
WebCTRL®

ALC

- Automated Logic's WebCTRL® is a building automation system that offers an intuitive user interface and powerful control features. Your building can be accessed from anywhere in the world using your favorite browser, eliminating the need for special software on the workstation or tablet.
- It uses a SOAP-like API with a username and password. SOAP is essentially XML in and XML out.

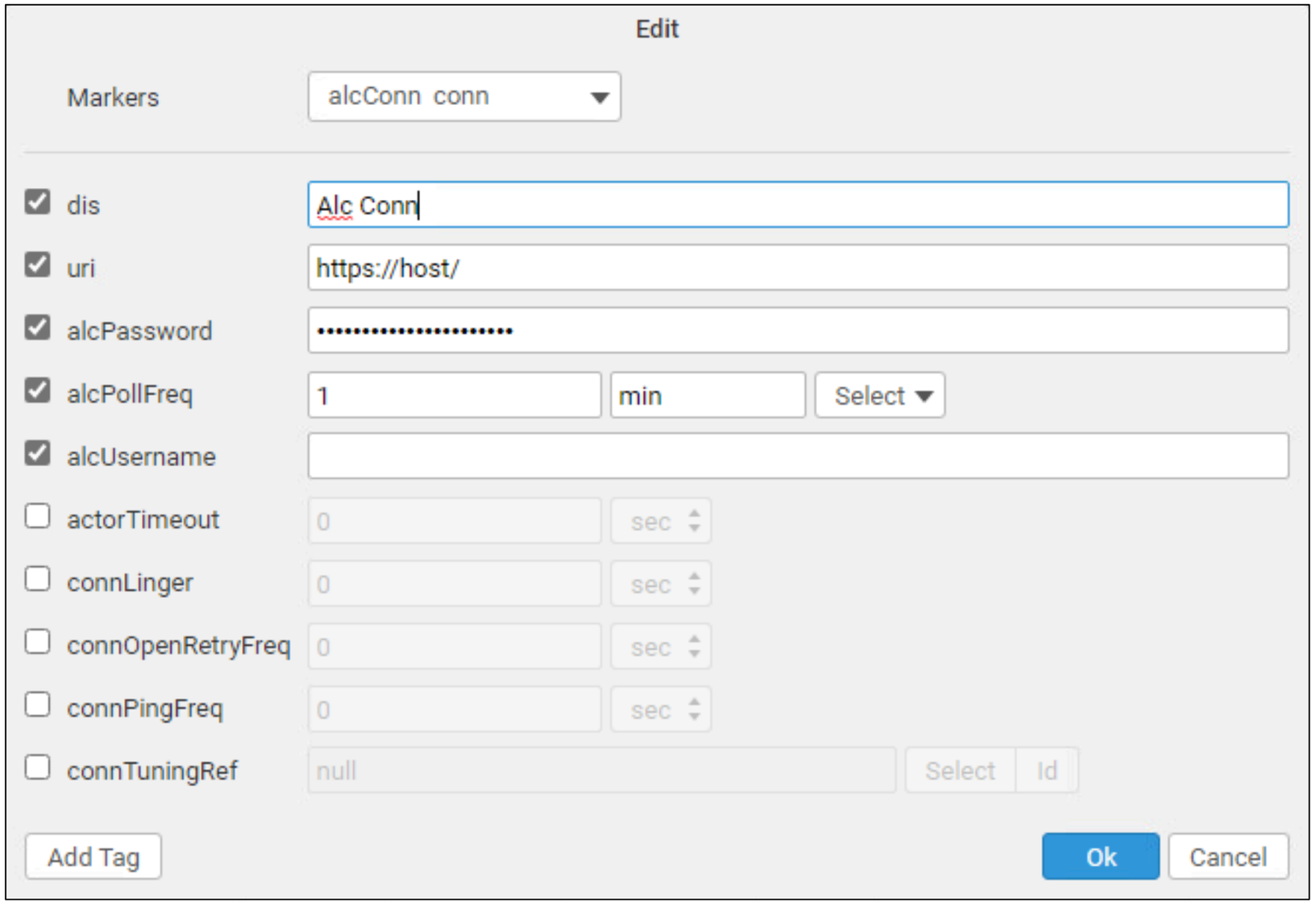
Licensing

- Each ALC connector extension is licensed to a single SkySpark Node Id
- Each ALC connector extension is licensed to a set maximum number of points
- If either of these states goes into fault, the issue must be corrected and SkySpark must be restarted

ALC Connectors in SkySpark

- Creating an ALC connector:
  - dis: Any Readable Name
  - uri: `https://host` ie `https://alc.net/`
  - alcUsername: This will be provided by your ALC Team
  - alcPassword: This will be provided by your ALC Team
  - The alcPollFreq tag determines the minimum amount of time before the cache can be refreshed and how often the curVal values will be updated (which is from where both polling with intervals and COV come). The default value of 1min is good for detecting COV changes. The curVals do not matter for trend log synching.
  - Of course, these records will have the alcConn and conn tags
- These can be created in the connector app

ALC Connector

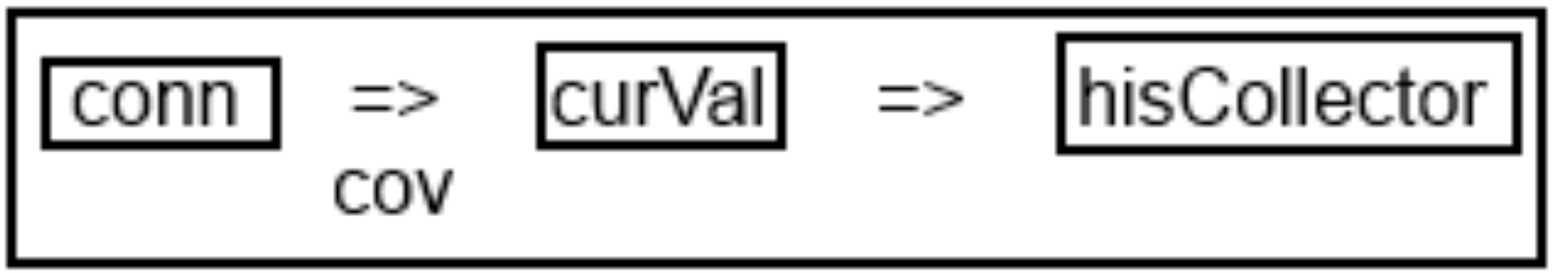


ALC Connector Set Up

- You must set up the API on the ALC side, then SkySpark will be able to talk to it. As long as the uri, username, and password are correct, it should work.

Polling of curVal

- alcPollFreq (duration)
  - How often you want SkySpark to poll the connector for COV
  - The default is 1min
  - It is a tuning parameter
  - Note that this determines how often the curVal in SkySpark changes. You would still need to set how often the data is collected with the hisCollect tags.



ALC Functions Part I

- alcSyncHis – sync history from a trend log
  - readAll(alcHis).alcSyncHis(null)
- alcSyncCur – force one or more points to sync
  - readAll(alcCur).alcSyncCur
- alcPing – send ping to connector to check connection
  - read(alcConn).alcPing
- alcHisRead – attempt to do a raw read from ALC without synching (will tell you if point is trended)
  - read(alcConn).alcHisRead(read(alcHis)) read(alcConn).alcHisRead("trees/geographic/pointAddr", today)
- alcVals – return a list of addresses and curVals (good for testing if an address exists)
  - read(alcConn).alcVals(readAll(alcCur)) read(alcConn).alcVals(null, ["pointAddr1", "pointAddr2"])
- alcLearnTool – this gets display names, but not actual point addresses – it's an ALC point list report
  - read(alcConn).alcLearnTool("trees/geographic")

ALC Functions Part 2

- alcWsdl – A WSDL is the instructions for a SOAP command. Defaults to Report. Can also use Eval, Trend, or System.
  - read(alcConn).alcWsdl("Eval")
- alcReport – Run any of the following reports. It takes a report and a location.

  ~schedule-instance ~effective-schedule ~point-list-report ~locked-value ~network-io ~test-and-balance ~equipment-checkout ~audit-log ~alarms ~alarm-source ~network-status ~module-version ~security-assignment ~alarm-messages ~alarm-actions ~trend-usage ~parameter-mismatch

  - readAll(alcConn).alcReport("~point-list-report", "trees/geographic")
- alcChildren – used for the builder learn, but get the children nodes for any node
  - readAll(alcConn).alcChildren("trees/geographic")

Getting Data for ALC Points

Manually or Automatically

- Collecting current values (from curVal):
  - Each point must have these tags:
    - alcConnRef (ref) which is a ref that points to the ALC Connector
    - alcCur (str) which points to the ALC point object
    - his (marker)
    - cur (marker)
    - hisCollectCov (marker or number) for cov collection OR
    - hisCollectInterval (duration) for polling
- Getting history data from Trend Logs:
  - Each point must have these tags:
    - alcConnRef (ref) which is a ref that points to the ALC Connector
    - alcHis (str) which points to the ALC Trend Log object
    - his (marker)
    - Run the alcSyncHis(timeRange) function
      - generally as a task
      - readAll(alcHis).alcSyncHis(null)

Note: we recommend also using a hisCollectInterval of 24hr when using a hisCollectCov to catch faulty sensors.

Writing to ALC Points from SkySpark

Manually or with Automated Tasks

- Writing/Sending current values to a ALC point:
  - Each point must have these tags:
    - alcConnRef (ref) which is a ref that points to the Hubitat Connector
    - alcWrite (string) which points to the Hubitat point object
    - writable (marker)
  - Note: The pointWrite() command is the only writing command that is truly supported. The pointAuto() function is only used to clear the writeVal on SkySpark's end. It does not do anything on the device's end.

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 the same in SkySpark and ALC
- Note: If a point kind is Bool, 0 and 1 will automatically be converted to false and true for curVals and trends in SkySpark. This conversion is not done on SkySpark writing back to ALC.

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