About `dab` CLI

The Data API builder CLI (**dab CLI** or `dab`) is a command line tool that streamlines the local development experience for applications using Data API builder.

Key Features of `dab` CLI

- Initialize the configuration file for REST and GraphQL endpoints
- Add new entities
- Update entity details
- Add/update entity relationships
- Configure roles and their permissions
- Configure cross-origin requests (CORS)
- Run the Data API builder engine

CLI command line

DAB CLI comes with an integrated help system. To get a list of what commands are available, use the `--help` option on the `dab` command.

```
dab --help
```

To get help on a specific command, use the `--help` option. For example, to learn more about the `init` command:

```
dab init --help
```

CLI command line verbs and options

`init`

Initializes the runtime configuration for the Data API builder runtime engine. It creates a new JSON file with the properties provided as options.

Syntax: `dab init [options]`

Example: `dab init --config "dab-config.MsSql.json" --database-type mssql --connection-string "Server=tcp:127.0.0.1,1433;User ID=sa;Password=REPLACEME;Connection Timeout=5;"`

Options	Option Required	Default Value	Value Required	Value Type	Descriptio
--database-type	true	-	true	String	Type of dat connect. Su values: mss cosmosdb_ cosmosdb_ mysql, pos
--connection-string	false	""	true	String	Connectior connect to database.
--cosmosdb_nosql-database	true when databaseType=cosmosdb_nosql	-	true	String	Database r Cosmos DE
--cosmosdb_nosql-container	false	-	true	String	Container r Cosmos DE
--graphql-schema	true when databaseType=cosmosdb_nosql	-	true	String	GraphQL s
--set-session-context	false	false	false	-	Enable sen to MsSql u session cor
--host-mode	false	production	true	String	Specify the mode - dev or product
--cors-origin	false	""	true	String	Specify the allowed ori
--auth.provider	false	StaticWebApps	true	String	Specify the Provider.
--rest.path	false	/api	true	String	Specify the endpoint's

Options	Option Required	Default Value	Value Required	Value Type	Descriptio
--rest.disabled	false	false	false	-	Disables RE endpoint fo entities.
--rest.enabled	false	true	true	-	Enables RE endpoint fo entities.
--rest.request-body-strict	false	true	true	-	Does not a extraneous request bo
--graphql.path	false	/graphql	true	String	Specify the endpoint's
--graphql.disabled	false	false	false	-	Disables Gr endpoint fo entities.
--graphql.enabled	false	true	true	-	Enables Gr endpoint fo entities.
--auth.audience	false	-	true	String	Identifies th recipients t JWT is inte
--auth.issuer	false	-	true	String	Specify the issued the
-c,--config	false	dab-config.json	true	String	Path to cor

add

Add new database entity to the configuration file. Make sure you already have a configuration file before executing this command, otherwise it returns an error.

Syntax: `dab add [entity-name] [options]`

Example:: `dab add Book -c "dab-config.MsSql.json" --source dbo.books --permissions "anonymous:*"`

Options	Option Required	Default Value	Value Required	Value Type	Description
-s,--source	true	-	true	String	Name of the source table or container.
--permissions	true	-	true	String	Permissions required to acce source table or container. Fo [role]:[actions]".
--source.type	false	table	true	String	Type of the database object. one of: [table, view, stored-procedure].
--source.params	false	-	true	String	Dictionary of parameters anc values for Source object."param1:val1,param2:' for Stored-Procedures.
--source.key-fields	true when `--source.type` is view	-	true	String	The field(s) to be used as pri keys for tables and views onl Comma separated values. Ex `source.key-fields "id,name`
--rest	false	case sensitive entity name.	true	String	Route for REST API. Example: `--rest: false` -> Disables RI calls for this entity. `--rest: true` -> Entity name becomes the rest path. `--rest: "customPathName"` -> Provided customPathName b the REST path.
--rest.methods	false	post	true	String	HTTP actions to be supporte stored procedure. Specify th as a comma separated list. V HTTP actions are:[get, post, p patch, delete].
--graphql	false	case sensitive	true	Bool/String	Entity type exposed for Grap Example: `--graphql: false` -> disable

Options	Option Required	Default Value	Value Required	Value Type	Description
		entity name			calls for this entity. `--graphql: true` -> Exposes entity for GraphQL with defa names. The singular form of entity name is considered fo query and mutation names. `--graphql: "customQueryNam` Lets the user customize the s and plural name for queries a mutations.
--graphql.operation	false	mutation	true	String	GraphQL operation to be sup for stored procedure. Valid operations are: [query, mutat
--fields.include	false	-	true	String	Fields with permission to acc
--fields.exclude	false	-	true	String	Fields excluded from the acti
--policy-database	false	-	true	String	Specify an OData style filter r is injected in the query sent t database.
-c,--config	false	dab-config.json	true	String	Path to config file.

`update`

Update the properties of any database entity in the configuration file.

Syntax: `dab update [entity-name] [options]`

Example: `dab update Publisher --permissions "authenticated:*"`

> **NOTE**
>
> `dab update` supports all the options that are supported by `dab add`. Additionally, it also supports the below listed options.

Options	Option Required	Default Value	Value Required	Value Type	Description
--relationship	false	-	true	String	Specify relationship between two entities. Provide the name of the relationship.
--cardinality	true when `--relationship` option is used	-	true	String	Specify cardinality between two entities. Could be one or many.
--target.entity	true when `--relationship` option is used	-	true	String	Another exposed entity that the source entity relates to.
--linking.object	false	-	true	String	Database object that is used to support an M:N relationship.
--linking.source.fields	false	-	true	String	Database fields in the linking object to connect to the related item in the source entity. Comma separated fields.
--linking.target.fields	false	-	true	String	Database fields in the linking object to connect to the related item in the target entity. Comma separated fields.
--relationship.fields	false	-	true	String	Specify fields to be used for mapping the entities. Example: `--relationship.fields "id:book_id"`. Here `id` represents column from sourceEntity, while `book_id` from targetEntity. Foreign keys are required between the underlying sources if not specified
-m,--map	false	-	true	String	Specify mappings between database fields and GraphQL and REST fields. Format: --map

Options	Option Required	Default Value	Value Required	Value Type	Description
					"backendName1:exposedName1, backendName2:exposedName2,...

export

Export the required schema as a file and save to disk based on the options.

Syntax: `dab export [options]`

Example: `dab export --graphql -o ./schemas`

Options	Option Required	Default Value	Value Required	Value Type	Description
--graphql	false	false	false	-	Export GraphQL schema.
-o,--output	true	-	true	String	Specify the directory to save the schema file.
-g,--graphql-schema-file	false	schema.graphql	true	String	Specify the name of the Graphql schema file.
-c,--config	false	dab-config.json	true	String	Path to config file.

start

Start the runtime engine with the provided configuration file for serving REST and GraphQL requests.

Syntax: `dab start [options]`

Example: `dab start`

Options	Option Required	Default Value	Value Required	Value Type	Description
--verbose	false	-	false	-	Specify logging level as informational.
--LogLevel	false	Debug when hostMode=development, else	true	String	Specify logging level as provided value. example: debug,

Options	Option Required	Default Value	Value Required	Value Type	Description
		Error when HostMode=Production			error, information, etc.
--no-https-redirect	false	false	true	String	Disables automatic https redirects.
-c,--config	false	dab-config.json	true	String	Path to config file.

> **NOTE**
> You can't use `--verbose` and `--LogLevel` at the same time. Learn more about different logging levels here.

Using Data API builder with two configuration files

You can maintain multiple pairs of baseline and environment specific configuration files to simplify management of your environment specific settings. The following steps demonstrate how to set up a base configuration file with two environment specific configuration files (**development** and **production**):

1. Create a base configuration file `dab-config.json` with all of your settings common across each of your environments.
2. Create two environemnt specific configuration files: `dab-config.development.json` and `dab-config.production.json`. These two configuration files should only include settings which differ from the base configuration file such as connection strings.
3. Next, set the `DAB_ENVIRONMENT` variable based on the environment configuration you want Data API builder to consume. For this example, set `DAB_ENVIRONMENT=development` so the `development` environment configuration file selected.
4. Start Data API builder with the command `dab start`. The engine checks the value of `DAB_ENVIRONMENT` and uses the base file `dab-config.json` and the environment specific file `dab-config.development.json`. When the engine detects the presence of both files, it merges the files into a third file: `dab-config.development.merged.json`.
5. (Optional) Set the `DAB_ENVIRONMENT` environment variable value to `production` when you want the production environment specific settings to be merged with your base configuration file.

NOTE

1. By default, **dab-config.json** is used to start the engine when the `DAB_ENVIRONMENT` environment variable isn't set.
2. A user provided config file is used regardless of the `DAB_ENVIRONMENT` environment variable's value. For example, the file `my-config.json` is used when Data API builder is started using `dab start -c my-config.json`

Getting Started