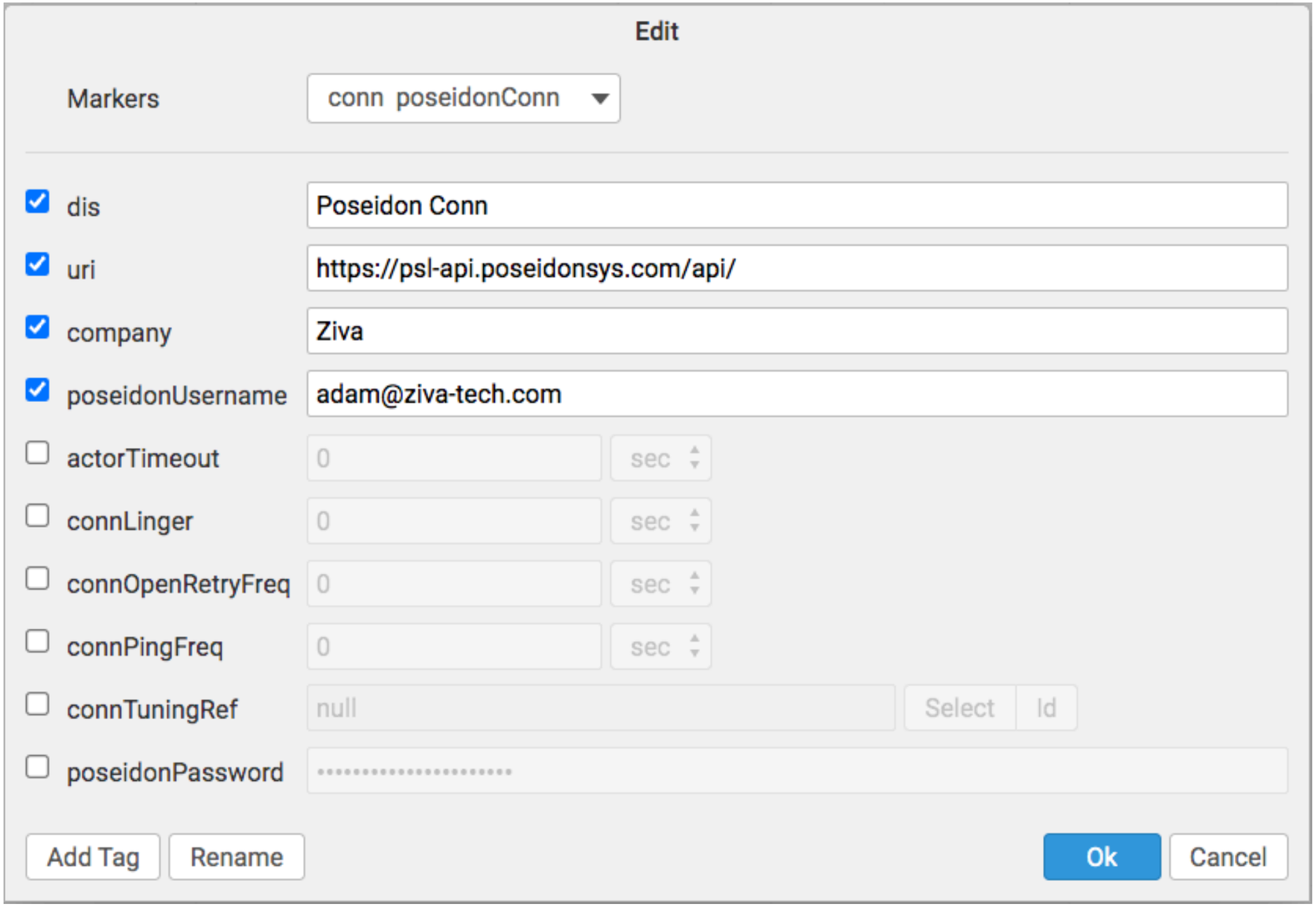
Poseidon Live

- Poseidon Live is a system that contains real-time oil monitoring sensors
- It uses a RESTful API with basic authentication to provide security

Poseidon Connectors in SkySpark

- Creating a Poseidon connector:
  - dis: Any Readable Name
  - uri: `https://psl-api.poseidonsys.com/api/`
  - company: Use the company name or abbreviation exactly as it was given to Poseidon
  - poseidonUsername: This will be provided by your Poseidon Team
  - poseidonPassword: This will be provided by your Poseidon Team
  - Of course, these records will have the poseidonConn and conn tags
- These can be created in the connector app

Poseidon Connector



Poseidon Functions pt 1

- poseidonCmd – send Poseidon a custom get request
  - read(poseidonConn).poseidonCmd
- poseidonCompanies – return a grid of all available companies for the Poseidon user
  - read(poseidonConn).poseidonCompanies
- poseidonLocations – return a grid of all available locations for the company set in the connector
  - read(poseidonConn).poseidonPoints
- poseidonAssets – return a grid of all available assets for a company with an optional location
  - read(poseidonConn).poseidonAssets("Richmond")
- poseidonPoints – return a grid of all available his points for a given company, location, and asset (use the * separator)
  - read(poseidonConn).poseidonPoints("Ziva*Richmond*Generator")
- poseidonSyncHis – sync history from a trend log
  - readAll(poseidonHis).poseidonSyncHis(null)
- poseidonPing – send ping to connector to check connection
  - read(poseidonConn).poseidonPing

Poseidon Functions part 2

- poseidonHisRead – read history directly from Poseidon
  - read(poseidonConn).poseidonHisRead(read(poseidonHis), today())
  - read(poseidonConn).poseidonHisRead("Ziva*Richmond*Generator*", today())
- poseidonHisReadEvents – return a list of events for a given asset by passing a point under that asset or that point's address
  - read(poseidonConn).poseidonHisReadEvents(read(poseidonHis), today())
  - read(poseidonConn).poseidonHisReadEvents("Ziva*Richmond*Generator*oil_temperature", today())
- poseidonAlarmDevices – return a list of all company-registered alarm devices
  - read(poseidonConn).poseidonAlarmDevices
- poseidonAlarmTriggers – return a list of all company alarm triggers
  - read(poseidonConn).poseidonAlarmTriggers

Poseidon View Functions

- poseidonDupFinder – find points with the same addresses (default true will add dupPoint tag to point)
  - poseidonDupFinder(true)
- poseidonMakePointsToBuckets – assigns tuning records to Poseidon curVal points (use poseidonTuningBuckets view)
  - Number of points per connTuningRec
  - poseidonMakePointsToBuckets(100)
- poseidonMakeTuningBuckets – makes Poseidon Tuning records (use poseidonTuningBuckets view)
  - Number of points per connTuningRec, starting pollFreq, max pollFreq
  - poseidonMakeTuningBuckets(100, 5min, 9min)
- poseidonViewStatus – see Poseidon points
  - poseidonViewStatus()
- poseidonViewSummary – see status of Poseidon points
  - poseidonViewSummary()
- poseidonViewTuning – display Poseidon tuning records
  - poseidonViewTuning()

Getting Data for Poseidon Points

Manually or Automatically

- Getting history data from Trend Logs:
  - Each point must have these tags:
    - poseidonConnRef (ref) which is a ref that points to the Poseidon Connector
    - poseidonHis (str) which points to the Poseidon Trend Log object
    - his (marker)
    - Run the poseidonSyncHis(timeRange) function
      - generally as a task
      - readAll(poseidonHis).poseidonSyncHis(null)

Basic Troubleshooting

- Note that it is normal for a connection to close when it has no points in a watch for a given period of time
- For larger systems, you want to set the poll frequency to a higher value to avoid network congestion
- Make sure time zones, kinds, and units are correct

Getting Points via Code

```
() => do
  grid: {}.toGrid
  pConn: read(poseidonConn)
  locs: pConn.poseidonLocations
  locs.each l => do
    pConn.poseidonAssets(l->dis).each a => do
      rows: pConn.poseidonPoints(a->learn)
      grid = grid.addRows(rows)
    end
  end
  grid.each r => commit(diff(null, r, {add}))
end
```

Run as a task in large systems to avoid a timeout error

Getting Points with GUI

- We recommend using the Builder App's drag and drop capability to select which points are of interest as there are often many superfluous points