Preliminary

Func

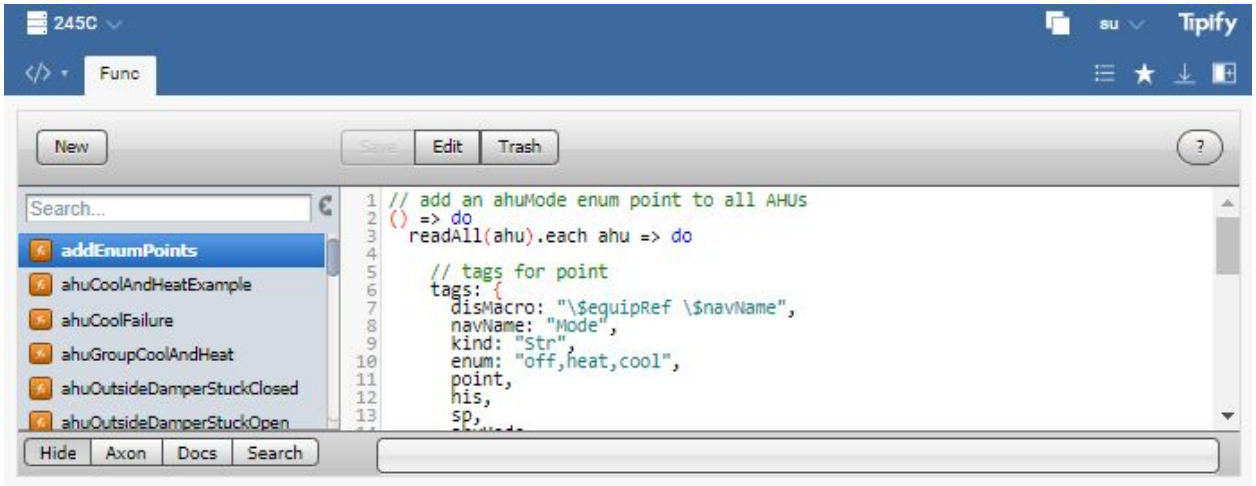
Contents

Introduction

This is an introduction to the Func app in Tipify™. It allows you to create Axon Functions that can be reused.

Overview

After you login to InferStack™, you will be presented with the Home screen. Select Func to start the app.

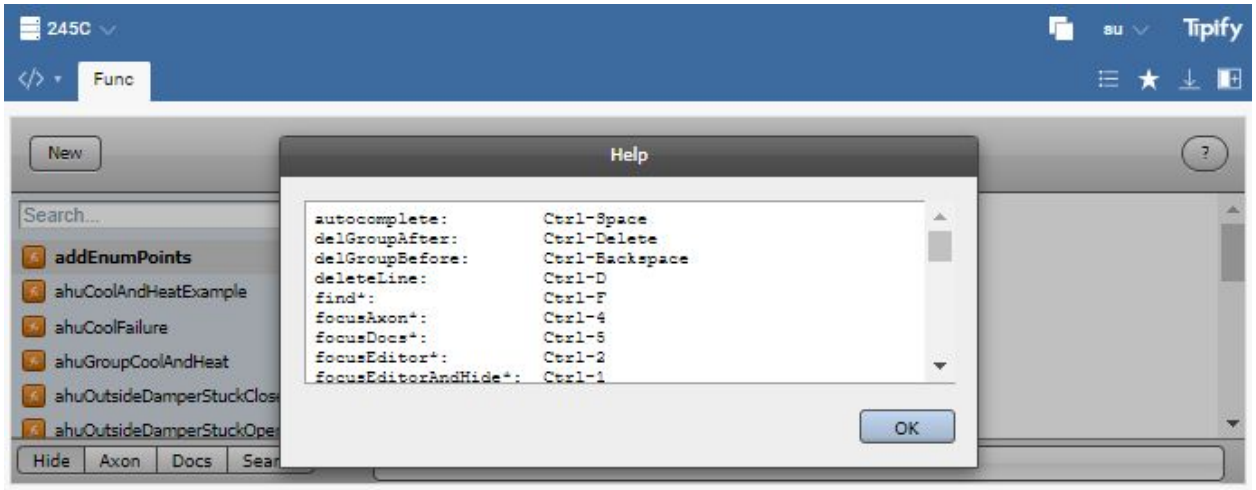


Func app includes a list of Functions on the left and a Main Menu across the top. There is a utility bar at the bottom that provides access to Axon, Docs, or Search.

Keyboard Navigation

There is a help button to the right ()which shows all the keyboard shortcuts.

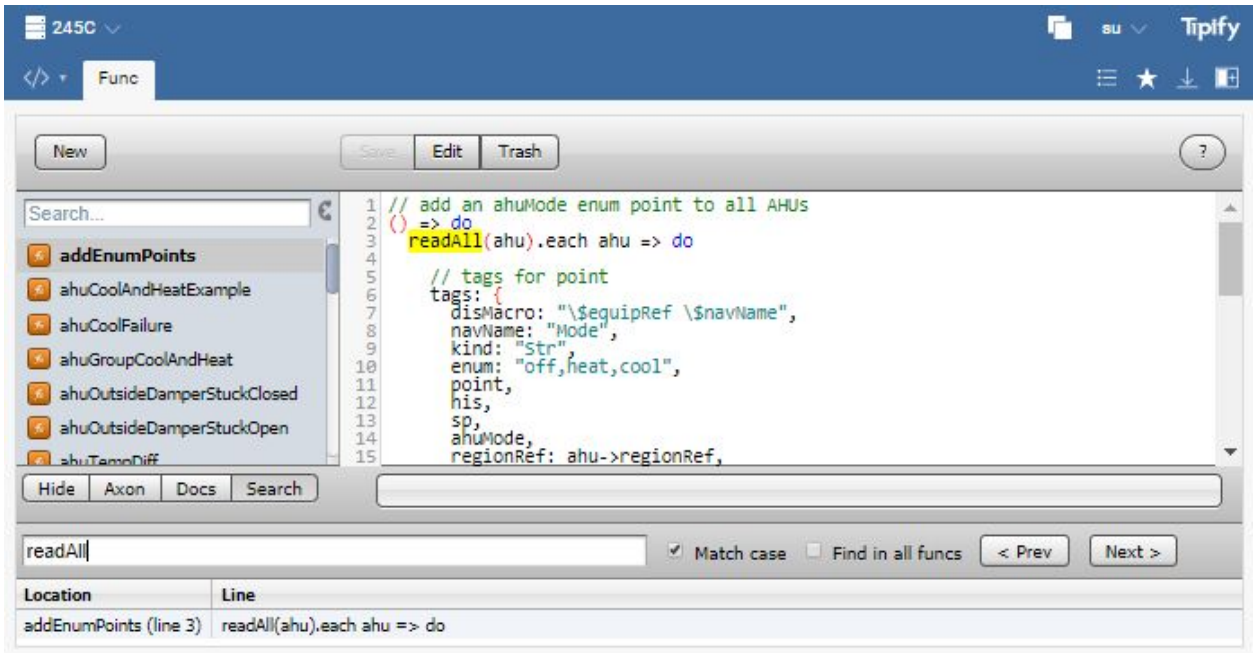
Preliminary



Keyboard Shortcuts

```
autocomplete:          Ctrl-Space
delGroupAfter:         Ctrl-Delete
delGroupBefore:        Ctrl-Backspace
deleteLine:            Ctrl-D
find*:                 Ctrl-F
focusAxon*:            Ctrl-4
focusDocs*:            Ctrl-5
focusEditor*:          Ctrl-2
focusEditorAndHide*:   Ctrl-1
focusFuncs*:           Ctrl-3
focusSearch*:          Ctrl-6
goDocEnd:              Ctrl-End
goDocStart:            Ctrl-Home
goGroupLeft:           Ctrl-Left
goGroupRight:          Ctrl-Right
goLineDown:            Ctrl-Down
goLineEnd:             Alt-Right
goLineStart:           Alt-Left
goLineUp:              Ctrl-Up
help*:                 F1
indentLess:            Shift-Tab
indentMore:            Tab
nextMark*:             Ctrl-]
prevMark*:             Ctrl-[
redo:                  Shift-Ctrl-Z, Ctrl-Y
redoSelection:         Alt-U
save*:                 Ctrl-S
selectAll:             Ctrl-A
showDocs*:             Ctrl-I
undo:                  Ctrl-Z
undoSelection:         Ctrl-U
unitPicker*:           Shift-Ctrl-U

* Denotes application level commands versus editor specific commands
```

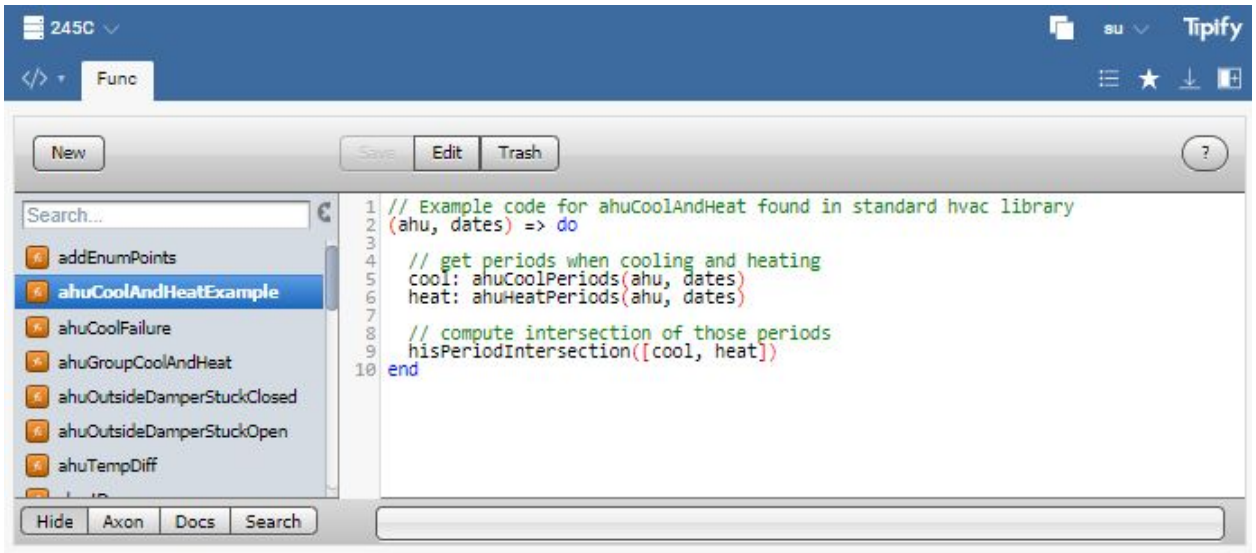
We've put a lot of work into allowing full navigation strictly by the keyboard. For example use Ctrl+F to search.



When you are done you can use Ctrl+1 to focus back to the editor and hide the search bar.

Select a Function

You can select a function by pressing the name in the left column.

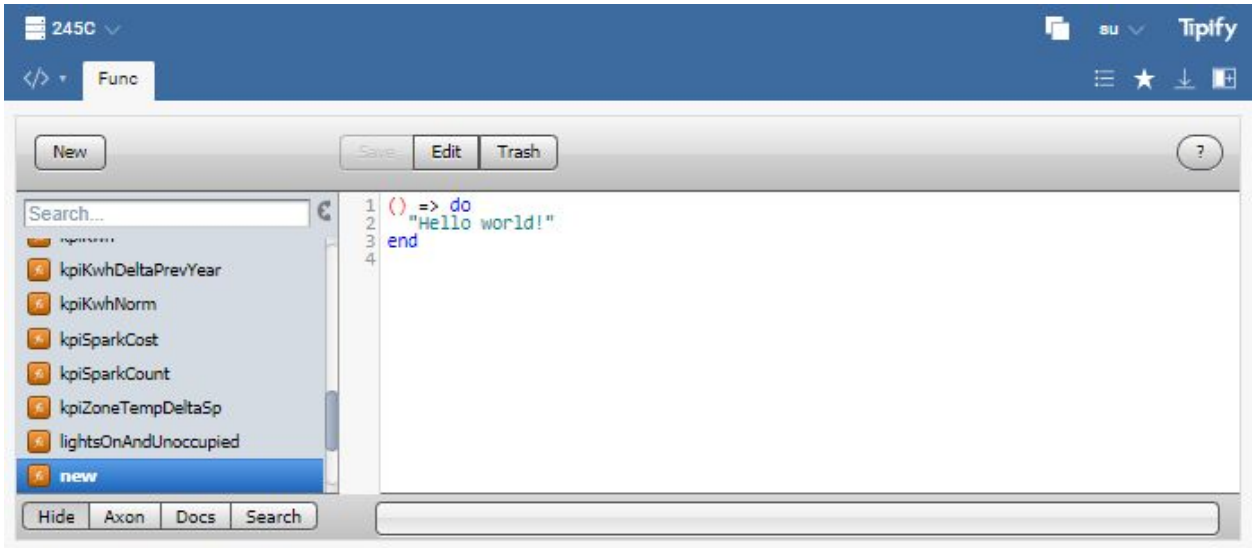


Main Menu

The Main Menu includes New, Save, Edit and Trash.

New

Press New to create a new function. Type a name for your function and press OK. It is created with a "Hello World!" application.



Function Name

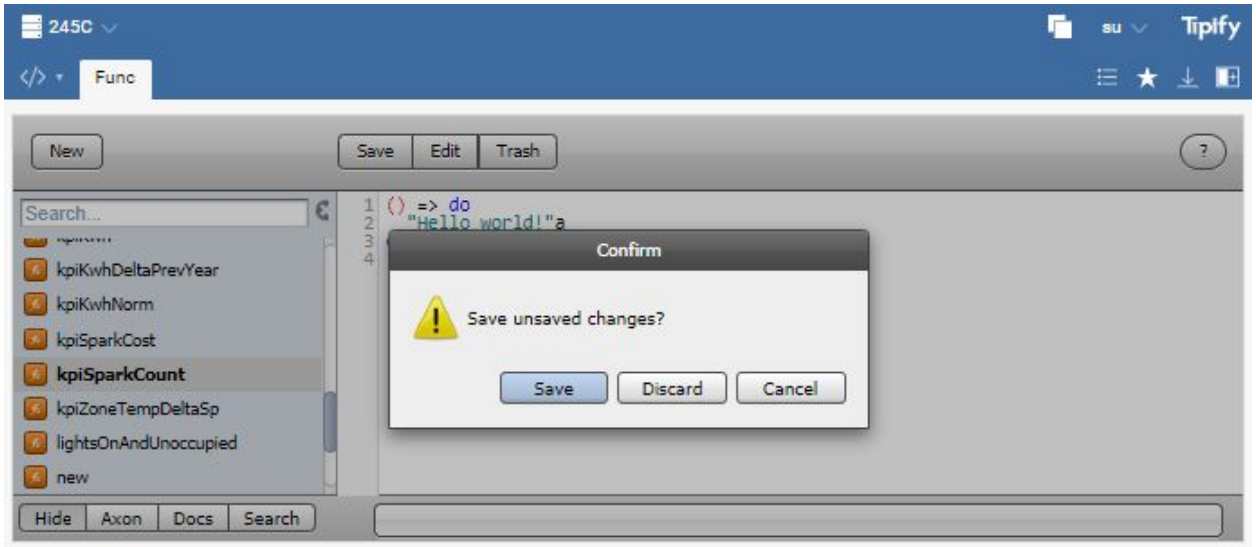
Functions must follow standard rules for naming. Programmatic names use `camelCaseNaming` as follows:

- first char must be ASCII lower case letter: `a` - `z`
- rest of chars must be ASCII letter or digit: `a` - `z`, `A` - `Z`, `0` - `9`, or `_`

See Naming Reference at https://skyfoundry.com/doc/docSkySpark/Folio#naming.

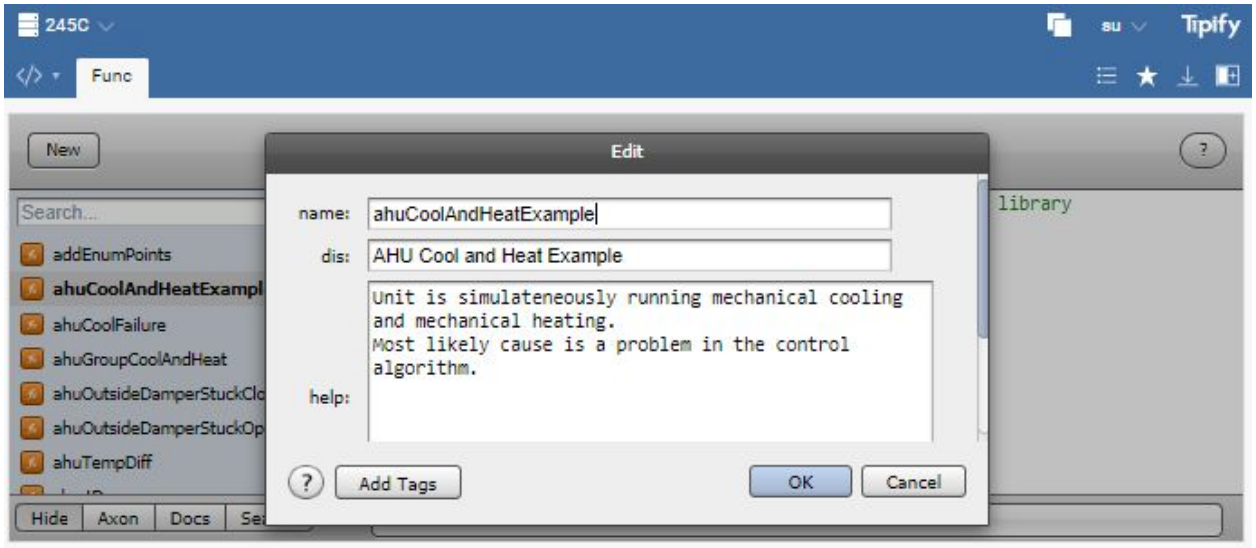
Save

Once you make any changes to the content of your function, the Save button will be enabled in the menu bar to save your changes. Press another function to discard your changes.



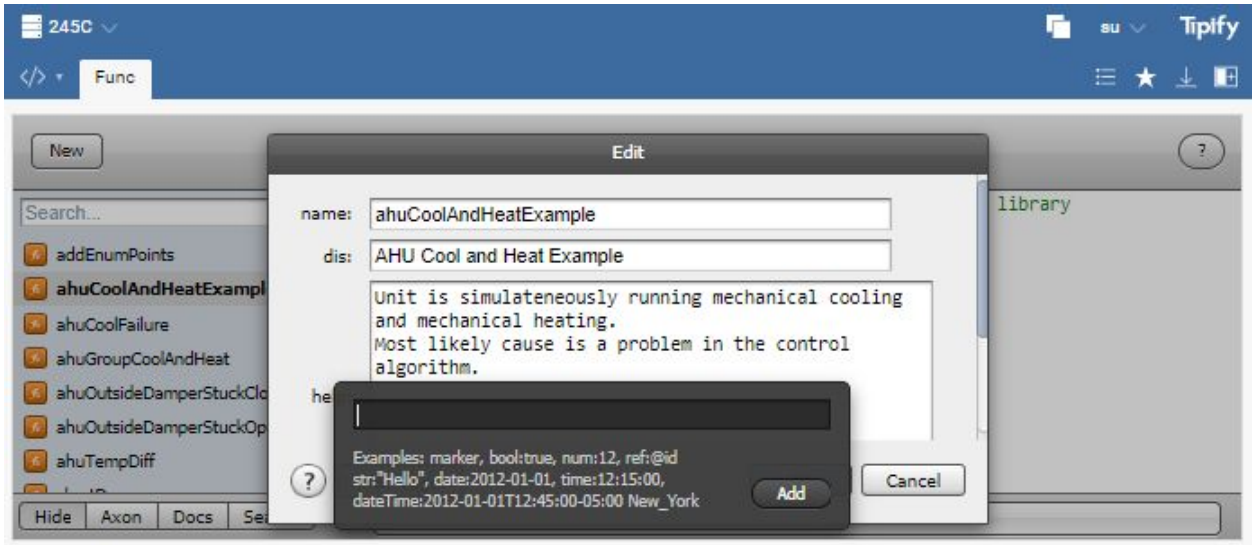
Edit

Select a function and press Edit to make changes to the name or tags.



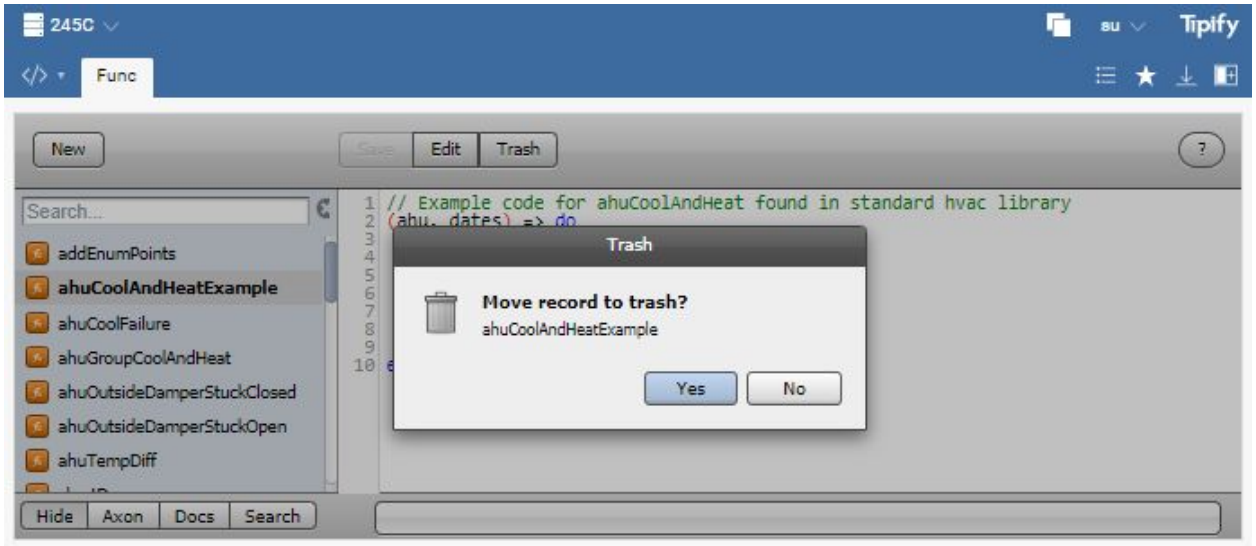
Marker and Property Tags

You can add any marker or property tags by pressing Add Tags.



Trash

Trash allows you to delete a Function. Select one in the left pane and press Trash.

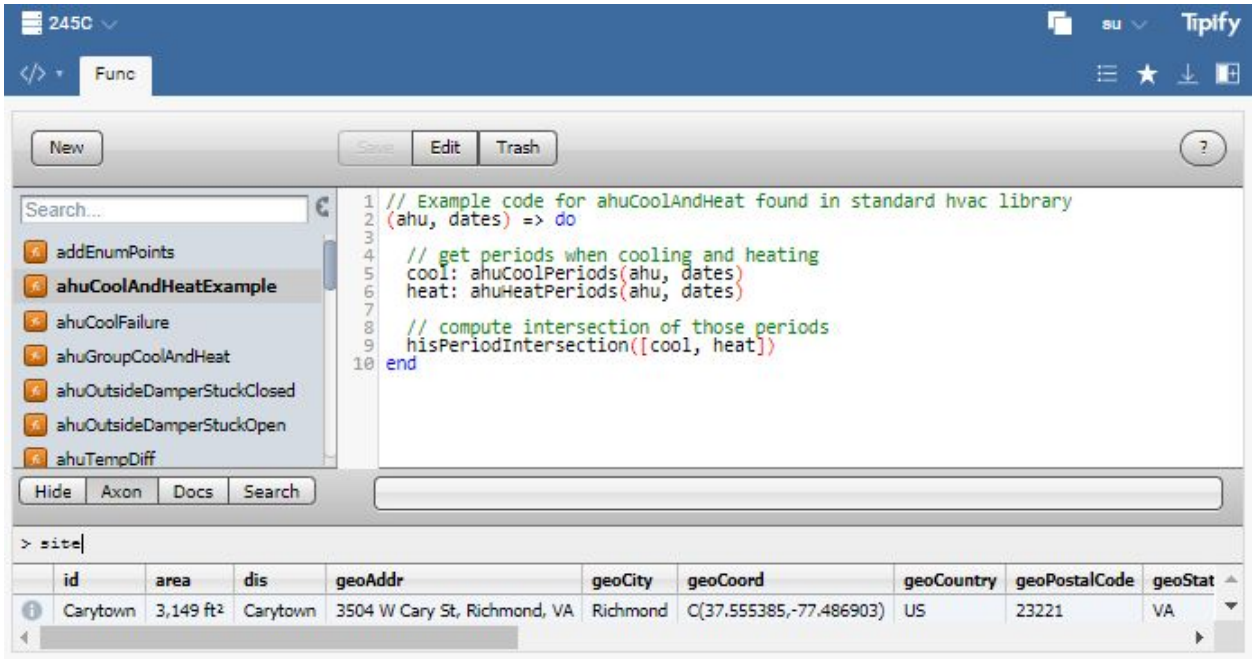


Utility Bar

The bottom of the FuncApp provides a utility bar that you can open to enable new features: Axon, Docs, or Search. Note that the save error message has been moved to the bottom too.

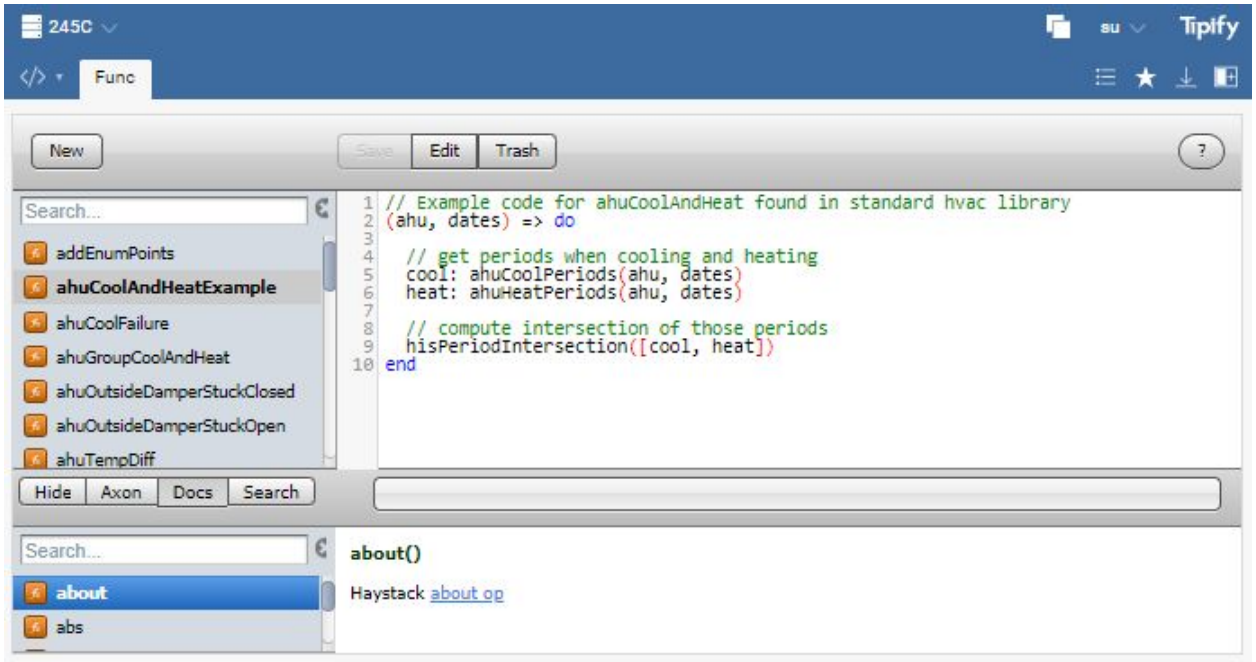
Axon

Axon allows you to make Axon requests while in the Func app.



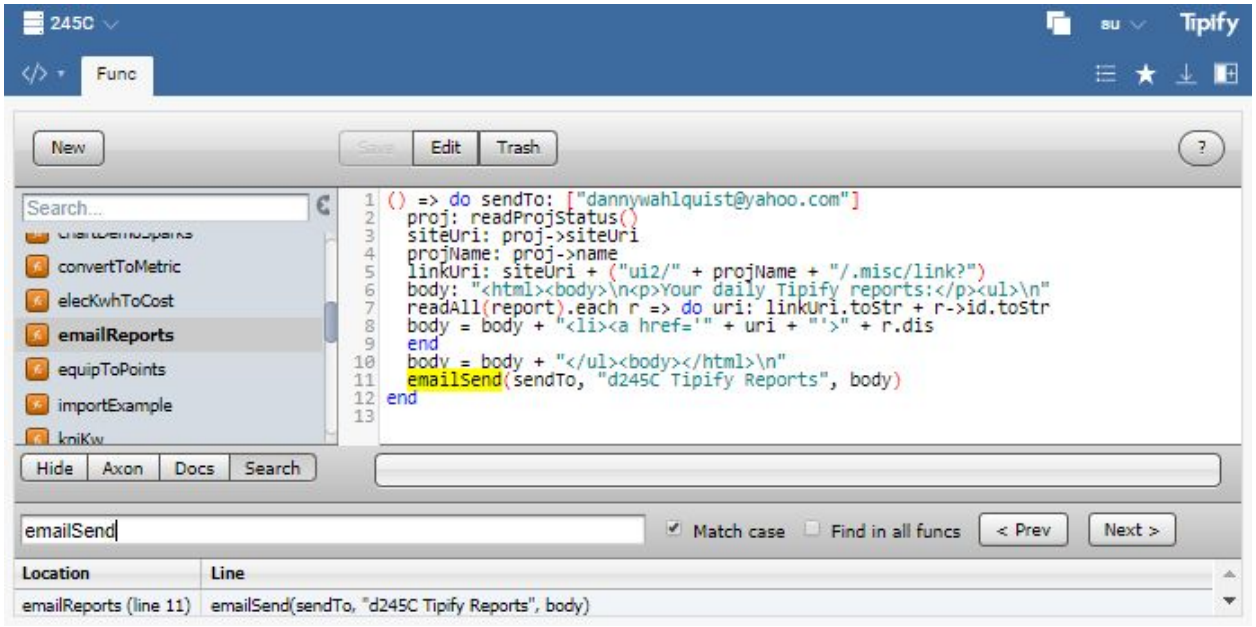
Docs

Docs provides access to reference information within the Func app.



Search

Search allows you to find text in the current or all funcs.



Migration Functions

Three migration functions are provided. These are used after migration from R2 to complete the appropriate connectors.

migrateAdioConns

This migrates adio connectors to pulse connectors.

migrateModbusConns

This add the serial baud rate to the modbus connector.

migratePulseConns

This migrates R2 pulse connectors.

Examples

This is a collection of example functions that are provided as examples of what can be accomplished.

emailReports

This function emails links to the Reports.

```
() => do sendTo: ["dwahlquist@intellastar.com"]
  proj: readProjStatus()
  siteUri: proj->siteUri
```

```
  projName: proj->name
  linkUri: siteUri + ("ui2/" + projName + "/.misc/link?")
  body: "<html><body>\n<p>Your daily Tipify reports:</p><ul>\n"
  readAll(report).each r => do uri: linkUri.toStr + r->id.toStr
      body = body + "<li><a href='" + uri + "'>" + r.dis
  end
  body = body + "</ul><body></html>\n"
  emailSend(sendTo, "245GM Tipify Reports", body)
end
```

oscillateLights

This function demonstrates control by oscillating selected commandable points.

```
() => do
  pts: readAll(point and writable and kind == "Bool" and not rocker)
  pts.each pt => do
    if (pt.has("curVal")) do
      val: not pt->curVal
      log("info", "test", "update " + pt->navName + " -> " + val)
      pointWrite(pt, val, 5, "oscillateLights func")
    else
      pointWrite(pt, true, 5, "init oscillateLights func")
    end
  end
```

totalizeKwh

Compute the total kWh consumed by a given meter over an interval of time.

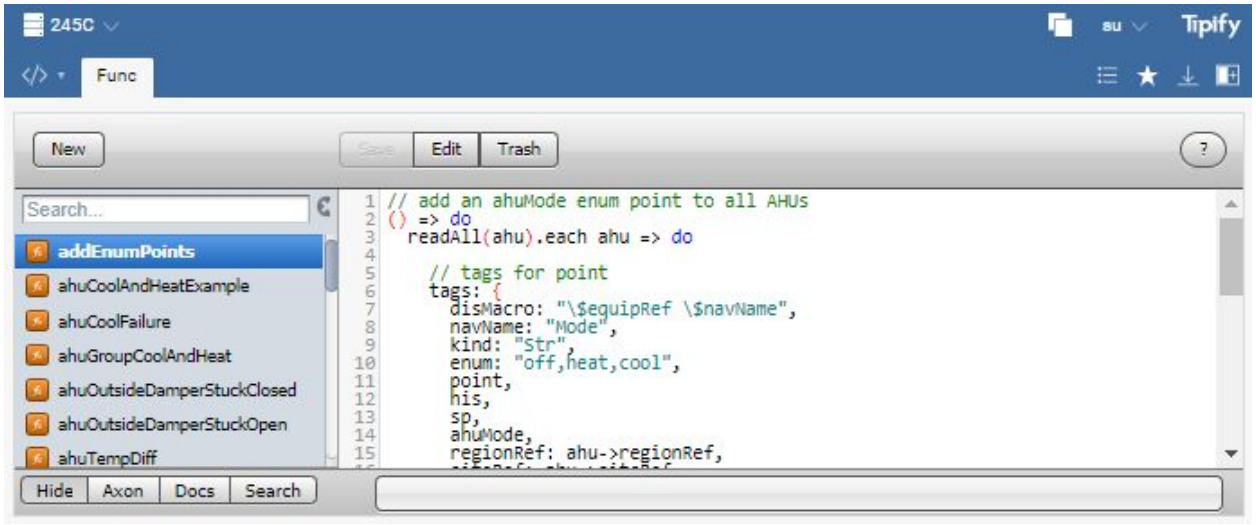
```
/*
  Compute the total kWh consumed by a given meter over an interval of time:
    - meter: Dict for elec meter record
    - tsStart: DateTime of inclusive starting time
    - tsEnd: DateTime of exclusive ending time
    - returns: Number in kWh
*/
(meter, tsStart, tsEnd) => do

  // get kwh point for meter
  point: read(energy and equipRef == meter->id)

  // read history for date and get hourly rollup of kWh
  his: hisRead(point, tsStart.date .. tsEnd.date).hisRollup(sum, 1hr)

  // filter out the times outside of our start/end time
  filteredHis: findAll(his) r => tsStart <= r->ts and r->ts < tsEnd

  // rollup the kWh for our filtered time period
  filteredHis.foldCol("v0", sum)
end
```

Calling and Declaring Functions

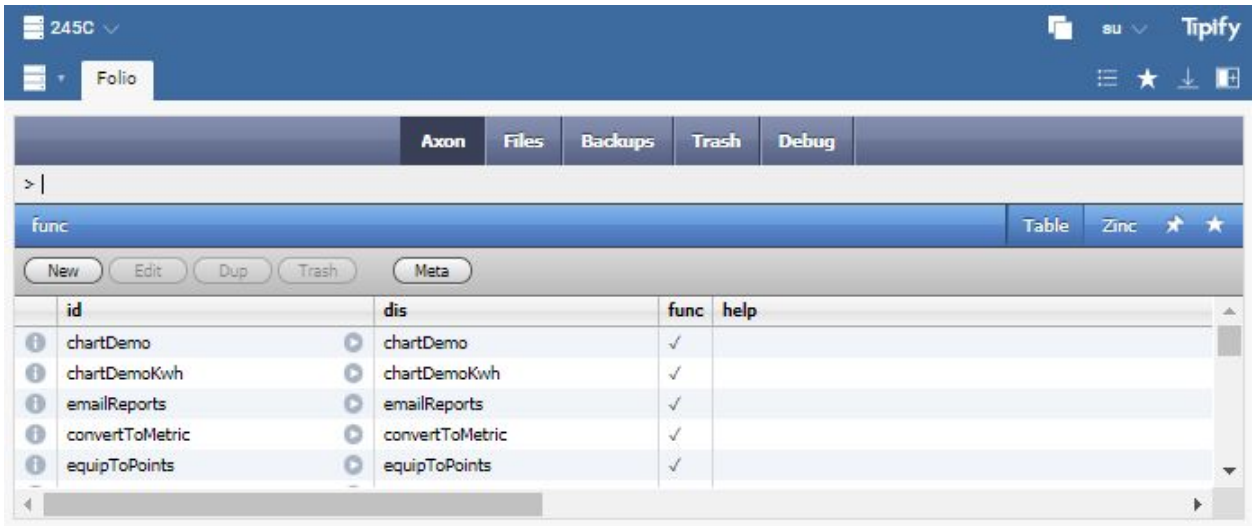
Top-level namespace

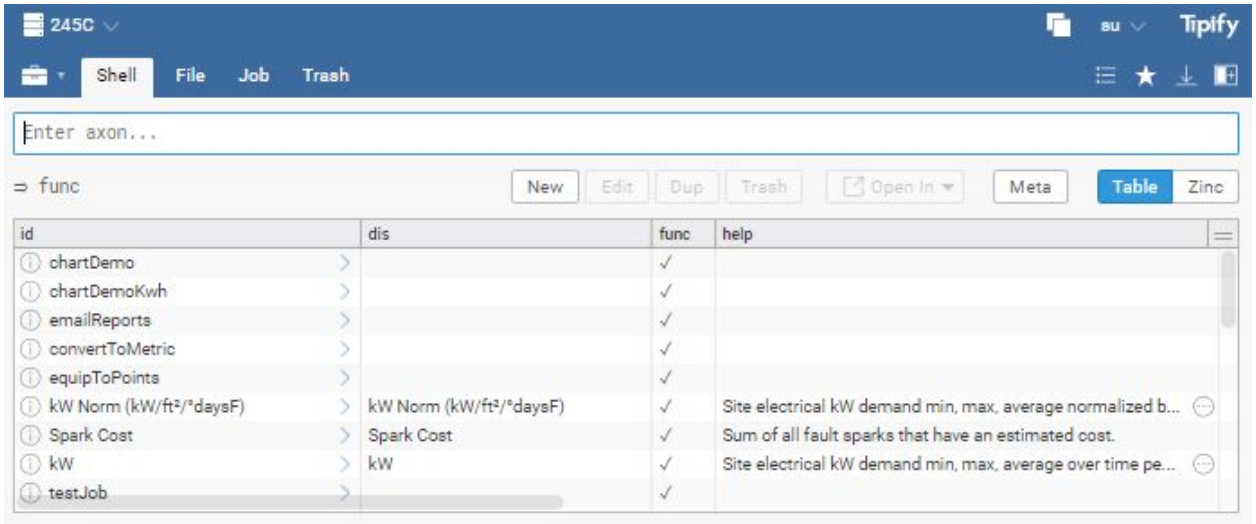
The top level namespace includes every record in database declared as a function. Extensions can also publish functions. Project based functions override and take priority over library functions.

You can see the project based functions in the Func app.



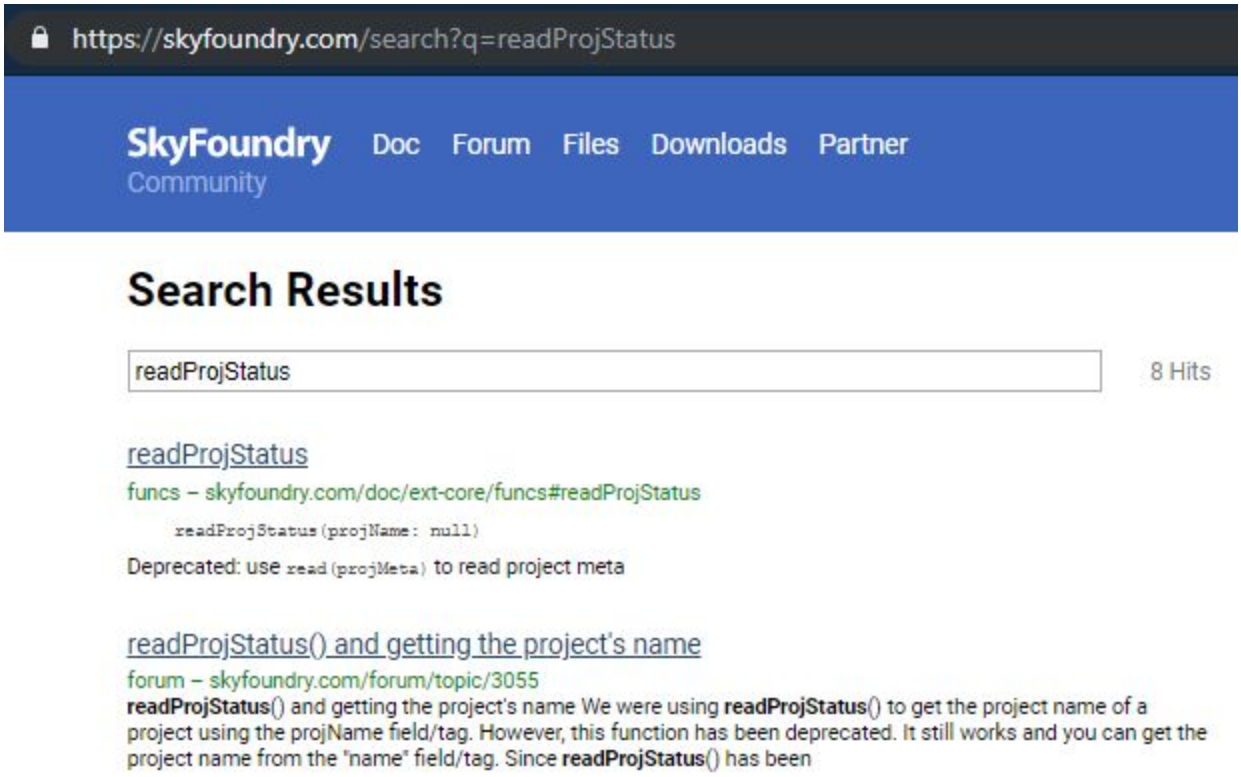
You can also see them by filtering on func in Folio or Tools - Shell.





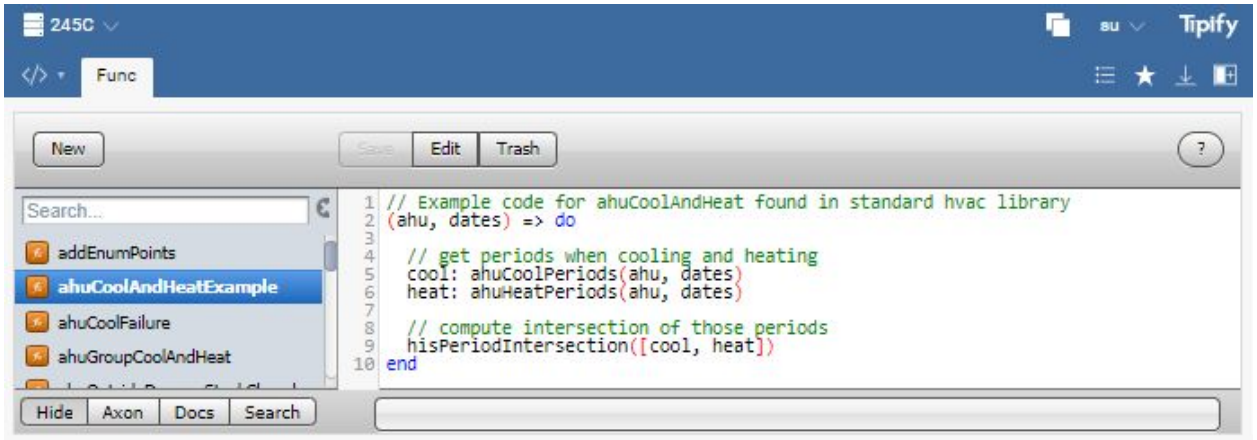
You can see the extension and library functions in Help. Select an extension to see its functions like in the core below.

If you are looking for a particular function you can use search the SkyFoundry portal to find it.



Declaring a Function

You can declare a function by using New in the Func app. An example is the ahuCoolAndHeatExample.

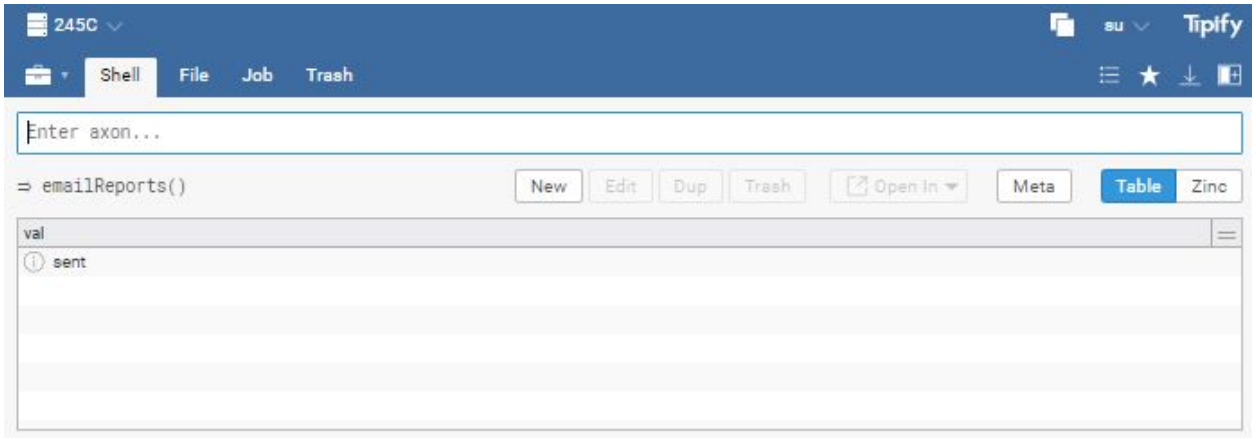


Function Arity

The arity is the number of parameters passed to the function. The ahuCoolAndHeatExample has two parameters.

Calling a function

To call a function add parentheses operator. I can enter emailReports() in Folio or Tools - Shell to run the function.



Dot Calls

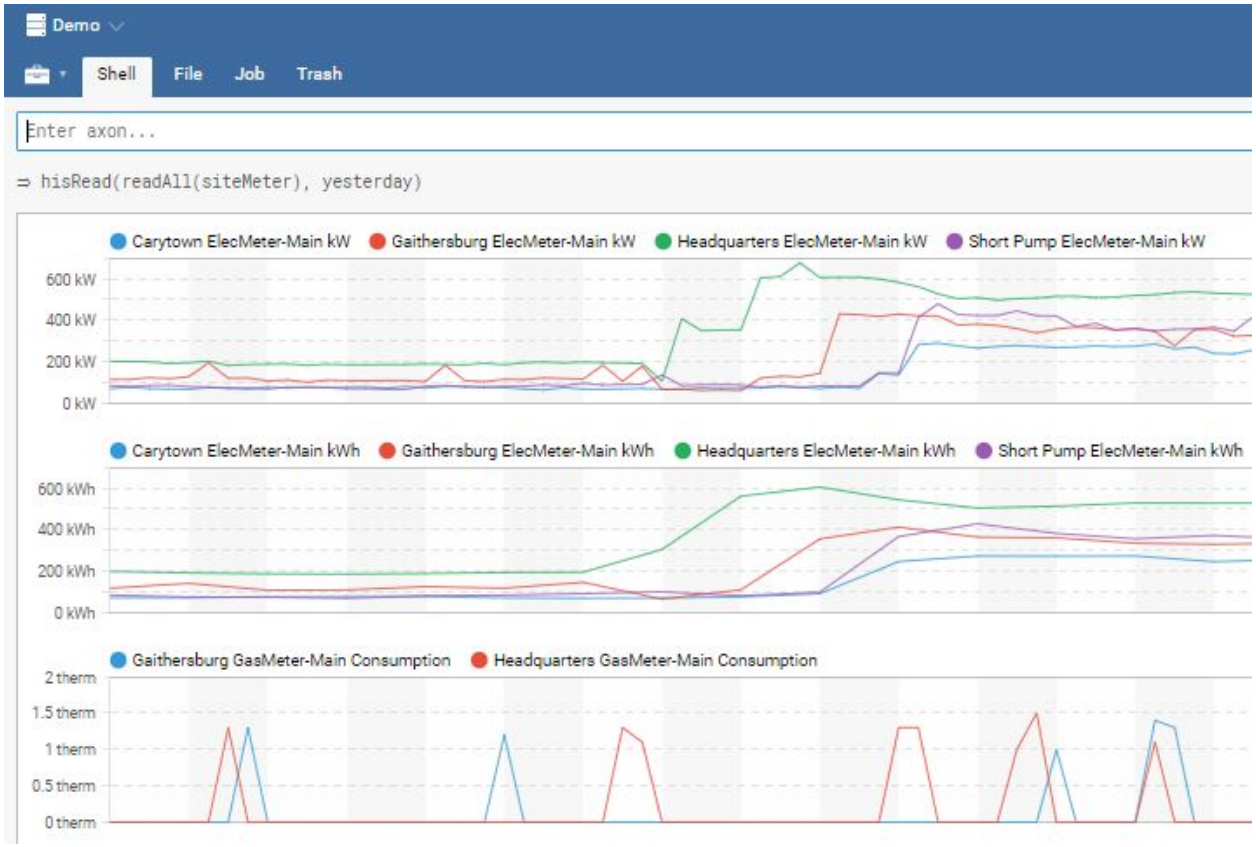
You can use results of one function in another by embedding them like date(now()) gets date from now. You could select the year using year(date(now())). This creates a nested list of parentheses. You could rewrite this using the dot operator as now().date().year(). If there are no parameters, you can leave off the parentheses like now().date.year

```
readAll(siteMeter).hisRead(yesterday)
```

Can be rewritten as

```
hisRead(readAll(siteMeter), yesterday)
```

Preliminary



Default Parameters

Parameters can be defaulted if not passed.

```
(x, y: "defy", z: "defz") => "x=" + x + " y=" + y + " z="+ z
```

Importing and Exporting Functions

Write a trio file to export and import your functions. To export all functions:

```
readAll(func).removeCols(["id","mod"]).ioWriteTrio(`io/funcs.trio`)
```

After moving the trio file to the io folder of the new project,

(Tools - File - create file, paste content & Save in T-Star)

you can import them:

```
ioReadTrio(`io/funcs.trio`).map row => commit(diff(null, row, {add}))
```

Reference

Reference information is available on your device or server in the Doc app or is searchable online at https://skyfoundry.com/doc/ .

The language documentation is in docLang Functions (https://skyfoundry.com/doc/docLang/Functions).

The Axon programming language is in docInferStack™ Axon (https://skyfoundry.com/doc/docSkySpark/Axon).

Standard Rules for Naming are defined at docInferStack™ Folio naming (https://skyfoundry.com/doc/docSkySpark/Folio#naming).