Developing a Source Connector

A Source is responsible for continuously reading data from a third party system and returning it in the form of an SDK Record
.

You need to implement the functions required by the Source interface and provide your own implementations. Information about individual functions are listed below.

source.go

This file provides the main functionality of your Source Connector.

Source Connector Lifecycle Functions

- Source
- Struct: Every Source implementation needs to include an UnimplementedSource
- to satisfy the interface. This allows us to potentially change the interface in the future while remaining backward compatible with existing Source implementations. This struct can be modified to add additional fields that can be accessed throughout the lifecycle of the Connector
- type
- Source
- struct
- {
- sdk
- .
- UnimplementedSource
- config SourceConfig
- lastPositionRead sdk
- .
- Position
- //nolint:unused // this is just an example
- watcher
- *
- fsnotify
- .
- Watcher
- recentlyCreated sync
- .
- Map
- // To keep track of recently created files
- createCooldown time
- .
- Duration
- // Cooldown period after a create event
- }
- NewSource()
- : A constructor function for your Source struct. Note that this is the same function that should be set as the value ofConnector.NewSource
- . The constructor should be used to wrap your Source in the default middleware. You can add additional middleware, but unless you have a very good reason, you should always include the default middleware.
- func
- NewSource
- (
-)
- sdk
- .
- Source
- {
- // Create Source and wrap it in the default middleware.
- return
- sdk
- .
- SourceWithMiddleware
- (
- &

- Source
- {
- }
- ,
- sdk
- .
- DefaultSourceMiddleware
- (
-)
- ...
-)
- }
- Parameters()
- : A map of named Parameters that describe how to configure the connector. This map is typically generates usingparamgen
- .
- func
- (
- s
- *
- Source
-)
- Parameters
- (
-)
- map
- [
- string
-]
- sdk
- .
- Parameter
- {
- return
- s
- .
- config
- .
- Parameters
- (
-)
- }
- Configure()
- : Validates and stores configuration data for the connector. Any complex validation logic should be implemented here.
- func
- (
- s
- *
- Source
-)
- Configure
- (
- ctx context
- .
- Context
- ,
- cfg
- map
- [
- string
-]
- string
-)
- error
- {
- sdk
- .
- Logger

- (
- ctx
-)
- .
- Info
- (
-)
- .
- Msg
- (
- "Configuring Source..."
-)
- err
- :=
- sdk
- .
- Util
- .
- ParseConfig
- (
- cfg
- ,
- &
- s
- .
- config
-)
- if
- err
- !=
- nil
- {
- return
- fmt
- .
- Errorf
- (
- "invalid config: %w"
- ,
- err
-)
- }
- // add custom validations here
- return
- nil
- }
- Open()
- : Prepares the connector to start producing records based on the last known successful position. If needed, the connector should open connections in this function.
- func
- (
- s
- *
- Source
-)
- Open
- (
- ctx context
- .
- Context
- ,
- pos sdk
- .
- Position
-)
- error
- {
- configDirPath

- :=
- s
- .
- config
- .
- Directory
- files
- ,
- err
- :=
- ioutil
- .
- ReadDir
- (
- configDirPath
-)
- if
- err
- !=
- nil
- {
- return
- fmt
- .
- Errorf
- (
- "error reading directory '%s': %w"
- ,
- configDirPath
- ,
- err
-)
- }
- for
- _
- ,
- f
- :=
- range
- files
- {
- sdk
- .
- Logger
- (
- ctx
-)
- .
- Info
- (
-)
- .
- Msgf
- (
- " - %s\n"
- ,
- f
- .
- Name
- (
-)
-)
- }
- w
- ,
- err
- :=
- fsnotify

- .
- NewWatcher
- (
-)
- if
- err
- !=
- nil
- {
- return
- fmt
- .
- Errorf
- (
- "error creating fsnotify watcher: %w"
- ,
- err
-)
- }
- s
- .
- watcher
- =
- w
- s
- .
- createCooldown
- =
- 2
- *
- time
- .
- Second
- return
- s
- .
- watcher
- .
- Add
- (
- s
- .
- config
- .
- Directory
-)
- }
- Read()
- : Gathers data from the configured data source and formats it into asdk.Record
- that is returned from the function. The returnedsdk.Record
- is queued into the pipeline to be consumed by a Destination connector.
- func
- (
- s
- *
- Source
-)
- Read
- (
- ctx context
- .
- Context
-)
- (
- sdk
- .
- Record
- ,

- error
-)
- {
- for
- {
- select
- {
- case
- event
- ,
- ok
- :=
- <-
- s
- .
- watcher
- .
- Events
- :
- if
- !
- ok
- {
- return
- sdk
- .
- Record
- {
- }
- ,
- fmt
- .
- Errorf
- (
- "events channel was closed"
-)
- }
- if
- event
- .
- Op
- &
- fsnotify
- .
- Create
- ==
- fsnotify
- .
- Create
- {
- sdk
- .
- Logger
- (
- ctx
-)
- .
- Info
- (
-)
- .
- Msgf
- (
- "Detected new file: %s"
- ,
- event
- .
- Name

-)
- // Mark the file as recently created to avoid processing if it is modified shortly after being created
- s
- .
- markAsRecentlyCreated
- (
- event
- .
- Name
-)
- // Read the newly created file
- fileContent
- ,
- err
- :=
- ioutil
- .
- ReadFile
- (
- event
- .
- Name
-)
- if
- err
- !=
- nil
- {
- return
- sdk
- .
- Record
- {
- }
- ,
- err
- }
- recordKey
- :=
- sdk
- .
- RawData
- (
- filepath
- .
- Base
- (
- event
- .
- Name
-)
-)
- recordValue
- :=
- sdk
- .
- RawData
- (
- fileContent
-)
- // Return a Record reflecting that a new file has been created
- return
- sdk
- .
- Util
- .
- Source
- .

- NewRecordCreate
- (
- sdk
- .
- Position
- (
- recordKey
-)
- ,
- map
- [
- string
-]
- string
- {
- MetadataFilePath
- :
- event
- .
- Name
- ,
- }
- ,
- recordKey
- ,
- recordValue
- ,
-)
- ,
- nil
- }
- // If the event is not a Create event, continue listening without doing anything
- continue
- case
- err
- ,
- ok
- :=
- <-
- s
- .
- watcher
- .
- Errors
- :
- if
- !
- ok
- {
- return
- sdk
- .
- Record
- {
- }
- ,
- fmt
- .
- Errorf
- (
- "errors channel was closed"
-)
- }
- return
- sdk
- .
- Record
- {

- }
- ,
- fmt
- .
- Errorf
- (
- "error on watcher: %w"
- ,
- err
-)
- case
- <-
- ctx
- .
- Done
- (
-)
- :
- return
- sdk
- .
- Record
- {
- }
- ,
- ctx
- .
- Err
- (
-)
- }
- }
- }
- Ack()
- : Ack signals to the implementation that the record with the supplied position was successfully processed.
- func
- (
- s
- *
- Source
-)
- Ack
- (
- ctx context
- .
- Context
- ,
- position sdk
- .
- Position
-)
- error
- {
- sdk
- .
- Logger
- (
- ctx
-)
- .
- Debug
- (
-)
- .
- Msg
- (
- "Record successfully processed"
-)

- return
- nil
- }
- Teardown()
- : Teardown signals to the connector that there will be no more calls to any other function. Any connections that were created in theOpen()
- function should be closed here.
- func
- (
- s
- *
- Source
-)
- Teardown
- (
- ctx context
- .
- Context
-)
- error
- {
- if
- s
- .
- watcher
- !=
- nil
- {
- err
- :=
- s
- .
- watcher
- .
- Close
- (
-)
- if
- err
- !=
- nil
- {
- // Log the error or handle it as needed
- sdk
- .
- Logger
- (
- ctx
-)
- .
- Error
- (
-)
- .
- Msgf
- (
- "Failed to close fsnotify watcher: %v"
- ,
- err
-)
- return
- fmt
- .
- Errorf
- (
- "failed to close fsnotify watcher: %w"
- ,
- err

-)
- }
- }
- return
- nil
- } Edit this page Previous Connector Specification Next Developing a Destination Connector