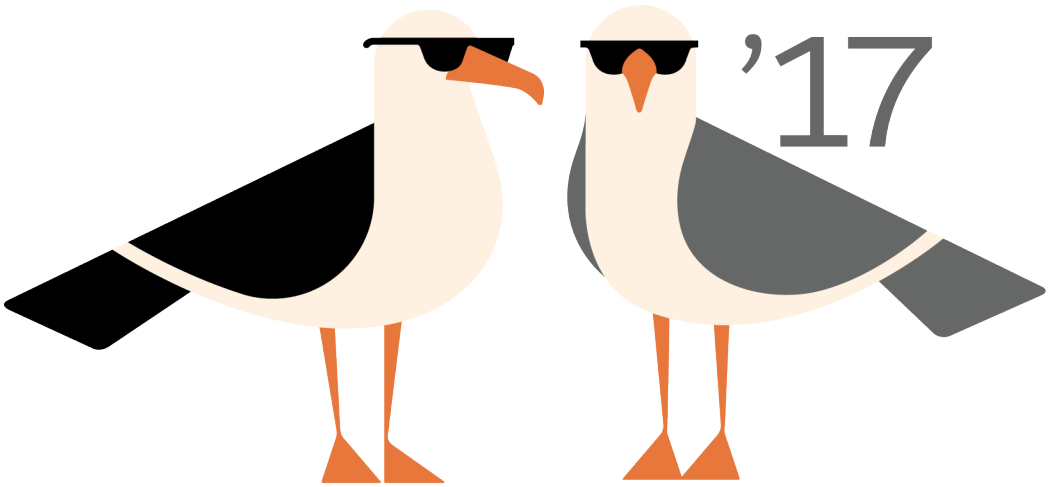
Salesforce CLI Command Reference (Beta)

Salesforce, Summer '17



@salesforcedocs

Last updated: August 9, 2017

CONTENTS

SALESFORCE CLI COMMAND REFERENCE (BETA)

The command reference contains information about the Salesforce CLI commands and their parameters.

Note: This release contains a beta version of Salesforce DX, which means it's a high-quality feature with known limitations. Salesforce DX isn't generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. We can't guarantee general availability within any particular time frame or at all. Make your purchase decisions only on the basis of generally available products and features. You can provide feedback and suggestions for Salesforce DX in the Salesforce DX Beta group in the Success Community.

`force` Namespace

Use commands in the `force` namespace to develop on the Force.com platform.

`alias` Commands

Use the alias commands to manage username aliases.

`alias:list`

Lists the aliases that sfdx can use for various commands and tasks.

Command Syntax

```
sfdx force:alias:list
   [--json]
   [--loglevel LOGLEVEL]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `alias:list`

Example:

```
$ sfdx force:alias:list
```

`alias:set`

Sets an alias that sfdx can use for various commands and tasks.

Command Syntax

sfdx force:alias:set

[--json]

[--loglevel LOGLEVEL]

Parameters

--json

Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `alias:set`

You can associate an alias with only one username at a time. If you've set an alias multiple times, the alias points to the most recent username.

To delete an alias, run "sfdx force:alias:set" with no username.

Examples:

```
$ sfdx force:alias:set DefaultOrg=me@my.org
```

```
$ sfdx force:alias:set DefaultOrg=me@my.org DevHubOrg=me@myhub.org
```

```
$ sfdx force:alias:set AliasToDelete=
```

apex Commands

Use the apex commands to create Apex classes, execute anonymous blocks, view your logs, and run and view the results of Apex tests.

apex:class:create

Creates an Apex class in the specified directory or the current working directory. If you don't explicitly set the API version, it defaults to the current API version. The .cls file and associated metadata file are created.

apex:execute

Executes one or more lines of anonymous Apex code, or executes the code in a local file.

apex:log:get

Fetches a specific debug log.

apex:log:list

Displays a list of debug log IDs, along with general information about the logs.

apex:test:report

Displays the test results for a specific test run.

apex:test:run

Runs Apex tests.

apex:class:create

Creates an Apex class in the specified directory or the current working directory. If you don't explicitly set the API version, it defaults to the current API version. The .cls file and associated metadata file are created.

Command Syntax

sfdx force:apex:class:create
-n CLASSNAME
[-t TEMPLATE]
[-d OUTPUTDIR]
[-a APIVERSION]
[--json]
[--loglevel LOGLEVEL]

Parameters

-n | --classname CLASSNAME
Required

The name of the new Apex class. The name can be up to 40 characters and must start with a letter.

Type: string

-t | --template TEMPLATE
Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: DefaultApexClass, ApexException, ApexUnitTest, InboundEmailService

Default value: DefaultApexClass

-d | --outputdir OUTPUTDIR
Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

-a | --apiversion APIVERSION
Optional

The API version of the created source.

Type: string

Permissible values are: 40.0, 39.0

Default value: 40.0

--json
Optional

Formats output as JSON.

Type: string

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `apex:class:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
$ sfdx force:apex:class:create -n MyClass
```

```
$ sfdx force:apex:class:create -n MyClass -d classes
```

`apex:execute`

Executes one or more lines of anonymous Apex code, or executes the code in a local file.

Command Syntax

`sfdx force:apex:execute`
[-f APEXCODEFILE]
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]

Parameters

`-f | --apexcodefile APEXCODEFILE`
Optional

Path to a local file that contains Apex code.

Type: file

`-u | --targetusername TARGETUSERNAME`
Optional

Username for the target org. Overrides the default target org.

Type: string

`--json`
Optional

Format output as JSON.

Type: flag

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `apex:execute`

Executes one or more lines of Apex code, or executes the code in a local file.

Before you enter code, run this command with no parameters to get a prompt.

From the prompt, all commands are executed in a single execute anonymous request.

For more information, see "Anonymous Blocks" in the Apex Developer Guide.

Examples:

```
$ sfdx force:apex:execute -f ~/test.apex
```

```
$ sfdx force:apex:execute
```

```
>> Start typing Apex code. Press the Enter key after each line,
```

```
>> then press CTRL+D when finished.
```

apex:log:get

Fetches a specific debug log.

Command Syntax

sfdx force:apex:log:get
-i LOGID
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]

Parameters

-i | --logid LOGID
Required

ID of the log to display.

Type: id

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `apex:log:get`

When you execute this command in a project, it fetches the specified log from your default scratch org.To get the IDs for your debug logs, run "sfdx force:apex:log:list".

Examples:

```
$ sfdx force:apex:log:get -i <log id>
```

```
$ sfdx force:apex:log:get -i <log id> -u me@my.org
```

`apex:log:list`

Displays a list of debug log IDs, along with general information about the logs.

Command Syntax

`sfdx force:apex:log:list`
[-u TARGETUSERNAME]

[--json]

[--loglevel LOGLEVEL]

Parameters

`-u | --targetusername TARGETUSERNAME`
Optional

Username for the target org. Overrides the default target org.

Type: string

`--json`
Optional

Format output as JSON.

Type: flag

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `apex:log:list`

When you execute this command in a project, it lists the log IDs for your default scratch org.

Examples:

```
$ sfdx force:apex:log:list
```

```
$ sfdx force:apex:log:list -u me@my.org
```

apex:test:report

Displays the test results for a specific test run.

Command Syntax

sfdx force:apex:test:report
-i TESTRUNID
[-c]
[-d OUTPUTDIR]
[-r RESULTFORMAT]
[-w WAIT]
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]
[--verbose]

Parameters

-i | --testrunid TESTRUNID
Required

ID of test run.

Type: id

-c | --codecoverage
Optional

Retrieves code coverage results.

Type: flag

-d | --outputdir OUTPUTDIR
Optional

Directory to store test run files.

Type: directory

-r | --resultformat RESULTFORMAT
Optional

Format to use when displaying test results. If you also specify the --json flag, --json overrides this parameter.

Type: string

Permissible values are: human, tap, junit, json

Default value: human

-w | --wait WAIT
Optional

Sets the streaming client socket timeout, in minutes.If the streaming client socket has no contact from the server for a number of minutes, the client exits. Specify a longer wait time if timeouts occur frequently.

Type: minutes

Default value: 6

-u | --targetusername TARGETUSERNAME

Optional

Username for the target org. Overrides the default target org.

Type: string

--json

Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

--verbose

Optional

Displays Apex test processing details. If json format is specified, processing details aren't displayed.

Type: flag

Help for `apex:test:report`

Displays test results for an enqueued or completed asynchronous Apex test run.

Examples:

```
$ sfdx force:apex:test:report -i <test run id>
```

```
$ sfdx force:apex:test:report -i <test run id> -r junit
```

```
$ sfdx force:apex:test:report -i <test run id> -c --json
```

`apex:test:run`

Runs Apex tests.

Command Syntax

sfdx force:apex:test:run

[-n CLASSNAMES]

[-s SUITENAMES]

[-c]

[-d OUTPUTDIR]

```
[-l TESTLEVEL]
[-r RESULTFORMAT]
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]
[--verbose]
```

Parameters

-n | --classnames CLASSNAMES
Optional

Comma-separated list of Apex test class names to execute. You can't specify both class names and suite names.

Type: string

-s | --suitenames SUITENAMES
Optional

Comma-separated list of Apex test suite names to execute. You can't specify both class names and suite names.

Type: string

-c | --codecoverage
Optional

Retrieves code coverage results.

Type: flag

-d | --outputdir OUTPUTDIR
Optional

Directory to store test run files.

Type: directory

-l | --testlevel TESTLEVEL
Optional

Specifies which tests to run, using one of these TestLevel enum values:

RunSpecifiedTests—Only the tests that you specify are run.

RunLocalTests—All tests in your org are run, except the ones that originate from installed managed packages.

RunAllTestsInOrg—All tests are in your org and in installed managed packages are run.

Type: string

Permissible values are: RunLocalTests, RunAllTestsInOrg, RunSpecifiedTests

-r | --resultformat RESULTFORMAT
Optional

Format to use when displaying test results. If you also specify the --json flag, --json overrides this parameter.

Type: string

Permissible values are: human, tap, junit, json

Default value: human

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

--verbose
Optional

Displays Apex test processing details. If json format is specified, processing details aren't displayed.

Type: flag

Help for `apex:test:run`

By default, runs all Apex tests in the org's namespace.

To run specific tests, specify the class names or suite names or set a --testlevel value.

Examples:

```
$ sfdx force:apex:test:run
```

```
$ sfdx force:apex:test:run -n MyClassTest,MyOtherClassTest -r human
```

```
$ sfdx force:apex:test:run -s MySuite,MyOtherSuite -c --json
```

```
$ sfdx force:apex:test:run -l RunLocalTests -d <path to outputdir> -u me@my.org
```

`auth` Commands

Use the auth commands to authorize a Salesforce org for use with the Salesforce CLI.

auth:jwt:grant
Authorizes a Salesforce org using the JWT flow.

auth:sfdxurl:store
Authorizes a Salesforce org using an SFDX auth URL.

auth:web:login
Authorizes a Salesforce org by opening a browser so you can log in through salesforce.com.

auth:jwt:grant

Authorizes a Salesforce org using the JWT flow.

Command Syntax

sfdx force:auth:jwt:grant
```
   -u USERNAME
   -f JWTKEYFILE
   -i CLIENTID
   [-r INSTANCEURL]
   [-d]
   [-s]
   [-a SETALIAS]
   [--json]
   [--loglevel LOGLEVEL]
```

Parameters

-u | --username USERNAME
Required

The authentication username.

Type: string

-f | --jwtkeyfile JWTKEYFILE
Required

Path to a file containing the private key.

Type: filepath

-i | --clientid CLIENTID
Required

The OAuth client ID (sometimes referred to as the consumer key).

Type: string

-r | --instanceurl INSTANCEURL
Optional

The login URL of the Salesforce instance that the org lives on.

Type: url

-d | --setdefaultdevhubusername
Optional

Sets the authenticated org as the default Dev Hub org for scratch org creation.

Type: flag

-s | --setdefaultusername
Optional

Sets the authenticated org as the default username that all commands run against.

Type: flag

-a | --setalias SETALIAS

Optional

Sets an alias for the authenticated org.

Type: string

--json

Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `auth:jwt:grant`

Authorizes a Salesforce org using a private key file that has been uploaded to a personal connected app.

Examples:

```
  $ sfdx force:auth:jwt:grant -u me@my.org -f <path to jwt key file> -i <oauth client id>
```

```
  $ sfdx force:auth:jwt:grant -u me@my.org -f <path to jwt key file> -i <oauth client id>
-s -a MyDefaultOrg
```

`auth:sfdxurl:store`

Authorizes a Salesforce org using an SFDX auth URL.

Command Syntax

sfdx force:auth:sfdxurl:store

```
-f SFDXURLFILE
[-d]
[-s]
[-a SETALIAS]
[--json]
[--loglevel LOGLEVEL]
```

Parameters

-f | --sfdxurlfile SFDXURLFILE
Required

Path to a file containing the SFDX URL.

Type: filepath

-d | --setdefaultdevhubusername
Optional

Sets the authenticated org as the default Dev Hub org for scratch org creation.

Type: flag

-s | --setdefaultusername
Optional

Sets the authenticated org as the default username that all commands run against.

Type: flag

-a | --setalias SETALIAS
Optional

Sets an alias for the authenticated org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `auth:sfdxurl:store`

Authorize a Salesforce org using an SFDX auth URL stored within a file.

The file must have use format "force://(<clientId>:<clientSecret>:)?<refreshToken>@<instanceUrl>", where "?" denotes an option value.

The file must contain only the URL or be a JSON file that has a top-level property named sfdxAuthUrl.

Examples:

```
$ sfdx force:auth:sfdxurl:store -f <path to sfdxAuthUrl file>
```

```
$ sfdx force:auth:sfdxurl:store -f <path to sfdxAuthUrl file> -s -a MyDefaultOrg
```

auth:web:login

Authorizes a Salesforce org by opening a browser so you can log in through salesforce.com.

Command Syntax

sfdx force:auth:web:login
[-i CLIENTID]
[-r INSTANCEURL]
[-d]
[-s]
[-a SETALIAS]
[--json]
[--loglevel LOGLEVEL]

Parameters

-i | --clientid CLIENTID
Optional

The OAuth client ID (sometimes referred to as the consumer key).

Type: string

-r | --instanceurl INSTANCEURL
Optional

The login URL of the Salesforce instance that the org lives on.

Type: url

-d | --setdefaultdevhubusername
Optional

Sets the authenticated org as the default Dev Hub org for scratch org creation.

Type: flag

-s | --setdefaultusername
Optional

Sets the authenticated org as the default username that all commands run against.

Type: flag

-a | --setalias SETALIAS
Optional

Sets an alias for the authenticated org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `auth:web:login`

To log in to a sandbox, set --instanceurl to https://test.salesforce.com.

Examples:

```
$ sfdx force:auth:web:login -a TestOrg1
```

```
$ sfdx force:auth:web:login -i <oauth client id>
```

```
$ sfdx force:auth:web:login -r https://test.salesforce.com
```

`config` Commands

Use the config commands to view and set your Salesforce CLI configuration values. Set your default Dev Hub and scratch org, and your default instance URL, either globally or at the project level.

config:get

Gets the Salesforce CLI configuration values for your default scratch org, your default Dev Hub org, your default instance URL, or any combination of the three.

config:list

Lists the config variables for sfdx.

config:set

Sets the local and global configuration variables for the Salesforce CLI.

`config:get`

Gets the Salesforce CLI configuration values for your default scratch org, your default Dev Hub org, your default instance URL, or any combination of the three.

Command Syntax

`sfdx force:config:get`

[--json]

[--loglevel LOGLEVEL]

[--verbose]

Parameters

`--json`
Optional

Format output as JSON.

Type: flag

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

`--verbose`
Optional

Emit additional command output to stdout.

Type: flag

Help for `config:get`

To see your default scratch org username, include "defaultusername".

To see your default Dev Hub, include "defaultdevhubusername".

To see your default instance URL, include "instanceUrl".

To see the locations where your values are set, include the --verbose flag.

Examples:

```
$ sfdx force:config:get defaultusername
```

```
$ sfdx force:config:get defaultusername defaultdevhubusername instanceUrl
```

```
$ sfdx force:config:get defaultusername defaultdevhubusername --verbose
```

`config:list`

Lists the config variables for sfdx.

Command Syntax

`sfdx force:config:list`
[--json]

[--loglevel LOGLEVEL]

Parameters

`--json`
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `config:list`

Lists the config variables that the Salesforce CLI uses for various commands and tasks.

`config:set`

Sets the local and global configuration variables for the Salesforce CLI.

Command Syntax

sfdx force:config:set

[-g]

[--json]

[--loglevel LOGLEVEL]

Parameters

-g | --global

Optional

Sets the configuration variables globally, so they can be used from any directory.

Type: flag

--json

Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `config:set`

Sets the configuration variables that the Salesforce CLI uses for various commands and tasks. Local variables apply only to your current project. Global variables apply in any directory.

Examples:

```
$ sfdx force:config:set defaultusername=me@my.org defaultdevhubusername=me@myhub.org
```

```
$ sfdx force:config:set defaultdevhubusername=me@myhub.org -g
```

`data` Commands

Use the data commands to manipulate records in your org. Commands are available to help you work with various APIs. Import CSV files with the Bulk API. Export and import data that includes master-detail relationships with the SObject Tree Save API. Perform simple CRUD operations on individual records with the REST API.

data:bulk:delete
Deletes a batch of records listed in a CSV file.

data:bulk:status
Polls the Bulk API for job status or batch status.

data:bulk:upsert
Creates a job and one or more batches for inserting new rows and updating existing rows by accessing the Bulk API.

data:record:create
Creates and inserts a record.

data:record:delete
Deletes a single record.

data:record:get
Displays a single record.

data:record:update
Updates a single record.

data:soql:query
Executes a SOQL query.

data:tree:export
Exports data from an org into sObject Tree format for force:data:import consumption.

data:tree:import
Imports data into an org using the SObject Tree Save API. This data can include master-detail relationships.

`data:bulk:delete`

Deletes a batch of records listed in a CSV file.

Command Syntax

`sfdx force:data:bulk:delete`
`-s SOBJECTTYPE`

```
-f CSVFILE
[-w WAIT]
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]
```

Parameters

-s | --sobjecttype SOBJECTTYPE
Required

The sObject type of the records you're deleting.

Type: string

-f | --csvfile CSVFILE
Required

The path to the CSV file that contains the IDs of the records to delete.

Type: file

-w | --wait WAIT
Optional

The number of minutes to wait for the command to complete before displaying the results.

Type: minutes

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `data:bulk:delete`

The file must be a CSV file with only one column: "Id".

One job can contain many batches, depending on the length of the CSV file.

Returns a job ID and a batch ID. Use these IDs to check job status with data:bulk:status.

Examples:

```
$ sfdx force:data:bulk:delete -s Account -f ./path/to/file.csv
```

```
$ sfdx force:data:bulk:delete -s MyObject__c -f ./path/to/file.csv
```

data:bulk:status

Polls the Bulk API for job status or batch status.

Command Syntax

sfdx force:data:bulk:status
-i JOBID
[-b BATCHID]
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]

Parameters

-i | --jobid JOBID
Required

The ID of the job you want to view or of the job whose batch you want to view.

Type: id

-b | --batchid BATCHID
Optional

The ID of the batch whose status you want to view.

Type: id

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `data:bulk:status`

Examples:

```
$ sfdx force:data:bulk:status -i 750xx000000005sAAA
```

```
$ sfdx force:data:bulk:status -i 750xx000000005sAAA -b 751xx000000005nAAA
```

`data:bulk:upsert`

Creates a job and one or more batches for inserting new rows and updating existing rows by accessing the Bulk API.

Command Syntax

`sfdx force:data:bulk:upsert`
`-s SOBJECTTYPE`
`-f CSVFILE`
`[-i EXTERNALID]`
`[-w WAIT]`
`[-u TARGETUSERNAME]`
`[--json]`
`[--loglevel LOGLEVEL]`

Parameters

`-s | --sobjecttype SOBJECTTYPE`
Required

The sObject type of the records you want to upsert.

Type: string

`-f | --csvfile CSVFILE`
Required

The path to the CSV file that defines the records to upsert.

Type: file

`-i | --externalid EXTERNALID`
Optional

The column name of the external ID. If not provided, an arbitrary ID is used.

Type: id

`-w | --wait WAIT`
Optional

The number of minutes to wait for the command to complete before displaying the results.

Type: minutes

`-u | --targetusername TARGETUSERNAME`
Optional

Username for the target org. Overrides the default target org.

Type: string

--json

Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `data:bulk:upsert`

Inserts or updates records from a CSV file.

One job can contain many batches, depending on the length of the CSV file.

Returns a job ID and a batch ID. Use these IDs to check job status with data:bulk:status.

For information about formatting your CSV file, see "Prepare CSV Files" in the Bulk API Developer Guide.

Examples:

```
$ sfdx force:data:bulk:upsert -s MyObject__c -f ./path/to/file.csv -i MyField__c
```

```
$ sfdx force:data:bulk:upsert -s MyObject__c -f ./path/to/file.csv -i Id -w 2
```

`data:record:create`

Creates and inserts a record.

Command Syntax

sfdx force:data:record:create

-s SOBJECTTYPE

-v VALUES

[-u TARGETUSERNAME]

[--json]

[--loglevel LOGLEVEL]

Parameters

-s | --sobjecttype SOBJECTTYPE

Required

The sObject type of the record you're creating.

Type: string

-v | --values VALUES
Required

The <fieldName>=<value> pairs you're creating.

Type: string

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `data:record:create`

The format of a field-value pair is <fieldName>=<value>.

Enclose all field-value pairs in one set of double quotation marks.

Enclose values that contain spaces in single quotes.

Examples:

```
   $ sfdx force:data:record:create -s Account -v "Name=Acme"
```

```
   $ sfdx force:data:record:create -s Account -v "Name='Universal Containers'"
```

```
   $ sfdx force:data:record:create -s Account -v "Name='Universal Containers'
Website=www.example.com"
```

`data:record:delete`

Deletes a single record.

Command Syntax

sfdx force:data:record:delete
`-s SOBJECTTYPE`

`[-i SOBJECTID]`

```
[-w WHERE]
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]
```

Parameters

-s | --sobjecttype SOBJECTTYPE
Required

The sObject type of the record you're deleting.

Type: string

-i | --sobjectid SOBJECTID
Optional

The ID of the record you're deleting.

Type: id

-w | --where WHERE
Optional

A list of <fieldName>=<value> pairs to search for.

Type: string

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `data:record:delete`

Specify an sObject type and either an ID or a list of <fieldName>=<value> pairs.

The format of a field-value pair is <fieldName>=<value>.

Enclose all field-value pairs in one set of double quotation marks.

Enclose values that contain spaces in single quotes.

Examples:

```
$ sfdx force:data:record:delete -s Account -i 001D000000Kv3dl
```

```
$ sfdx force:data:record:delete -s Account -w "Name=Acme"
```

```
$ sfdx force:data:record:delete -s Account -w "Name='Universal Containers'"
```

```
$ sfdx force:data:record:delete -s Account -w "Name='Universal Containers' Phone='(123) 456-7890'"
```

data:record:get

Displays a single record.

Command Syntax

sfdx force:data:record:get
-s SOBJECTTYPE
[-i SOBJECTID]
[-w WHERE]
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]

Parameters

-s | --sobjecttype SOBJECTTYPE
Required

The sObject type of the record you're retrieving.

Type: string

-i | --sobjectid SOBJECTID
Optional

The ID of the record you're retrieving.

Type: id

-w | --where WHERE
Optional

A list of <fieldName>=<value> pairs to search for.

Type: string

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json

Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `data:record:get`

Specify an sObject type and either an ID or a list of <fieldName>=<value> pairs.

The format of a field-value pair is <fieldName>=<value>.

Enclose all field-value pairs in one set of double quotation marks.

Enclose values that contain spaces in single quotes.

Examples:

```
   $ sfdx force:data:record:get -s Account -i 001D000000Kv3dl
```

```
   $ sfdx force:data:record:get -s Account -w "Name=Acme"
```

```
   $ sfdx force:data:record:get -s Account -w "Name='Universal Containers'"
```

```
   $ sfdx force:data:record:get -s Account -w "Name='Universal Containers' Phone='(123)
456-7890'"
```

`data:record:update`

Updates a single record.

Command Syntax

`sfdx force:data:record:update`

```
   -s SOBJECTTYPE
   [-i SOBJECTID]
   [-w WHERE]
   -v VALUES
   [-u TARGETUSERNAME]
   [--json]
   [--loglevel LOGLEVEL]
```

Parameters

-s | --sobjecttype SOBJECTTYPE
Required

The sObject type of the record you're updating.

Type: string

-i | --sobjectid SOBJECTID
Optional

The ID of the record you're updating.

Type: id

-w | --where WHERE
Optional

A list of <fieldName>=<value> pairs to search for.

Type: string

-v | --values VALUES
Required

The <fieldName>=<value> pairs you're updating.

Type: string

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `data:record:update`

The format of a field-value pair is <fieldName>=<value>.

Enclose all field-value pairs in one set of double quotation marks.

Enclose values that contain spaces in single quotes.

Examples:

```
  $ sfdx force:data:record:update -s Account -i 001D000000Kv3dl -v "Name=NewAcme"
```

```
  $ sfdx force:data:record:update -s Account -w "Name='Old Acme'" -v "Name='New Acme'"
```

```
  $ sfdx force:data:record:update -s Account -i 001D000000Kv3dl -v "Name='Acme III'
Website=www.example.com"
```

data:soql:query

Executes a SOQL query.

Command Syntax

sfdx force:data:soql:query

-q QUERY

[-t]

[-u TARGETUSERNAME]

[--json]

[--loglevel LOGLEVEL]

Parameters

-q | --query QUERY

Required

SOQL query to execute.

Type: string

-t | --usetoolingapi

Optional

Execute the query using Tooling API.

Type: flag

-u | --targetusername TARGETUSERNAME

Optional

Username for the target org. Overrides the default target org.

Type: string

--json

Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `data:soql:query`

When you execute this command in a project, it executes the query against the data in your default scratch org.

Examples:

```
$ sfdx force:data:soql:query -q "SELECT Id, Name, Account.Name FROM Contact"
```

```
$ sfdx force:data:soql:query -q "SELECT Id, Name FROM Account WHERE ShippingState IN
('CA', 'NY')"
```

```
$ sfdx force:data:soql:query -q "SELECT Name FROM ApexTrigger" -t
```

`data:tree:export`

Exports data from an org into sObject Tree format for force:data:import consumption.

Command Syntax

`sfdx force:data:tree:export`
`-q QUERY`

`[-p]`

`[-x PREFIX]`

`[-d OUTPUTDIR]`

`[-u TARGETUSERNAME]`

`[--json]`

`[--loglevel LOGLEVEL]`

Parameters

`-q | --query QUERY`
Required

SOQL query statement or the path of the file containing a SOQL query statement to retrieve the records to export.

Type: string

`-p | --plan`
Optional

Generates mulitple sObject tree files and a plan definition file for aggregated import.

Type: flag

`-x | --prefix PREFIX`
Optional

Prefix of generated files.

Type: string

-d | --outputdir OUTPUTDIR
Optional

Directory to store generated files.

Type: directory

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `data:tree:export`

Generates JSON files for use with the force:data:tree:import command.

Examples:

```
   $ sfdx force:data:tree:export -q "SELECT Id, Name, (SELECT Name, Address__c FROM
Properties__r) FROM Broker__c"
```

```
   $ sfdx force:data:tree:export -q <path to file containing soql query> -x export-demo
-d /tmp/sfdx-out -p
```

`data:tree:import`

Imports data into an org using the SObject Tree Save API. This data can include master-detail relationships.

Command Syntax

sfdx force:data:tree:import
[-f SOBJECTTREEFILES]

[-p PLAN]

[-c CONTENTTYPE]

[--configihelp]

[-u TARGETUSERNAME]

[--json]

```
[--loglevel LOGLEVEL]
```

Parameters

-f | --sobjecttreefiles SOBJECTTREEFILES
Optional

Ordered paths of JSON files containing a collection of record trees to insert. Either --sobjecttreefiles or --plan is required.

Type: filepath

-p | --plan PLAN
Optional

Path to plan to insert multiple data files that have master-detail relationships. Either --sobjecttreefiles or --plan is required.

Type: filepath

-c | --contenttype CONTENTTYPE
Optional

If the data file extension is not .json, provide the content type (applies to all files).

Type: string

--confighelp
Optional

Displays the schema information for the configuration file. If you use this option, all other options, except --json, are ignored.

Type: flag

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `data:tree:import`

To generate JSON files for use with force:data:tree:import, run "sfdx force:data:tree:export".

Examples:

```
$ sfdx force:data:tree:import -p data/accounts-contacts-plan.json -u me@my.org
```

```
$ sfdx force:data:tree:import -f data/accounts-only.json data/contacts-only-1.json
```

```
$ sfdx force:data:tree:import -p ./test/data/accounts-contacts-plan.json
```

doc Commands

Use to display help for force commands.

doc:commands:display

Displays help for force commands.

Command Syntax

sfdx force:doc:commands:display
[--json]
[--loglevel LOGLEVEL]

Parameters

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for doc:commands:display

Displays help for force commands.

`doc:commands:list`

Displays a list of force commands.

Command Syntax

`sfdx force:doc:commands:list`
[`--json`]
[`--loglevel LOGLEVEL`]

Parameters

`--json`
Optional

Format output as JSON.

Type: flag

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `doc:commands:list`

Displays a list of force commands and their descriptions.

`lightning` Commands

Use the lightning commands to create and test Lightning component bundles.

lightning:app:create
Creates a Lightning app bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

lightning:component:create
Creates a Lightning component bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

lightning:event:create
Creates a Lightning event bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

lightning:interface:create
Creates a Lightning interface bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

lightning:test:create

Creates a Lightning test in the specified directory or the current working directory. The .resource file and associated metadata file are created.

lightning:test:run

Runs Lightning component tests. The Lightning Testing Service (LTS) unmanaged package must be installed in your org. For details, see the LTS documentation.

`lightning:app:create`

Creates a Lightning app bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

Command Syntax

```
sfdx force:lightning:app:create
   -n APPNAME
   [-t TEMPLATE]
   [-d OUTPUTDIR]
   [-a APIVERSION]
   [--json]
   [--loglevel LOGLEVEL]
```

Parameters

`-n | --appname APPNAME`
Required

The Lightning app name. The name can be up to 40 characters and must start with a letter.

Type: string

`-t | --template TEMPLATE`
Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: DefaultLightningApp

Default value: DefaultLightningApp

`-d | --outputdir OUTPUTDIR`
Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

`-a | --apiversion APIVERSION`
Optional

The API version of the created source.

Type: string

Permissible values are: 40.0, 39.0

Default value: 40.0

--json
Optional

Formats output as JSON.

Type: string

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `lightning:app:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
$ sfdx force:lightning:app:create -n myapp
```

```
$ sfdx force:lightning:app:create -n myapp -d lightning
```

`lightning:component:create`

Creates a Lightning component bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

Command Syntax

sfdx force:lightning:component:create
-n COMPONENTNAME

[-t TEMPLATE]

[-d OUTPUTDIR]

[-a APIVERSION]

[--json]

[--loglevel LOGLEVEL]

Parameters

-n | --componentname COMPONENTNAME
Required

The Lightning component name. The name can be up to 40 characters and must start with a letter.

Type: string

-t | --template TEMPLATE
Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: DefaultLightningCmp

Default value: DefaultLightningCmp

-d | --outputdir OUTPUTDIR
Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

-a | --apiversion APIVERSION
Optional

The API version of the created source.

Type: string

Permissible values are: 40.0, 39.0

Default value: 40.0

--json
Optional

Formats output as JSON.

Type: string

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `lightning:component:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
$ sfdx force:lightning:component:create -n mycomponent
```

```
$ sfdx force:lightning:component:create -n mycomponent -d lightning
```

`lightning:event:create`

Creates a Lightning event bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

Command Syntax

```
sfdx force:lightning:event:create
   -n EVENTNAME
   [-t TEMPLATE]
   [-d OUTPUTDIR]
   [-a APIVERSION]
   [--json]
   [--loglevel LOGLEVEL]
```

Parameters

-n | --eventname EVENTNAME
Required

The Lightning event name. The name can be up to 40 characters and must start with a letter.

Type: string

-t | --template TEMPLATE
Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: DefaultLightningEvt

Default value: DefaultLightningEvt

-d | --outputdir OUTPUTDIR
Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

-a | --apiversion APIVERSION
Optional

The API version of the created source.

Type: string

Permissible values are: 40.0, 39.0

Default value: 40.0

--json
Optional

Formats output as JSON.

Type: string

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `lightning:event:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
$ sfdx force:lightning:event:create -n myevent
```

```
$ sfdx force:lightning:event:create -n myevent -d lightning
```

`lightning:interface:create`

Creates a Lightning interface bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

Command Syntax

sfdx force:lightning:interface:create
-n INTERFACENAME
[-t TEMPLATE]
[-d OUTPUTDIR]
[-a APIVERSION]
[--json]
[--loglevel LOGLEVEL]

Parameters

-n | --interfacename INTERFACENAME
Required

The Lightning interface name. The name can be up to 40 characters and must start with a letter.

Type: string

-t | --template TEMPLATE
Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: DefaultLightningIntf

Default value: DefaultLightningIntf

-d | --outputdir OUTPUTDIR
Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

-a | --apiversion APIVERSION
Optional

The API version of the created source.

Type: string

Permissible values are: 40.0, 39.0

Default value: 40.0

--json
Optional

Formats output as JSON.

Type: string

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `lightning:interface:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
$ sfdx force:lightning:interface:create -n myinterface
```

```
$ sfdx force:lightning:interface:create -n myinterface -d lightning
```

`lightning:test:create`

Creates a Lightning test in the specified directory or the current working directory. The .resource file and associated metadata file are created.

Command Syntax

sfdx force:lightning:test:create
-n TESTNAME

[-t TEMPLATE]

```
[-d OUTPUTDIR]
[--json]
[--loglevel LOGLEVEL]
```

Parameters

-n | --testname TESTNAME
Required

The name of the new Lightning test. The name can be up to 40 characters and must start with a letter.

Type: string

-t | --template TEMPLATE
Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: DefaultLightningTest

Default value: DefaultLightningTest

-d | --outputdir OUTPUTDIR
Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

--json
Optional

Formats output as JSON.

Type: string

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `lightning:test:create`

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
$ sfdx force:lightning:test:create -n MyLightningTest
```

```
$ sfdx force:lightning:test:create -n MyLightningTest -d lightningTests
```

`lightning:test:run`

Runs Lightning component tests. The Lightning Testing Service (LTS) unmanaged package must be installed in your org. For details, see the LTS documentation.

Command Syntax

`sfdx force:lightning:test:run`
[-a APPNAME]
[-d OUTPUTDIR]
[-r RESULTFORMAT]
[-f CONFIGFILE]
[-o]
[-t TIMEOUT]
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]

Parameters

-a | --appname APPNAME
Optional

Name of your Lightning test application. The name is case insensitive, and ".app" is optional, so "Test" and "test.app" are equivalent.

Default value: Test.app

Type: string

-d | --outputdir OUTPUTDIR
Optional

Directory path to store test run artifacts: log files, test results, etc.

Type: directory

-r | --resultformat RESULTFORMAT
Optional

Format to use when displaying test results. If you also specify the --json flag, --json overrides this parameter.

Type: string

Permissible values are: human, tap, junit, json

Default value: human

-f | --configfile CONFIGFILE
Optional

Path to a test configuration file to configure WebDriver and other settings. For details, see the LTS documentation.

Type: filepath

-o | --leavebrowseropen
Optional

Leaves browser open after the test finishes so that you can view the test suite results.

Type: flag

-t | --timeout TIMEOUT

Optional

Time, in milliseconds, to wait for the results element to be present in the DOM, before failing and moving on to the next test.

Type: number

Default value: 20000

-u | --targetusername TARGETUSERNAME

Optional

Username for the target org. Overrides the default target org.

Type: string

--json

Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `lightning:test:run`

Examples:

```
$ sfdx force:lightning:test:run
```

```
$ sfdx force:lightning:test:run -a tests -r human
```

```
$ sfdx force:lightning:test:run -f config/myConfigFile.json -d testResultFolder
```

`limits` Commands

Use the limits commands to view your org's limits and how close you are to reaching them.

`limits:api:display`

Displays remaining and maximum calls and events for your org.

Command Syntax

```
sfdx force:limits:api:display
   [-u TARGETUSERNAME]
   [--json]
   [--loglevel LOGLEVEL]
```

Parameters

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `limits:api:display`

When you execute this command in a project, it provides limit information for your default scratch org.

Examples:

```
$ sfdx force:limits:api:display
```

```
$ sfdx force:limits:api:display -u me@my.org
```

`mdapi` Commands

Use the mdapi commands to retrieve and deploy Metadata API–formatted files that represent components in an org or to convert Metadata API–formatted source into the format used in Salesforce DX projects.

mdapi:convert
Converts source retrieved from Metadata API into the format used in Salesforce DX projects.

mdapi:deploy
Deploys file representations of components into an org by creating or updating the components they represent. You can deploy and retrieve up to 10,000 files or 400 MB (39 MB compressed) at one time. The default target username is the admin user for the default scratch org.

mdapi:retrieve

Uses Metadata API to retrieve a .zip of XML files that represent metadata from the targeted org. The default target username is the admin user for the default scratch org. You can retrieve and deploy up to 10,000 files or 400 MB (39 MB compressed) at one time.

`mdapi:convert`

Converts source retrieved from Metadata API into the format used in Salesforce DX projects.

Command Syntax

```
sfdx force:mdapi:convert
   -r ROOTDIR
   [-d OUTPUTDIR]
   [--json]
   [--loglevel LOGLEVEL]
```

Parameters

`-r | --rootdir ROOTDIR`
Required

The root directory that contains the source you retrieved using Metadata API.

Type: directory

`-d | --outputdir OUTPUTDIR`
Optional

The directory to store your source in after it's converted to the Salesforce DX format. Can be an absolute or relative path.

Type: directory

`--json`
Optional

Format output as JSON.

Type: flag

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `mdapi:convert`

This command must be run in a project.

To work with source that you retrieved via Metadata API using the Salesforce DX tools, convert the source to the Salesforce DX format using "sfdx force:mdapi:convert".

To convert the source back to the Metadata API format, so that you can deploy it using "sfdx force:mdapi:deploy", run "sfdx force:source:convert".

Examples:

```
$ sfdx force:mdapi:convert -r <path to source>
```

```
$ sfdx force:mdapi:convert -r <path to source> -d <path to outputdir>
```

mdapi:deploy

Deploys file representations of components into an org by creating or updating the components they represent. You can deploy and retrieve up to 10,000 files or 400 MB (39 MB compressed) at one time. The default target username is the admin user for the default scratch org.

Command Syntax

sfdx force:mdapi:deploy
[-c]
[-d DEPLOYDIR]
[-w WAIT]
[-i JOBID]
[-l TESTLEVEL]
[-r RUNTESTS]
[-e ROLLBACKONERROR]
[-f ZIPFILE]
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]
[--verbose]

Parameters

-c | --checkonly
Optional

Validates the deployed metadata and runs all Apex tests, but prevents the deployment from being saved to the org.

Type: boolean

-d | --deploydir DEPLOYDIR
Optional

The root of the directory tree that contains the files to deploy. The root must contain a valid package.xml file describing the entities in the directory structure. Required to initiate a deployment if you don't use --zipfile. If you specify both --zipfile and --deploydir, a zip file of the contents of the --deploydir directory is written to the location specified by --zipfile.

Type: directory

-w | --wait WAIT
Optional

The number of minutes to wait for the command to complete. The default is –1 (no limit). 0

Type: minutes

-i | --jobid JOBID
Optional

The job ID (asyncId) of the deployment you want to check. Use with -w to resume waiting.

Type: id

-l | --testlevel TESTLEVEL
Optional

Specifies which level of deployment tests to run. Valid values are:

NoTestRun—No tests are run. This test level applies only to deployments to development environments, such as sandbox, Developer Edition, or trial orgs. This test level is the default for development environments.

RunSpecifiedTests—Runs only the tests that you specify in the --runtests option. Code coverage requirements differ from the default coverage requirements when using this test level. Executed tests must comprise a minimum of 75% code coverage for each class and trigger in the deployment package. This coverage is computed for each class and trigger individually and is different than the overall coverage percentage.

RunLocalTests—All tests in your org are run, except the ones that originate from installed managed packages. This test level is the default for production deployments that include Apex classes or triggers.

RunAllTestsInOrg—All tests in your org are run, including tests of managed packages.

If you don't specify a test level, the default behavior depends on the contents of your deployment package. For more information, see "Running Tests in a Deployment" in the Metadata API Guide.

Type: string

Permissible values are: NoTestRun, RunSpecifiedTests, RunLocalTests, RunAllTestsInOrg

-r | --runtests RUNTESTS
Optional

Lists the Apex classes containing the deployment tests to run. Use this parameter when you set --testlevel to RunSpecifiedTests.

Type: string

-e | --rollbackonerror ROLLBACKONERROR
Optional

Indicates whether any failure causes a complete rollback of the deploy operation. The default is true. If set to false, the operation performs actions that don't have errors and returns errors for the remaining actions. You must set this parameter to true if you are deploying to a production org.

Type: boolean

Default value: true

-f | --zipfile ZIPFILE
Optional

The path to the .zip file of metadata files to deploy. Required to initiate a deployment if you do not use --deploydir. If you specify both --zipfile and --deploydir, a zip file of the contents of the --deploydir directory is written to the location specified by --zipfile.

Type: filepath

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

--verbose
Optional

Indicates that you want verbose output from the deploy operation.

Type: flag

Help for `mdapi:deploy`

Specify the location of the files to deploy as a .zip file or by the root of the directory tree containing the files. To check the status of a deployment, specify its job ID.

The default value of --rollbackonerror is true, but the corresponding parameter in the Metadata API deploy() call defaults to false.

`mdapi:retrieve`

Uses Metadata API to retrieve a .zip of XML files that represent metadata from the targeted org. The default target username is the admin user for the default scratch org. You can retrieve and deploy up to 10,000 files or 400 MB (39 MB compressed) at one time.

Command Syntax

sfdx force:mdapi:retrieve
[-a APIVERSION]

[-w WAIT]

-r RETRIEVETARGETDIR

[-k UNPACKAGED]

[-d SOURCEDIR]

[-p PACKAGENAMES]

[-s]

[-i JOBID]

```
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]
[--verbose]
```

Parameters

-a | --apiversion APIVERSION
Optional

Use to override the default, which is the latest version supported by your CLI plug-in, with the version in your package.xml file.

Type: number

-w | --wait WAIT
Optional

The number of minutes to wait for the command to complete. -1

Type: minutes

-r | --retrievetargetdir RETRIEVETARGETDIR
Required

The root of the directory structure where the retrieved zip or metadata files are put.

Type: directory

-k | --unpackaged UNPACKAGED
Optional

The complete path for the manifest file that specifies the components to retrieve.

Type: filepath

-d | --sourcedir SOURCEDIR
Optional

The source directory to use instead of the default manifest specified in sfdx-config.json.

Type: directory

-p | --packagenames PACKAGENAMES
Optional

A comma-separated list of package names to retrieve.

Type: string

-s | --singlepackage
Optional

Specifies whether only a single package is being retrieved (true) or more than one package (false).

Type: flag

-i | --jobid JOBID
Optional

The job ID (asyncId) of the retrieve you want to check. You must specify a --retrievetargetdir. Use with --wait to resume waiting.

Type: id

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

--verbose
Optional

Indicates that you want verbose output from the retrieve operation.

Type: flag

Help for `mdapi:retrieve`

The default target username is the admin user for the default scratch org. You can retrieve and deploy up to 10,000 files or 400 MB (39 MB compressed) at one time.

`org` Commands

Use the org commands to manage the orgs you use with Salesforce DX. Create and delete scratch orgs, list your created and authorized orgs, and open orgs in your browser.

org:create
Creates a scratch org using values specified in a configuration file or key=value pairs that you specify on the command line. Values specified on the command line override values in the configuration file.

org:delete
Marks a scratch org for deletion.

org:display
Gets the description for the current or target org.

org:list
Lists all orgs that the Salesforce CLI has created or authenticated to.

org:open
Opens an org in your browser.

`org:create`

Creates a scratch org using values specified in a configuration file or key=value pairs that you specify on the command line. Values specified on the command line override values in the configuration file.

Command Syntax

`sfdx force:org:create`
[-f DEFINITIONFILE]
[-n]
[-i CLIENTID]
[-s]
[-a SETALIAS]
[-w WAIT]
[-v TARGETDEVHUBUSERNAME]
[--json]
[--loglevel LOGLEVEL]

Parameters

`-f | --definitionfile DEFINITIONFILE`
Optional

Path to a scratch org definition file. Either --definitionfile or --definitionjson is required.

Type: filepath

`-n | --nonamespace`
Optional

Creates the scratch org with no namespace. Useful when using a scratch org to test installations of packages with namespaces.

Type: flag

`-i | --clientid CLIENTID`
Optional

Connected app consumer key, as configured in your Dev Hub.

Type: string

`-s | --setdefaultusername`
Optional

Sets the created org as the default username.

Type: flag

`-a | --setalias SETALIAS`
Optional

Set an alias for for the created scratch org.

Type: string

`-w | --wait WAIT`
Optional

Sets the streaming client socket timeout, in minutes.If the streaming client socket has no contact from the server for a number of minutes, the client exits. Specify a longer wait time if timeouts occur frequently.

Type: minutes

Default value: 6

-v | --targetdevhubusername TARGETDEVHUBUSERNAME

Optional

A username for the target Dev Hub org. Overrides default Dev Hub org.

Type: string

--json

Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `org:create`

To set up a connected app for your new scratch org, specify the value that was returned when you created a connected app in your Dev Hub org as --clientid.

Examples:

```
   $ sfdx force:org:create -f config/enterprise-scratch-def.json -a TestOrg1
```

```
   $ sfdx force:org:create -a MyDevOrg -s -v me@myhub.org edition=Developer
```

```
   $ sfdx force:org:create -f config/enterprise-scratch-def.json -a OrgWithOverrides
username=testuser1@mycompany.org
```

`org:delete`

Marks a scratch org for deletion.

Command Syntax

sfdx force:org:delete

[-p]

-u TARGETUSERNAME

[--json]

[--loglevel LOGLEVEL]

Parameters

-p | --noprompt
Optional

No prompt to confirm deletion.

Type: flag

-u | --targetusername TARGETUSERNAME
Required

Username for the target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `org:delete`

To mark the org for deletion without being prompted to confirm, specify --noprompt.

Examples:

```
$ sfdx force:org:delete -u me@my.org
```

```
$ sfdx force:org:delete -u MyOrgAlias -p
```

`org:display`

Gets the description for the current or target org.

Command Syntax

sfdx force:org:display
[-u TARGETUSERNAME]

[--json]

[--loglevel LOGLEVEL]

[--verbose]

Parameters

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

--verbose
Optional

Emit additional command output to stdout.

Type: flag

Help for `org:display`

Output includes your access token, client ID, connected status, org ID, instance URL, username, and alias, if applicable. Use --verbose to include the SFDX auth URL.

Examples:

```
$ sfdx force:org:display
```

```
$ sfdx force:org:display -u me@my.org
```

```
$ sfdx force:org:display -u TestOrg1 --json
```

```
$ sfdx force:org:display -u TestOrg1 --json > tmp/MyOrgDesc.json
```

`org:list`

Lists all orgs that the Salesforce CLI has created or authenticated to.

Command Syntax

sfdx force:org:list
[--all]

[--clean]

[-p]

```
[--json]
[--loglevel LOGLEVEL]
```

Parameters

--all
Optional

Lists all authenticated orgs, including expired, deleted, and unknown-status scratch orgs.

Type: flag

--clean
Optional

Remove all local org authorizations for deleted or expired orgs.

Type: flag

-p | --noprompt
Optional

Do not prompt for confirmation.

Type: flag

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `org:list`

Examples:

```
$ sfdx force:org:list
```

```
$ sfdx force:org:list --verbose --json
```

```
$ sfdx force:org:list --verbose --json > tmp/MyOrgList.json
```

`org:open`

Opens an org in your browser.

Command Syntax

sfdx force:org:open
[-p PATH]
[-r]
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]

Parameters

-p | --path PATH
Optional

Navigation URL path (not including domain).

Type: string

-r | --urlonly
Optional

Displays a navigation URL, but doesn't launch your browser.

Type: flag

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `org:open`

Opens your default scratch org, or another org that you specify.

To open a specific page, specify the portion of the URL after "yourInstance.salesforce.com/" as --path.

For example, specify "--path one/one.app" to open Lightning Experience, or specify "--path /apex/YourPage" to open a Visualforce page.

To generate a URL but not launch your browser, specify --urlonly.

Examples:

```
$ sfdx force:org:open
```

```
$ sfdx force:org:open -u me@my.org
```

```
$ sfdx force:org:open -u MyTestOrg1
```

```
$ sfdx force:org:open -r -p one/one.app
```

`package` Commands

Use the package commands to install managed and unmanaged packages.

`package:install`

Installs a package in the target org.

Command Syntax

`sfdx force:package:install`
`-i ID`
`[-w WAIT]`
`[-k INSTALLATIONKEY]`
`[-u TARGETUSERNAME]`
`[--json]`
`[--loglevel LOGLEVEL]`

Parameters

`-i | --id ID`
Required

ID of the package to install (starts with 04t).

Type: id

`-w | --wait WAIT`
Optional

Maximum number of minutes to wait for installation status. The default is 0.

Type: minutes

`-k | --installationkey INSTALLATIONKEY`
Optional

Installation key for installing a key-protected package. The default is null.

Type: string

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package:install`

Supply the ID of the package you want to install. Installs to the default username org unless you supply the username for a different target org.

Examples:

```
$ sfdx force:package:install -i 04t6A0000004eytQAA
```

```
$ sfdx force:package:install -i 04t6A0000004eytQAA -u me@my.org
```

`package:install:get`

Retrieves the status of a package install request.

Command Syntax

sfdx force:package:install:get
-i REQUESTID

[-u TARGETUSERNAME]

[--json]

[--loglevel LOGLEVEL]

Parameters

-i | --requestid REQUESTID
Required

The ID of the PackageInstallRequest.

Type: id

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package:install:get`

Displays the status of a package install request.

`package1` Commands

Use the package1 commands to create and view package versions in your Dev Hub org.

`package1:version:create`

Creates a package version in the release org.

Command Syntax

sfdx force:package1:version:create
```
-i PACKAGEID

-n NAME
```

```
[-d DESCRIPTION]
[-v VERSION]
[-m]
[-r RELEASENOTESURL]
[-p POSTINSTALLURL]
[-k INSTALLATIONKEY]
[-w WAIT]
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]
```

Parameters

-i | --packageid PACKAGEID
Required

ID of the metadata package (starts with 033) of which you're creating a new version.

Type: id

-n | --name NAME
Required

Package version name.

Type: string

-d | --description DESCRIPTION
Optional

Package version description.

Type: string

-v | --version VERSION
Optional

Package version in major.minor format, for example, 3.2.

Type: string

-m | --managedreleased
Optional

Creates a managed package version. To create a beta version, don't include this parameter.

Type: flag

-r | --releasenotesurl RELEASENOTESURL
Optional

Release notes URL.

Type: url

-p | --postinstallurl POSTINSTALLURL
Optional

Post install URL.

Type: url

-k | --installationkey INSTALLATIONKEY
Optional

Installation key for creating the key-protected package. The default is null.

Type: string

-w | --wait WAIT
Optional

Minutes to wait for the package version to be created. The default is 2 minutes.

Type: number

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package1:version:create`

This command must be run in a project.

The package version is based on the contents of the specified metadata package. Omit -m if you want to create an unmanaged package version.

`package1:version:create:get`

Retrieves the status of a PackageUploadRequest.

Command Syntax

```
sfdx force:package1:version:create:get
   -i REQUESTID
   [-u TARGETUSERNAME]
   [--json]
   [--loglevel LOGLEVEL]
```

Parameters

-i | --requestid REQUESTID
Required

The ID of the PackageUploadRequest.

Type: id

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package1:version:create:get`

Displays the status of a package upload request.

`package1:version:display`

Displays detailed information about an individual package version.

Command Syntax

`sfdx force:package1:version:display`
`-i PACKAGEVERSIONID`

`[-u TARGETUSERNAME]`

`[--json]`

`[--loglevel LOGLEVEL]`

Parameters

-i | --packageversionid PACKAGEVERSIONID
Required

ID (starts with 04t) of the metadata package version whose details you want to display.

Type: id

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package1:version:display`

This command must be run in a project.

You can view the metadata package ID, name, release state, and build number.

`package1:version:list`

Lists the versions for the specified package or all packages in the org.

Command Syntax

```
sfdx force:package1:version:list
   [-i PACKAGEID]
   [-u TARGETUSERNAME]
   [--json]
   [--loglevel LOGLEVEL]
```

Parameters

-i | --packageid PACKAGEID
Optional

Metadata package ID (starts with 033) whose package versions you want to list. If not specified, shows all versions for all packages (managed and unmanaged) in the org.

Type: id

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

`--json`
Optional

Format output as JSON.

Type: flag

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package1:version:list`

This command must be run in a project.

If a metadata package ID is specified, lists all versions of the specified package. Otherwise, lists all package versions for the org. For each package version, the list includes the package version ID, metadata package ID, name, version number, and release state.

`package2` Commands (Pilot)

Use the package2 commands to create, install, and manage second-generation packages.

Note: We provide Second-Generation Packaging to selected customers through a pilot program that requires agreement to specific terms and conditions. To be nominated to participate in the program, contact Salesforce. Pilot programs are subject to change, and we can't guarantee acceptance. Second-Generation Packaging isn't generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. We can't guarantee general availability within any particular time frame or at all. Make your purchase decisions only on the basis of generally available products and features.

`package2:create` (Pilot)

Creates a second-generation package.

Command Syntax

```
sfdx force:package2:create
   -n NAME
   [-d DESCRIPTION]
   [-s NAMESPACE]
   [-v TARGETDEVHUBUSERNAME]
   [--json]
   [--loglevel LOGLEVEL]
```

Parameters

`-n | --name NAME`
Required

Name of the package2 to create.

Type: string

`-d | --description DESCRIPTION`
Optional

Description of the package2.

Type: string

`-s | --namespace NAMESPACE`
Optional

Global namespace for the package2.

Type: string

-v | --targetdevhubusername TARGETDEVHUBUSERNAME
Optional

A username for the target Dev Hub org. Overrides default Dev Hub org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package2:create`

This command must be run in a project.

First, use this command to create a package2. Then create a package2 version.

Examples:

```
   $ sfdx force:package2:create --name PackageName --namespace MyNamespace --description
'My New Package'
```

Run 'sfdx force:package2:list' to list all package2 packages in the dev hub org.

`package2:installed:list` (Pilot)

Lists all subscriber package2 versions installed in the target org.

Command Syntax

sfdx force:package2:installed:list
[-u TARGETUSERNAME]

[--json]

[--loglevel LOGLEVEL]

Parameters

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package2:installed:list`

This command must be run in a project.

Lists all subscriber package2 versions installed in the target org.

`package2:list` (Pilot)

Lists all second-generation packages in the Dev Hub org.

Command Syntax

```
sfdx force:package2:list
   [-v TARGETDEVHUBUSERNAME]
   [--json]
   [--loglevel LOGLEVEL]
```

Parameters

-v | --targetdevhubusername TARGETDEVHUBUSERNAME
Optional

A username for the target Dev Hub org. Overrides default Dev Hub org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package2:list`

You can view the namespace, ID, and subscriber package2 ID for each package2.

`package2:manifest:create` (Pilot)

Creates a second-generation package manifest file in JSON format based on the files in the specified package2 source directory.

Command Syntax

```
sfdx force:package2:manifest:create
   -d DIRECTORY
   [-v TARGETDEVHUBUSERNAME]
   [--json]
   [--loglevel LOGLEVEL]
```

Parameters

`-d | --directory DIRECTORY`
Required

Directory for reading the package2 contents and generating the package2 manifest JSON file.

Type: directory

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`
Optional

A username for the target Dev Hub org. Overrides default Dev Hub org.

Type: string

`--json`
Optional

Format output as JSON.

Type: flag

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package2:manifest:create`

This command must be run in a project.

The manifest must be in the directory specified when you create a package2 version.

`package2:members:list` (Pilot)

Lists all subscriber package2 members in the target org.

Command Syntax

`sfdx force:package2:members:list`
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]

Parameters

`-u | --targetusername TARGETUSERNAME`
Optional
Username for the target org. Overrides the default target org.
Type: string

`--json`
Optional
Format output as JSON.
Type: flag

`--loglevel LOGLEVEL`
Optional
The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.
Type: string
Permissible values are: trace, debug, info, warn, error, fatal
Default value: error

Help for `package2:members:list`

This command must be run in a project.

Lists the history of each package2 member: the package2 version when a member was first added (minimum), the latest package version that contains the member (current), and the latest version that contained any deleted members (maximum). The Subject Manageable State shows the state of the object referenced by the package2 member: beta, deleted, deprecated, installed, released, or unmanaged.

`package2:version:create` (Pilot)

Creates a second-generation package (package2) version in the Dev Hub org.

Command Syntax

`sfdx force:package2:version:create`
-i PACKAGE2ID
-d DIRECTORY

```
[-b BRANCH]
[-t TAG]
[-w WAIT]
[-v TARGETDEVHUBUSERNAME]
[--json]
[--loglevel LOGLEVEL]
```

Parameters

-i | --package2id PACKAGE2ID
Required

ID of parent package2 (starts with 0Ho).

Type: id

-d | --directory DIRECTORY
Required

The directory that contains the manifest, descriptor, and contents of the package2 version.

Type: directory

-b | --branch BRANCH
Optional

The package2 version's branch.

Type: string

-t | --tag TAG
Optional

The package2 version's tag.

Type: string

-w | --wait WAIT
Optional

The number of minutes to wait for the package2 version to be created.

Type: minutes

Default value: 0

-v | --targetdevhubusername TARGETDEVHUBUSERNAME
Optional

A username for the target Dev Hub org. Overrides default Dev Hub org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package2:version:create`

This command must be run in a project.

The package2 version is based on the manifest, descriptor, and package2 contents in the specified directory.

To retrieve details about a package2 version create request, including status and package2 version id (05i), run "sfdx force:package2:version:create:get -i 08c...".

To list package2 version creation requests in the org, run "sfdx force:package2:version:create:list".

Examples:

```
   $ sfdx force:package2:version:create --package2id 0Ho... --directory common --tag
'Release 1.0.0' --branch master
```

```
   $ sfdx force:package2:version:create -i 0Ho... -d common
```

`package2:version:create:get` (Pilot)

Retrieves a second-generation package version creation request in the Dev Hub org.

Command Syntax

`sfdx force:package2:version:create:get`
`-i PACKAGE2CREATEREQUESTID`

`[-v TARGETDEVHUBUSERNAME]`

`[--json]`

`[--loglevel LOGLEVEL]`

Parameters

`-i | --package2createrequestid PACKAGE2CREATEREQUESTID`
Required

The ID of the package2 version creation request you want to display.

Type: id

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`
Optional

A username for the target Dev Hub org. Overrides default Dev Hub org.

Type: string

`--json`
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package2:version:create:get`

Specify the request ID for which you want to view details. If applicable, the command displays errors related to the request.

Examples:

```
$ sfdx force:package2:version:create:get --package2createrequestid 08c...
```

To show all requests in the org, run "sfdx package2:version:create:list".

`package2:version:create:list` (Pilot)

Lists all requests to create second-generation package (package2) versions in the Dev Hub org.

Command Syntax

sfdx force:package2:version:create:list
[-c CREATEDLASTDAYS]

[-s STATUS]

[-v TARGETDEVHUBUSERNAME]

[--json]

[--loglevel LOGLEVEL]

Parameters

-c | --createdlastdays CREATEDLASTDAYS
Optional

Lists the requests made in the last specified number of days, starting at 00:00:00 of first day to now. Use 0 for today.

Type: number

-s | --status STATUS
Optional

Filters the list based on the status of version creation requests.

Type: string

Permissible values are: Queued, InProgress, Success, Error

-v | --targetdevhubusername TARGETDEVHUBUSERNAME
Optional

A username for the target Dev Hub org. Overrides default Dev Hub org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package2:version:create:list`

Shows the details of each request to create a package2 version that's run in the Dev Hub org.

All filter parameters are applied using the AND logical operator (not OR).

To get information about a specific request, run "sfdx force:package2:version:create:get" and supply the request ID.

Examples:

```
$ sfdx force:package2:version:create:list
```

```
$ sfdx force:package2:version:create:list --createdlastdays 3
```

```
$ sfdx force:package2:version:create:list --status Error
```

```
$ sfdx force:package2:version:create:list -s InProgress
```

```
$ sfdx force:package2:version:create:list -c 3 -s Success
```

`package2:version:get` (Pilot)

Retrieves a package version in the Dev Hub org

Command Syntax

sfdx force:package2:version:get
-i PACKAGE2VERSIONID

[-v TARGETDEVHUBUSERNAME]

[--json]

[--loglevel LOGLEVEL]

Parameters

-i | --package2versionid PACKAGE2VERSIONID
Required

The package2 version ID (starts with 05i).

Type: id

-v | --targetdevhubusername TARGETDEVHUBUSERNAME
Optional

A username for the target Dev Hub org. Overrides default Dev Hub org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package2:version:get`

This command must be run in a project.

Examples:

```
$ sfdx force:package2:version:get --package2versionid 05i...
```

To update package version values, run "sfdx force:package2:version:update".

`package2:version:install` (Pilot)

Installs a second-generation subscriber package version in the target org.

Command Syntax

sfdx force:package2:version:install
-i SUBSCRIBERPACKAGE2VERSIONID

[-u TARGETUSERNAME]

[--json]

[--loglevel LOGLEVEL]

Parameters

-i | --subscriberpackage2versionid SUBSCRIBERPACKAGE2VERSIONID
Required

The ID of the subscriber package2 version to install (starts with 04t).

Type: id

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package2:version:install`

This command must be run in a project.

Installs a second-generation subscriber package version in the target org. To view the error messages, open the Dev Hub org, select the Setup menu, enter Deployment in the Quick Find box, and select Deployment Status.

Examples:

```
$ sfdx force:package2:version:install --subscriberpackage2versionid 04t...
```

```
$ sfdx force:package2:version:install -i 04t... -u <username of target org>
```

To find the ID for the subscriber package2 version, run "sfdx package2:version:list".

To list the org's installed subscriber package2 versions, run "package2:installed:list".

`package2:version:list` (Pilot)

Lists all package2 versions in the Dev Hub org.

Command Syntax

sfdx force:package2:version:list
[-c CREATEDLASTDAYS]

[-m MODIFIEDLASTDAYS]

[-i PACKAGE2IDS]

[-r]

[-o ORDERBY]

[-v TARGETDEVHUBUSERNAME]

[--concise]

```
[--json]
[--loglevel LOGLEVEL]
[--verbose]
```

Parameters

-c | --createdlastdays CREATEDLASTDAYS
Optional

Lists the requests made in the last specified number of days, starting at 00:00:00 of first day to now. Use 0 for today.

Type: number

-m | --modifiedlastdays MODIFIEDLASTDAYS
Optional

Lists the items modified in the last given number of days, starting at 00:00:00 of first day to now. Use 0 for today.

Type: number

-i | --package2ids PACKAGE2IDS
Optional

Filters results on the specified comma-delimited package2 IDs (start with 0Ho).

Type: string

-r | --released
Optional

Displays released versions only (IsBeta=false).

Type: flag

-o | --orderby ORDERBY
Optional

Orders the list by the specified package2 version fields.

Type: string

-v | --targetdevhubusername TARGETDEVHUBUSERNAME
Optional

A username for the target Dev Hub org. Overrides default Dev Hub org.

Type: string

--concise
Optional

Displays limited package2 version details.

Type: flag

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

--verbose

Optional

Display extended package2 versions detail.

Type: flag

Help for `package2:version:list`

Displays details of each package2 version in the org.

Use --concise or --verbose to display limited or additional details, respectively.

All filter parameters are applied using the AND logical operator (not OR).

Examples:

```
   $ sfdx force:package2:version:list --verbose --createdlastdays 3 --released --orderby
PatchVersion
```

```
   $ sfdx force:package2:version:list --package2ids 0Ho000000000000,0Ho000000000001
--released --modifiedlastdays 0
```

```
   $ sfdx force:package2:version:list --released
```

```
   $ sfdx force:package2:version:list --concise --modifiedlastdays 0
```

```
   $ sfdx force:package2:version:list --concise -c 3 -r
```

`package2:version:uninstall` (Pilot)

Uninstalls a subscriber package2 version in the target org.

Command Syntax

`sfdx force:package2:version:uninstall`

`-i SUBSCRIBERPACKAGE2VERSIONID`

`[-u TARGETUSERNAME]`

`[--json]`

`[--loglevel LOGLEVEL]`

Parameters

`-i | --subscriberpackage2versionid SUBSCRIBERPACKAGE2VERSIONID`

Required

The ID of the subscriber package2 version to uninstall (starts with 04t).

Type: id

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package2:version:uninstall`

This command must be run in a project.

Specify the subscriber package2 version ID.

Examples:

```
$ sfdx force:package2:version:uninstall --subscriberpackage2versionid 04t...
```

```
$ sfdx force:package2:version:uninstall -i 04t... -u <username of target org>
```

To list the org's installed subscriber package2 versions, run "package2:installed:list".

`package2:version:update` (Pilot)

Updates a second-generation package version in the Dev Hub org.

Command Syntax

sfdx force:package2:version:update
-i PACKAGE2VERSIONID

[-n NAME]

[-d DESCRIPTION]

[-b BRANCH]

[-t TAG]

[-s]

[-v TARGETDEVHUBUSERNAME]

[--json]

[--loglevel LOGLEVEL]

Parameters

-i | --package2versionid PACKAGE2VERSIONID
Required

The package2 version ID (starts with 05i).

Type: id

-n | --name NAME
Optional

The package2 version name.

Type: string

-d | --description DESCRIPTION
Optional

The package2 version description.

Type: string

-b | --branch BRANCH
Optional

The package2 version branch.

Type: string

-t | --tag TAG
Optional

The package2 version tag.

Type: string

-s | --setasreleased
Optional

Sets the package2 version as released. Second-generation packages can’t be changed to beta after they've been released.

Type: flag

-v | --targetdevhubusername TARGETDEVHUBUSERNAME
Optional

A username for the target Dev Hub org. Overrides default Dev Hub org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `package2:version:update`

This command must be run in a project.

Specify a new value for each option you want to update.

Examples:

```
$ sfdx force:package2:version:update --package2versionid 05i... --setasreleased
```

```
$ sfdx force:package2:version:update --i 05i... -b master -t 'Release 1.0.7'
```

To display details about a package2 version, run "sfdx package2:version:get".

`project` Commands

Use the project commands to set up a project in the Salesforce DX format.

project:create
Creates a Salesforce DX project in the specified directory or the current working directory. The command creates the necessary configuration files and folders.

project:upgrade
Updates project configuration and definition files to the latest format.

`project:create`

Creates a Salesforce DX project in the specified directory or the current working directory. The command creates the necessary configuration files and folders.

Command Syntax

`sfdx force:project:create`
`-n PROJECTNAME`
`[-t TEMPLATE]`
`[-d OUTPUTDIR]`
`[-s NAMESPACE]`
`[-p DEFAULTPACKAGEDIR]`
`[--json]`
`[--loglevel LOGLEVEL]`

Parameters

`-n | --projectname PROJECTNAME`
Required

The name for the new project. Any valid folder name is accepted.

Type: string

`-t | --template TEMPLATE`
Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: Defaultsfdx-project.json

Default value: Defaultsfdx-project.json

-d | --outputdir OUTPUTDIR
Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

-s | --namespace NAMESPACE
Optional

The namespace associated with this project and any connected scratch orgs.

Type: string

-p | --defaultpackagedir DEFAULTPACKAGEDIR
Optional

The default package directory name. Metadata items such as classes and Lightning bundles are placed inside this folder.

Type: string

Default value: force-app

--json
Optional

Formats output as JSON.

Type: string

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `project:create`

Default values are used if the template, namespace, defaultpackagedir, and outputdir aren't supplied.

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
$ sfdx force:project:create --projectname mywork
```

```
$ sfdx force:project:create --projectname mywork --defaultpackagedir myapp
```

`project:upgrade`

Updates project configuration and definition files to the latest format.

Command Syntax

sfdx force:project:upgrade
[-f]
[--json]
[--loglevel LOGLEVEL]

Parameters

-f | --forceupgrade
Optional
Run all upgrades, even if the project definition files have already been upgraded.
Type: flag

--json
Optional
Format output as JSON.
Type: flag

--loglevel LOGLEVEL
Optional
The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.
Type: string
Permissible values are: trace, debug, info, warn, error, fatal
Default value: error

Help for **project:upgrade**

Examples:

$ sfdx force:project:upgrade

$ sfdx force:project:upgrade -f

schema Commands

Use the schema commands to view and edit the metadata for the standard and custom objects in your org.

schema:sobject:describe
Displays the metadata for a standard or custom object.

schema:sobject:list
Lists all objects, custom objects, or standard objects in the org.

schema:sobject:describe

Displays the metadata for a standard or custom object.

Command Syntax

sfdx force:schema:sobject:describe
[-s SOBJECTTYPE]
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]

Parameters

-s | --sobjecttype SOBJECTTYPE
Optional
The API name of the object to describe.
Type: string

-u | --targetusername TARGETUSERNAME
Optional
Username for the target org. Overrides the default target org.
Type: string

--json
Optional
Format output as JSON.
Type: flag

--loglevel LOGLEVEL
Optional
The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.
Type: string
Permissible values are: trace, debug, info, warn, error, fatal
Default value: error

Help for **schema:sobject:describe**

Examples:

```
$ sfdx force:schema:sobject:describe -s Account
```

```
$ sfdx force:schema:sobject:describe -s MyObject__c
```

schema:sobject:list

Lists all objects, custom objects, or standard objects in the org.

Command Syntax

sfdx force:schema:sobject:list
-c SOBJECTTYPECATEGORY

```
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]
```

Parameters

-c | --sobjecttypecategory SOBJECTTYPECATEGORY
Required

The type of objects to list: all, custom, or standard.

Type: string

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `schema:sobject:list`

Examples:

```
$ sfdx force:schema:sobject:list -c all
```

```
$ sfdx force:schema:sobject:list -c custom
```

```
$ sfdx force:schema:sobject:list -c standard
```

`source` Commands

Use the source commands to push and pull source to and from your scratch orgs, to see synchronization changes between your project and scratch orgs, and to prepare your source for deployment to non-scratch orgs.

source:convert
Converts source in a Salesforce DX project into source that you can deploy using Metadata API.

source:open

Opens the specified Lightning Page in Lightning App Builder. Lightning Page files have the suffix .flexipage-meta.xml, and are stored in the flexipages directory. If you specify a different type of file, this command opens your org's home page.

source:pull

Pulls changed source from the scratch org to your project to keep them in sync.

source:push

Pushes changed source from your project to the scratch org to keep them in sync.

source:status

Lists changes that have been made locally, in a scratch org, or both.

source:convert

Converts source in a Salesforce DX project into source that you can deploy using Metadata API.

Command Syntax

sfdx force:source:convert
[-r ROOTDIR]
[-d OUTPUTDIR]
[-n PACKAGENAME]
[--json]
[--loglevel LOGLEVEL]

Parameters

-r | --rootdir ROOTDIR
Optional

The directory that contains the source to convert.

Type: directory

-d | --outputdir OUTPUTDIR
Optional

The output directory to export the Metadata API source to.

Type: directory

-n | --packagename PACKAGENAME
Optional

The name of the package to associate with the Metadata API source.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `source:convert`

This command must be run in a project.

To convert Salesforce DX–formatted source into a format that you can deploy using Metadata API, run "sfdx force:source:convert". Then deploy the source using "sfdx force:mdapi:deploy".

To convert Metadata API–formatted source into the Salesforce DX format, run "sfdx force:mdapi:convert".

Examples:

```
$ sfdx force:source:convert -r <path to source>
```

```
$ sfdx force:source:convert -r <path to source> -d <path to output dir>
```

`source:open`

Opens the specified Lightning Page in Lightning App Builder. Lightning Page files have the suffix .flexipage-meta.xml, and are stored in the flexipages directory. If you specify a different type of file, this command opens your org's home page.

Command Syntax

sfdx force:source:open
-f SOURCEFILE

[-r]

[-u TARGETUSERNAME]

[--json]

[--loglevel LOGLEVEL]

Parameters

-f | --sourcefile SOURCEFILE
Required

File to edit.

Type: file

-r | --urlonly
Optional

Generate a navigation URL path, but don't launch a browser-based editor.

Type: flag

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `source:open`

This command must be run in a project.

If Force.com IDE 2 is installed, the file opens in an embedded browser within the IDE. If Force.com IDE 2 isn't installed, the file opens in your default browser.

If no browser-based editor is available for the selected file, this command opens your org's home page.

To generate a URL for the browser-based editor but not open the editor, use --urlonly.

Examples:

```
$ sfdx force:source:open -f Property_Record_Page.flexipage-meta.xml
```

```
$ sfdx force:source:open -f Property_Record_Page.flexipage-meta.xml -r
```

`source:pull`

Pulls changed source from the scratch org to your project to keep them in sync.

Command Syntax

sfdx force:source:pull
[-w WAIT]

[-f]

[-u TARGETUSERNAME]

[--json]

[--loglevel LOGLEVEL]

Parameters

-w | --wait WAIT
Optional

The number of minutes to wait for the command to complete and display results to the terminal window. If the command continues to run after the wait period, the CLI returns control of the terminal window to you. The default is 33 minutes.

Type: minutes

Default value: 33

-f | --forceoverwrite
Optional

Runs the pull command even if conflicts exist. Changes in the scratch org overwrite changes in the project.

Type: flag

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `source:pull`

This command must be run in a project.

If the command detects a conflict, it displays the conflicts but does not complete the process. After reviewing the conflict, rerun the command with the --forceoverwrite parameter.

`source:push`

Pushes changed source from your project to the scratch org to keep them in sync.

Command Syntax

sfdx force:source:push
[-w WAIT]

[-g]

[-f]

```
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]
```

Parameters

-w | --wait WAIT
Optional

Number of minutes to wait for the command to complete and display results to the terminal window. If the command continues to run after the wait period, the CLI returns control of the terminal window to you. The default is 33 minutes.

Type: minutes

Default value: 33

-g | --ignorewarnings
Optional

Completes the deployment even if warnings are generated.

Type: flag

-f | --forceoverwrite
Optional

Runs the push command even if conflicts exist. Changes in the project overwrite changes in the scratch org.

Type: flag

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `source:push`

This command must be run in a project.

If the command detects a conflict, it displays the conflicts but does not complete the process. After reviewing the conflict, rerun the command with the --forceoverwrite parameter.

source:status

Lists changes that have been made locally, in a scratch org, or both.

Command Syntax

sfdx force:source:status
[-a]
[-l]
[-r]
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]

Parameters

-a | --all
Optional
Lists all the changes that have been made.
Type: flag

-l | --local
Optional
Lists the changes that have been made locally.
Type: flag

-r | --remote
Optional
Lists the changes that have been made in the scratch org.
Type: flag

-u | --targetusername TARGETUSERNAME
Optional
Username for the target org. Overrides the default target org.
Type: string

--json
Optional
Format output as JSON.
Type: flag

--loglevel LOGLEVEL
Optional
The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.
Type: string
Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `source:status`

This command must be run in a project.

Examples:

```
$ sfdx force:source:status -l
```

```
$ sfdx force:source:status -r
```

```
$ sfdx force:source:status -a
```

```
$ sfdx force:source:status -a -u me@my.org --json
```

`user` Commands

Perform user-related admin tasks.

user:password:generate

Generates a password for a scratch org.

user:permset:assign

Assigns a named permission set to the admin user of an org.

`user:password:generate`

Generates a password for a scratch org.

Command Syntax

`sfdx force:user:password:generate`
[`-u TARGETUSERNAME`]

[`-v TARGETDEVHUBUSERNAME`]

[`--json`]

[`--loglevel LOGLEVEL`]

Parameters

`-u | --targetusername TARGETUSERNAME`
Optional

Username for the target org. Overrides the default target org.

Type: string

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`
Optional

A username for the target Dev Hub org. Overrides default Dev Hub org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `user:password:generate`

Generates a random password and sets it for the user of a scratch org.

If you haven't set a default Dev Hub, or if your scratch org isn't associated with your default Dev Hub, --targetdevhubusername is required.

To see a password that was previously generated, run "sfdx force:org:display".

`user:permset:assign`

Assigns a named permission set to the admin user of an org.

Command Syntax

sfdx force:user:permset:assign
```
-n PERMSETNAME
[-u TARGETUSERNAME]
[--json]
[--loglevel LOGLEVEL]
```

Parameters

-n | --permsetname PERMSETNAME
Required

The name of the permission set to assign.

Type: string

-u | --targetusername TARGETUSERNAME
Optional

Username for the target org. Overrides the default target org.

Type: string

--json
Optional

Format output as JSON.

Type: flag

--loglevel LOGLEVEL

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `user:permset:assign`

Examples:

```
$ sfdx force:user:permset:assign -n DreamHouse
```

```
$ sfdx force:user:permset:assign -n DreamHouse -u me@my.org
```

```
$ sfdx force:user:permset:assign -n DreamHouse -u TestOrg1
```

`visualforce` Commands

Use the visualforce commands to create Visualforce pages and components.

visualforce:component:create

Creates a Visualforce component in the specified directory or the current working directory. The command creates the .component file and associated metadata file.

visualforce:page:create

Creates a Visualforce page in the specified directory or the current working directory. The command creates the .page file and associated metadata file.

`visualforce:component:create`

Creates a Visualforce component in the specified directory or the current working directory. The command creates the .component file and associated metadata file.

Command Syntax

sfdx force:visualforce:component:create

[-t TEMPLATE]

[-d OUTPUTDIR]

-n COMPONENTNAME

[-a APIVERSION]

-l LABEL

[--json]

[--loglevel LOGLEVEL]

Parameters

-t | --template TEMPLATE
Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: DefaultVFComponent

Default value: DefaultVFComponent

-d | --outputdir OUTPUTDIR
Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

-n | --componentname COMPONENTNAME
Required

The Visualforce component name. The name can be up to 40 characters and must start with a letter.

Type: string

-a | --apiversion APIVERSION
Optional

The API version of the created source.

Type: string

Permissible values are: 40.0, 39.0

Default value: 40.0

-l | --label LABEL
Required

The label saved in the metadata for the Visualforce component.

Type: string

--json
Optional

Formats output as JSON.

Type: string

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `visualforce:component:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Name and label are required.

Examples:

```
$ sfdx force:visualforce:component:create -n mycomponent -l mylabel
```

```
$ sfdx force:visualforce:component:create -n mycomponent -l mylabel -d components
```

visualforce:page:create

Creates a Visualforce page in the specified directory or the current working directory. The command creates the .page file and associated metadata file.

Command Syntax

sfdx force:visualforce:page:create
[-t TEMPLATE]
[-d OUTPUTDIR]
-n PAGENAME
[-a APIVERSION]
-l LABEL
[--json]
[--loglevel LOGLEVEL]

Parameters

-t | --template TEMPLATE
Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: DefaultVFPage

Default value: DefaultVFPage

-d | --outputdir OUTPUTDIR
Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

-n | --pagename PAGENAME
Required

The Visualforce page name. The name can be up to 40 characters and must start with a letter.

Type: string

-a | --apiversion APIVERSION
Optional

The API version of the created source.

Type: string

Permissible values are: 40.0, 39.0

Default value: 40.0

-l | --label LABEL
Required

The label saved in the metadata for the Visualforce page.

Type: string

--json
Optional

Formats output as JSON.

Type: string

--loglevel LOGLEVEL
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: string

Permissible values are: trace, debug, info, warn, error, fatal

Default value: error

Help for `visualforce:page:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Name and label are required.

Examples:

```
$ sfdx force:visualforce:page:create -n mypage -l mylabel
```

```
$ sfdx force:visualforce:page:create -n mypage -l mylabel -d pages
```