.

Parameter	Description
`--file-path <file>.json`	Enter a path to a file with a `.json` extension containing the list of users to remove and the scoped roles to remove them from: `[ { "id": "<UserID>", "role": "<RoleID>" }, { "id": "<UserID>", "role": "<RoleID>" } ]`

For example, to remove `BBAXTER` from both the `Sales_Modeler` and `Sales_Integrator` roles in the `SALES_US` space, enter the following command and press `Return`:

```
datasphere spaces users remove --space SALES_US --file-path remove.json
```

Where the file `remove.json` contains the following:

```
[
            {
                "id": "BBAXTER",
                "role": "Sales_Modeler"
            },
            {
                "id": "BBAXTER",
                "role": "Sales_Integrator"
            }
        ]
```

4.3 Managing Database Users via the Command Line

You can use the `datasphere` command line interface to create, update and delete space database users, and to manage passwords and certificates.

This topic contains the following sections:

- Prerequisites [page 43]
- List Database Users [page 43]
- Create Database Users [page 44]
- Reset a Password [page 46]
- Switch to X.509 Authentication [page 47]
- Update Database Users [page 48]
- Delete Database Users [page 49]

Prerequisites

To manage database users and their passwords or certificates via the command line, you must have a scoped role that grants you access to your space with the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *Spaces* (`-RU-----`) - To open and update your space in the *Space Management* tool.
- *Space Files* (`-R------`) - To view objects in your space.

The *DW Space Administrator* role template, for example, grants these privileges. For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

To browse the available commands, enter the following and press `Return`:

```
datasphere dbusers
```

> **Note**
>
> This feature is not supported for file spaces (spaces with a storage type of *SAP HANA Data Lake Files*).

For general information about working with database users in SAP Datasphere, see Integrating Data via Database Users/Open SQL Schemas.

List Database Users

You can list the database users in your space, and optionally write them to a file.

Enter the following command and press `Return`:

```
datasphere dbusers list
     --space <id>
    [--output <file>.json]
```

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

For example, to list the database users in the SALES space, enter the following:

```
datasphere dbusers list --space SALES
```

Create Database Users

You can create a database user by providing a definition in a JSON file or an input string.

Enter the following command and press `Return`:

```
datasphere dbusers create
    --space <id>
    --databaseuser <id>
    --file-path <name>.json|--input '<stringified-json>'
    [--output <file>.json]
```

> **Note**
>
> We recommend that, for reasons of security, you specify to receive the output in a file.

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--databaseuser <id>`	Enter the name of your database user (*Database User Name Suffix*).

Parameter	Description
`--file-path <name>`	Enter a path to a file with a `.json` extension containing your database user definition. This sample file will create a database user with a password-based authentication, with no access rights, and with audit logs retained for 7 days: `{ "consumption": { "consumptionWithGrant": false, "spaceSchemaAccess": false, "scriptServerAccess": false, "scriptServerAccessWithGrant":false "localSchemaAccess": false, "hdiGrantorForCupsAccess": false }, "ingestion": { "auditing": { "dppRead": { "isAuditPolicyActive": false, "retentionPeriod": 7 }, "dppChange": { "isAuditPolicyActive": false, "retentionPeriod": 7 } } } }`  This sample file will create a database user with a certificate-based authentication, with no access rights and with audit logs retained for 7 days:  `{ "consumption": { "consumptionWithGrant": false, "spaceSchemaAccess": false, "scriptServerAccess": false, "scriptServerAccessWithGrant":false, "enablePasswordLifetime": false, "localSchemaAccess": false, "hdiGrantorForCupsAccess": false }, "x509Config":{ "x509Enabled":true, "subjectName":"<subject distinguished name>", "issuerName":"<issuer distinguished name>" }, "ingestion": { "auditing": { "dppRead": { "isAuditPolicyActive": false, "retentionPeriod": 7 }, "dppChange": { "isAuditPolicyActive": false, "retentionPeriod": 7 } } } }`

Parameter	Description
`--input '<stringified-json>'`	[optional] Provide your definition in stringified json format instead of via the `--file-path` option. **Note** We recommend that you: • Strip unnecessary spaces, tabs, newline, and return characters (`\t`, `\n`, and `\r`). • Use double-quotes consistently to surround names and values within the JSON code. • Surround the whole string in single-quotes. • Be aware of the escape characters available in your shell environment and when it is necessary to use them.
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

For example, to create the database user `JEFF` in the SALES space, provide an appropriate json file and enter the following:

```
datasphere dbusers create --space SALES --databaseuser JEFF --file-path
jeff_defn.json --output jeff.json
```

Reset a Password

To reset a database user password, enter the following command and press `Return`:

```
datasphere dbusers password reset
    --space <id>
    --databaseuser <id>
    [--output <file>.json]
```

Note

We recommend that, for reasons of security, you specify to receive the output in a file.

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--databaseuser <id>`	Enter the name of your database user (*Database User Name Suffix*).
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

For example, to reset the password of the JEFF user in the SALES, and pretty-print it to the file `jeff.json`, enter the following:

```
datasphere dbusers password reset --space SALES --databaseuser JEFF --output
"jeff.json"
```

Switch to X.509 Authentication

If a database user has password-based authentication, you can switch to certificate-based authentication by providing an updated definition in a JSON file or an input string.

> **Note**
>
> Once done, you cannot switch back to a password-based authentication.

> **Note**
>
> Once you've uploaded a certificate for a database user, you cannot upload another certificate later for this user. Also, you cannot use the same certificate for several database users.

Enter the following command and press `Return`:

```
datasphere dbusers update
    --space <id>
    --databaseuser <id>
    --file-path <name>.json|--input '<stringified-json>'
    [--output <file>.json]
```

> **Note**
>
> We recommend that, for reasons of security, you specify to receive the output in a file.

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--databaseuser <id>`	Enter the name of your database user (*Database User Name Suffix*).
`--file-path <name>`	Enter a path to a file with a `.json` extension containing your database user definition.

Parameter	Description
`--input '<stringified-json>'`	[optional] Provide your definition in stringified json format instead of via the `--file-path` option. **Note** We recommend that you: • Strip unnecessary spaces, tabs, newline, and return characters (`\t`, `\n`, and `\r`). • Use double-quotes consistently to surround names and values within the JSON code. • Surround the whole string in single-quotes. • Be aware of the escape characters available in your shell environment and when it is necessary to use them.
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

For example, to switch the database user `JEFF` in the SALES space from password-based to certificate-based authentication, provide an appropriate json file and enter the following:

```
datasphere dbusers update --space SALES --databaseuser JEFF --file-path
jeff_user_cert.json --output jeff.json
```

Where the file `jeff_user_cert.json` contains the following information about the certificate to upload:

```
{
   "x509Config":{
     "x509Enabled":true,
     "subjectName":"CN=MyUserupdate, O=MyCompany, C=DE",
     "issuerName":"CN=MyCA, O=MyCompany, C=DE"}
}
```

Update Database Users

You can update a database user by providing an updated definition in a JSON file or an input string.

Enter the following command and press `Return`:

```
datasphere dbusers update
    --space <id>
    --databaseuser <id>
    --file-path <name>.json|--input '<stringified-json>'
    [--output <file>.json]
```

Note

We recommend that, for reasons of security, you specify to receive the output in a file.

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--databaseuser <id>`	Enter the name of your database user (*Database User Name Suffix*).
`--file-path <name>`	Enter a path to a file with a `.json` extension containing your database user definition.
`--input '<stringified-json>'`	[optional] Provide your definition in stringified json format instead of via the `--file-path` option. **Note** We recommend that you: • Strip unnecessary spaces, tabs, newline, and return characters (`\t`, `\n`, and `\r`). • Use double-quotes consistently to surround names and values within the JSON code. • Surround the whole string in single-quotes. • Be aware of the escape characters available in your shell environment and when it is necessary to use them.
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

For example, to update the database user JEFF in the SALES space, provide an appropriate json file and enter the following:

```
datasphere dbusers update --space SALES --databaseuser JEFF --file-path
jeff_user_newdef.json --output jeff.json
```

Delete Database Users

You can delete a database user.

Enter the following command and press `Return`:

```
datasphere dbusers delete
    --space <id>
    --databaseuser <id>
    [--force]
```

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--databaseuser <id>`	Enter the name of your database user (*Database User Name Suffix*).

Parameter	Description
`--force`	[optional] Suppress the *Confirm Deletion* dialog and delete the database user without confirmation.

For example, to delete the database user `JEFF` in space `SALES`, and suppress the *Confirm Deletion* dialog, enter:

```
datasphere dbusers delete --space SALES --databaseuser JEFF --force
```

4.4 The Space Definition File Format

Space properties are set and retrieved in the space definition file format and stored as a `.json` file. A space definition file must not exceed 25MB.

- Space Properties [page 50]
- Members [page 53]
- Database Users [page 53]
- HDI Containers [page 53]
- Table, View, and Data Access Control Definitions [page 54]

Space Properties

Users with the *DW Administrator* role can create spaces and set any space properties (see Create a Space) using the following syntax:

```
{
  "<SPACE_ID>": {
    "spaceDefinition": {
      "version": "1.0.4",
      "label": "<Space_Name>",
      "assignedStorage": <bytes>,
      "assignedRam": <bytes>,
    "longDescription": "<Space_Description>",
      "auditing": {
        "dppRead": {
          "retentionPeriod": <days>,
          "isAuditPolicyActive": true|false
        },
        "dppChange": {
          "retentionPeriod": <days>,
          "isAuditPolicyActive": true|false
        }
      },
      "allowConsumption": true|false,
      "enableDataLake": true|false,
      "members": [],
      "dbusers": {},
      "hdicontainers": {},
      "workloadClass": {
        "totalStatementMemoryLimit": {
```

```
          "value": 0,
          "unit": "Gigabyte|Percent"
        },
        "totalStatementThreadLimit": {
          "value": 0,
          "unit": "Counter|Percent"
    }
  }
}
```

> **Note**
>
> Users with the *DW Space Administrator* role cannot create spaces, but they can set space properties except `SPACE_ID`, `assignedStorage` and `assignedRam`.

Parameters are set as follows:

Parameter	Space Property	Description
`<SPACE_ID>`	Space ID	[required] Enter the technical name of the space. Can contain a maximum of 20 uppercase letters or numbers and must not contain spaces or special characters other than _ (underscore). Unless advised to do so, must not contain prefix _SYS and should not contain prefixes: DWC_, SAP_ (See Rules for Technical Names).
`version`	-	[required] Enter the version of the space definition file fomat. This must always be set to `1.0.4`.
`label`	Space Name	Enter the business name of the space. Can contain a maximum of 30 characters, and can contain spaces and special characters. Default value: `<Space_ID>`
`assignedStorage`	Disk (GB)	Enter the amount of storage allocated to the space in bytes. You can enter any value between `100000000` bytes (100MB) and the total storage size available in the tenant. Default value: `2000000000` bytes (2GB) > **Note** > To set no size limit for the space (and disable the *Enable Space Quota* option), enter `0` for both this parameter and `assignedRam`.
`assignedRam`	Memory (GB)	Enter the amount of ram allocated to the space in bytes. You can enter any value between `100000000` bytes (100MB) and the total storage size available in the tenant. Default value: `1000000000` bytes (1GB)
`longDescription`	Description	Enter a description for the space. Can contain a maximum of 4 000 characters.

Parameter	Space Property	Description

`dppRead.isAuditPolicyActive`
`dppRead.retentionPeriod`
`dppChange.isAuditPolicyActive`
`dppChange.retentionPeriod`
	Enable Audit Log for Read Operations
	Keep Logs for <n> Days
	Enable Audit Log for Change Operations
	Keep Logs for <n> Days
		Enter the audit logging policy for read and change operations and the number of days that the logs are retained. you can retain logs for any period between `7` and `10000` days.
		Default values: `false`, `30`, `false`, `30`

`allowConsumption`	Expose for Consumption by Default	Choose the default setting for the *Expose for Consumption* property for views created in this space. Default value: `false`
`enableDataLake`	Use This Space to Access the Data Lake	Enable access to the SAP HANA Cloud data lake. Only one space can connect to the data lake. Default value: `false`
`members`	User Assignment	See Members [page 53].
`dbusers`	Database Users	See Database Users [page 53].
`hdicontainers`	HDI Containers	See HDI Containers [page 53].

For example, the following file will create a new space, with all default properties:

```
{
  "NEWSPACE": {
    "spaceDefinition": {
      "version": "1.0.4"
    }
  }
}
```

> **Note**
>
> If a property is not set it will receive the default value (on creation) or will keep its current value (on update).

This second file will update `NEWSPACE` by modifying the *Space Name* and increasing the *Disk (GB)* and *In-Memory (GB)* allocations:

```
{
  "NEWSPACE": {
    "spaceDefinition": {
      "version": "1.0.4",
      "label": "My New Space",
      "assignedStorage": 6000000000,
      "assignedRam": 5000000000
    }
  }
}
```

> **Note**
>
> The following properties are not supported when creating, reading, or updating spaces using `datasphere`:
>
> - *Connections*
> - *Time Data*
> - *Space Status* and other run-time properties

Members

See Managing Space Users via the Command Line [page 38].

Database Users

The `dbusers` area of the space definition file is deprecated and ignored. To create, update, or delete database users, use the `datasphere dbusers` commands (see Managing Database Users via the Command Line [page 42]).

HDI Containers

Users with the *DW Administrator*, *DW Space Administrator*, or *DW Integrator* role can associate HDI containers to a space (see Exchanging Data with SAP HANA for SQL Data Warehousing HDI Containers) using the following syntax:

```
{
  ...
  "hdicontainers":{
    "<Container_Name>":{}
    },
```

Parameters are set as follows:

Parameter	Space Property	Description
`<Container_Name>`	*HDI Container Name*	[required] Enter the name of an HDI container that is associated with your SAP Datasphere instance and which is not assigned to any other space.

For example, the following file will associate two HDI containers to `NEWSPACE`:

```
{
  "NEWSPACE": {
    "spaceDefinition": {
      "version": "1.0.4",
      "hdicontainers": {
        "MyHDIContainer": {},
```

```
        "MyOtherContainer": {},
      }
    }
  }
}
```

When updating HDI containers, you must always list all HDI containers that you want to have assigned to the space. To delete an HDI container, remove them from the `hdicontainers` section.

Table, View, and Data Access Control Definitions

Users with the *DW Administrator* or *DW Space Administrator* role can add tables and views, and data access controls to a space using the standard CSN syntax (see Core Data Services Schema Notation (CSN)). Users with the *DW Modeler* role can add tables and views.

> **Note**
>
> You can also use the `objects` commands, which support a wider selection of object types, to read and write objects to your space (see Manage Modeling Objects and Tasks via the Command Line [page 60]).

For example, the following file will create a table with two columns in `NEWSPACE`:

```
{
  "NEWSPACE": {
    "spaceDefinition": {
      "version": "1.0.4"
    },
    "definitions": {
      "Products": {
        "kind": "entity",
        "elements": {
          "Product ID": {
            "type": "cds.Integer64",
            "key": true,
            "notNull": true
          },
          "Product Name": {
            "type": "cds.String",
            "length": 5000
          }
        }
      }
    }
  }
}
```

> **Note**
>
> To obtain more complex examples, read existing objects from a space into a file using the `--definitions` option (see Read a Space [page 35]).

4.5 Managing Priorities and Statement Limits for Spaces or Groups via the Command Line

You can use the SAP Datasphere `datasphere` command line interface to set priorities and statement limits for spaces or groups.

Prerequisites

To set priorities and statement limits for spaces or groups, you must have a global role that grants you the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *System Information* (`-RU-----`) - To access the *Configuration* area in the *System* tool.

The *DW Administrator* global role, for example, grants these privileges. For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

To browse the available commands, enter the following and press `Return`:

```
datasphere workload
```

For general information about priorities and statement limits, see Set Priorities and Statement Limits for Spaces or Groups.

List Priorities and Statement Limits for Spaces or Groups

To list priorities and statement limits for spaces or groups available on the tenant, enter the following command and press `Return`:

```
datasphere workload list
       [--output file.json]
```

Complete the parameters as follows:

Parameter	Description
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

Update Priorities and Statement Limits for Spaces or Groups

To update priorities and statement limits for spaces or groups available on the tenant, enter the following command and press `Return`:

```
datasphere workload update
    --file-path <file>.json|--input '<stringified-json>'
```

Complete the parameters as follows:

Parameter	Description

`--file-path <file>.json`	Enter a path to a file with a `.json` extension containing the list of space or group properties to update.
	`{`
	`  "assignment": "SPACE" | "GROUP",`
	`  "workloadClasses": [`
	`  {`
	`  "group": "<GroupName>", // relevant only if assignment is GROUP`
	`  "spaceId": "<SpaceID>", // relevant only if assignment is SPACE`
	`  "priority": <PriorityNumber>,`
	`  "workloadType": "<WorkloadType>",`
	`  "totalStatementThreadLimit": <Value>,`
	`  "totalStatementThreadLimitUnit": "Counter|Percent",`
	`  "totalStatementMemoryLimit": <Value>,`
	`  "totalStatementMemoryLimitUnit": "Gigabyte|Percent"`
	`  }`
	`  ]`
	`}`

In this example, you set the following priorities and statements limits to 2 spaces:

- Set the space SALES_EU the space priority from 1 to 2 and change the workload type from `custom` to `default`.
- Set the space SALES_US the space priority from 5 to 8 and change the workload type from `default` to `custom`, with the totalStatementThreadLimit parameter to 50%, and the totalStatementMemoryLimit parameter to 80%.

Enter the following command and press `Return`:

```
datasphere workload update --file-path update.json
```

Where the file `update.json` contains the following:

```
{
    "assignment": "SPACE",
    "workloadClasses": [
        {
            "group": null,
            "spaceId": "SALES_EU",
            "priority": 2,
            "workloadType": "default",
        },
        {
            "group": null,
```

```
            "spaceId": "SALES_US",
            "priority": 8,
            "workloadType": "custom",
            "totalStatementThreadLimit": 50,
            "totalStatementThreadLimitUnit": "Percent",
            "totalStatementMemoryLimit": 80,
            "totalStatementMemoryLimitUnit": "Percent"
        },
    ]
}
```

In this example, the default priority and statements limits of the group SQL Access are applied: To change the priority and apply custom statement limits, enter the following command and press `Return`:

```
datasphere workload update --file-path update.json
```

Where the file `update.json` contains the following:

```
{
    "assignment": "GROUP",
    "workloadClasses": [
        {
            "group": "SQL Access",
            "spaceId": null,
            "priority": 4,
            "workloadType": "custom",
            "totalStatementThreadLimit": 50,
            "totalStatementThreadLimitUnit": "Percent",
            "totalStatementMemoryLimit": 70,
            "totalStatementMemoryLimitUnit": "Percent"
    }
    ]
}
```

Switch Workload Distribution from Space to Group or Group to Space

You can switch the workload distribution between space and group.

In this example, the workload distribution is organized by space:

```
{
    "assignment": "SPACE",
    "workloadClasses": [
        {
            "group": null,
            "spaceId": "<SpaceID>",
            "priority": <PriorityNumber>,
            "workloadType": "<WorkloadType>",
            "totalStatementThreadLimit": <Value>,
            "totalStatementThreadLimitUnit": "Counter|Percent",
            "totalStatementMemoryLimit": <Value>,
            "totalStatementMemoryLimitUnit": "Gigabyte|Percent"
        }
    ]
}
```

To switch the workload distribution from space to group, enter the following command and press `Return`:

```
datasphere workload update --file-path update.json
```

Where the file `update.json` contains the following:

```
{
    "assignment": "GROUP",
    "workloadClasses": [
    ]
}
```

The default settings are applied to the groups.

The Workload Management File Format

The priorities and statement limits for spaces or groups are set and retrieved in the workload management definition file format and stored as a .json file.

You can set any priority and statement limits properties using the following syntax:

```
{
    "assignment": "SPACE" | "GROUP",
    "workloadClasses": [
        {
            "group": "<GroupName>", // relevant only if assignment is GROUP
            "spaceId": "<SpaceID>", // relevant only if assignment is SPACE
            "priority": <PriorityNumber>,
            "workloadType": "<WorkloadType>",
            "totalStatementThreadLimit": <Value>,
            "totalStatementThreadLimitUnit": "Counter|Percent",
            "totalStatementMemoryLimit": <Value>,
            "totalStatementMemoryLimitUnit": "Gigabyte|Percent"
        }
    {
            "group": "<GroupName>", // relevant only if assignment is GROUP
            "spaceId": "<SpaceID>", // relevant only if assignment is SPACE
            "priority": <PriorityNumber>,
            "workloadType": "<WorkloadType>",
            "totalStatementThreadLimit": <Value>,
            "totalStatementThreadLimitUnit": "Counter|Percent",
            "totalStatementMemoryLimit": <Value>,
            "totalStatementMemoryLimitUnit": "Gigabyte|Percent"
        }
    ]
}
```

Parameters are set as follows:

Parameter	Space Property	Description
`<spaceId>`	Space ID	[required] Choose between Space ID or Group Enter the technical name of the space. Can contain a maximum of 20 uppercase letters or numbers and must not contain spaces or special characters other than _ (underscore). Unless advised to do so, must not contain prefix _SYS and should not contain prefixes: DWC_, SAP_ (See Rules for Technical Names).

Parameter	Space Property	Description
`<group>`	Group	[required] Choose between Space ID or Group Enter the name of the group. To view the list of groups, see Set Priorities and Statement Limits for Spaces or Groups.
`priority`	Priority	Enter the prioritization of the space or group when querying the database. You can enter a value from 1 (lowest priority) to 8 (highest priority).
`workloadType`	Total Statement Memory Limit	Choose between: • `default` - The default configuration provides generous resource limits, while preventing any single space or group from overloading the system. • `custom` - The custom configuration enables you to specify the value for statement limits to set maximum total thread and memory limits that statements running concurrently in the space or group can consume.

`totalStatementMemoryLimit`
 	
`totalStatementMemoryLimitUnit`
	Total Statement Memory Limit
	
	GB/%
		If you've chosen `custom` for the `workloadType`, you can set these parameters. Enter the maximum number (or percentage) of GBs of memory that statements running concurrently in the space or group can consume. You can enter any value or percentage between 0 (no limit) and the total amount of memory available in your tenant.

`totalStatementThreadLimit`
 	
`totalStatementThreadLimitUnit`
	Total Statement Thread Limit
	
	Threads/%
		If you've chosen `custom` for the `workloadType`, you can set these parameters. Enter the maximum number (or percentage) of threads that statements running concurrently in the space or group can consume. You can enter any value or percentage between 0 (no limit) and the total number of threads available in your tenant.

To view the default values for the above parameters, see Set Priorities and Statement Limits for Spaces or Groups.

5 Manage Modeling Objects and Tasks via the Command Line

Users with a modeler role can use the `datasphere` command line interface to list, create, update, and delete modeling objects.

Objects	Command
Modeling Objects	`datasphere objects` See Managing Modeling Objects via the Command Line [page 60].
Tasks	`datasphere tasks` See Running and Managing Tasks via the Command Line [page 74].

5.1 Managing Modeling Objects via the Command Line

You can use the `datasphere` command line interface to list, read, create, update, and delete modeling objects.

This topic contains the following sections:

Prerequisites

To work with modeling objects on the command line, you must have a scoped role that grants you access to a space with the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *Space Files* (`CRUD----`) - To create, read, update, and delete objects in your spaces.
- *Data Warehouse Data Builder* (`CRUD--S-`) - To create, edit and delete *Data Builder* objects.
- *Data Warehouse Data Access Control* (`CRUD----`) - To create, edit and delete these objects.
- *Data Warehouse Business Builder* (`CRUD----`) - To create, edit and delete *Business Builder* objects.
- *Data Warehouse Business Entity* (`CRUD----`) - To create, edit and delete these objects.
- *Data Warehouse Fact Model* (`CRUD----`) - To create, edit and delete these objects.
- *Data Warehouse Consumption Model* (`CRUD----`) - To create, edit and delete these objects.
- *Data Warehouse Authorization Scenario* (`CRUD----`) - To create, edit and delete these objects.

The *DW Modeler* scoped role template, for example, grants these privileges.For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

To browse the available commands, enter the following and press `Return`:

```
datasphere objects <object-type>
```

The following object types are available:

Object Type	Documentation
`remote-tables`	Importing Tables and Views from Sources
`local-tables`	Creating a Local Table
`views`	Creating a Graphical View Creating an SQL View **Note** Both types of views are accessible via the `views` command.
`intelligent-lookups`	Creating an Intelligent Lookup
`er-models`	Creating an Entity-Relationship Model
`types` `contexts`	These objects cannot be created manually in SAP Datasphere, but may be imported as necessary along with remote tables or content packages.
`data-flows`	Creating a Data Flow
`replication-flows`	Creating a Replication Flow

Object Type	Documentation
`transformation-flows`	Creating a Transformation Flow
`task-chains`	Creating a Task Chain
`analytic-models`	Creating an Analytic Model
`business-entities`	Creating a Business Entity
`fact-models`	Creating a Fact Model
`consumption-models`	Creating a Consumption Model
`data-access-controls`	Securing Data with Data Access Controls
`packages`	Creating Packages to Export

> **Note**
>
> You cannot manage folders or add modeling objects to folders with the `create` and `update` commands. For information about adding objects to a package, see Managing Packages via the Command Line [page 78].

List Objects in a Space

You can list all the objects of a particular type in a space and optionally write the output to a file.

Enter the following command and press `Return`:

```
datasphere objects <object-type> list
    --space <id>
    [--technical-names <name>[,...]]
    [--output <file>.json]
    [--select "<property[,...]>"]
    [--filter "<property operator value>[ and|or ...]"]
    [--top <n>]
    [--skip <n>]
```

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--technical-names <name>[,...]`	[optional] Specify the objects to include in your list by technical name, separated by commas.

Parameter	Description
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.
`--select "<property>[,...]"`	[optional] Specify the properties to include in your list, separated by commas, from among the following: • `technicalName` • `businessName` • `type` • `semanticUsage` • `status` • `createdOn` • `changedOn` • `deployedOn` • `createdBy` • `changedBy` • `defaultFileName` - The `.json` filename is calculated by combining the `technicalName` and the `type` and has a maximum length of 250 characters. Default: `technicalName`
`--filter "<property operator value>[ and|or ...]"`	[optional] Specify a filter condition using the standard OData filter syntax. You can use the following comparison operators: • `eq` - Equal to • `ne` - Not equal to • `gt` - Greater than • `lt` - Less than • `ge` - Greater than or equal to • `le` - Less than or equal to Dates must be entered in the format `'YYYY-MM-DD'`. You can combine filter conditions using the `and` and `or` keywords and control priority by grouping conditions with parentheses. For example, to list only objects that were deployed after 31 December 2022, and have a semantic usage of `Fact` or `Dimension`, enter `--filter "deployedOn gt '2022-12-31' and (semanticUsage eq Fact or semanticUsage eq Dimension)"`
`--top <n>`	[optional] List only the first `<n>` objects, up to a maximum of 200. Default: `25`
`--skip <n>`	[optional] Skip the first `<n>` objects.

For example, to list all the views in space `MySpace` showing their technical and business names as well as their semantic usage and status and write them to a file, enter:

```
datasphere objects views list --space MySpace --select
"technicalName,businessName,semanticUsage,status" --output MySpaceViews.json
```

You can filter lists on the following properties:

Property	Description

`technicalName`
 	
`businessName`
	Enter an object name. You must use single or double quotes if the name contains spaces.
	
	For example, to list only the objects with technical name `Table_1` and `Table_2`, enter:
	
	`--filter "technicalName eq Table_1 or technicalName eq Table_2"`

`type`	Enter a valid object type: • `LocalTable` • `RemoteTable` • `View` • `IntelligentLookup` • `ReplicationFlow` • `TransformationFlow` • `DataFlow` • `TaskChain` • `ERModel` • `AnalyticModel` • `DataAccessControl` • `BusinessEntity` • `FactModel` • `ConsumptionModel` For example, to list only views, enter: `--filter "type eq View"`
`semanticUsage`	Enter a valid semantic usage: • `Fact` • `RelationalDataset` • `Dimension` • `Hierarchy` • `HierarchyWithDirectory` • `Text` • `AnalyticalDataset` For example, to list only objects with a semantic usage of `Dimension`, enter: `--filter "semanticUsage eq Dimension"`

Property	Description
`status`	Enter a valid status: • `NotDeployed` • `Deployed` • `ChangesToDeploy` • `DesignTimeError` • `DesignTimeError` • `PendingDeployment` For example, to list only objects with a status of Deployed, enter: `--filter "status eq Deployed"`

`createdOn`
`changedOn`
`deployedOn`
	Enter a date in the format `'YYYY-MM-DD'`.
	For example, to list only objects deployed after 31 December 2022, enter:
	`--filter "deployedOn gt '2022-12-31'"`

`createdBy` `changedBy`	Enter a valid username. For example, to list only objects last changed by `Lisa`, enter: `--filter "changedBy eq Lisa"`

Read Objects

You can read the CSN/JSON definition of an object and optionally write it to a file.

Enter the following command and press `Return`:

```
datasphere objects <object-type> read
    --space <id>
    --technical-name <name>
    [--output <file>.json]
    [--accept <format>]
```

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--technical-name <name>`	Enter the *Technical Name* of the object.

Parameter	Description
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. Enter the option without any value (`--output`) to write the output to a file with the `defaultFileName` for the object. If you do not include this option, the output will be printed to the command line.
`--accept <format>`	[optional] Specify the format to return the object definition in. You can choose between: • `application/vnd.sap.datasphere.object.content+json` - [default] Most complete definition of the object, which may include undeployed design-time changes and unresolved associations (following import of the object). • `application/vnd.sap.datasphere.object.content.design-time+json` - Current design-time version of the object, which may include undeployed changes, but excludes any unresolved associations. • `application/vnd.sap.datasphere.object.content.run-time+json` - Current run-time version of the object, which excludes any undeployed design-time changes.

For example, to read the definition of the remote table `MyTable` in space `MySpace` and write it to the default file name, enter:

```
datasphere objects remote-tables read --space MySpace --technical-name MyTable
--output
```

> ⓘ Note
>
> For local tables with delta capture enabled, you must use the `--accept` option when reading the object, in order to obtain all the delta capture objects in the output file:
>
> ```
> datasphere objects local-tables read
>   --space MySpace
>   --technical-name MyDeltaTable
>   --accept application/vnd.sap.datasphere.object.content.design-time+json
>   --output MyDeltaTable.json
> ```
>
> The file created can then be written back into a space in the normal way:
>
> ```
> datasphere objects local-tables create
>   --space MySpace2
>   --technical-name MyDeltaTable
>   --file-path MyDeltaTable.json
> ```

Create Objects

You can create an object by providing a definition in a JSON file or an input string.

> ⓘ Note
>
> For information about considerations when creating or updating objects, see Object Dependencies [page 70] and Object Definition Syntax [page 72].

Enter the following command and press `Return`:

```
datasphere objects <object-type> create
    --space <id>
    --technical-name <name>
    --file-path <name>.json|--input '<stringified-json>'
    [--custom-validation-options <option[,option]>]    [--save-anyway]
    [--no-deploy]
```

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--technical-name <name>`	Enter the *Technical Name* of the object.
`--file-path <name>`	[optional] Enter a path to a file with a `.json` extension containing your object definition.
`--input '<stringified-json>'`	[optional] Provide your definition in stringified json format instead of via the `--file-path` option. **Note** We recommend that you: • Strip unnecessary spaces, tabs, newline, and return characters (`\t`, `\n`, and `\r`). • Use double-quotes consistently to surround names and values within the JSON code. • Surround the whole string in single-quotes. • Be aware of the escape characters available in your shell environment and when it is necessary to use them. For example, to create a simple table, `T1` in `MySpace`, you could enter: `datasphere objects local-tables create --space MySpace --technical-name T1 --input '{"definitions":{"T1":{"kind":"entity","@EndUserText.label":"T1","@ObjectModel.modelingPattern":{"#":"DATA_STRUCTURE"},"@ObjectModel.supportedCapabilities":[{"#":"DATA_STRUCTURE"}],"elements":{"ID":{"@EndUserText.label":"ID","type":"cds.String","length":100,"key":true,"notNull":true},"Name":{"@EndUserText.label":"Name","type":"cds.String","length":100}}}}}'`
`--custom-validation-options <option[,option]>`	[optional] Provide a comma-separated list of custom validation options: • `allowRevertReleaseState:true` - Allows users with the *Spaces.Update* privilege (included in the *DW Space Administrator* role) to override warnings and import objects with invalid release states (see Releasing Stable Views for Consumption).
`--save-anyway`	[optional] Save the object, even if there are validation warnings.
`--no-deploy`	[optional] Do not deploy the object after saving.

For example, to create a new local table in space `MySpace`, enter:

```
datasphere objects local-tables create --space MySpace --technical-name MyTable
--file-path MyTableDefn.json
```

Update Objects

You can update an object by providing a new definition in a JSON file or an input string.

> **Note**
>
> For information about considerations when creating or updating objects, see Object Dependencies [page 70] and Object Definition Syntax [page 72].

Enter the following command and press `Return`:

```
datasphere objects <object-type> update
    --space <id>
    --technical-name <name>
    --file-path <name>.json|--input '<stringified-json>'
    [--custom-validation-options <option[,option]>]
    [--save-anyway]
    [--no-deploy]
```

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--technical-name <name>`	Enter the *Technical Name* of the object.
`--file-path <name>`	[optional] Enter a path to a file with a `.json` extension containing your object definition.

Parameter	Description
`--input '<stringified-json>'`	[optional] Provide your definition in stringified json format instead of via the `--file-path` option. **Note** We recommend that you: • Strip unnecessary spaces, tabs, newline, and return characters (`\t`, `\n`, and `\r`). • Use double-quotes consistently to surround names and values within the JSON code. • Surround the whole string in single-quotes. • Be aware of the escape characters available in your shell environment and when it is necessary to use them. For example, to create a simple table, `T1` in `MySpace`, you could enter: `datasphere objects local-tables create --space MySpace --technical-name T1 --input '{"definitions":{"T1":{"kind":"entity","@EndUserText.label":"T1","@ObjectModel.modelingPattern":{"#":"DATA_STRUCTURE"},"@ObjectModel.supportedCapabilities":[{"#":"DATA_STRUCTURE"}],"elements":{"ID":{"@EndUserText.label":"ID","type":"cds.String","length":100,"key":true,"notNull":true},"Name":{"@EndUserText.label":"Name","type":"cds.String","length":100}}}}}'`
`--custom-validation-options <option[,option]>`	[optional] Provide a comma-separated list of custom validation options: • `allowRevertReleaseState:true` - Allows users with the *Spaces.Update* privilege (included in the *DW Space Administrator* role) to override warnings and import objects with invalid release states (see Releasing Stable Views for Consumption).
`--save-anyway`	[optional] Save the object, even if there are validation warnings.
`--no-deploy`	[optional] Do not deploy the object after saving.

For example, to update the `MyTable` local table, enter:

```
datasphere objects local-tables update --space MySpace --technical-name MyTable
--file-path MyTableDefn.json
```

Delete Objects

You can delete an object.

Enter the following command and press `Return`:

```
datasphere objects <object-type> delete
    --space <id>
    --technical-name <name>
    [--delete-anyway]
```

```
    [--force]
```

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--technical-name <name>`	Enter the *Technical Name* of the object.
`--delete-anyway`	[optional] Delete the object, even if other objects depend on it.
`--force`	[optional] Suppress the *Confirm Deletion* dialog and delete the object without confirmation.

For example, to delete the local table `MyTable` in space `MySpace`, and suppress the *Confirm Deletion* dialog, enter:

```
datasphere objects local-tables delete --space MySpace --technical-name MyTable
--force
```

Object Dependencies

When creating (see Create Objects [page 66]) or updating (Update Objects [page 68]) objects, any sources or other objects that your object depends on must already be present in the space, or your action will fail.

For example, if `View_A` has two tables (`Table_B` and `Table_C`) and another view (`View_D`) as its sources, then these three objects must all be present in the space before you can create `View_A`.

> **Note**
>
> When you export a view (or other object), via CSN export (see Importing and Exporting Objects in CSN/JSON Files), the exported CSN file contains the view and all its dependencies. CSN files generated in this way cannot be used directly with the command line. You must provide one CSN file for each object (excluding any dependencies) and create the sources of the view before creating the view itself.

The dependencies between object types are as follows:

Object Type	Dependencies
Remote Tables See Import Remote Tables.	A remote table depends on its connection.

Object Type	Dependencies

Local Tables
 	
See Creating a Local Table.
	A local table without delta capture does not have dependencies on any other objects. For local tables with delta capture enabled, you must use the `--accept` option when reading the object, in order to obtain all the delta capture objects in the output file:
	
	`datasphere objects local-tables read --space MySpace --technical-name MyDeltaTable --accept application/ vnd.sap.datasphere.object.content.design-time+json --output MyDeltaTable.json`
	
	The file created can then be written back into a space in the normal way:
	
	`datasphere objects local-tables create --space MySpace2 --technical-name MyDeltaTable --file-path MyDeltaTable.json`

Flows See Creating a Data Flow, Creating a Replication Flow, and Creating a Transformation Flow.	A data flow, replication flow, or transformation flow depends on all its sources and its target tables.
Views See Creating a Graphical View and Creating an SQL View.	A view depends on all its sources and any used data access controls.
Intelligent Lookups See Creating an Intelligent Lookup.	An intelligent lookup depends on its input and lookup entities.
Analytic Models See Creating an Analytic Model.	An analytic model depends on its fact and dimension sources.
E/R Models See Creating an Entity-Relationship Model.	An E/R model depends on the objects that it visualizes.
Data Access Controls See Securing Data with Data Access Controls.	The definition of a data access control contains the definition of its permissions entity. When you export a data access control, the permissions entity is exported too.
Task Chains See Creating a Task Chain.	A task chain depends on the objects that it automates.

Object Type	Dependencies
Business Entities / Business Entity Versions See Creating a Business Entity.	A business entity depends on its source data entity and any authorization scenarios.
Fact Models See Creating a Fact Model.	A fact model depends on all its source fact models and business entities.
Consumption Models See Creating a Consumption Model.	A consumption model depends on all its source fact models and business entities.
Authorization Scenarios See Creating an Authorization Scenario.	An authorization scenario depends on its data access control.
Packages See Creating Packages to Export.	A package depends on all its required packages and the objects it contains.

> **Note**
>
> You cannot manage folders or add modeling objects to folders with the `create` and `update` commands. For information about adding objects to a package, see Managing Packages via the Command Line [page 78].

Object Definition Syntax

You can obtain detailed information about object syntax by reading an appropriate object from your space (see Read Objects [page 65]). The principle sections are:

Section	Description
`definitions`	Contains the structure and core metadata of the object.
`editorSettings`	Contains information about the editor associated with the object including, in the case of graphical objects, a serialization of the diagram.
`sharing`	[tables and views] Contains a list of the spaces to which the object is shared (see Share Objects to Other Spaces [page 73]).
`i18n`	Contains translations of metadata grouped by language (see Translate Model, Entity, and Column Names [page 73]).

Share Objects to Other Spaces

Local tables, remote tables, and views can be shared from their space to other spaces, in which they can be used as sources for flows and views (see Sharing Entities and Task Chains to Other Spaces).

The `sharing` section lists the spaces to which the entity is shared. You can modify this section to add or remove spaces to which the table or viewed is shared.

In this example, the object, a local table, is shared to two spaces, SALES_EU and SALES_US:

```
"sharing": {
    "LocalTable":[
        "targetSpaces": [
            "SALES_EU",
            "SALES_US"
        ]
    ]
}
```

Translate Model, Entity, and Column Names

Business names in your exposed objects can be translated to support display languages in SAP Analytics Cloud (see Translating Metadata for SAP Analytics Cloud).

The `i18n` section contains translations of the model or entity business name, and the business names of its measures and attributes, grouped by language. You can modify this section to edit translations, or add or remove languages and their translations.

In this example, the local table's business name, and the business name of its two attributes are provided in English and French:

```
"i18n": {
    "en":[
        "LocalTable@ENDUSERTEXT.LABEL": "Departments",
        "LocalTable#ID@ENDUSERTEXT.LABEL": "Department ID",
        "LocalTable#ID@ENDUSERTEXT.LABEL": "Department Name"
    ],
    "fr":[
       "LocalTable@ENDUSERTEXT.LABEL": "Départements",
        "LocalTable#ID@ENDUSERTEXT.LABEL": "Numéro de département",
        "LocalTable#ID@ENDUSERTEXT.LABEL": "Nom du département"
    ]
}
```

Modify Generated Objects

Certain objects can be generated in SAP Datasphere. These objects can be modified in only limited ways:

- Time Dimensions - You can only modify business names and values in the `timesettings` section. See Create Time Data and Dimensions.
- Currency Conversion Objects - Only modifications to the business names of these objects are supported. See Enabling Currency Conversion with TCUR* Tables and Views.

5.2 Running and Managing Tasks via the Command Line

You can use the `datasphere` command line interface to run and manage certain tasks.

This topic contains the following sections:

Prerequisites

To work with tasks and task chains on the command line, you must have a scoped role that grants you access to a space with the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *Space Files* (`-R------`) - To read objects in your spaces.
- *Data Warehouse Data Integration* (`-RU-----`) - To run tasks and task chains.

The *DW Integrator* scoped role template, for example, grants these privileges.For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

To browse the available commands, enter the following and press `Return`:

```
datasphere tasks
```

For general information about working with tasks in SAP Datasphere, see Managing and Monitoring Data Integration.

Run Task Chains

You can run, retry, and cancel a task chain, including providing values for input parameters and return the task ID.

To run a task chain, enter the following command and press `Return`:

```
datasphere tasks chains run
     --space <id>
     --object <technical_name>
```

```
      [--input-parameters '<stringified-json>']
```

To retry a task chain, enter the following command and press `Return`:

```
datasphere tasks chains retry
  --space <id>
  --object <id>
```

To cancel a task chain, enter the following command and press `Return`:

```
datasphere tasks chains cancel
  --space <id>
  --log <id>
```

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--object <technical_name>`	Specify the technical name of the task chain to run.
`--object <id>`	Enter the ID of the object.
`--log<id>`	Enter the ID of the log for the run.
`--input-parameters '<stringified-json>'`	[optional] Enter the names and values of input parameters in stringified json format. For example: `'{"CATEGORY":"A","FISCAL_YEAR":"2025"}'`

For example to run the task chain `LoadChain` in space `MySpace`, while setting the parameters `YEAR`and `DEPT` to `2026` and `Sales`respectively, enter:

```
datasphere tasks chains run --space MySpace --object LoadChain --input-
parameters '{"YEAR":"2026","DEPT":"Sales"}'
```

For more information about running and monitoring task chains, see Monitoring Task Chains.

Run Replication Flows

You can operate replication flows and individual replication objects.

To operate a replication flow or replication object, enter the following command and press `Return`:

```
datasphere tasks replication-flows <action>
  --space <id>
  —-technical-name <technical_name>
```

Complete the parameters as follows:

Parameter	Description
`<action>`	Enter the desired action: • `status [--object <id>]` - Obtain the current status of the replication flow. • `run` - Start a replication flow run. • `pause` - Pause a running replication flow. • `resume` - Resume a paused replication flow. • `stop` - Stop the replication flow. • `stop --no-cleanup` - [scheduled flows with delta loads] Stop the replication flow but do not delete any already replicated data. • `pause-object --object <id>` - Pause the specified running replication object. • `resume-object --object <id>` - Resume the specified paused replication object. • `restart-object --object <id>` - Restart the specified replication object.
`--space <id>`	Enter the *Space ID* of the space.
`--technical-name <technical_name>`	Specify the technical name of the replication flow to run.

For example to run the replication flow `RepSales` in space `MySpace`, enter:

```
datasphere tasks replication-flows run --space MySpace --technical-name RepSales
```

To obtain the status of object `SalesHeader` in replication flow `RepSales`, enter:

```
datasphere tasks replication-flows status --space MySpace --technical-name
RepSales --object SalesHeader
```

For more information about running and monitoring replication flows, see Monitoring Flows.

Obtain Task Logs

You can obtain log information for a run at various levels of detail.

To get a list of logs for all the runs of an object, enter the following command and press `Return`:

```
datasphere tasks logs list
  --space <id>
  --object <technical_name>
```

To get the status or details of a particular run, enter the following command and press `Return`:

```
datasphere tasks logs get
  --space <id>
  --log-id <id>
  [--info-level status | details]
```

To get the extended details of a particular run, enter the following command and press `Return`:

```
datasphere tasks logs get-extended
  --space <id>
  --log-id <id>
```

```
  [--accept <format>]
```

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--object <technical_name>`	Enter the technical name of the object.
`--log-id <id>`	Enter the ID of the log for the run.
`--info-level status | details`	[optional] Select whether to return: • `status` - [default] Returns `RUNNING`, `COMPLETED`, or `FAILED`. • `details` - Returns full status information including, for task chains, detailed logs for subtasks.
`--accept <format>`	[optional] Select whether to return: • `application/vnd.sap.datasphere.task.log.status+json` - [default] Return the status as a string. • `application/vnd.sap.datasphere.task.log.status.object+json` - Return the status object if no accept header is specified. • `application/vnd.sap.datasphere.task.log.details+json` - Returns detailed logs of including messages and, for task chains, detailed logs for subtasks. • `application/vnd.sap.datasphere.task.log.details.extended+json` - Return extended logs including all detailed log info and message details.

For more information about reviewing task logs, see Managing and Monitoring Data Integration.

Manage Consent for Scheduled Tasks

You can give or revoke consent for SAP Datasphere to run scheduled tasks on your behalf and obtain your current consent status.

Enter the following command and press `Return`:

```
datasphere tasks consent give|revoke|get
  [--verbose]
```

The commands act as follows:

Command	Description
`give`	Gives consent for SAP Datasphere to run scheduled tasks on your behalf. In verbose mode, returns one of the following response codes: • 201 (consent created) • 400 (consent already given)

Command	Description
`revoke`	Revokes consent. In verbose mode, returns one of the following response codes: • 200 (consent revoked) • 404 (unable to find and revoke consent)
`get`	Returns one of the following: • GIVEN plus data provided and expiry date • EXPIRED

For more information about task consent, see Scheduling Data Integration Tasks.

5.3 Managing Packages via the Command Line

You can use the `datasphere` command line interface to add required packages and objects to a package and to export the package to the Content Network.

This topic contains the following sections:

Prerequisites

To create and export packages, you must have a combination of a global role and a scoped role:

- A global role that grants you the following privileges:
  - *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
  - *Lifecycle* (`-R---MS-`) - To use the *Transport* apps.
- A scoped role that grants you access to a space with the following privileges:
  - *Spaces* (`-RU-----`) - To update your spaces and their properties.
  - *Space Files* (`CRUD----`) - To create, read, update, and delete objects in your spaces.
  - *Data Warehouse Data Builder* (`-RU-----`) - To view and update *Data Builder* objects (and any other relevant object privileges to allow you to update other types of objects contained in the package).

The *DW Space Administrator* role template, for example, grants this combination of privileges. For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

To browse the available commands, enter the following and press `Return`:

```
datasphere objects packages
```

List, Read, Create, Update, and Delete Packages

Use the standard modeling object commands to work with packages:

- List Objects in a Space [page 62]
- Read Objects [page 65]
- Create Objects [page 66]
- Update Objects [page 68]
- Delete Objects [page 69]

Specify the Contents of a Package

A package must specify its required packages and the objects it contains, as well as Content Network metadata.

In the following example, `My_Package` has two required packages and contains three objects. When exported, it will create version `0.0.1` of the package:

```
{
  "repositoryPackages": {
    "My_Package": {
      "requires": [
        "Required_Package_1",
        "Required_Package_2"
      ],
      "_kind": "repositoryPackage",
      "_packageProperties": {
        "@EndUserText.label": "My Package",
        "purpose": "Demo Package",
        "contentDelivery": {
          "category": "myContent",
          "semantic_version": "0.0.1"
        }
      }
    }
  },"repositoryPackagesAttachedObjects": {
    "My_Package":[
            {
              "technicalName": "My_Table"
            },
            {
              "technicalName": "My_View"
            },
            {
              "technicalName": "My_Analytic_Model"
            },
    ]
}
```

Complete your package definition as follows:

`requires`	Specify the list of technical names of any packages that contain objects that the objects you add to your package depend on as part of their lineage.
`_kind`	Must be set to `repositoryPackage`.
`@EndUserText.label`	Enter a descriptive name to help users identify the package.
`purpose`	Provide a business purpose for the package.
`category`	Must be set to `myContent`.
`semantic_version`	Enter the version number of the package in the format `1.0.0` (major version, minor version, patch number). You can re-export your package with the same version number to overwrite the current exported version or change the version to export a package update. The version number must always go up and never down.
`repositoryPackagesAttachedObjects`	Specify the objects contained in the package as an array of JSON objects in the following format: `{ "technicalName": "My_Table" }`

When your package definition is complete, use `create` (see Create Objects [page 66]) or `update` (see Update Objects [page 68]) to write it to your tenant.

Export Packages

You can export a package to the Content Network object and optionally write it to a file.

Enter the following command and press `Return`:

```
datasphere objects packages export
    --space <id>
    --technical-name <name>
    [--allow-overwrite]
```

Complete the parameters as follows:

Parameter	Description
`--space <id>`	Enter the *Space ID* of the space.
`--technical-name <name>`	Enter the *Technical Name* of the package.
`--allow-overwrite`	[optional] Use this option to specify that you want to overwrite the existing version of the package with the same semantic version number. To create a new version of the package, specify a higher semantic version number. If you do not specify this option and have not increased the semantic version number, then your export will fail.

For example, to export a new version of the package `My_Package` in space `MySpace` to the Content Network, enter:

```
datasphere objects packages export --space MySpace --technical-name My_Package
```

6 Managing the Data Marketplace via the Command Line

Users with a modeler role can use the `datasphere` command line interface to manage the Data Marketplace.

Objects	Command
Data Marketplace data providers	`datasphere marketplace providers` See Managing Data Marketplace Data Providers via the Command Line [page 82].
Data Marketplace data products	`datasphere marketplace products` or `datasphere marketplace products-by-provider` See Managing Data Marketplace Data Products via the Command Line [page 101].
Data Marketplace licences	`datasphere marketplace licenses-by-provider` See Managing Data Marketplace Licenses via the Command Line [page 130].
Data Marketplace data product releases	`datasphere marketplace releases` See Managing Data Marketplace Releases via the Command Line [page 143].
Data Marketplace contexts	`datasphere marketplace contexts-by-provider` See Managing Data Marketplace Contexts via the Command Line [page 149].

6.1 Managing Data Marketplace Data Providers via the Command Line

You can use the SAP Datasphere command line interface, `datasphere`, to manage Data Marketplace data providers, including batch updating properties such as contact email addresses or visibility. You can also list all data providers that a user has access to.

This topic contains the following sections:

- datasphere marketplace providers create [page 84]
- datasphere marketplace providers overwrite [page 85]
- datasphere marketplace providers update [page 86]
- datasphere marketplace providers keys generate [page 86]
- datasphere marketplace providers keys list [page 87]
- datasphere marketplace providers keys delete [page 87]

Prerequisites

To manage marketplace objects on the command line, you must have a scoped role that grants you access to a space with the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *Data Warehouse Data Builder* (`CRUD---S`) - To create, edit and delete data providers.
- *Space Files* (`CRUD----`) - To create, read, update, and delete objects in your spaces.
- *Spaces* (`-RU-----`) - To modify the space properties.

The *DW Space Administrator* role template, for example, grants these privileges.For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

To browse the available commands, enter the following and press `Return`:

```
datasphere marketplace providers
```

To get access to a data provider the user needs to be member of the data provider profile. If a `contentAggregatorID` is given as parameter, only data provider profiles that are managed by that content aggregator are returned.

datasphere marketplace providers list

Returns a simplified list of all data providers that the current user has access to.

```
datasphere marketplace providers list
```

If you want a detailed representation of a certain data provider, use the `read` command.

Parameter	Description
`--accept <accept>`	Format to return the content in. Choices: • application/vnd.sap.marketplace.providers.list+json (default) • application/vnd.sap.marketplace.providers.details+json

datasphere marketplace providers read

Reads the metadata of a certain data provider after specifying its UUID or its content aggregator UUID.

```
datasphere marketplace providers read
```

Complete the parameters as follows:

datasphere marketplace providers create

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace providers create

Creates a new data provider or a new managed data provider (managed by a content aggregator) based on a configuration following the data provider definition file format and stored as a `.json` file. See The Data Provider Definition File Format [page 88] for more information.

```
datasphere marketplace providers create
```

Specify the full path to the input `.json` file, for example `C:\temp\mydataproviderdefintion.json`.

To create a managed data provider specify the `contentAggregatorID` in the request body.

> **Note**
>
> Managed providers can only be created if you are a member of the assigned content aggregator profile.

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your data provider definition.
`--input <input>`	Specifies input as string to use for the command (optional).

datasphere marketplace providers overwrite

```
datasphere marketplace providers overwrite
```

> **Note**
>
> If you want to update specific properties only, use the `update` command.

Parameter	Overwrites all properties of the specified data provider with the provided data inDescription
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your data provider definition.
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--input <input>`	Specifies input as string to use for the command (optional).

datasphere marketplace providers update

Updates only selected properties of the specified data provider which are defined in the provided data provider definition file.

```
datasphere marketplace providers update
```

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your data provider definition.
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--input <input>`	Specifies input as string to use for the command (optional).

datasphere marketplace providers keys generate

Generates activation keys for a specified data provider.

```
datasphere marketplace providers keys generate
```

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing the keys.

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--input <input>`	Specifies input as string to use for the command (optional).

datasphere marketplace providers keys list

Returns a list of all the existing activation keys for a specified data provider.

```
datasphere marketplace providers keys list
```

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace providers keys delete

Deletes activation keys for a specified data provider.

```
datasphere marketplace providers keys delete
```

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing the keys.
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--input <input>`	Specifies input as string to use for the command (optional).

6.1.1 The Data Provider Definition File Format

The properties of a data provider definition are set and retrieved in the data provider definition file format and stored as a .json file.

A data provider definition file must not exceed 25MB, and can contain the following information:

Data Provider Properties

Users with the **DW Modeler** role can create data providers and set any data provider properties (see Managing Data Marketplace Data Providers via the Command Line [page 82]) using the following syntax:

Sample Code

```
{
  "name": "<string>",
  "contentAggregatorProviderID": "<string>",
  "logo": "<object>",
  "description": "<string>",
  "homepageUrl": "<string>",
  "linkedinUrl": "<string>",
  "regionalCoverages": [
    "<string>",
    "<string>", ...
  ],
  "dataCategories": [
    "<string>",
    "<string>", ...
  ],
  "industries": [
```

```
    "<string>",
    "<string>", ...
  ],
  "sapApplications": [
    "<string>",
    "<string>", ...
  ],
  "contactEmail": "<string>",
  "sapEmail": "<string>",
  "country_code": "<contry_code>",
  "zipCode": "<string>",
  "city": "<string>",
  "address1": "<string>",
  "address2": "<string>",
  "phoneNumber": "<string>",
  "shipments": [
    "Direct|External|OpenSql"
  ],
  "marketplaceVisibilities": [
  "public|private|internal|UclFormations"
  ]
}
```

Parameters are set as follows:

Technical Parameter	Mapped Name of Data Provider Profile Parameter used in Data Sharing Cockpit	Description
`<name>`	*Name*	The name of the data provider.
`<logo>`	*Company Logo*	The logo of the data provider company. It must be delivered as an object with maximum 5000 characters, Base64 encoded. The following mime-types are supported: bmp, svg+xml, gif, jpeg, png, tiff
`<description>`	*Description*	Share some information about the data provider: the products you offer, your claim and your vision.
`<homepageUrl>`	*Homepage URL*	The URL of the data provider's homepage.
`<linkedinUrl>`	*LinkedIn*	The URL to the data provider's LinkedIn profile.

Technical Parameter	Mapped Name of Data Provider Profile Parameter used in Data Sharing Cockpit	Description
`<regionalCoverages>`	*Regional Coverage*	String values for each region the data is applicable to. Multiple values can be specified. `[Global, Europe, North America, South America, Central America, Asia, Africa, Afghanistan, Aland Islands, Albania, Algeria, American Samoa, Andorra, Angola, Anguilla, Antarctica, Antigua and Barbuda, Argentina, Armenia, Aruba, Australia, Austria, Azerbaijan, Bahamas, Bahrain, Bangladesh, Barbados, Belarus, Belgium, Belize, Benin, Bermuda, Bhutan, Bolivia, Bonaire, Sint Eustatius and Saba, Bosnia and Herzegovina, Botswana, Bouvet Island, Brazil, British Indian Ocean Territory, Brunei Darussalam, Bulgaria, Burkina Faso, Burundi, Cambodia, Cameroon, Canada, Cape Verde, Cayman Islands, Central African Republic, Chad, Chile, China, Christmas Island, Cocos (Keeling) Islands, Colombia, Comoros, Congo, Congo, The Democratic Republic of, Cook Islands, Costa Rica, Cote d'Ivoire, Croatia, Cuba, CuraÃ§ao, Cyprus, Czechia, Denmark, Djibouti, Dominica, Dominican Republic, Egypt, El Salvador, Equatorial`

Technical Parameter	Mapped Name of Data Provider Profile Parameter used in Data Sharing Cockpit	Description
		Guinea, Eritrea, Estonia, Ethiopia, Falkland Islands (Malvinas), Faroe Islands, Fiji, Finland, France, French Guiana, French Polynesia, French Southern Territories, Gabon, Gambia, Georgia, Germany, Ghana, Gibraltar, Greece, Greenland, Grenada, Guadeloupe, Guam, Guatemala, Guernsey, Guinea, Guinea-Bissau, Guyana, Haiti, Heard and Mc Donald Islands, Holy See (Vatican City State), Honduras, Hong Kong, Hungary, Iceland, India, Indonesia, Iran, Islamic Republic of, Iraq, Ireland, Isle of Man, Israel, Italy, Jamaica, Japan, Jersey, Jordan, Kazakstan, Kenya, Kiribati, Korea, Democratic People's Republic of, Korea, Republic of, Kosovo (temporary code), Kuwait, Kyrgyzstan, Lao, People's Democratic Republic, Latvia, Lebanon, Lesotho, Liberia, Libyan Arab Jamahiriya, Liechtenstein, Lithuania, Luxembourg, Macao, Macedonia, Madagascar, Malawi, Malaysia, Maldives, Mali, Malta, Marshall Islands, Martinique, Mauritania, Mauritius, Mayotte, Mexico, Micronesia,

Technical Parameter	Mapped Name of Data Provider Profile Parameter used in Data Sharing Cockpit	Description
		Federated States of, Moldova, Republic of, Monaco, Mongolia, Montenegro, Montserrat, Morocco, Mozambique, Myanmar, Namibia, Nauru, Nepal, Netherlands, Netherlands Antilles, New Caledonia, New Zealand, Nicaragua, Niger, Nigeria, Niue, Norfolk Island, Northern Mariana Islands, Norway, Oman, Pakistan, Palau, Palestinian Territory, Occupied, Panama, Papua New Guinea, Paraguay, Peru, Philippines, Pitcairn, Poland, Portugal, Puerto Rico, Qatar, Republic of Serbia, Reunion, Romania, Russia Federation, Rwanda, Saint BarthŽlemy, Saint Helena, Saint Kitts & Nevis, Saint Lucia, Saint Martin, Saint Pierre and Miquelon, Saint Vincent and the Grenadines, Samoa, San Marino, Sao Tome and Principe, Saudi Arabia, Senegal, Serbia and Montenegro, Seychelles, Sierra Leone, Singapore, Sint Maarten, Slovakia, Slovenia, Solomon Islands, Somalia, South Africa, South Georgia & The South Sandwich Islands, South Sudan, Spain, Sri Lanka, Sudan, Suriname, Svalbard and Jan Mayen, Swaziland, Sweden,

Technical Parameter	Mapped Name of Data Provider Profile Parameter used in Data Sharing Cockpit	Description
		Switzerland, Syrian Arab Republic, Taiwan, Province of China, Tajikistan, Tanzania, United Republic of, Thailand, Timor-Leste, Togo, Tokelau, Tonga, Trinidad and Tobago, Tunisia, Turkey, Turkish Rep N Cyprus, Turkmenistan, Turks and Caicos Islands, Tuvalu, Uganda, Ukraine, United Arab Emirates, United Kingdom, United States, United States Minor Outlying Islands, Uruguay, Uzbekistan, Vanuatu, Venezuela, Vietnam, Virgin Islands, British, Virgin Islands, U.S., Wallis and Futuna, Western Sahara, Yemen, Zambia, Zimbabwe]

Technical Parameter	Mapped Name of Data Provider Profile Parameter used in Data Sharing Cockpit	Description
`<dataCategories>`	*Data Category*	Coded values for each data category. Multiple values can be specified. `[C001 - Benchmarking Data, C011 - Company Data, C021 - Countries, Regions & Cities Data, C031 - Culture & Sports Data, C041 - Environmental & Weather Data, C051 - Finance & Economy Data, C061 - Geospatial Data, C071 - Health Data, C081 - Hospitality, Travel & Tourism Data, C091 - Industry-Specific Data, C101 - Innovation & Trend Data, C111 - Legal & Justice Data, C121 - Market & Consumer Data, C131 - Media & Entertainment Data, C141 - Mobility Data, C151 - Natural Resources & Energy Data, C161 - Political Data, C171 - Product & Services Data, C181 - Public Sector & Society Data, C191 - Science & Technology Data, C201 - Social Media, News & Communication Data, C206 - Sustainability Data, C211 - Transport & Logistics Data, C221 - Web, IoT & Device Data, C231 - Other Data Categories]`

Technical Parameter	Mapped Name of Data Provider Profile Parameter used in Data Sharing Cockpit	Description
`<industries>`	*Industry*	String values for each industry the data is applicable to. Multiple values can be specified. `[All Industries, Energy and Natural Ressources, Energy and Natural Ressources:Building Products, Energy and Natural Ressources:Chemicals, Energy and Natural Ressources:Mill Products, Energy and Natural Ressources:Mining, Energy and Natural Ressources:Oil and Gas, Energy and Natural Ressources:Utilities, Financial Services, Financial Services:Banking, Financial Services:Insurance, Consumer Industries, Consumer Industries:Agribusiness, Consumer Industries:Consumer Products, Consumer Industries:Fashion, Consumer Industries:Life Sciences, Consumer Industries:Retail, Consumer Industries:Wholesale Distribution, Discrete Industries, Discrete Industries:Aerospace and Defense, Discrete Industries:Automotive, Discrete Industries:High`

Technical Parameter	Mapped Name of Data Provider Profile Parameter used in Data Sharing Cockpit	Description
		Tech, Discrete Industries:Industrial Machinery and Components, Service Industries, Service Industries:Airlines, Service Industries:Engineering, Construction, and Operations, Service Industries:Media, Service Industries:Professional Services, Service Industries:Railways, Service Industries:Sports & Entertainment, Service Industries:Telecommunications, Service Industries:Travel and Transportation, Public Services, Public Services:Defense and Security, Public Services:Future Cities, Public Services:Healthcare, Public Services:Higher Education and Research, Public Services:Public Sector]

Technical Parameter	Mapped Name of Data Provider Profile Parameter used in Data Sharing Cockpit	Description
`<sapApplications>`	*SAP Application*	String values for each SAP Application the data is applicable to. Multiple values can be specified. `[HR & People Engagement, HR & People Engagement:Employee in HR and People Engagements, HR & People Engagement:Employee Experience Management, HR & People Engagement:Core HR and Payroll, HR & People Engagement:Talent Management, HR & People Engagement:HR Analytics and Workforce Planning, CRM and Customer Experience, CRM and Customer Experience:Customer Data, CRM and Customer Experience:Marketing, CRM and Customer Experience:Commerce, CRM and Customer Experience:Sales, CRM and Customer Experience:Service, ERP & Finance, ERP & Finance:SAP S/4HANA, ERP & Finance:ERP for Small and Midsize Enterprises, ERP & Finance:Financial Planning and Analysis, ERP & Finance:Accounting and Financial Close, ERP & Finance:Treasury Management, ERP & Finance:Accounts Receivable, Billing & Revenue Management, ERP`

Technical Parameter	Mapped Name of Data Provider Profile Parameter used in Data Sharing Cockpit	Description
		& Finance:Cybersecurity, Governance, Risk & Compliance, Network & Spend Management, Network & Spend Management:Supplier Managemetn, Network & Spend Management:Strategic Sourcing, Network & Spend Management:Procurement, Network & Spend Management:Services Procurement and Contingent Workforce, Network & Spend Management:Selling and Fulfillment, Network & Spend Management:Travel and Expense, Business Technology Plattform, Business Technology Plattform:Database and Data Management, Business Technology Plattform:Application Development and Integration, Business Technology Plattform:Analytics, Business Technology Plattform:Intelligent Technologies, Digital Supply Chain, Digital Supply Chain:Supply Chain Planning, Digital Supply Chain:Supply Chain Logistics, Digital Supply Chain:Manufacturing, Digital Supply Chain:R&D / Engineering, Digital Supply Chain:Asset Management, Experience Management, Experience

Technical Parameter	Mapped Name of Data Provider Profile Parameter used in Data Sharing Cockpit	Description
		`Management:Brand Experience, Experience Management:Customer Experience, Experience Management:Product Experience, Experience Management:Employee Experience ]`
`<contactEmail>`	*Email*	Email address all consumer contacts are sent to if the consumer uses the *Contact Provider* button in the Data Provider page.
`<sapEmail>`	*SAP Email*	An email address for the internal communication between the data provider and SAP to get credentials from your Data Marketplace system. This email is for internal usage, consumers won't see it.
`<country_code>`	*Country*	The country where the data provider is located.
`<zipCode>`	*ZIP Code*	The data provider's ZIP code.
`<city>`	*City*	The city as part of the address.
`<address1>`	*First Address*	The business address of the data provider.
`<address2>`	*Second Address*	A second business address if needed.
`<phoneNumber>`	*Phone Number*	A business telephone number, where customers can reach the data provider.
`<shipments>`	*Data Shipment*	The shipment types which the data provider supports: • `<Direct>` • `<External>` • `<OpenSql>`

Technical Parameter	Mapped Name of Data Provider Profile Parameter used in Data Sharing Cockpit	Description
`<marketplaceVisibilities>`	*Marketplace Visibility*	The visibility of the data provider's profile in Data Marketplace: • `<public>`: The profile and data products will be visible for everyone in the *Public Data Marketplace* context owned by SAP. • `<private>`: The data provider can create and participate in contexts of type Private Data Products, Private Data Exchange, or Data Shop. The profile is only visible to users that are members of the data provider's contexts. • `<internal>`: Internal visibility of the data provider profile is a prerequisite to join or create internal Contexts. An internal context is intended for data products that should only be visible within a company. • `<UclFormations>`: this visibility allows you to create custom data products on a Delta Share runtime.

For example, the following file will create a new data provider definition:

Sample Code

```
{
  "name": "Example Provider",
  "contentAggregatorProviderID": "string",
  "logo": "image object",
  "description": "Lorem Ipsum description of my Provider",
  "homepageUrl": "www.mydataprovidercompany.com",
  "linkedinUrl": "www.linkedin.com/mydataprovidercompany",
  "regionalCoverages": [
    "Germany",
    "France"
  ],
  "dataCategories": [
    "C001",
    "C031"
  ],
  "industries": [
    "Financial Services",
    "Energy and Natural Ressources"
  ],
  "sapApplications": [
    "HR & People Engagement",
    "ERP & Finance"
  ],
  "contactEmail": "max.user@companymail.com",
```

```
  "sapEmail": "max.user@sap.com",
  "country_code": "DE",
  "zipCode": "12345",
  "city": "Walldorf",
  "address1": "Dietmar-Hopp Allee 16",
  "address2": "Address 2",
  "phoneNumber": "+40(0) 151 123456",
  "shipments": [
    "OpenSql"
  ],
  "marketplaceVisibilities": [
    "public"
  ]
}
```

Use the `.json` file format to store the data product definition. It is needed for the upload which is part of the `create` command for example.

> **Note**
>
> You must remove the empty properties from the `.json` file because empty properties cause an error (`422 Unprocessable Entities`).

You can find more information on data provider profiles in the Data Provider's Guide under Maintaining your Data Provider Profile.

6.2 Managing Data Marketplace Data Products via the Command Line

You can use the SAP Datasphere command line interface, `datasphere`, to manage and orchestrate Data Marketplace data products, including batch creation, updates of properties such as pricing, status, or context assignments, and deletion. You can also list all data products that belong to a certain data provider.

This topic contains the following sections:

- datasphere marketplace products-by-provider overwrite [page 109]
- datasphere marketplace products-by-provider update [page 110]
- datasphere marketplace products-by-provider change-lifecycle-status [page 111]

Prerequisites

To manage marketplace objects on the command line, you must have a scoped role that grants you access to a space with the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *Data Warehouse Data Builder* (`CRUD---S`) - To create, edit and delete data providers.
- *Space Files* (`CRUD----`) - To create, read, update, and delete objects in your spaces.
- *Spaces* (`-RU-----`) - To modify the space properties.

The *DW Space Administrator* role template, for example, grants these privileges.For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

To browse the available commands, enter the following and press `Return`:

```
datasphere marketplace products
```

In environments where you are managing data products for multiple providers, you can use the following command:

```
datasphere marketplace products-by-provider
```

> **Note**
>
> New releases of data products cannot be created via the command line. You must use the Data Sharing Cockpit (see The Data Sharing Cockpit).

datasphere marketplace products list

Returns a simple list of all existing data products assigned to your user.

```
datasphere marketplace products list
```

Listed are all data products of all data providers and content aggregators you are a member of.

Parameter	Description
`--accept <accept>`	Format to return the content in. Choices: • application/vnd.sap.marketplace.providers.list+json (default) • application/vnd.sap.marketplace.providers.details+json

datasphere marketplace products read

Lists the properties of a single data product. You need to specify the data products UUID.

```
datasphere marketplace products read
```

Use the `list` command to get all available UUIDs.

Parameter	Description
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

datasphere marketplace products create

Creates a new data product based on a configuration following the data product definition file format and stored as a `.json` file. See The Data Product Definition File Format [page 112] for more information.

```
datasphere marketplace products create
```

Specify the full path to the input `.json` file, for example `C:\temp\mydataproductdefinition.json`.

The new data product is created in status Draft. You can change the status with the command `change-lifecycle-status`.

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your data product definition. Alternative: `--file-path Filename.json`
`--input <input>`	Specifies input as string to use for the command (optional).

datasphere marketplace products overwrite

Overwrites all properties of the specified data product with the provided data in the data product definition file.

```
datasphere marketplace products overwrite
```

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your data product definition. Alternative: `--file-path Filename.json`
`--input <input>`	Specifies input as string to use for the command (optional).
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

Specify the full path to the input `.json` file, for example `C:\temp\mydataproductdefinition.json`.

> **Note**
>
> If you want to update specific properties only, use the `update` command.

datasphere marketplace products update

Updates only selected properties of the specified data product which are defined in the provided data product definition file.

```
datasphere marketplace products update
```

You need to specify the data products UUID.

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your data product definition. Alternative: `--file-path Filename.json`
`--input <input>`	Specifies input as string to use for the command (optional).
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

Specify the full path to the input `.json` file, for example `C:\temp\mydataproductdefinition.json`.

datasphere marketplace products install

Installs a data product for a consumer after checking that:

- The consumer has a valid license (if the data product is not a free data product).
- The product status is listed.
- The shipment type is either "Direct" or "Integrated Delivery".
- The consumer must belong to the context assigned to the data product. If a user doesn't have the required context membership to trigger the installation, a 404 error is returned.

```
datasphere marketplace products install
```

Parameter	Description
`--space-id <space-id>`	Space identifier: this is the technical name of the space. *Example*: "MY_SPACE_1"
`--license-key <license-key>`	License key UUID. Optional: this parameter is required if the data product is a licensed product.
`--update-type <update-type>`	Optional parameter to define whether you want the data product to be updated automatically and immediately ("Immediate"), or manually ("Manual", default value).
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

datasphere marketplace products change-lifecycle-status

With this command you can change the lifecycle status of a data product. A newly created data product is automatically set to status Draft.

```
datasphere marketplace products change-lifecycle-status
```

Select one of the following status from the list:

- Listed
- Delisted
- Deactivated

Than specify the data products UUID.

Parameter	Description
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

Parameter	Description
`--lifecycle-status <lifecycle-status>`	New lifecycle status.

datasphere marketplace products delete

Deletes an existing data product.

```
datasphere marketplace products delete
```

Parameter	Description
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

Confirm the intention to delete the data product.

datasphere marketplace products-by-provider list

Returns a list of data products of a certain data provider.

```
datasphere marketplace products-by-provider list
```

Parameter	Description
`--accept <accept>`	Format to return the content in. Choices: • application/vnd.sap.marketplace.providers.list+json (default) • application/vnd.sap.marketplace.providers.details+json

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace products-by-provider read

Lists the properties of a single data product. You need to specify the technical ID and data provider UUID or content aggregator UUID.

```
datasphere marketplace products-by-provider read
```

Parameter	Description
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace products-by-provider create

Creates a new data product for a specified data provider based on a configuration following the data product definition file format and stored as a `.json` file. See The Data Product Definition File Format [page 112] for more information.

```
datasphere marketplace products-by-provider create
```

Specify the full path to the input `.json` file, for example `C:\temp\mydataproductdefinition.json`.

The new data product is created in status Draft. You can change the status with the command `change-lifecycle-status`.

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your data product definition. Alternative: `--file-path Filename.json`
`--input <input>`	Specifies input as string to use for the command (optional).
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace products-by-provider delete

Deletes an existing data product of a specified data provider.

```
datasphere marketplace products-by-provider delete
```

Confirm the intention to delete the data product.

Parameter	Description
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace products-by-provider overwrite

Overwrites all properties of the specified data product with the provided data in the data product definition file.

```
datasphere marketplace products-by-provider overwrite
```

> **Note**
>
> If you want to update specific properties only, use the `update` command.

Parameter	Description
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your data product definition. Alternative: `--file-path Filename.json`
`--input <input>`	Specifies input as string to use for the command (optional).

datasphere marketplace products-by-provider update

Updates only selected properties of the specified data product which are defined in the provided data product definition file.

```
datasphere marketplace products-by-provider update
```

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your data product definition. Alternative: `--file-path Filename.json`
`--input <input>`	Specifies input as string to use for the command (optional).
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace products-by-provider change-lifecycle-status

With this command you can change the lifecycle status of a data product. A newly created data product is automatically set to status Draft.

```
datasphere marketplace products-by-provider change-lifecycle-status
```

Select one of the following status from the list:

- Listed
- Delisted
- Deactivated

Then specify the data products UUID.

Parameter	Description
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.
`--lifecycle-status <lifecycle-status>`	New lifecycle status.

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

6.2.1 The Data Product Definition File Format

Properties of a data product are set and retrieved in the space definition file format and stored as a `.json` file.

A data product definition file must not exceed 25MB, and can contain the following information:

Data Product Properties

Users with the **DW Modeler** role can create data products and set any data product properties (see Managing Data Marketplace Data Products via the Command Line [page 101]) using the following syntax:

Sample Code

```
{
  "dataProviderProductID": "<string>",
  "contentAggregatorProductID": "<string>",
  "name": "<string>",
  "description": "<string>",
  "space": "<string>",
  "pricingModel": "OneTime|Monthly",
  "pricingDescription": "<string>",
  "price": "<number>",
  "pricePerMonth": "<number>",
  "priceCurrencyCode": "<string>",
  "licenseKeyUrl": "<URL>",
  "regionalCoverages": [
    "<string>",
    "<string>", ...
  ],
  "dataCategories": [
    "<string>",
    "<string>", ...
  ],
  "industries": [
    "<string>",
```

```
    "<string>", ...
  ],
  "sapApplications": [
    "<string>",
    "<string>", ...
  ],
  "shipments": [
    "OpenSql|External|Direct"
  ],
  "productArtifacts": [
    {
      "name": "<string>",
      "dataFilter": "<string>",
      "columns": [
        {
          "name": "<string>"
        }
      ]
    }
  ],
  "dataDocumentation": [
    {
      "name": "<string>",
      "description": "<string>",
      "blobData": "<string>",
      "mimeType": "pptx|ppt|doc|htm|html|pdf|xls|xlsx"
    }
  ],
  "additionalDataDocumentation": [
    {
      "name": "<string>",
      "description": "<string>",
      "blobData": "<string>",
      "mimeType": "pptx|ppt|doc|htm|html|pdf|xls|xlsx"
    }
  ],
  "sampleBlobs": [
    {
      "name": "<string>",
      "description": "<string>",
      "blobData": "<string>",
      "mimeType": "json"
    }
  ],
  "images": [
    {
      "name": "<string>",
      "description": "<string>",
      "blobData": "<string>",
      "mimeType": "bmp|svg|gif|jfif|jpe|jpeg|jpg|png|tif|tiff"
    }
  ],
  "legalDocuments": [
    {
      "name": "<string>",
      "description": "<string>",
      "blobData": "<string>",
      "mimeType": "pptx|ppt|doc|htm|html|pdf|xls|xlsx"
    }
  ],
  "termsOfUse": "string",
  "sizeCategory": "S|M|L|XL|XXL|XXXL",
  "contractType": "Free|LicenseKey|OnRequest",
  "deliveryMode": "OneTime|Full",
  "deliveryPattern": "Daily|Weekly|Biweekly|Monthly|Quarterly|Yearly|Other",
  "deliveryPatternDescription": "<string>",
  "assignedContexts": [
    "<string>",
```

```
    "<string>", ...
  ],
}
```

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
`<dataProviderProductID>`	not set manually in Data Sharing Cockpit	The unique ID of the given product in the system of the data provider.
`<contentAggregatorProductID>`	not set manually Data Sharing Cockpit	The unique ID of the given product in the system of the content aggregator (if the corresponding data provider is managed by a content aggregator).
`<name>`	*Name*	Enter a name for your data product.
`<description>`	*Description*	Enter precise and meaningful information about your data product: for example, what kind of data it contains and what it's used for.
`<space>`	*Artifact Space*	Enter the technical name of the space that contains the referenced data artifacts of the product. This is a mandatory property if your shipment type is set to `Direct`. The user who calls the API must have access to the space (must be assigned to space).
`<pricingModel>`	*Pricing Model*	Select the pricing model for your data product: `One Time` or `Monthly`. This setting is only relevant for contract type `LicenseKey`.
`<pricingDescription>`	*Pricing Description*	Explain the pricing to your consumers. Mandatory setting for `<contractType>` `LicenseKey`, otherwise optional. Verbal description of the pricing model and/or metric and potentially also the price.
`<price>`	*Price*	State the one-time price. Mandatory setting for `<pricingModel>` `OneTime`, otherwise optional. The associated currency also has to be provided in (`priceCurrencyCode`).
`<pricePerMonth>`	*Price per Month*	State the monthly price. Mandatory setting for `<pricingModel>` `Monthly`, otherwise optional. Gives the monthly product subscription price. The associated currency also has to be provided (`priceCurrencyCode`).

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
`<priceCurrencyCode>`	*Currency Code*	Enter the currency code to be used, for example: EUR, USD. Mandatory setting for `<pricingModel>` `Monthly` or `OneTime`, otherwise optional. For valid currency codes see the drop-down list in the Data Sharing Cockpit under Create New Data Provider Profile.
`<licenseKeyUrl>`	*URL for License Key Purchase*	Enter the URL to the shop of the data provider where consumers can purchase licenses for the data product.

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
`<regionalCoverages>`	*Regional Coverage*	String values for each region the data is applicable to. Multiple values can be specified. `[Global, Europe, North America, South America, Central America, Asia, Africa, Afghanistan, Aland Islands, Albania, Algeria, American Samoa, Andorra, Angola, Anguilla, Antarctica, Antigua and Barbuda, Argentina, Armenia, Aruba, Australia, Austria, Azerbaijan, Bahamas, Bahrain, Bangladesh, Barbados, Belarus, Belgium, Belize, Benin, Bermuda, Bhutan, Bolivia, Bonaire, Sint Eustatius and Saba, Bosnia and Herzegovina, Botswana, Bouvet Island, Brazil, British Indian Ocean Territory, Brunei Darussalam, Bulgaria, Burkina Faso, Burundi, Cambodia, Cameroon, Canada, Cape Verde, Cayman Islands, Central African Republic, Chad, Chile, China, Christmas Island, Cocos (Keeling) Islands, Colombia, Comoros, Congo, Congo, The Democratic Republic of, Cook Islands, Costa Rica, Cote d'Ivoire, Croatia, Cuba, CuraÃ§ao, Cyprus, Czechia, Denmark, Djibouti, Dominica, Dominican Republic, Egypt, El Salvador, Equatorial Guinea, Eritrea, Estonia,`

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
		Ethiopia, Falkland Islands (Malvinas), Faroe Islands, Fiji, Finland, France, French Guiana, French Polynesia, French Southern Territories, Gabon, Gambia, Georgia, Germany, Ghana, Gibraltar, Greece, Greenland, Grenada, Guadeloupe, Guam, Guatemala, Guernsey, Guinea, Guinea-Bissau, Guyana, Haiti, Heard and Mc Donald Islands, Holy See (Vatican City State), Honduras, Hong Kong, Hungary, Iceland, India, Indonesia, Iran, Islamic Republic of, Iraq, Ireland, Isle of Man, Israel, Italy, Jamaica, Japan, Jersey, Jordan, Kazakstan, Kenya, Kiribati, Korea, Democratic People's Republic of, Korea, Republic of, Kosovo (temporary code), Kuwait, Kyrgyzstan, Lao, People's Democratic Republic, Latvia, Lebanon, Lesotho, Liberia, Libyan Arab Jamahiriya, Liechtenstein, Lithuania, Luxembourg, Macao, Macedonia, Madagascar, Malawi, Malaysia, Maldives, Mali, Malta, Marshall Islands, Martinique, Mauritania, Mauritius, Mayotte, Mexico, Micronesia, Federated States of,

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
		Moldova, Republic of, Monaco, Mongolia, Montenegro, Montserrat, Morocco, Mozambique, Myanmar, Namibia, Nauru, Nepal, Netherlands, Netherlands Antilles, New Caledonia, New Zealand, Nicaragua, Niger, Nigeria, Niue, Norfolk Island, Northern Mariana Islands, Norway, Oman, Pakistan, Palau, Palestinian Territory, Occupied, Panama, Papua New Guinea, Paraguay, Peru, Philippines, Pitcairn, Poland, Portugal, Puerto Rico, Qatar, Republic of Serbia, Reunion, Romania, Russia Federation, Rwanda, Saint BarthŽlemy, Saint Helena, Saint Kitts & Nevis, Saint Lucia, Saint Martin, Saint Pierre and Miquelon, Saint Vincent and the Grenadines, Samoa, San Marino, Sao Tome and Principe, Saudi Arabia, Senegal, Serbia and Montenegro, Seychelles, Sierra Leone, Singapore, Sint Maarten, Slovakia, Slovenia, Solomon Islands, Somalia, South Africa, South Georgia & The South Sandwich Islands, South Sudan, Spain, Sri Lanka, Sudan, Suriname, Svalbard and Jan Mayen, Swaziland, Sweden, Switzerland, Syrian Arab

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
		Republic, Taiwan, Province of China, Tajikistan, Tanzania, United Republic of, Thailand, Timor-Leste, Togo, Tokelau, Tonga, Trinidad and Tobago, Tunisia, Turkey, Turkish Rep N Cyprus, Turkmenistan, Turks and Caicos Islands, Tuvalu, Uganda, Ukraine, United Arab Emirates, United Kingdom, United States, United States Minor Outlying Islands, Uruguay, Uzbekistan, Vanuatu, Venezuela, Vietnam, Virgin Islands, British, Virgin Islands, U.S., Wallis and Futuna, Western Sahara, Yemen, Zambia, Zimbabwe]

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
`<dataCategories>`	*Data Category*	Coded values for each data category. Multiple values can be specified. `[C001 - Benchmarking Data, C011 - Company Data, C021 - Countries, Regions & Cities Data, C031 - Culture & Sports Data, C041 - Environmental & Weather Data, C051 - Finance & Economy Data, C061 - Geospatial Data, C071 - Health Data, C081 - Hospitality, Travel & Tourism Data, C091 - Industry-Specific Data, C101 - Innovation & Trend Data, C111 - Legal & Justice Data, C121 - Market & Consumer Data, C131 - Media & Entertainment Data, C141 - Mobility Data, C151 - Natural Resources & Energy Data, C161 - Political Data, C171 - Product & Services Data, C181 - Public Sector & Society Data, C191 - Science & Technology Data, C201 - Social Media, News & Communication Data, C206 - Sustainability Data, C211 - Transport & Logistics Data, C221 - Web, IoT & Device Data, C231 - Other Data Categories]`

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
`<industries>`	*Industry*	String values for each industry the data is applicable to. Multiple values can be specified. `[All Industries, Energy and Natural Ressources, Energy and Natural Ressources:Building Products, Energy and Natural Ressources:Chemicals, Energy and Natural Ressources:Mill Products, Energy and Natural Ressources:Mining, Energy and Natural Ressources:Oil and Gas, Energy and Natural Ressources:Utilities, Financial Services, Financial Services:Banking, Financial Services:Insurance, Consumer Industries, Consumer Industries:Agribusiness, Consumer Industries:Consumer Products, Consumer Industries:Fashion, Consumer Industries:Life Sciences, Consumer Industries:Retail, Consumer Industries:Wholesale Distribution, Discrete Industries, Discrete Industries:Aerospace and Defense, Discrete Industries:Automotive, Discrete Industries:High Tech, Discrete`

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
		Industries:Industrial Machinery and Components, Service Industries, Service Industries:Airlines, Service Industries:Engineering, Construction, and Operations, Service Industries:Media, Service Industries:Professional Services, Service Industries:Railways, Service Industries:Sports & Entertainment, Service Industries:Telecommunications, Service Industries:Travel and Transportation, Public Services, Public Services:Defense and Security, Public Services:Future Cities, Public Services:Healthcare, Public Services:Higher Education and Research, Public Services:Public Sector]

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
`<sapApplications>`	*SAP Application*	String values for each SAP Application the data is applicable to. Multiple values can be specified. `[HR & People Engagement, HR & People Engagement:Employee in HR and People Engagements, HR & People Engagement:Employee Experience Management, HR & People Engagement:Core HR and Payroll, HR & People Engagement:Talent Management, HR & People Engagement:HR Analytics and Workforce Planning, CRM and Customer Experience, CRM and Customer Experience:Customer Data, CRM and Customer Experience:Marketing, CRM and Customer Experience:Commerce, CRM and Customer Experience:Sales, CRM and Customer Experience:Service, ERP & Finance, ERP & Finance:SAP S/4HANA, ERP & Finance:ERP for Small and Midsize Enterprises, ERP & Finance:Financial Planning and Analysis, ERP & Finance:Accounting and Financial Close, ERP & Finance:Treasury Management, ERP & Finance:Accounts Receivable, Billing & Revenue Management, ERP & Finance:Cybersecurity,`

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
		Governance, Risk & Compliance, Network & Spend Management, Network & Spend Management:Supplier Managemetn, Network & Spend Management:Strategic Sourcing, Network & Spend Management:Procurement, Network & Spend Management:Services Procurement and Contingent Workforce, Network & Spend Management:Selling and Fulfillment, Network & Spend Management:Travel and Expense, Business Technology Plattform, Business Technology Plattform:Database and Data Management, Business Technology Plattform:Application Development and Integration, Business Technology Plattform:Analytics, Business Technology Plattform:Intelligent Technologies, Digital Supply Chain, Digital Supply Chain:Supply Chain Planning, Digital Supply Chain:Supply Chain Logistics, Digital Supply Chain:Manufacturing, Digital Supply Chain:R&D / Engineering, Digital Supply Chain:Asset Management, Experience Management, Experience Management:Brand

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
		`Experience, Experience Management:Customer Experience, Experience Management:Product Experience, Experience Management:Employee Experience ]`
`<shipments>`	*Data Shipment*	Specify the shipment types you are offering. These shipment types are used if a customer filters offerings for a distinct shipment type. It is particularly important if the data provider has no products listed yet as only the shipment types given here are used for the search. Select one of the following shipment types: • *Direct* The data is copied directly into the space that the consumers select. • *External* The data is delivered by sharing files outside SAP Datasphere. • *OpenSql* The consumers need to provide an Open SQL Schema to you, using the Data Inbox. Once activated, the data is accessible through the Data Builder and the provided schema appears as a source.
`<productArtifacts>`	*Views*	Specify one or multiple views from the given `<space>` that should be contained in the data product. Selecting tables is not supported. If `<shipments>` is set to `Direct` in combination with the lifecycle status `Listed`, this is a mandatory property. For all other shipment types the product artifacts have to be left out. You can also specify filters to be applied to the view.

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
`<dataDocumentation>`	*Data Documentation*	Specify a file containing data documentation. String Array
`<additionalDataDocumentation>`	*Additional Data Documentation*	A Base64 encoded file containing additional data documentation. Supported MIME types: pptx, ppt, doc, htm, html, pdf, xls, xlsx.
`<sampleBlobs>`	*Sample Data*	A `.json` file containing sample data.
`<images>`	*Image*	An image representing the data product. It has to be Base64 encoded and of the following MIME types: bmp, svg, gif, jfif, jpe, jpeg, jpg, png, tif, tiff
`<legalDocuments>`	*Legal Description*	A Base64 encoded file containing legal information. Supported MIME types: pptx, ppt, doc, htm, html, pdf, xls, xlsx.
`<termsOfUse>`	*Terms & Conditions*	Written, unformatted description of the product's terms of use. Terms of use are shown and confirmed by the customer during product activation
`<sizeCategory>`	*Size Category*	This parameter is an information on how many records can be approximately expected in the data set. It should give a rough size estimation also in terms of consumed data volume in the target space. S = Less than 1,000 records M = Between 1,000 and 100,000 records L = Between 100,000 and 1,000,000 XL = between 1,000,000 and 10,000,000 records XXL = between 10,000,000 records and 100,000,000 records XXXL = more than 100,000,000 records

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
`<contractType>`	*Contract Type*	Defines how access to the product is managed/granted. `OnRequest` is the mandatory contract type for all products that have shipment type `OpenSql`" or `ExternalDelivery`. `Free`": Customer can activate the product without any further authorization by the Data Provider. `LicenseKey`: The customer will need a valid license to key to authorize the product activation. The Data Provider can create a license through the Data Sharing Cockpit and give a customer access through activation keys which he can generate for this license. It's also advised to specify the URL to the external shop (`<licenseKeyUrl>`) to purchase license keys, if available.

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
`<deliveryMode>`	*Delivery Mode*	Controls if processes are activated that allow pushing regular data updates to customers of the given product: • `OneTime`: The data set is static and the customer should not expect updates on the data. Still the customer can trigger a refresh of the data if this should be necessary for any reason. • `Full`: The data set is periodically refreshed. The refresh pattern is described by the `<DeliveryPattern>` and `<DeliveryPatternDescription>`. The `<DeliveryMode>` is only relevant if `<ShipmentType>` is set to `Direct`. **Note** If you create a data product in full delivery using the CLI, you cannot create a release with the CLI. Make sure to manually create a release in the *Publishing Management* section of the *Data Sharing Cockpit*. Indeed, a release is mandatory if you want to list your data product for instance.
`<deliveryPattern>`	*Delivery Pattern*	Provides information on the update pattern of the data. Additionally, more precise information (e.g. on the time of the update or on the weekday etc.) can be given in the `<DeliveryPatternDescription>`. This parameter is mandatory for `<DeliveryMode>` = `Full`. Possible values: `<Daily>`, `<Weekly>`, `<Biweekly>`, `<Monthly><Quarterly>`, `<Yearly>`, `<Other>`

Parameter	Mapped Name of Data Product Parameter used in Data Sharing Cockpit	Description
`<deliveryPatternDescription>`	*Delivery Pattern Description*	Free text that provides additional information when exactly a customer can expect an update of the product data. Example: "Data is delivered on the 2nd workday of a week at 7pm CET"
`<assignedContexts>`	*Context*	Names of contexts in which the given data product is visible.

For example, the following code snippet defines a new data product:

Sample Code

```
{
  "dataProviderProductID": "My_Data_Product",
  "contentAggregatorProductID": "CW_4711",
  "name": "Sales Sample Data for SAP",
  "description": "Lorem Ipsum description of my Provider",
  "space": "SAMPLE_SPACE",
  "pricingModel": "OneTime",
  "pricingDescription": "string",
  "price": 599.99,
  "pricePerMonth": 9.99,
  "priceCurrencyCode": "string",
  "licenseKeyUrl": "http://mysapdatalicenseshop.com",
  "regionalCoverages": [
    "Germany",
    "France"
  ],
  "dataCategories": [
    "C001",
    "C031"
  ],
  "industries": [
    "Financial Services",
    "Energy and Natural Ressources"
  ],
  "sapApplications": [
    "HR & People Engagement",
    "ERP & Finance"
  ],
  "shipments": [
    "OpenSql"
  ],
  "productArtifacts": [
    {
      "name": "My_View",
      "dataFilter": "country = 'france' and year = '2019'",
      "columns": [
        {
          "name": "string"
        }
      ]
    }
  ],
  "dataDocumentation": [
    {
      "name": "My_DataDocumentation_File.pdf",
      "description": "My data documentation file",
      "blobData": "string",
```

```
      "mimeType": "doc"
    }
  ],
  "additionalDataDocumentation": [
    {
      "name": "My_Documentation_File.pdf",
      "description": "My additional data documentation file",
      "blobData": "string",
      "mimeType": "doc"
    }
  ],
  "sampleBlobs": [
    {
      "name": "My_File.json",
      "description": "My sample data",
      "blobData": "string",
      "mimeType": "json"
    }
  ],
  "images": [
    {
      "name": "My_Image_File.jpg",
      "description": "My image file",
      "blobData": "string",
      "mimeType": "jpg"
    }
  ],
  "legalDocuments": [
    {
      "name": "My_Legal_File.pdf",
      "description": "My legal document",
      "blobData": "string",
      "mimeType": "doc"
    }
  ],
  "termsOfUse": "The terms of use are complex.",
  "sizeCategory": "S",
  "contractType": "Free",
  "deliveryMode": "OneTime",
  "deliveryPattern": "Weekly",
  "deliveryPatternDescription": "Data is delivered on the 2nd workday of a
week at 7pm CET",
  "assignedContexts": [
    "My Private Context"
  ]
}
```

Use the `.json` file format to store the data product definition. It is needed for the upload which is part of the `create` command for example.

You can find more information on creating data products in the Data Provider's Guide under Creating Marketplace Data Products.

6.3 Managing Data Marketplace Licenses via the Command Line

You can use the SAP Datasphere command line interface, `datasphere`, to create and manage Data Marketplace licenses in the data sharing cockpit.

This topic contains the following sections:

Prerequisites

To manage marketplace objects on the command line, you must have a scoped role that grants you access to a space with the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *Data Warehouse Data Builder* (`CRUD---S`) - To create, edit and delete data providers.
- *Space Files* (`CRUD----`) - To create, read, update, and delete objects in your spaces.
- *Spaces* (`-RU-----`) - To modify the space properties.

The *DW Space Administrator* role template, for example, grants these privileges.For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

For more information about managing your licenses and generating activation keys for consumers to ensure compliant access to your data products, see Managing Licenses.

datasphere marketplace licenses-by-provider list

Returns a simplified list of all the licenses available for your data provider profile.

```
datasphere marketplace licenses-by-provider list
```

Parameter	Description
`--accept <accept>`	Format to return the content in. Choices: • application/vnd.sap.marketplace.licenses.list+json (default) • application/vnd.sap.marketplace.licenses.details+json
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace licenses-by-provider create

Creates a new license.

```
datasphere marketplace licenses-by-provider create
```

The new license is created in status Draft. You can change the status with the command `change-lifecycle-status`.

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your license definition.
`--input <input>`	Specifies input as string to use for the command (optional).

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace licenses-by-provider read

Lists the properties of a license. You must specify the license UUID.

```
datasphere marketplace licenses-by-provider read
```

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--license-identifier <license-identifier>`	License UUID.

datasphere marketplace licenses-by-provider update

Updates properties of a specified license.

```
datasphere marketplace licenses-by-provider update
```

You must specify the license UUID.

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your license definition.
`--input <input>`	Specifies input as string to use for the command (optional).
`--license-identifier <license-identifier>`	License UUID.
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace licenses-by-provider overwrite

Overwrites all the properties of a specified license.

```
datasphere marketplace licenses-by-provider overwrite
```

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your license definition.
`--input <input>`	Specifies input as string to use for the command (optional).
`--license-identifier <license-identifier>`	License UUID.

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

> **Note**
>
> If you want to update specific properties only, use the `update` command.

datasphere marketplace licenses-by-provider delete

Deletes an existing license.

```
datasphere marketplace licenses-by-provider delete
```

Parameter	Description
`--license-identifier <license-identifier>`	License UUID.
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace licenses-by-provider change-lifecycle-status

Changes the lifecycle status of a license from Draft to Active, or vice versa. A newly created license is automatically set to status Draft.

```
datasphere marketplace licenses-by-provider change-lifecycle-status
```

Select the Active (or Draft) status, then specify the license UUID.

Parameter	Description
`--license-identifier <license-identifier>`	License UUID.
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace licenses-by-provider keys list

Returns a list of all the existing activation keys for a specified license.

```
datasphere marketplace licenses-by-provider keys list
```

Parameter	Description
`--license-identifier <license-identifier>`	License UUID.

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace licenses-by-provider keys generate

Generates activation keys for a specified license.

```
datasphere marketplace licenses-by-provider keys generate
```

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--license-identifier <license-identifier>`	License UUID.
`--file-path <path>`	Enter a path to a file with a `.json` extension containing the keys.
`--input <input>`	Specifies input as string to use for the command (optional).

datasphere marketplace licenses-by-provider keys delete

Deletes an activation key from a specified license.

```
datasphere marketplace licenses-by-provider keys delete
```

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--license-identifier <license-identifier>`	License UUID.
`--file-path <path>`	Enter a path to a file with a `.json` extension containing the keys.
`--input <input>`	Specifies input as string to use for the command (optional).

datasphere marketplace licenses-by-provider products add

Add data products to a license.

Parameter	Description
`--license-identifier <license-identifier>`	License UUID.

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your license definition.
`--input <input>`	Specifies input as string to use for the command (optional).

datasphere marketplace licenses-by-provider products delete

Remove data products from a license.

Parameter	Description
`--license-identifier <license-identifier>`	License UUID.
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your license definition.
`--input <input>`	Specifies input as string to use for the command (optional).

datasphere marketplace licenses-by-provider keys assign-users

Assigns users to a license key.

When a consumer installs the product for the first time, the marketplace service verifies the user account's eligibility based on the registered email address. It then generates a key and activates the data product in the background. This allows the consumer to immediately install a data product that requires a license key, without needing to enter the license key.

```
datasphere marketplace licenses-by-provider keys assign-users
```

Parameter	Description
`--license-identifier <license-identifier>`	License UUID.
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

Parameter	Description
`--filePath <user-file>`	Enter a path to a file with a .json extension containing the required information for the users you want to assign (email address, keyUUID, tenant URL). Sample payload: `[{ "email" :"john.doe@sap.com", "keyUUID":"1b9e9f0b-266e-4422-9354-5647c45c5dd5", "tenantURL": "dwc-master-hc-datama-cons.master.hanacloudservices.cloud.sap" }, { "email" :"jane.smith@sap.com", "keyUUID":"e5faa991-6a16-44a5-8380-a1f168d2b93d", "tenantURL": "dwc-master-hc-datama-cons2.master.hanacloudservices.cloud.sap" }]`
`--input <input>`	Specifies input as string to use for the command (optional).

6.3.1 The License Definition File Format

The properties of a license definition are set and retrieved in the space definition file format and stored as a .json file.

A license definition file must not exceed 25MB, and can contain the following information:

License Properties

Users with the **DW Modeler** role can create licenses and set any license properties (see Managing Data Marketplace Licenses via the Command Line [page 130]) using the following syntax:

Sample Code

```
{
"providerUUID": "<string>",
"reference": "<string>",
"company": "<string>",
"domains": ["<string>", "<string>", ...],
"validUntil": "<string>",
"scope": ["<string>", "<string>", ...]
}
```

Parameters are set as follow:

Technical Parameter	Mapped Name of License Parameter used in Data Sharing Cockpit	Description
`<providerUUID>`	*Data Provider*	Enter the unique identifier (UUID) of the data provider (string in UUID format).
`<reference>`	*Reference*	Enter a descriptive name, a contract number, or any other reference information (string).
`<company>`	*Company*	Enter the name of the company that is going to use the license (string).
`<domains>`	*Domains*	Enter one or more domains to restrict the validity of the license (array of strings, each string representing a domain).
`<validUntil>`	*Valid Until*	Enter the last day of the validity of the license (string in date-time format).
`<scope>`	*License Scope*	Select the data products that you want to include in the license scope (array of strings, each string representing a product UUID to be included in the license scope).

For example, the following code snippet defines a new license:

Sample Code

```
{
"providerUUID": "66c9b1e6-1d84-4f1d-8f98-1c1e26f54b8d",
"reference": “Example Reference",
"company": "Example Company",
"domains": ["example.com", "anotherexample.com"],
"validUntil": "2023-02-16T16:20:20.698Z",
"scope": ["dcccaddb-e343-4830-b0e3-8a0233309c24", "77d0a15d-a68b-4e6d-
b7ea-262b16b03cce"]
}
```

Use the `.json` file format to store the license definition. It is needed for the upload which is part of the `create` command for example.

You can find more information on licenses in the Data Provider's Guide under Managing Licenses.

6.4 Managing Data Marketplace Releases via the Command Line

You can use the SAP Datasphere command line interface, `datasphere`, to publish and manage Data Marketplace data product releases.

This topic contains the following sections:

Prerequisites

To manage marketplace objects on the command line, you must have a scoped role that grants you access to a space with the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *Data Warehouse Data Builder* (`CRUD---S`) - To create, edit and delete data providers.
- *Space Files* (`CRUD----`) - To create, read, update, and delete objects in your spaces.
- *Spaces* (`-RU-----`) - To modify the space properties.

The *DW Space Administrator* role template, for example, grants these privileges.For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

The publishing management within Data Marketplace allows controlled data updates and shipment of these updates to your consumers. For each data product update, you create a new release. For more information about Data Marketplace releases, see Publishing New Releases.

datasphere marketplace releases list

Returns a simplified list of all the releases of a data product.

```
datasphere marketplace releases list
```

Parameter	Description
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.
`--accept <accept>`	Format to return the content in. Choices: • application/vnd.sap.marketplace.releases.list+json (default) • application/vnd.sap.marketplace.releases.details+json

datasphere marketplace releases create

Creates a new release.

```
datasphere marketplace releases create
```

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your release definition.
`--input <input>`	Specifies input as string to use for the command (optional).
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

datasphere marketplace releases read

Lists the properties of a release. You must specify the release UUID.

```
datasphere marketplace releases read
```

Parameter	Description
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.
`--release-identifier <release-identifier>`	Release UUID.

datasphere marketplace releases update

Updates properties of a specified release for a data product.

```
datasphere marketplace releases update
```

You must specify the data product UUID and the release UUID.

Parameter	Description
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your release definition.
`--input <input>`	Specifies input as string to use for the command (optional).
`--release-identifier <release-identifier>`	Release UUID.

datasphere marketplace releases overwrite

Overwrites all the properties of a specified release.

```
datasphere marketplace releases overwrite
```

Parameter	Description
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your release definition.
`--input <input>`	Specifies input as string to use for the command (optional).
`--release-identifier <release-identifier>`	Release UUID.

> **Note**
>
> If you want to update specific properties only, use the `update` command.

datasphere marketplace releases delete

Deletes an existing release.

```
datasphere marketplace releases delete
```

Parameter	Description
`--release-identifier <release-identifier>`	Release UUID.
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

datasphere marketplace releases publish

Publishes a release to make data product updates available to the consumers.

```
datasphere marketplace releases publish
```

Parameter	Description
`--release-identifier <release-identifier>`	Release UUID.
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

datasphere marketplace releases lock

Locks a certain release of a data product.

```
datasphere marketplace releases lock
```

Parameter	Description
`--release-identifier <release-identifier>`	Release UUID.
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

datasphere marketplace releases unlock

Unlocks a certain release of a data product.

```
datasphere marketplace releases unlock
```

Parameter	Description
`--release-identifier <release-identifier>`	Release UUID.
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

datasphere marketplace releases lock-all

Locks all the releases of a data product.

```
datasphere marketplace releases lock-all
```

Parameter	Description
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

datasphere marketplace releases unlock-all

Unlocks all the releases of a data product.

```
datasphere marketplace releases unlock-all
```

Parameter	Description
`--data-product-uuid <data-product-uuid>`	Data product UUID. You can use the `datasphere marketplace products list` command to find the available data products and get their UUIDs.

6.4.1 The Release Definition File Format

The properties of a release definition are set and retrieved in the space definition file format and stored as a .json file.

A release definition file must not exceed 25MB, and can contain the following information:

Release Properties

Users with the **DW Modeler** role can create releases and set any release properties (see Managing Data Marketplace Releases via the Command Line [page 143]) using the following syntax:

Sample Code

```
{
"dataContained": "<string>",
"comment": "<string>",
"from": "<string>",
"to": "<string>",
"isLocked": "<boolean>",
"isPublished": "<boolean>"
}
```

Parameters are set as follow:

Technical Parameter	Mapped Name of Release Parameter used in Data Sharing Cockpit	Description
`<dataContained>`	*Data Contained*	Enter a description of the data contained in the release (string).
`<comment>`	*Comment*	Enter a comment for the release (string).
`<from>`	*Date Range*	Enter the start date of the release (string in date-time format).
`<to>`	*Date Range*	Enter the end date of the release (string in date-time format).
`<isLocked>`	*Locked*	Indicate if the release is locked or not (boolean).

Technical Parameter	Mapped Name of Release Parameter used in Data Sharing Cockpit	Description
`<isPublished>`		Indicate if the release is published or not (boolean).

For example, the following code snippet defines a new release:

Sample Code

```
{
"dataContained": "Data Contained Description",
"comment": "Release Comment",
"from": "2023-02-16T16:20:20.698Z",
"to": "2023-02-16T16:20:20.698Z",
"isLocked": false,
"isPublished": false
}
```

Use the `.json` file format to store the release definition. It is needed for the upload which is part of the `create` command for example.

You can find more information on releases in the Data Provider's Guide under Publishing New Releases.

6.5 Managing Data Marketplace Contexts via the Command Line

You can use the SAP Datasphere command line interface, `datasphere`, to create and manage Data Marketplace contexts.

This topic contains the following sections:

Prerequisites

To manage marketplace objects on the command line, you must have a scoped role that grants you access to a space with the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *Data Warehouse Data Builder* (`CRUD---S`) - To create, edit and delete data providers.
- *Space Files* (`CRUD----`) - To create, read, update, and delete objects in your spaces.
- *Spaces* (`-RU-----`) - To modify the space properties.

The *DW Space Administrator* role template, for example, grants these privileges.For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

Use contexts to restrict the visibility of your data provider profile and your data products to selected users only. For more information about Data Marketplace contexts, see Using Contexts to Realize Public, Private, and Internal Offers.

datasphere marketplace contexts-by-provider list

Returns a list of contexts for a data provider.

```
datasphere marketplace contexts-by-provider list
```

Parameter	Description
`--accept <accept>`	Format to return the content in. Choices: • application/vnd.sap.marketplace.contexts.list+json (default) • application/vnd.sap.marketplace.contexts.details+json

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace contexts-by-provider create

Creates a new context for a specified data provider.

```
datasphere marketplace contexts-by-provider create
```

Specify the full path to the input `.json` file, for example `C:\temp\mycontextdefinition.json`.

The new context is created in status Draft. You can activate the context with the command `datasphere marketplace context-by-provider change-lifecycle-status`.

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your context definition.
`--input <input>`	Specifies input as string to use for the command (optional).
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace contexts-by-provider read

Lists the properties of a context for a specific data provider. You must specify the context UUID and the data provider UUID.

```
datasphere marketplace contexts-by-provider read
```

Parameter	Description
`--context-identifier <context-identifier>`	Context UUID.
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"

datasphere marketplace contexts-by-provider update

Updates selected properties of a specified context.

```
datasphere marketplace contexts-by-provider update
```

Parameter	Description
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your context definition.
`--input <input>`	Specifies input as string to use for the command (optional).

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--context-identifier <context-identifier>`	Context UUID.

datasphere marketplace contexts-by-provider overwrite

Overwrites all properties of the specified context with the provided data in the context definition file.

```
datasphere marketplace contexts-by-provider overwrite
```

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--context-identifier <context-identifier>`	Context UUID.
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your context definition.
`--input <input>`	Specifies input as string to use for the command (optional).

datasphere marketplace contexts-by-provider delete

Deletes an existing context of a specified data provider.

```
datasphere marketplace contexts-by-provider delete
```

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--context-identifier <context-identifier>`	Context UUID.

datasphere marketplace contexts-by-provider change-lifecycle-status

Changes the lifecycle status of a context to Active. A newly created context is automatically set to status Draft: before consumers can use their activation keys to join a context in the Data Marketplace, you have to activate the context.

```
datasphere marketplace contexts-by-provider change-lifecycle-status
```

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--context-identifier <context-identifier>`	Context UUID.

datasphere marketplace contexts-by-provider join

With this command you join an existing context. After joining you can see all data products listed in this context.

```
datasphere marketplace contexts-by-provider join
```

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your context definition.
`--input <input>`	Specifies input as string to use for the command (optional).

datasphere marketplace contexts-by-provider leave

With this command you leave a context. If you leave a context, you won't be able to see the data products listed in this context anymore.

```
datasphere marketplace contexts-by-provider leave
```

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your context definition.
`--input <input>`	Specifies input as string to use for the command (optional).

datasphere marketplace contexts-by-provider keys list

Returns a list of all the existing activation keys for a specified context.

```
datasphere marketplace contexts-by-provider keys list
```

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--context-identifier <context-identifier>`	Context UUID.

datasphere marketplace contexts-by-provider keys generate

Generates activation keys for a specified context.

```
datasphere marketplace contexts-by-provider keys generate
```

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--context-identifier <context-identifier>`	Context UUID.
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your context definition.
`--input <input>`	Specifies input as string to use for the command (optional).

datasphere marketplace contexts-by-provider keys delete

Deletes an activation key from a specified context.

```
datasphere marketplace contexts-by-provider keys delete
```

Parameter	Description
`--provider-identifier <provider-identifier>`	Provider identifier. You can either: • Provide the UUID of the data provider or of the content aggregator. • Provide the UUID of a content aggregator and a `contentAggregatorProviderID` separated by a colon. Example: List ["0b0dfe8e-f974-407d-8987-6d863b2c5e83", "f97dfe8e-f9a4-4d2d-8187-ed863bdc5e88:rubix-b2b-risk-assessment-amp-monitoring-solutions-all-cou-rubix-data-sciences"]"
`--context-identifier <context-identifier>`	Context UUID.
`--file-path <path>`	Enter a path to a file with a `.json` extension containing your context definition.
`--input <input>`	Specifies input as string to use for the command (optional).

6.5.1 The Context Definition File Format

The properties of a context definition are set and retrieved in the space definition file format and stored as a .json file.

A context definition file must not exceed 25MB, and can contain the following information:

Context Properties

Users with the **DW Modeler** role can create contexts and set any context properties (see Managing Data Marketplace Contexts via the Command Line [page 149]) using the following syntax:

Sample Code

```
{
"contextOwnerUUID": "<string>",
"contextUUID": "<string>",
```

```
"contextType": "<string>",
"contextName": "<string>",
"description": "<string>",
"logo": {
"name": "<string>",
"blobData": "<string>",
"mimeType": "<string>"
},
"status": "<string>",
"domains": ["<string>", "<string>", ...],
}
```

Parameters are set as follow:

Technical Parameter	Mapped Name of License Parameter used in Data Sharing Cockpit	Description
`<contextOwnerUUID>`		Enter the unique identifier of the context owner (string in UUID format).
`<contextUUID>`		Enter the unique identifier of the context (string in UUID format).
`<contextType>`	*Type*	Select one of the following types (string): • Public Data Marketplace: `<PublicDataMarketplace>` • Data Shop: `<DataShop>` • Private Data Products: `<PrivateDataProducts>` • Private Data Exchange: `<PrivateDataExchange>` • Internal Data Marketplace: `<InternalDataMarketPlace>`
`<contextName>`	*Name*	Enter a unique name for the context (string).
`<description>`	*Description*	Enter a description for the context (string).

Technical Parameter	Mapped Name of License Parameter used in Data Sharing Cockpit	Description
`<logo>`		Define the logo of the context. It is an object with the following properties: • `<name>`: the file name of the image, including its extension (string). • `<blobData>`: the Base64 encoded data of the image (string). • `<mimeType>` the MIME type of the image (string). Possible values: `<image/jpeg>`, `<image/png>`, `<image/bmp>`, `<image/gif>`, `<image/svg+xml>`, `<image/tiff>`.
`<status>`		Define the status of the context (string): `<Draft>`, `<Active>`, `<Expired>`.
`<domains>`	Domains	Enter one or more domains (array of strings, each string representing a domain).

For example, the following code snippet defines a new context:

Sample Code

```
{
"contextOwnerUUID": "66c9b1e6-1d84-4f1d-8f98-1c1e26f54b8d",
"contextUUID": "b7e4e6a3-4e1c-4e8f-a7a8-6e1e26f54b8d",
"contextType": "PrivateDataExchange",
"contextName": "My Context",
"description": "My Context Description",
“logo”: {
“name”:”My_file.jpg”,
“blobData”: ”YXNkYXNkYXNkc2RhZA==",
“mimeType”:”image/jpeg”
},
"status": "Draft",
"domains": ["example.com", "anotherexample.com"]
}
```

Use the `.json` file format to store the context definition. It is needed for the upload which is part of the `create` command for example.

You can find more information on contexts in the Data Provider's Guide under Using Contexts to Realize Public, Private, and Internal Offers.

7 Manage Connectivity via the Command Line

Users with an administrator role can use the `datasphere` command line interface to manage TLS server certificates. Users with an integrator role can list, read, validate, and delete connections and can, additionally, create and edit connections for specific connection types.

Objects	Command
Certificates	`datasphere configuration certificates` See Managing TLS Server Certificates via the Command Line [page 161].
Connections	`datasphere spaces connections` See Managing Connections via the Command Line [page 163].

7.1 Managing TLS Server Certificates via the Command Line

You can use the `datasphere` command line interface to list, upload, and delete TLS server certificates.

This topic contains the following sections:

- Prerequisites [page 161]
- List TLS Certificates [page 162]
- Upload TLS Certificates [page 162]
- Delete TLS Certificates [page 163]

Prerequisites

To manage TLS Server Certificates on the command line, you must have a global role that grants you the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *System Information* (`-RU----`) - To access the *Configuration* area in the *System* tool.

The *DW Administrator* role template, for example, grants these privileges. For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

To browse the available commands, enter the following and press `Return`:

```
datasphere configuration certificates
```

For information about managing certificates in SAP Datasphere, see Manage Certificates.

List TLS Certificates

You can list all TLS server certificates uploaded to SAP Datasphere.

Enter the following command and press `Return`:

```
datasphere configuration certificates list
```

Upload TLS Certificates

You can upload a TLS server certificate.

Enter the following command and press `Return`:

```
datasphere configuration certificates upload
    --description <description>
    --file-path<path> | --input <input>
```

Complete the parameters as follows:

Parameter	Description
`--description <description>`	Enter a description to provide intelligible information on the certificate, for example to point out to which connection type the certificate applies.
`--file-path<path>`	[optional] Enter a path to a certificate file with a supported file extension - .pem (privacy-enhanced mail), .crt, or .cer.
`--input <input>`	[optional] Provide your certificate definition in stringified json format instead of via the `--file-path` option.

For example, to upload a TLS Server certificate for SAP SuccessFactors connections, enter:

```
datasphere configuration certificates upload --description SAP Success Factors
--file-path MySAPSuccssFactorsCertificate.pem
```

Delete TLS Certificates

You can delete a certificate.

Enter the following command and press `Return`:

```
datasphere configuration certificates delete
    --fingerprint <fingerprint>
```

Complete the parameters as follows:

Parameter	Description
`--fingerprint <fingerprint>`	Enter the fingerprint of the certificate that you want to delete. You can retrieve the fingerprint via the command `datasphere configuration certificates list`.

7.2 Managing Connections via the Command Line

You can use the `datasphere` command line interface to read, list, validate, create, edit, and delete connections.

This topic contains the following sections:

Prerequisites

To work with connections on the command line, you must have a scoped role that grants you access to a space with the following privileges:

- *Data Warehouse General* (`-R------`) - To access SAP Datasphere.
- *Data Warehouse Connection* (`CRUD----`) - To create, edit, validate, or delete a connection.
- *Space Files* (`CRUD----`) - To create, read, update, and delete objects in your spaces.

The *DW Space Administrator* and *DW Integrator* role templates, for example, grant these privileges. For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

To specify another location ID than the default location when using Cloud Connector for your connection, you must have a role that grants the following privilege:

- *Connection* (`-R------`)

If you need to select a location ID, ask your tenant administrator to assign your user to a global role (with license type SAP Datasphere) that includes the *Connection.Read* privilege. For more information, see Privileges and Permissions and Standard Roles Delivered with SAP Datasphere.

You must, in addition:

- Install the `datasphere` command line interface (see Install or Update the SAP Datasphere Command Line Interface [page 7]).
- Log in to your SAP Datasphere tenant (see Log into the Command Line Interface via an OAuth Client [page 8]).

To browse the available commands, enter the following and press `Return`:

```
datasphere spaces connections
```

For information about working with connections in SAP Datasphere, see Integrating Data via Connections.

Connection Types Supported for Creating and Editing Connections

Creating and editing connections via the command line is supported for the following connection types:

Connection Type	Type ID in the Command Line	More Information
Amazon Athena	ATHENA	Amazon Athena (Connection Definition File Format) [page 169]
Amazon Redshift	REDSHIFT	Amazon Redshift (Connection Definition File Format) [page 171]
Amazon Simple Storage Service	S3	Amazon Simple Storage Service (Connection Definition File Format) [page 171]
Apache Kafka	KAFKA	Apache Kafka (Connection Definition File Format) [page 172]
Cloud Data Integration	CDI	Cloud Data Integration (Connection Definition File Format) [page 173]
Confluent	CONFLUENT	Confluent (Connection Definition File Format) [page 174]
Generic JDBC	GENERICJDBC	Generic JDBC (Connection Definition File Format) [page 176]
Generic OData	ODATA	Generic OData (Connection Definition File Format) [page 176]
Generic SFTP	SFTP	Generic SFTP (Connection Definition File Format) [page 177]
Google BigQuery	BIGQUERY	Google BigQuery (Connection Definition File Format) [page 178]

Connection Type	Type ID in the Command Line	More Information
Google Cloud Storage	GCS	Google Cloud Storage (Connection Definition File Format) [page 179]
Hadoop Distributed File System	HDFS	Hadoop Distributed File System (Connection Definition File Format) [page 179]
Microsoft Azure Blob Storage	WASB	Microsoft Azure Blob Storage (Connection Definition File Format) [page 180]
Microsoft Azure Data Lake Store Gen2	ADL	Microsoft Azure Data Lake Store Gen2 (Connection Definition File Format) [page 181]
Microsoft Azure SQL Database	AZURESQL	Microsoft Azure SQL Database (Connection Definition File Format) [page 182]
Microsoft SQL Server	MSSQL	Microsoft SQL Server (Connection Definition File Format) [page 182]
Oracle	ORACLEDB	Oracle (Connection Definition File Format) [page 183]
SAP ABAP	ABAP	SAP ABAP (Connection Definition File Format) [page 184]
SAP BW	SAPBW	SAP BW (Connection Definition File Format) [page 185]
SAP BW/4HANA Model Transfer	SAPBWMODELTRANSFER	SAP BW/4HANA Model Transfer (Connection Definition File Format) [page 186]
SAP ECC	SAPECC	SAP ECC (Connection Definition File Format) [page 187]
SAP HANA	HANA	SAP HANA (Connection Definition File Format) [page 188]
SAP HANA Cloud, Data Lake Files	HDL_FILES	SAP HANA Cloud, Data Lake Files (Connection Definition File Format) [page 189]
SAP HANA Cloud, Data Lake Relational Engine	HDLDB	SAP HANA Cloud, Data Lake Relational Engine (Connection Definition File Format) [page 190]
SAP SuccessFactors	SAPSF	SAP SuccessFactors (Connection Definition File Format) [page 190]
SAP S/4HANA Cloud	SAPS4HANACLOUD	SAP S/4HANA Cloud (Connection Definition File Format) [page 193]
SAP S/4HANA On-Premise	SAPS4HANAOP	SAP S/4HANA On-Premise (Connection Definition File Format) [page 194]

> **Note**
>
> Connections based on custom connection types are not supported by the SAP Datasphere command line interface. You cannot list, read, create, edit, delete, or validate these connections via the command line interface.

List Connections in a Space

You can list all the connections in a space and optionally write the output to a file.

Enter the following command and press `Return`:

```
datasphere spaces connections list
                --space <space>
                [--accept <accept>]
                [--details]
                [--name]
                [--features ]
                [--top <n>]
                [--skip <n>]
                [--output file.json]
```

Complete the parameters as follows:

Parameter	Description
`--space <space>`	Enter the *Space ID* of the space.
`--accept <accept>`	[optional] Specify the format to return the connections definition in. You can choose between: • `"application/vnd.sap.datasphere.space.connections.list+json"` - [default] • `"application/vnd.sap.datasphere.space.connections.details+json"`
`--details`	[optional] List the connections with all their details except for credentials.
`--name`	[optional] List only the connections' technical names.
`--features`	[optional] List the connections with their technical names and their information about enabled and disabled features.
`--top <n>`	[optional] List only the first `<n>` connections (according to their creation date), up to a maximum of 200. Default: `10`
`--skip <n>`	[optional] Skip the first `<n>` connections. Default: `0`
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

For example, to list all the connections in space `MySpace` showing their technical and business names as well as their type id, creation date, creator, and replication status and write them to a file, enter:

```
datasphere spaces connections list --space MySpace --output
MySpaceConnections.json
```

Read Connection Details

You can read the JSON definition of a connection (without its credentials) in a space and optionally write the output to a file.

Enter the following command and press `Return`:

```
datasphere spaces connections get
    --space <space>
    --name <name>
    [--output file.json]
```

Complete the parameters as follows:

Parameter	Description
`--space <space>`	Enter the *Space ID* of the space.
`--name <name>`	[optional] Enter the technical name of the connection.
`--output <file>.json`	[optional] Enter a path to a `.json` file to write the output to. If you do not include this option, the output will be printed to the command line.

For example, to read the definition of the connection `MyConnection` in space `MySpace` write it to the default file name, enter:

```
datasphere spaces connections list --space MySpace --name MyConnection --output
MySpaceConnections.json
```

Create Connections

You can create a connection in a space by providing a definition in a JSON file or an input string. For an overview of supported connection types and information about the required file format, see Connection Types Supported for Creating and Editing Connections [page 164].

Enter the following command and press `Return`:

```
datasphere spaces connections create
    --space <space>
    --type-id <connection type id>
    --file-path<path> | --input <input>
```

Complete the parameters as follows:

Parameter	Description
`--space <space>`	Enter the *Space ID* of the space.
`--type-id <connection type id>`	Enter the connection type id (see Connection Types Supported for Creating and Editing Connections [page 164]).
`--file-path<path>`	[optional] Enter a path to a file with a .json extension containing your connection definition.
`--input <input>`	[optional] Provide your connection definition in stringified json format instead of via the `--file-path` option.

For example, to create the SAP SuccessFactors connection `MyConnection` in space `MySpace`, enter:

```
datasphere spaces connections create --space MySpace --type-id SAPSF --file-path
MySFConnectionDefinition.json
```

Validate Connections

You can validate a connection in a space.

Enter the following command and press `Return`:

```
datasphere spaces connections validate
    --space <space>
    --name <name>
```

Complete the parameters as follows:

Parameter	Description
`--space <space>`	Enter the *Space ID* of the space.
`--name <name>`	Enter the technical name of the connection.

For example, to validate the connection `MyConnection` in space `MySpace`, enter:

```
datasphere spaces connections validate --space MySpace --name MyConnection
```

Edit Connections

You can edit a connection in a space by providing a new definition in a JSON file or an input string. For an overview of supported connection types and information about the required file format, see Connection Types Supported for Creating and Editing Connections [page 164].

> **Note**
>
> The update file must include the existing optional parameters that you have used when creating the connection, otherwise the parameters will be set to their default values.

Enter the following command and press `Return`:

```
datasphere spaces connections edit
    --space <space>
    --name <name>
    --file-path<path> | --input <input>
```

Complete the parameters as follows:

Parameter	Description
`--space <space>`	Enter the *Space ID* of the space.
`--name <name>`	Enter the technical name of the connection.
`--file-path<path>`	[optional] Enter a path to a file with a .json extension containing your connection definition.
`--input <input>`	[optional] Provide your connection definition in stringified json format instead of via the `--file-path` option.

For example, to edit the SAP SuccessFactors connection `MyConnection` in space `MySpace`, enter:

```
datasphere spaces connections edit --space MySpace --name MySFConnection --file-
path MySFConnectionDefinition.json
```

Delete Connections

You can delete a connection in a space.

Enter the following command and press `Return`:

```
datasphere spaces connections delete
    --space <space>
    --name <name>
    [--force]
```

Complete the parameters as follows:

Parameter	Description
`--space <space>`	Enter the *Space ID* of the space.
`--name <name>`	Enter the technical name of the connection.
`--force`	[optional] Suppress the *Confirm Deletion* dialog and delete the connection without confirmation.

For example, to delete the connection `MyConnection` in space `MySpace` and suppress the *Confirm Deletion* dialog, enter:

```
datasphere spaces connections delete --space MySpace --name MyConnection --force
```

7.3 Amazon Athena (Connection Definition File Format)

Amazon Athena connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Amazon Athena connections and set any connections properties (see Amazon Athena Connections) using connection type specific syntax. To obtain detailed information about

the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax:

Sample Code

```
{
    "name": "<technical name>",
    "region": "us-east-2",
    "workgroup": "primary",
    "authType": "APIKey",
    "accessKey": "<access key>",
    "secretKey": "<secret key>"
}
```

Parameters are set as follows:

Parameter	Connection Property	Description
`name`	*Technical Name*	[required] Enter the technical name of the connection. The technical name can only contain alphanumeric characters and underscores (_). It cannot start or end with underscore (_). The name must be unique within the space. **Note** Once the object is saved, the technical name can no longer be modified.
`region`	*Region*	[required] Enter the AWS region in your Amazon Athena regional endpoint that you use to make your requests. For example, `us-west-2`.
`workgroup`	*Workgroup*	[required] Enter the name of the workgroup. Amazon Athena uses workgroups to control query access and costs. The default workgroup is `primary`.
`authType`	n/a	[required] Enter **`APIKey`**.
`accessKey`	*Access Key*	[required] Enter the access key ID of the user that the application must use to authenticate.
`secretKey`	*Secret Key*	[required] Enter the secret access key of the user that the application must use to authenticate.
`businessName`	*Business Name*	[optional] Enter a descriptive name to help users identify the object. This name can be changed at any time.
`description`	*Description*	[optional] Provide more information to help users understand the object.
`package`	*Package*	[optional] Enter an existing package to facilitate transport between tenants (see Transporting Content Between Tenants). Default value: **none**

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.4 Amazon Redshift (Connection Definition File Format)

Amazon Redshift connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Amazon Redshift connections and set any connections properties (see Amazon Redshift Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax:

Sample Code

```
{
    "authType": "Basic",
      "name": "<technical name>",
      "databaseName": "<database name>",
      "host": "<host name>",
      "port": "<port number>",
      "security": {
          "useSSL": "true",
          "validateCertificate": "true"
    },
      "username":"<user name>",
      "password": "<password>",
      "remoteTables": {
        "dataProvisioningAgent": "<Data Provisioning Agent name>"
    }
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.5 Amazon Simple Storage Service (Connection Definition File Format)

Amazon Simple Storage Service connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Amazon Simple Storage Service connections and set any connections properties (see Amazon Simple Storage Service Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for encryption with AWS key management service key::

Sample Code

```
{
    "name": "<technical name>",
    "authType": "<description>",
    "protocol": "https",
    "endpoint": "<URL of the Amazon S3 server without protocol prefix>",
    "accessKey": "<access key>",
    "secretKey": "<secret key>",
    "rootPath": "<root path starting with character slash>",
    "encryptionKeyType": "AWS-KMS",
    "encryptionKeyARN": "<KMS key ARN>",
    "assumeRole": "true",
    "roleARN": "<role ARN>",
    "serverSideEncryption": "true",
    "durationSeconds": 3000
}
```

Example syntax with parameters for S3-managed encryption and using Assume Role::

Sample Code

```
{
    "name": "<technical name>",
    "description": "<description>",
    "authType": "APIKey",
    "protocol": "https",
    "endpoint": "<URL of the Amazon S3 server without protocol prefix>",
    "accessKey": "<access key>",
    "secretKey": "<secret key>",
    "rootPath": "<root path starting with character slash>",
    "encryptionKeyType": "S3-MANAGED",
    "assumeRole": "true",
    "roleARN": "<role ARN>",
    "serverSideEncryption": "true",
    "roleSessionName": "<name to uniquely identify the assumed role session>",
    "durationSeconds": 3600,
    "externalId": "<external ID>",
    "rolePolicy": "<IAM policy in JSON format>"
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.6 Apache Kafka (Connection Definition File Format)

Apache Kafka connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Apache Kafka connections and set any connections properties (see Apache Kafka Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for authentication type *Kerberos with Username and Password*::

Sample Code

```
{
    "KAFKABrokers": "<comma-separated list of brokers in the format
host:port>",
    "name": "<technical name>",
    "authType": "SASLKerberosNamedUser",
    "username": "<user name>",
    "password": "<password>",
    "kerberosConfig": "<content of the krb5.conf configuration file>",
      "kerberosRealm": "<realm defined for the Kafka Kerberos broker>",
      "kerberosServiceName": "<Kerberos service used by the Kafka broker>",
      "mtls": "false"
  }
```

Example syntax with parameters for an Apache Kafka cluster that is behind the firewall and no authentication::

Sample Code

```
{
    "KAFKABrokers":"<comma-separated list of brokers in the format
host:port>",
    "name": "<technical name>",
    "authType": "NoAuth",
    "cloudConnector": {
        "useCloudConnector": "true",
        "cloudConnectorLocation": "",
        "virtualDestination": "auto"
    },
    "tls": "false"
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.7 Cloud Data Integration (Connection Definition File Format)

Cloud Data Integration connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Cloud Data Integration connections and set any connections properties (see Cloud Data Integration Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for basic authentication::

Sample Code

```
{
    "name": "<technical name>",
    "businessName": "<business name>",
    "authType": "Basic",
    "username":"<user name>",
    "password":"<password>",
    "url":"<protocol>://<host><service path>",
    "remoteTables": {
        "dataProvisioningAgent": "<Data Provisioning Agent name>"
    }
}
```

Example syntax with parameters for authentication with X.509 client certificates::

Sample Code

```
{
    "name": "<technical name>",
    "businessName": "<business name>",
    "authType": "ClientCertificate",
    "x509ClientCertificate": "-----BEGIN CERTIFICATE-----..-----END
CERTIFICATE-----",
    "x509ClientPrivateKey": "-----BEGIN PRIVATE KEY-----..-----END PRIVATE
KEY-----",
    "url":"<protocol>://<host><service path>",
    "remoteTables": {
        "dataProvisioningAgent": "<Data Provisioning Agent name>"
    }
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.8 Confluent (Connection Definition File Format)

Confluent connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Confluent connections and set any connections properties (see Confluent Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for Confluent Cloud::

Sample Code

```
{
    "name": "<technical name>",
    "description": "<description>",
```

```
    "systemType": "confluentCloud",
    "authType": "PLAIN",
    "KAFKABrokers":"<comma-separated list of brokers in the format
host:port>",
    "username": "<user name>",
    "password": "<password>",
    "schemaRegistry": {
        "url": "<URL of the Schema Registry service in the format protocol://
host:port>",
        "authType": "Basic",
        "username": "<user name>",
        "password": "<password>"
    }
 }
```

Example syntax with parameters for Confluent Platform::

Sample Code

```
{
  "name": "<technical name>",
  "description": "<description>",
  "systemType": "confluentPlatform",
  "KAFKABrokers": "<comma-separated list of brokers in the format host:port>",
  "cloudConnector": {
    "useCloudConnector": "true",
    "cloudConnectorLocation": "",
    "virtualDestination": "manual",
    "virtualHost": "<virtual host in Cloud Connector>",
    "virtualPort": "<virtual port in Cloud Connector>"
  },
  "authType": "SASLKerberosKeytab",
  "username": "<user name used to connect to the Kerberos service>",
  "kerberosConfig": "<content of the krb5.conf configuration file>",
  "kerberosRealm": "<realm defined for the Kafka Kerberos broker>",
  "kerberosServiceName": "<Kerberos service used by the Kafka broker>",
  "kerberosKeyTab": "<content of keytab file>",
  "clientKey": "-----BEGIN PRIVATE KEY-----...-----END CERTIFICATE-----",
  "mtls": "true",
  "schemaRegistry": {
    "url": "<URL of the Schema Registry service in the format protocol://
host:port>",
    "authType": "Basic",
    "username": "<user name used to connect to the Schema Registry>",
    "password": "<password>",
    "cloudConnector": {
      "useCloudConnector": "true",
      "cloudConnectorLocation": "",
      "virtualDestination": "manual",
      "virtualHost": "<Schema Registry virtual host in Cloud Connector>",
      "virtualPort": "<Schema Registry virtual port in Cloud Connector>"
    }
  }
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.9 Generic JDBC (Connection Definition File Format)

Generic JDBC connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Generic JDBC connections and set any connections properties (see Generic JDBC Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax:

Sample Code

```
{
    "name": "<technical name>",
    "businessName": "<business name>",
    "driverClass":"<DBC driver class for the database you are using>",
    "url":"<URL for the JDBC driver>",
    "username":"<user name>",
    "password":"<password>",
    "authType": "Basic",
    "remoteTables": {
        "dataProvisioningAgent": "<Data Provisioning Agent name>"
    }
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.10 Generic OData (Connection Definition File Format)

Generic OData connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Generic OData connections and set any connections properties (see Generic OData Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for OData V2 and OAuth 2.0 authentication with *Client Credentials* grant type:

Sample Code

```
{
    "name": "<technical name>",
    "businessName": "<business name>",
    "description": "<description>",
    "url":"<OData service provider URL>",
```

```
    "version":"V2",
    "authType":"OAuth2",
    "oauth2GrantType": "client_credentials",
    "oauth2TokenEndpoint":"<API endpoint to use to request an access token>",
    "clientId":"<client id>",
    "clientSecret":"<client secret>"
}
```

Example syntax with parameters for OData V4 , OAuth 2.0 authentication with *Client Credentials* grant type, and custom HTTP header:

Sample Code

```
{
    "name": "<technical name>",
    "businessName": "<business name>",
    "description": "<description>",
    "url": "<OData service provider URL>",
    "version": "V4",
    "authType":"OAuth2",
    "oauth2GrantType": "client_credentials",
    "oauth2TokenEndpoint": "<API endpoint to use to request an access token>",
    "oauth2TokenRequestContentType": "json",
    "oauth2Scope":"<string>",
    "clientId": "<client id>",
    "clientSecret": "<client secret>",
    "httpHeaders":[{ "name": "<header name>", "value": "<header value>" }]
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.11 Generic SFTP (Connection Definition File Format)

Generic SFTP connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Generic SFTP connections and set any connections properties (see Generic SFTP Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for an SFTP server in the public cloud with basic authentication:

Sample Code

```
{
    "name": "<technical name>",
    "category": "cloud",
    "host":"<host name of the SFTP server>",
    "port":"<port number of the SFTP server>",
    "hostKey":"<public SSH host key (public key of the SFTP server, not the
key fingerprint)>",
```

```
    "authType":"Basic",
    "username":"<user name>",
    "password":"<password>"
}
```

Example syntax with parameters for an SFTP server in your local network:

Sample Code

```
{
    "name": "<technical name>",
    "description": "<description>",
    "category": "onpremise",
    "host":"<host name of the SFTP server>",
    "port":"<port number of the SFTP server>",
    "hostKey":"<public SSH host key (public key of the SFTP server, not the
key fingerprint)>",
    "rootPath": "<root path starting with character slash>",
    "authType":"SSH",
    "user": "<user name>",
      "key": "-----BEGIN RSA PRIVATE KEY-----...-----END RSA PRIVATE KEY-----
\n",
      "passphrase": "<passphrase>",
      "cloudConnector": {
        "useCloudConnector": "true",
        "cloudConnectorLocation": "",
        "virtualDestination": "auto"
    }
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.12 Google BigQuery (Connection Definition File Format)

Google BigQuery connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Google BigQuery connections and set any connections properties (see Google BigQuery Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for a BigQuery project enabling datasets from the EU location additionally to Google's default location:

Sample Code

```
{
    "name": "<technical name>",
    "businessName": "<business name>",
    "authType": "APIKey",
```

```
    "project": "<ID of the Google Cloud project>",
    "location": "Europe",
    "key": "<content of the json key file that is used for authentication>"
}
```

> **Note**
>
> If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.13 Google Cloud Storage (Connection Definition File Format)

Google Cloud Storage connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Google Cloud Storage connections and set any connections properties (see Google Cloud Storage Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax:

> **Sample Code**

```
{
    "name": "<technical name>",
    "authType": "APIKey",
    "project": "<ID of the Google Cloud project>",
    "rootPath": "<root path starting with character slash>",
    "key": "<content of the json key file that is used for authentication>"
}
```

> **Note**
>
> If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.14 Hadoop Distributed File System (Connection Definition File Format)

Hadoop Distributed File System connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Hadoop Distributed File System connections and set any connections properties (see Hadoop Distributed File System Connections) using connection type specific

syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax:

Sample Code

```
{
    "name": "<technical name>",
    "authType": "Kerberos",
    "endpoint": "WEBHDFS",
    "host": "<host name>",
    "port": "<port number>",
    "rootPath": "<root path>",
    "username": "<user name>",
    "krb5Conf": "krb5Conf",
    "kerberosKeyTab": "<content of keytab file>",
    "custom": {
        "useCustomParameters": "false"
    }
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.15 Microsoft Azure Blob Storage (Connection Definition File Format)

Microsoft Azure Blob Storage connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Microsoft Azure Blob Storage connections and set any connections properties (see Microsoft Azure Blob Storage Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax:

Sample Code

```
{
  "authType": "SharedKey",
  "name": "<technical name>",
  "protocol": "wasbs",
  "rootPath": "<root path starting with character slash>",
  "accountKey": "<account key>",
  "accountName": "<account name>",
  "endpointSuffix": "core.windows.net"
}
```

> **Note**
>
> If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.16 Microsoft Azure Data Lake Store Gen2 (Connection Definition File Format)

Microsoft Azure Data Lake Store Gen2 connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Microsoft Azure Data Lake Store Gen2 connections and set any connections properties (see Microsoft Azure Data Lake Storage Gen2 Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for OAuth 2.0 authentication with grant type *Client Credentials with X.509 Client Certificate*:

> **Sample Code**

```
{
    "name": "<technical name>",
    "authType": "OAuth2",
    "clientSecret": "<client secret>",
    "clientId": "<client id>",
    "oauth2TokenEndpoint": "<token endpont to use to request an access
token>",
    "oauth2GrantType": "client_credentials",
    "accountName": "<storage account name>"
}
```

Example syntax with parameters for OAuth 2.0 authentication with grant type *User Name and Password*:

> **Sample Code**

```
{
    "name": "<technical name>",
    "authType": "OAuth2",
    "username":"<user name>",
    "password":"<password>",
    "oauth2ClientEndpoint": "<client endpont to use to request an access
token>",
    "oauth2GrantType": "password",
    "accountName": "<storage account name>"
}
```

> **Note**
>
> If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.17 Microsoft Azure SQL Database (Connection Definition File Format)

Microsoft Azure SQL Database connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Microsoft Azure SQL Database connections and set any connections properties (see Microsoft Azure SQL Database Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax:

Sample Code

```
{
  "name": "<technical name>",
  "serverName": "<host name of the Azure server>",
  "port": "1433",
  "version": "12.0",
  "databaseName": "<database name>",
  "username":"<user name>",
  "password":"<password>"
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.18 Microsoft SQL Server (Connection Definition File Format)

Microsoft SQL Server connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Microsoft SQL Server connections and set any connections properties (see Microsoft SQL Server Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for enabling remote tables by selecting a Data Provisioning Agent (including advanced properties):

Sample Code

```
{
  "name": "<technical name>",
```

```
  "description": "<description>",
  "serverName": "<Microsoft SQL Server name>",
  "port": "<Microsoft SQL Server port number>",
  "databaseName": "<Microsoft SQL Server database name>",
  "username": "<user name>",
  "password": "<password>",
  "useSSL": "false",
  "version":"Microsoft SQL Server 2022",
  "remoteTables": {
    "dataProvisioningAgent": "<Data Provisioning Agent name>",
    "connectionPoolSize": 10,
    "triggersRecordPrimaryKeysOnly": true
  }
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.19 Oracle (Connection Definition File Format)

Oracle connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create Oracle connections and set any connections properties (see Oracle Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for enabling remote tables by selecting a Data Provisioning Agent (including advanced properties):

Sample Code

```
{
  "name": "<technical name>",
  "businessName": "<business name>",
  "description": "<description>",
  "host": "<host name or IP address on which the remote Oracle database is
running>",
  "port": "<Oracle database server port number>",
  "databaseNameOrSID": "<Oracle database name>",
  "serviceName": "<service name of Oracle database>",
  "username": "<user name>",
  "password": "<password>",
  "useSSL": "false",
  "version":"Oracle 12c",
  "remoteTables": {
      "dataProvisioningAgent": "<Data Provisioning Agent name>",
      "connectionPoolSize": 10,
    "triggersRecordPrimaryKeysOnly": true
  }
}
```

Example syntax with parameters for an on-premise Oracle database supporting data flows by using Cloud Connector and remote tables by selecting a Data Provisioning Agent (including advanced properties):

Sample Code

```
{
  "name": "<technical name>",
  "description": "<description>",
  "host": "<host name or IP address on which the remote Oracle database is
running>",
  "port": "<Oracle database server port number>",
  "databaseNameOrSID": "<Oracle database name>",
  "serviceName": "<service name of Oracle database>",
  "username": "<user name>",
  "password": "<password>",
  "useSSL": "false",
  "version":"Oracle 12c",
  "remoteTables": {
      "dataProvisioningAgent": "<Data Provisioning Agent name>",
      "connectionPoolSize": 10,
    "triggersRecordPrimaryKeysOnly": true
  },
  "cloudConnector": {
        "useCloudConnector": "true",
        "cloudConnectorLocation": "",
        "virtualDestination": "manual",
        "virtualHost":"<virtual host in Cloud Connector>",
        "virtualPort":"<virtual port in Cloud Connector>"
    }
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.20 SAP ABAP (Connection Definition File Format)

SAP ABAP connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create SAP ABAP connections and set any connections properties (see SAP ABAP Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for an on-premise system using load balancing, supporting data flows by using Cloud Connector and remote tables by selecting a Data Provisioning Agent (including advanced properties):

Sample Code

```
{
    "protocol": "RFC",
    "sapLogonConnectionType": "messageServer",
```

```
    "messageServer": "<message server used for load balancing>",
    "messageServerPort": "<numerical port>",
    "messageServerGroup": "PUBLIC",
    "systemId": "<system id>",
    "client": "<client number>",
    "cloudConnector": {
        "useCloudConnector": "true",
        "cloudConnectorLocation": "",
        "virtualDestination": "manual",
        "virtualHost":"<virtual host in Cloud Connector>",
        "virtualPort":"<virtual numerical port in Cloud Connector>"
    },
    "username": "<user name>",
    "password": "<password>",
    "authType": "Basic",
    "remoteTables": {
    "dataProvisioningOption": "dpAgent",
        "dataProvisioningAgent": "<Data Provisioning Agent name>",
        "rfcSerialization": "rowBased"
    },
    "name": "<technical name>"
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.21 SAP BW (Connection Definition File Format)

SAP BW connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create SAP BW connections and set any connections properties (see SAP BW Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for load balancing and supporting data flows by using Cloud Connector and remote tables by selecting a Data Provisioning Agent:

Sample Code

```
{
  "name": "<technical name>",
  "businessName": "<business name>",
  "description": "<description>",
  "authType": "Basic",
  "messageServer": "<message server used for load balancing>",
  "sapLogonConnectionType": "messageServer",
  "messageServerPort": "<numerical port>",
  "messageServerGroup": "PUBLIC",
  "systemId": "<system id>",
  "client": "<client number>",
  "cloudConnector": {
        "useCloudConnector": "true",
        "cloudConnectorLocation": "",
```

```
        "virtualDestination": "manual",
        "virtualHost": "<virtual host in Cloud Connector>",
        "virtualPort": "<virtual numerical port in Cloud Connector>"
      },
  "username":"<user name>",
  "password":"<password>",
  "remoteTables": {
    "dataProvisioningAgent": "<Data Provisioning Agent name>"
  }
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.22 SAP BW/4HANA Model Transfer (Connection Definition File Format)

SAP BW/4HANA Model Transfer connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create SAP BW/4HANA Model Transfer connections and set any connections properties (see SAP BW/4HANA Model Transfer Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax:

Sample Code

```
{
    "name": "<technical name>",
    "authType": "Basic",
      "tunnelName": "<technical name of the live data connection of type
tunnel>",
      "schema": "<schema name>",
      "host": "<host name>",
      "port": "<port name>",
      "remoteTables": {
        "dataProvisioningAgent": "<Data Provisioning Agent name>"
      },
      "security": {
      "useSSL": "true",
      "validateCertificate": "true"
    },
      "username": "<user name>",
      "password": "<password>"
}
```

> **Note**
>
> If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.23 SAP ECC (Connection Definition File Format)

SAP ECC connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create SAP ECC connections and set any connections properties (see SAP ECC Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for remote table support by selecting a Data Provisioning Agent:

Sample Code

```
{
  "name": "<technical name>",
  "description": "<description>",
  "authType": "Basic",
  "applicationServer": "<application server>",
  "sapLogonConnectionType": "applicationServer",
  "systemNumber": "<system instance number>",
  "client": "<client number>",
  "username":"<user name>",
  "password":"<password>",
  "remoteTables": {
    "dataProvisioningAgent": "<Data Provisioning Agent name>"
  }
}
```

Example syntax with parameters for using load balancing, supporting data flows by using Cloud Connector and remote tables by selecting a Data Provisioning Agent:

Sample Code

```
{
  "name": "<technical name>",
  "businessName": "<business name>",
  "description": "<description>",
  "authType": "Basic",
  "messageServer": "<message server used for load balancing>",
  "sapLogonConnectionType": "messageServer",
  "messageServerPort": "<numerical port>",
  "messageServerGroup": "PUBLIC",
  "systemId": "<system id>",
  "client": "<client number>",
  "cloudConnector": {
        "useCloudConnector": "true",
        "cloudConnectorLocation": "",
        "virtualDestination": "manual",
        "virtualHost": "<virtual host in Cloud Connector>",
        "virtualPort": "<virtual numerical port in Cloud Connector>"
```

```
      },
  "username":"<user name>",
  "password":"<password>",
  "remoteTables": {
    "dataProvisioningAgent": "<Data Provisioning Agent name>"
  }
}
```

> **Note**
>
> If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.24 SAP HANA (Connection Definition File Format)

SAP HANA connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create SAP HANA connections and set any connections properties (see SAP HANA Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for SAP HANA Cloud using basic authentication and data access *Remote Only*:

> **Sample Code**

```
{
    "name": "<technical name>",
    "description": "<description>",
    "host":"<fully qualified host name or IP address on which the remote SAP
HANA server is running>",
    "port":"443",
    "authType": "Basic",
    "category": "cloud",
    "username": "<user name>",
        "password":"<password>",
    "remoteTables": {
        "dataAccess": "federationOnly"
    }
}
```

Example syntax with parameters for SAP HANA on-premise using basic authentication, encrypted communication, supporting remote tables by selecting a Data Provisioning Agent (including advanced properties), and supporting data and replication flows by using Cloud Connector:

> **Sample Code**

```
{
    "name": "<technical name>",
    "description": "<description>",
    "host":"<fully qualified host name or IP address on which the remote SAP
HANA server is running>",
```

```
    "port":"<TCP SQL port number of the remote SAP HANA server>",
    "authType": "Basic",
    "username": "<user name>",
    "password": "<password>",
    "category": "onpremise",
    "sslEncryption": {
        "enable": "true",
        "validate": "false"
    },
    "remoteTables": {
        "dataProvisioningOption": "dpAgent",
        "dataProvisioningAgent": "<Data Provisioning Agent name>",
        "conn_pool_size": "10",
        "record_pk_only": "true"
    },
    "cloudConnector": {
        "useCloudConnector": "true",
        "cloudConnectorLocation": "empty",
        "virtualDestination": "auto"
    }
}
```

> **Note**
>
> If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.25 SAP HANA Cloud, Data Lake Files (Connection Definition File Format)

SAP HANA Cloud, Data Lake Files connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create SAP HANA Cloud, Data Lake Files connections and set any connections properties (see SAP HANA Cloud, Data Lake Files Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax:

> **Sample Code**

```
{
  "name": "<technical name>",
  "dataAccessLevel": "fileSystem",
  "host": "<host name>",
  "rootPath": "",
  "authType": "KeyStore",
  "keyStoreFile": "<Base64-encoded value of the client keystore file>",
  "keyStorePassword": "<client keystore password>"
}
```

> **Note**
>
> If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.26 SAP HANA Cloud, Data Lake Relational Engine (Connection Definition File Format)

SAP HANA Cloud, Data Lake Relational Engine connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create SAP HANA Cloud, Data Lake Relational Engine connections and set any connections properties (see SAP HANA Cloud, Data Lake Relational Engine Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax:

> **Sample Code**

```
{
    "name": "<technical name>",
    "host": "<host name>",
    "port": "<port name>",
    "username": "<user name>",
    "password": "<password>",
    "authType": "Basic"
}
```

> **Note**
>
> If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.27 SAP SuccessFactors (Connection Definition File Format)

SAP SuccessFactors connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create SAP SuccessFactors connections and set any connections properties (see SAP SuccessFactors Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for OData V4 and basic authentication:

Sample Code

```
{
"name": "<technical name>",
"businessName" : "<business name>", //optional
"description":"<description>",//optional
"url": "https://<SAP SuccessFactors API Server>/odatav4/<supported SAP
SuccessFactors service group>",
"version" :"V4",
"authType": "Basic",
"username": "<user name>",
"password":"<password>"
}
```

Example syntax with parameters for OData V2 and OAuth2 authentication:

Sample Code

```
{
    "name":"<technical name>",
    "businessName":"<business name>",//optional
    "description":"<description>", //optional
    "url": "https://<SAP SuccessFactors API Server>/odata/v2/",
    "version": "V2",
    "authType": "OAuth2",
    "oauth2GrantType":"saml_bearer",
    "oauth2TokenEndpoint": "https://oauth2TokenEndpoint.com/oauth/token",
    "oauth2CompanyId": "<SAP SuccessFactors company ID>",
    "clientId": "<client id>",
    "samlAssertion":"<SAML assertion>"
}
```

Parameters are set as follows:

Parameter	Connection Property	Description
`name`	*Technical Name*	[required] Enter the technical name of the connection. The technical name can only contain alphanumeric characters and underscores (_). It cannot start or end with underscore (_). The name must be unique within the space. **Note** Once the object is saved, the technical name can no longer be modified.
`businessName`	*Business Name*	[optional] Enter a descriptive name to help users identify the object. This name can be changed at any time.
`description`	*Description*	[optional] Provide more information to help users understand the object.
`package`	*Package*	[optional] Enter an existing package to facilitate transport between tenants (see Transporting Content Between Tenants). Default value: **none**

Parameter	Connection Property	Description
`url`	*URL*	[required] Enter the OData service provider URL of the SAP SuccessFactors service that you want to access. The syntax for the URL is: • For V2: **`<SAP SuccessFactors API Server>/odata/v2`** (providing a supported SAP SuccessFactors service group **`/<service group>`** is optional) • For V4: **`<SAP SuccessFactors API Server>/odatav4/<supported SAP SuccessFactors service group>`**
`version`	*Version*	[required] Enter the OData version used to implement the SAP SuccessFactors OData service (**`V2`** or **`V4`**; see the URL).
`authType`	*Authentication Type*	[required] Enter the authentication type to use to connect to the OData endpoint. You can enter: • **`Basic`** for basic authentication with user name and password • **`OAuth2`** **Note** Access to APIs based on HTTP Basic Authentication will be deleted on November 12, 2027. We strongly recommend to adopt the *OAuth 2.0* authentication type for your connections. For more information, see: • Deprecation of Basic Authentication for APIs in *SAP SuccessFactors What's New Viewer* • SAP Note 3774454
`oauth2GrantType`	*OAuth Grant Type*	If you have entered **`OAuth2`** as `authType`: [required] Enter **`SAML Bearer`** as the grant type used to retrieve an access token.
`oauth2TokenEndpoint`	*OAuth Token Endpoint*	If you have entered **`OAuth2`** as `authType`: [required] Enter the SAP SuccessFactors API endpoint to use to request an access token: **`<SAP SuccessFactors API Server>/oauth/token`**.
`oauth2Scope`	*OAuth Scope*	If you have entered **`OAuth2`** as `authType`: [optional] Enter the OAuth scope, if applicable.
`oauth2CompanyId`	*OAuth Company ID*	If you have entered **`OAuth2`** as `authType`: [required] Enter the SAP SuccessFactors company ID (identifying the SAP SuccessFactors system on the SAP SuccessFactors API server) to use to request an access token.

Parameter	Connection Property	Description
`clientId`	*Client ID*	If you have entered **`OAuth2`** as `authType`: [required] Enter the API key received when registering SAP Datasphere as OAuth2 client application in SAP SuccessFactors.
`samlAssertion`	*SAML Assertion*	If you have entered **`OAuth2`** as `authType`: [required] Enter a valid SAML assertion that has been generated for authentication. **Note** If the SAML assertion expires, the connection gets invalid until you update the connection with a new valid SAML assertion.
`username`	*User Name*	If you have entered **`Basic`** as `authType`: [required] Enter the user name in `<username@companyID>` format.
`password`	*Password*	If you have entered **`Basic`** as `authType`: [required] Enter the password.

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.28 SAP S/4HANA Cloud (Connection Definition File Format)

SAP S/4HANA Cloud connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create SAP S/4HANA Cloud connections and set any connections properties (see SAP S/4HANA Cloud Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters for basic authentication and remote table support using a Data Provisioning Agent for federation and replication:

Sample Code

```
{
    "applicationServer": "myXXXXX-api.s4hana.ondemand.com",
    "port": "443",
    "username": "<user name>",
    "password": "<password>",
    "authType": "Basic",
```

```
    "remoteTables": {
        "dataProvisioningOption": "dpAgent",
        "dataProvisioningAgent": "<Data Provisioning Agent name>"
    },
    "name": "<technical name>",
    "client": "<client number>"
}
```

Example syntax with parameters for authentication with X.509 client certificate or certificate chain and remote table support using a Data Provisioning Agent for federation and replication:

Sample Code

```
{
    "applicationServer": "myXXXXX-api.s4hana.cloud.sap",
    "port": "443",
    "authType": "ClientCertificate",
    "x509ClientCertificate": "<certificate or certificate chain>",
    "x509ClientPrivateKey": "-----BEGIN PRIVATE KEY-----...-----END PRIVATE
KEY-----",
    "remoteTables": {
        "dataProvisioningOption": "dpAgent",
        "dataProvisioningAgent": "<Data Provisioning Agent name>"
    },
    "name": "<technical name>",
    "client": "<client number>"
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

7.29 SAP S/4HANA On-Premise (Connection Definition File Format)

SAP S/4HANA On-Premise connection properties are set and retrieved in the connection definition file format and stored as a `.json` file. A connection definition file must not exceed 25MB.

Users with the *DW Integrator* role can create SAP S/4HANA On-Premise connections and set any connections properties (see SAP S/4HANA On-Premise Connections) using connection type specific syntax. To obtain detailed information about the connection syntax, you can read an appropriate connection from your space and write the output to a file using the `--get` option (see Read Connection Details [page 166]).

Example syntax with parameters supporting using the ABAP SQL service for remote tables:

Sample Code

```
{
    "sapLogonConnectionType": "applicationServer",
    "applicationServer": "<application server>",
    "systemNumber": "<system instance number>",
    "systemId": "<system id>",
    "client": "<client number>",
```

```
    "cloudConnector": {
        "useCloudConnector": "true",
        "cloudConnectorLocation": "",
        "virtualDestination": "manual",
        "virtualHost":"<virtual host in Cloud Connector>",
        "virtualPort":"<virtual port defined in the Cloud Connector system
mapping for the HTTP/HTTPS protocol which has been created for remote tables
via ABAP SQL>"
    },
    "username": "<user name>",
    "password": "<password>",
    "authType": "Basic",
    "remoteTables": {
        "dataProvisioningOption": "direct"
    },
    "name": "<technical name>"
}
```

Example syntax with parameters for using load balancing, supporting flows and model import by using Cloud Connector and remote tables and model import by selecting a Data Provisioning Agent (including advanced properties) :

Sample Code

```
{
    "sapLogonConnectionType": "messageServer",
    "messageServer": "<message server used for load balancing>",
    "messageServerPort": "<numerical port>",
    "messageServerGroup": "PUBLIC",
    "systemId": "<system id>",
    "client": "<client number>",
    "cloudConnector": {
        "useCloudConnector": "true",
        "cloudConnectorLocation": "",
        "virtualDestination": "manual",
        "virtualHost":"<virtual host in Cloud Connector>",
        "virtualPort":"<virtual numerical port in Cloud Connector>"
    },
    "username": "<user name>",
    "password": "<password>",
    "authType": "Basic",
    "remoteTables": {
        "dataProvisioningOption": "dpAgent",
        "dataProvisioningAgent": "<Data Provisioning Agent name>",
        "rfcSerialization": "rowBased"
    },
    "name": "<technical name>"
}
```

Note

If a parameter isn't set, it will receive the default value (when creating or editing the connection). Credentials have always to be specified.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap