Use Adapter SDK

This section contains the following topics:

Prerequisites

To start using the Workflow Adapter SDK, you need to install a **Golang** environment, Protocol buffers, dedicated **go** plugins and download the **Adapter SDK** contained in the CWM software package.

Install Go

To develop and test an adapter, you need to install the **Golang** environment. Follow the installation instructions dedicated for your OS: https://grpc.io/docs/protoc-installation/.

Install Protocol buffers

To define an adapter interface and generate the input and output parameters, you need the Protobufs compiler. Follow the installation instructions dedicated for your OS: https://grpc.io/docs/protoc-installation/. Note that you need at least version **3.15** (proto3).

Install go plugins

Step 1 Install additional protocol compiler plugins for **go**:

```
go install google.golang.org/protobuf/cmd/protoc-gen-go@v1.28
go install google.golang.org/grpc/cmd/protoc-gen-go-grpc@v1.2
```

Step 2 Install protocol compiler plugin for **JSON schema**:

```
go install github.com/chrusty/protoc-gen-jsonschema/cmd/protoc-gen-jsonschema@latest
```

Step 3 Update your system PATH so that the `protoc` compiler can find the plugins:

```
export PATH="$PATH:$(go env GOPATH)/bin
```

Get CWM Adapter SDK

Go to Cisco Software Download page to download the CWM Software Package, where the Adapter SDK resides.

Include the location of cwm-sdk-binaries by setting the environment variable path:

```
export PATH=/path/to/cwm-sdk-binaries:$PATH
```

Note Remember to replace the /path/to/ with your actual path.

Overview of commands

The Adapter SDK application offers the following set of commands for managing an adapter:

- `cwm-sdk create-adapter` - use it to create a go module with a package and the corresponding .proto files).
- `cwm-sdk extend-adapter` - use it to add a new feature to an existing adapter (go package and .proto files).
- `make generate-model` - generate activities, input and output (go code).
- `make generate-code` - update activities, input and output (go code).
- `cwm-sdk upgrade-adapter` - upgrade the adapter to match CWM.
- `cwm-sdk create-installable` - create an archive installable by CWM.

Create a new adapter

To create an adapter, open a terminal and from the `cwmsdk` repository directory, run:

```
cwm-sdk create-adapter [options] -product <product-name>
```

Options

These are the options you can add to the `create-adapter` command:

- `-exclude-resource` - skip creation of the `.resource.proto` file from template.
- `-go-module` *string* - provide name for the module assigned to the go.mod file (default: "www.cisco.com/cwm/adapters/<vendor>/<adapter-name>").
- `-feature` *string* - provide name for the go package assigned to activities (default: "<adapter-name>").
- `-location` *string* - point to adapter location (default: current directory).
- `-os-architecture` *string* - define architecture in which adapter is developed. Valid options are: 'linux','mac-intel','mac-arm' and 'windows' (default: "linux").
- `-vendor` *string* - provide unique name for the company creating the adapter (default "cisco").

- `-product` *string* - provide name for the go module corresponding to the product name you create an adapter for (required).

Output

Once the command is executed, verify the generated output inside the new adapter directory:

- `<adapter-name>/go/go.mod`
- `<adapter-name>/proto/<vendor\>.<module\>.<package\>.adapter.proto`
- `<adapter-name>/proto/<vendor\>.<module\>.<package\>.resource.proto` (if `-exclude-resource` option wasn't used)
- `<adapter-name>/Makefile`

Extend adapter with features

To add a feature (a **go package**) for an adapter, open a terminal and from the `cwmsdk` repository directory, run:

```
cwm-sdk extend-adapter [options] -feature <feature_name>
```

Options

- `-exclude-resource` - skip creation of the `.resource.proto` file from template.
- `-location` *string* - point to the location of the adapter to be extended by the new package (default: current directory).

Output

Once the command is executed, verify the generated output inside the new adapter directory:

- `<adapter-name>/proto/<vendor>.<module>.<package>.adapter.proto`
- `<adapter-name>/proto/<vendor>.<module>.<package>.resource.proto` (if `-exclude-resource` option wasn't used)

Generate input and output

To generate the input and output files for the adapter, go to the root directory of your adapter and run:

```
make generate-model
```

Output

Once the command is executed, verify the generated output inside the adapter directory:

- `go/<feature\>/<vendor>.<product>.<feature>.adapter.pb.go`
- `go/common/<vendor>.<product>.common.adapter.pb.go`

The `.pb.go` files contain **go** structs defining the input and output parameters of the adapter. It should not be altered manually.

Generate activities

To generate the previously defined activities, go to the root directory of your adapter and run: `make generate-code`

Output

Once the command is executed, verify the generated output inside the adapter directory:

- go/<package>/activities.go

The `activities.go` file contains stubs for the gRPCs defined in the `.adapter.proto`. Once generated, you can add functionality to the activities by defining the message .

Upgrade an adapter

To upgrade the **go module** to contain matching versions for go and required imports, open a terminal and from the `cwmsdk` repository directory, run:

"Linux" `cwm-sdk upgrade-adapter [options]`

Options

- `-cwm-version` *string* - provide the version of CWM to upgrade to (default is latest).
- `-location` *string* - point to location of adapter to upgrade (default: current directory).

Output

- go/go.mod

The `go.mod` file module will be modifed allowing the adapter to be installed correctly.

Release an installable adapter

To create an archive for installing your adapter for different operating systems, open a terminal and from the `cwmsdk` repository directory, run:

"Linux" `cwm-sdk create-installable [options]`

This generates code based on the proto file.

Options

- `-location` *string* - point to location for the adapter installable file (default ".").

Output

- out/<vendor>-<product>-v<X.Y.Z>.tar.gz

The generated archive contains the adapter go module and proto files. The go module is modified using the go vendor command in order to not have any external dependencies.