

# Func

## Contents

<b>Introduction</b>	<b>2</b>
<b>Overview</b>	<b>2</b>
<b>Keyboard Navigation</b>	<b>2</b>
Keyboard Shortcuts	3
Select a Function	4
<b>Main Menu</b>	<b>5</b>
New	5
Function Name	5
Save	5
Edit	6
Marker and Property Tags	6
Trash	7
Utility Bar	7
Axon	7
Docs	8
Search	8
Migration Functions	9
migrateAdioConns	9
migrateModbusConns	9
migratePulseConns	9
<b>Examples</b>	<b>9</b>
emailReports	9
oscillateLights	10
totalizeKwh	10
Calling and Declaring Functions	11
<b>Top-level namespace</b>	<b>11</b>
Declaring a Function	13
Function Arity	13
Calling a function	14
Dot Calls	14
Default Parameters	15
Importing and Exporting Functions	15

## Reference

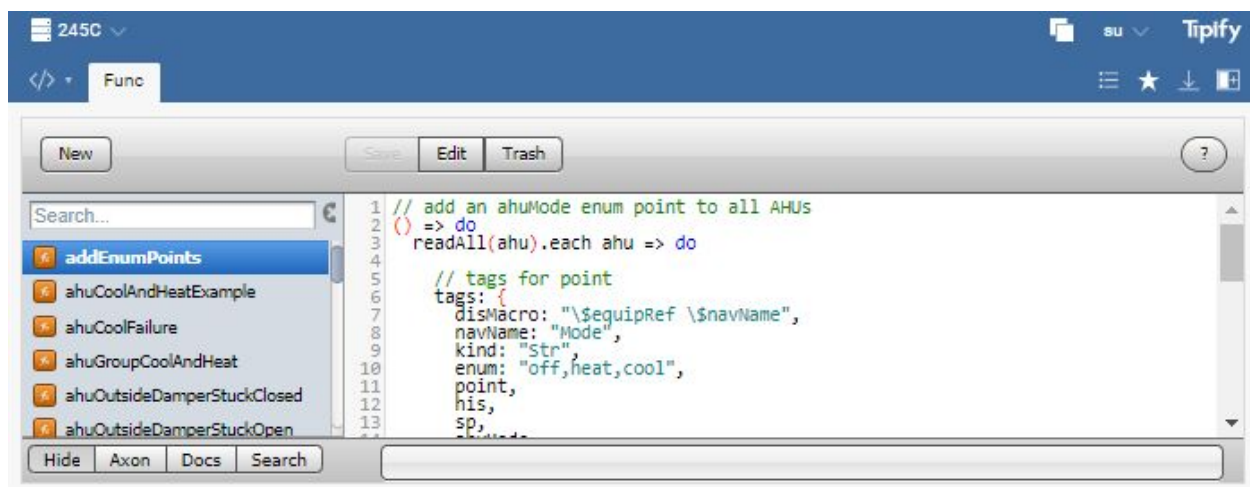
16

## Introduction

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

## Overview

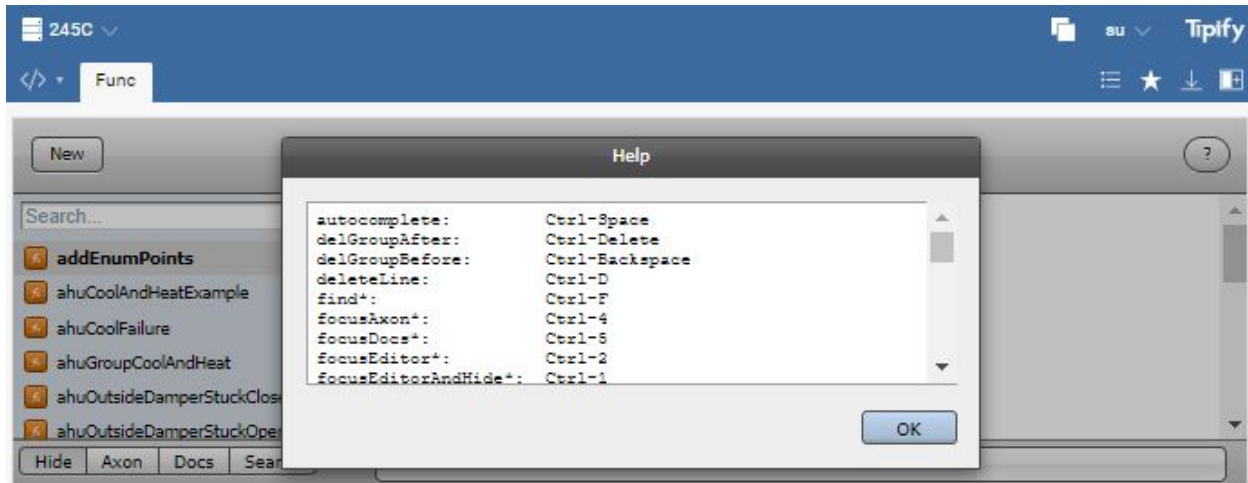
After you login to InferStack™, you will be presented with the Home screen. Select Func to start the app.



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.

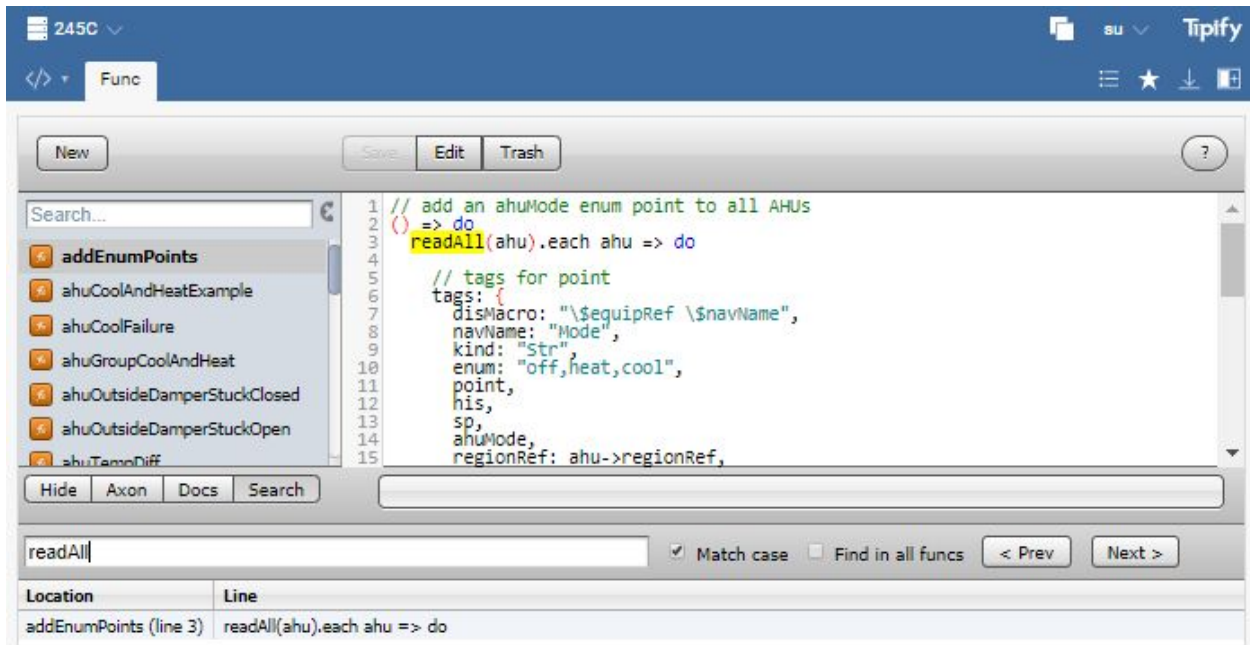


## Keyboard Shortcuts

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

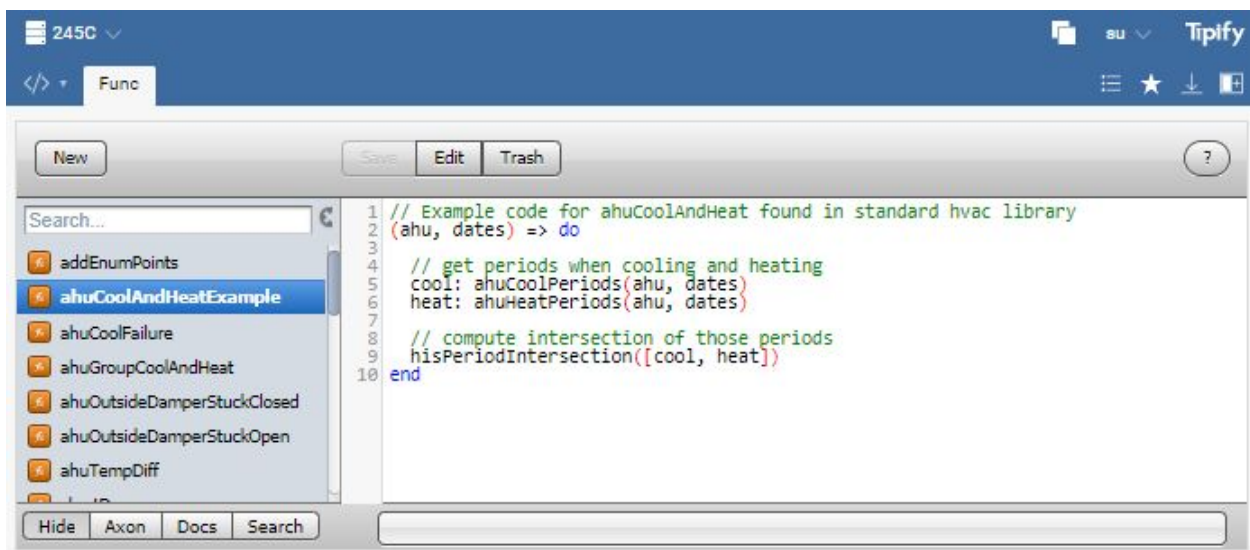
We've put a lot of work into allowing full navigation strictly by the keyboard. For example use Ctrl+F to search.



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.

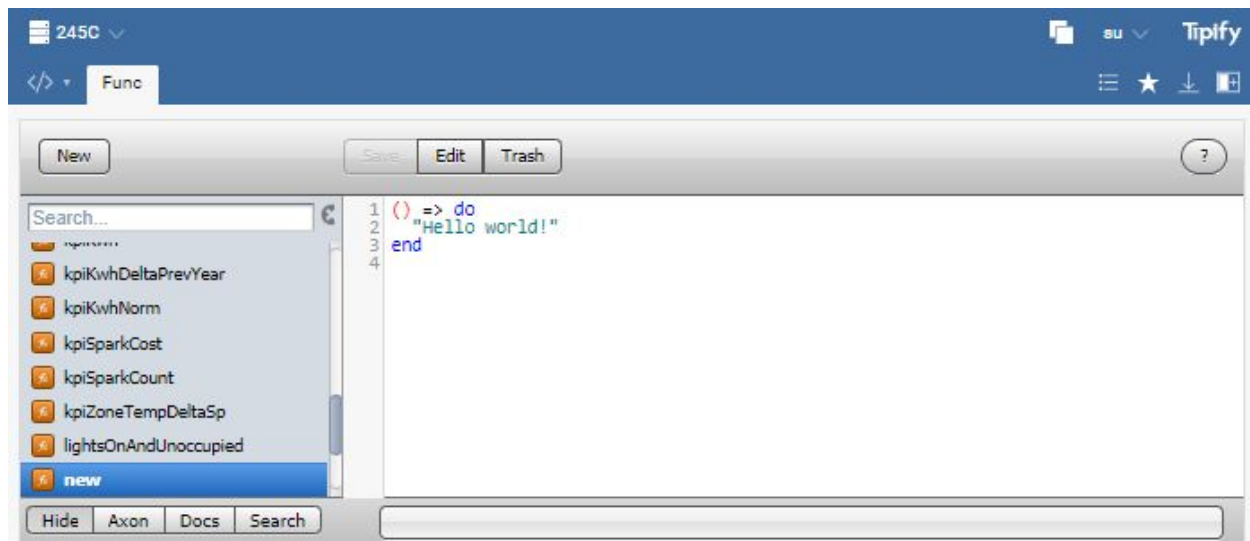


## Main Menu

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

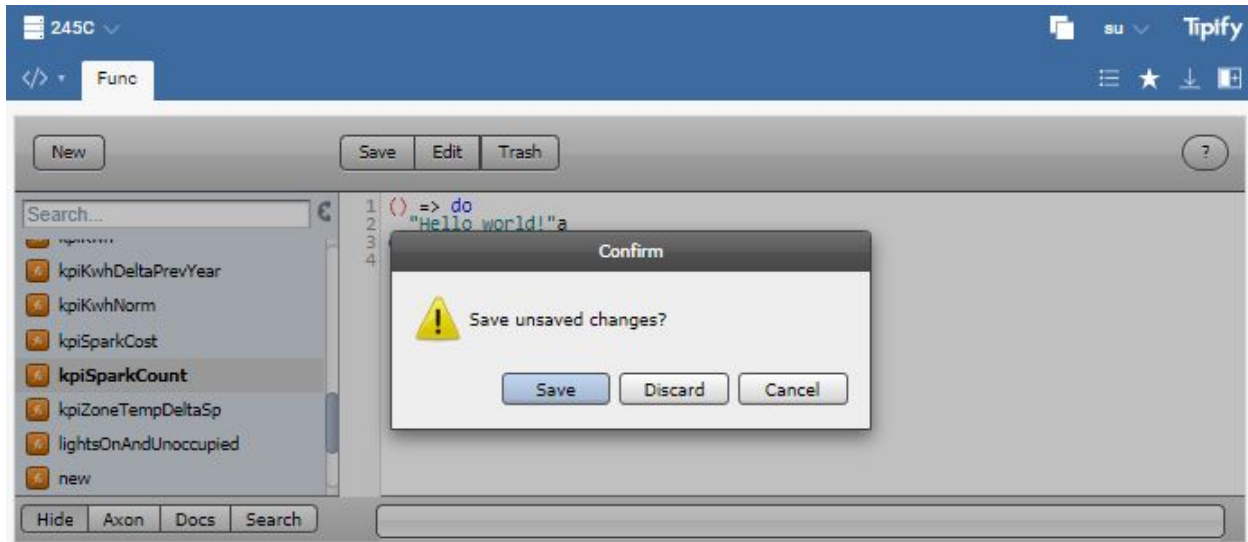
Functions must follow standard rules for naming. Programmatic names use [camelCaseNaming](#) as follows:

- 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>.

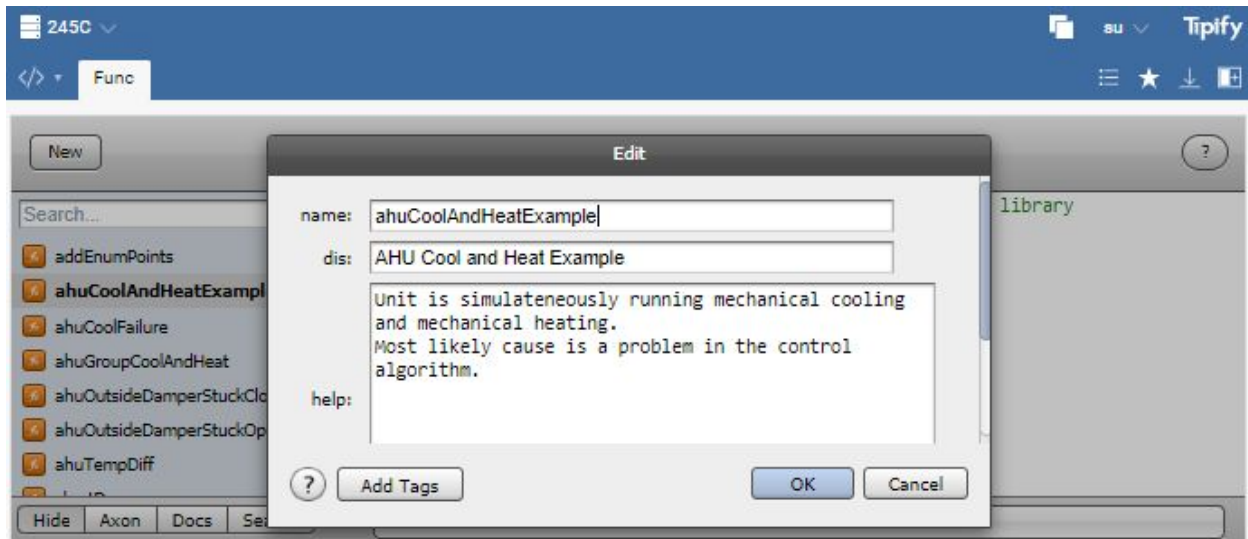
## Save

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.



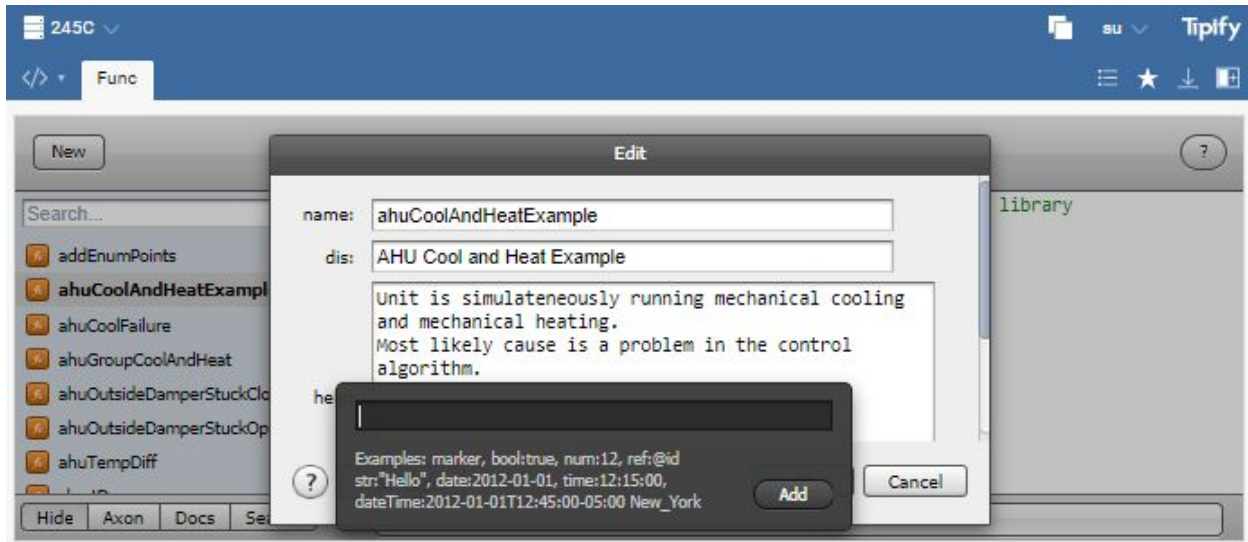
## Edit

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



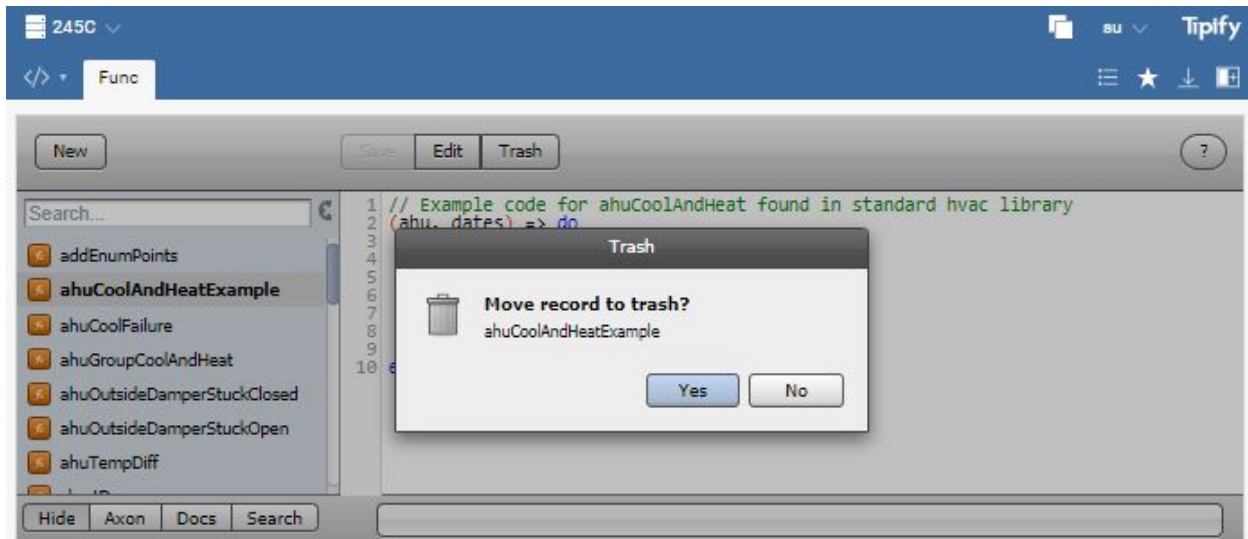
## Marker and Property Tags

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.

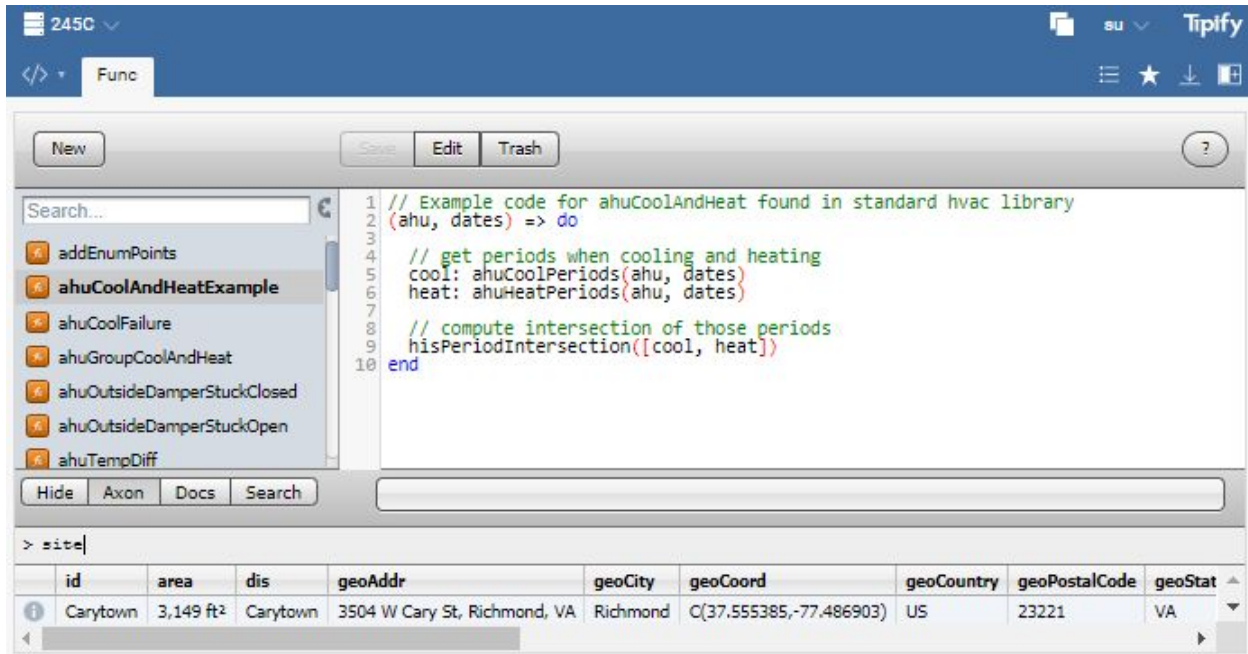


## Utility Bar

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.

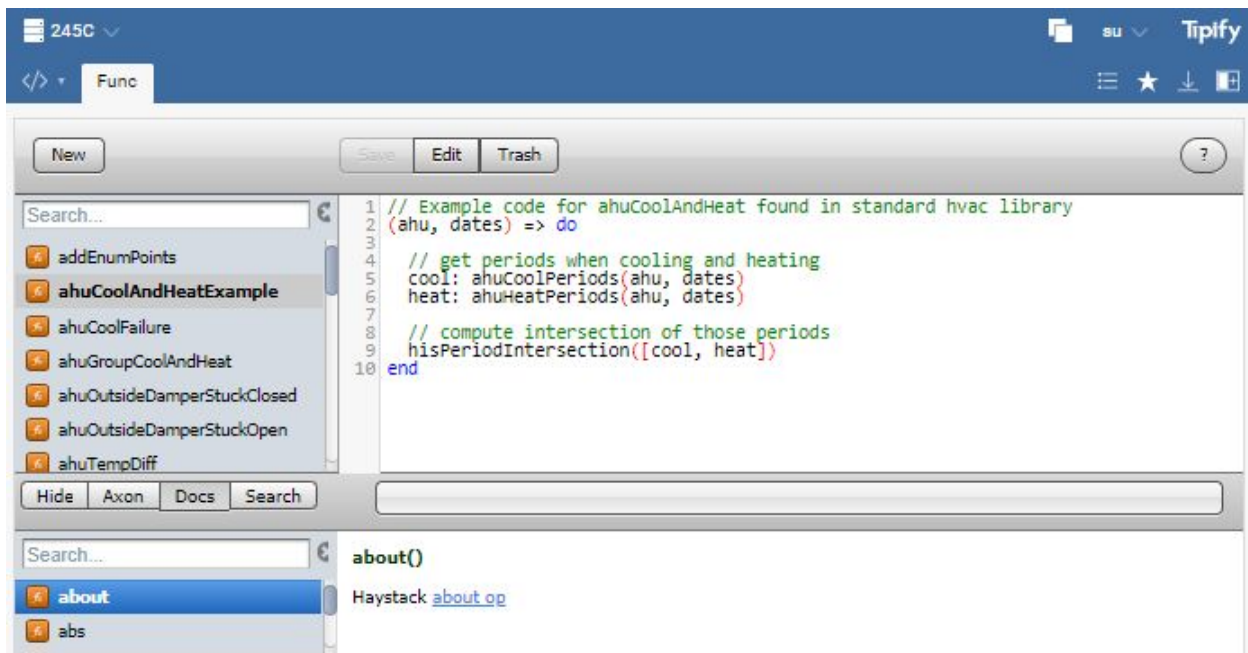
## Axon

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



## Docs

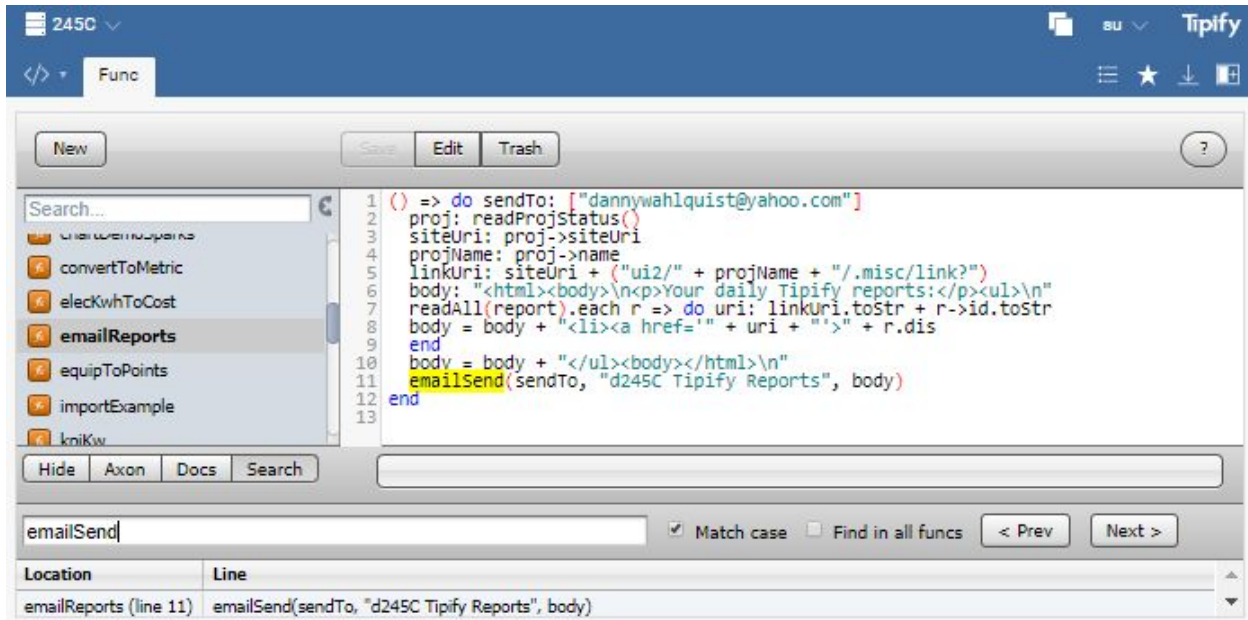
Docs provides access to reference information within the Func app.



## 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
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

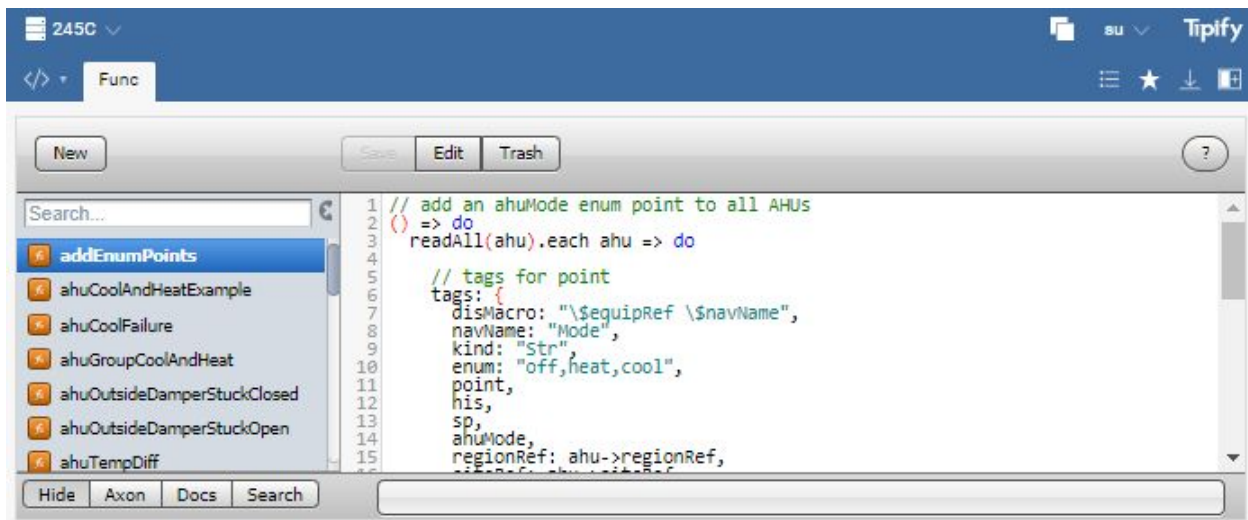
  // rollup the kWh for our filtered time period
  filteredHis.foldCol("v0", sum)
end
```

## Calling and Declaring Functions

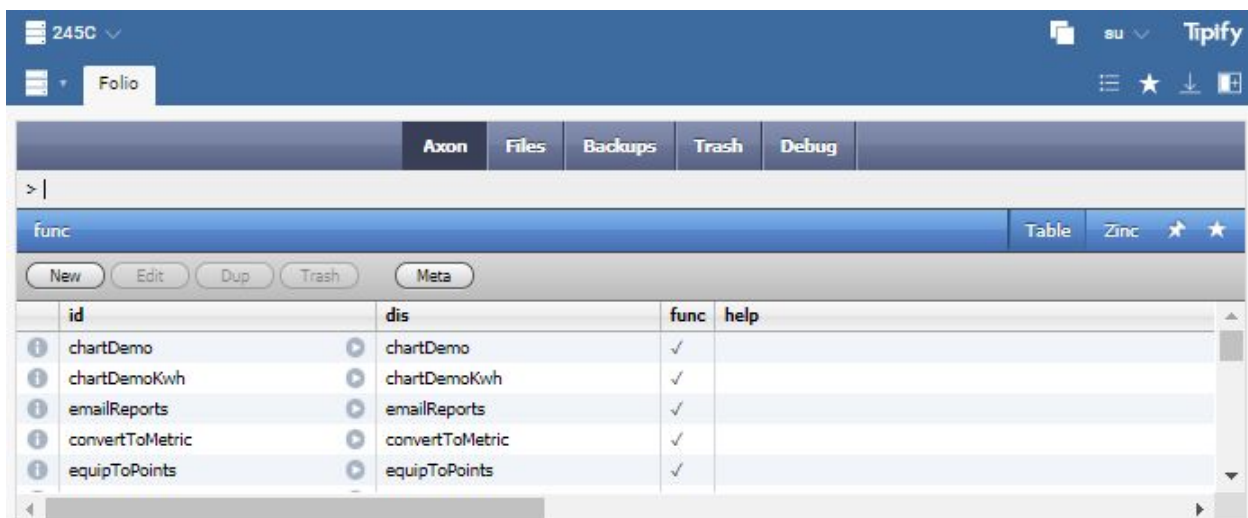
### Top-level namespace

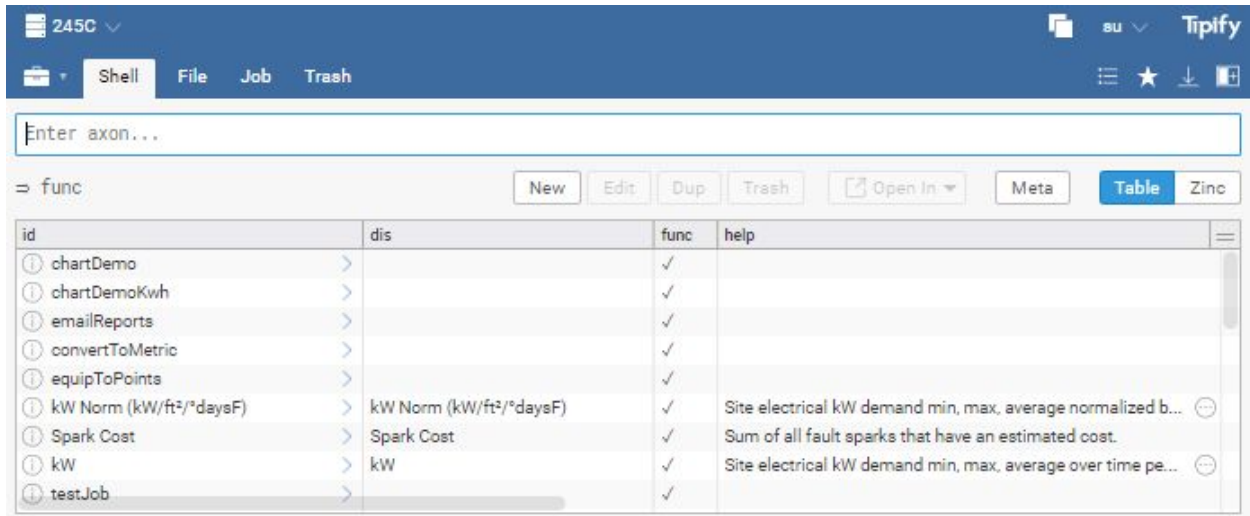
The top level namespace includes every record in database declared as a function. Extensions can also publish functions. Project based functions override and take priority over library functions.

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.

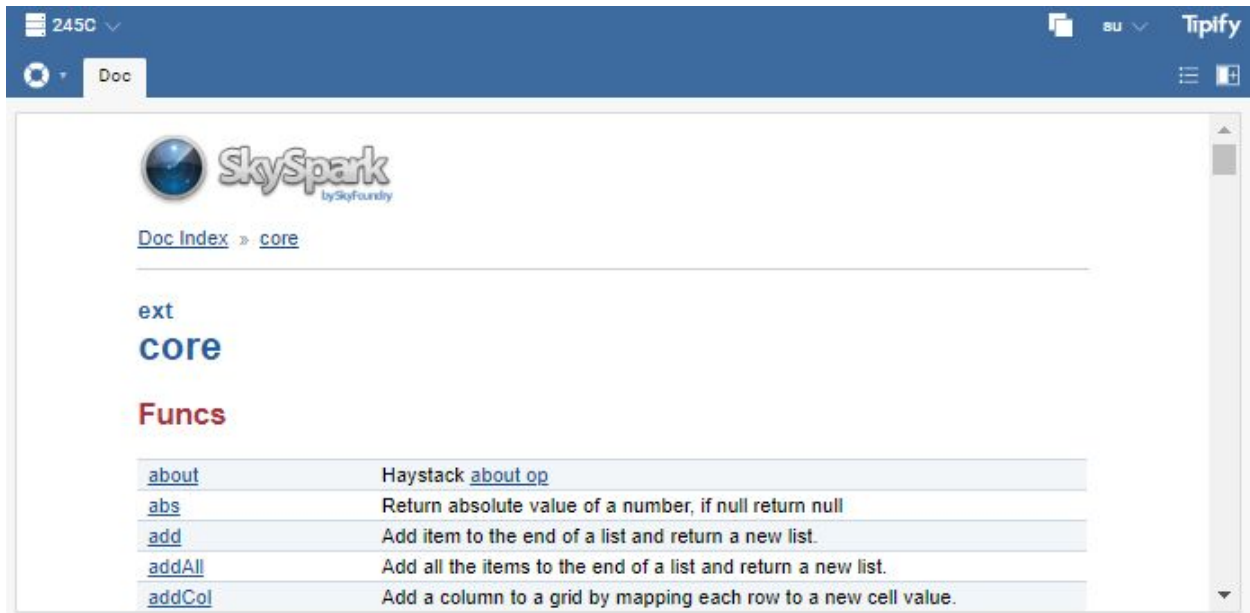




The screenshot shows the Tipify application interface. At the top, there is a navigation bar with a menu icon, the text '245C', and the user 'su'. Below this is a sub-header with 'Shell', 'File', 'Job', and 'Trash'. A search bar contains the text 'Enter axon...'. Below the search bar is a toolbar with buttons for 'New', 'Edit', 'Dup', 'Trash', 'Open In', 'Meta', 'Table', and 'Zinc'. The main content area displays a table with the following data:

id	dis	func	help
chartDemo		✓	
chartDemoKwh		✓	
emailReports		✓	
convertToMetric		✓	
equipToPoints		✓	
kW Norm (kW/ft²/*daysF)	kW Norm (kW/ft²/*daysF)	✓	Site electrical kW demand min, max, average normalized b...
Spark Cost	Spark Cost	✓	Sum of all fault sparks that have an estimated cost.
kW	kW	✓	Site electrical kW demand min, max, average over time pe...
testJob		✓	

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



The screenshot shows the SkySpark documentation page. The page header includes the SkySpark logo and the text 'by SkyFoundry'. Below the logo is a navigation breadcrumb: 'Doc Index > core'. The page content is organized into sections: 'ext', 'core', and 'Funcs'. Under the 'Funcs' section, there is a list of functions with their descriptions:

Funcs	Description
<a href="#">about</a>	Haystack <a href="#">about op</a>
<a href="#">abs</a>	Return absolute value of a number, if null return null
<a href="#">add</a>	Add item to the end of a list and return a new list.
<a href="#">addAll</a>	Add all the items to the end of a list and return a new list.
<a href="#">addCol</a>	Add a column to a grid by mapping each row to a new cell value.

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

https://skyfoundry.com/search?q=readProjStatus

SkyFoundry Community Doc Forum Files Downloads Partner

## Search Results

readProjStatus 8 Hits

[readProjStatus](#)  
 funcs – skyfoundry.com/doc/ext-core/funcs#readProjStatus  
 readProjStatus(projName: null)  
 Deprecated: use read(projMeta) to read project meta

[readProjStatus\(\) and getting the project's name](#)  
 forum – skyfoundry.com/forum/topic/3055  
 readProjStatus() and getting the project's name We were using readProjStatus() to get the project name of a project using the projName field/tag. However, this function has been deprecated. It still works and you can get the project name from the "name" field/tag. Since readProjStatus() has been

## Declaring a Function

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

```

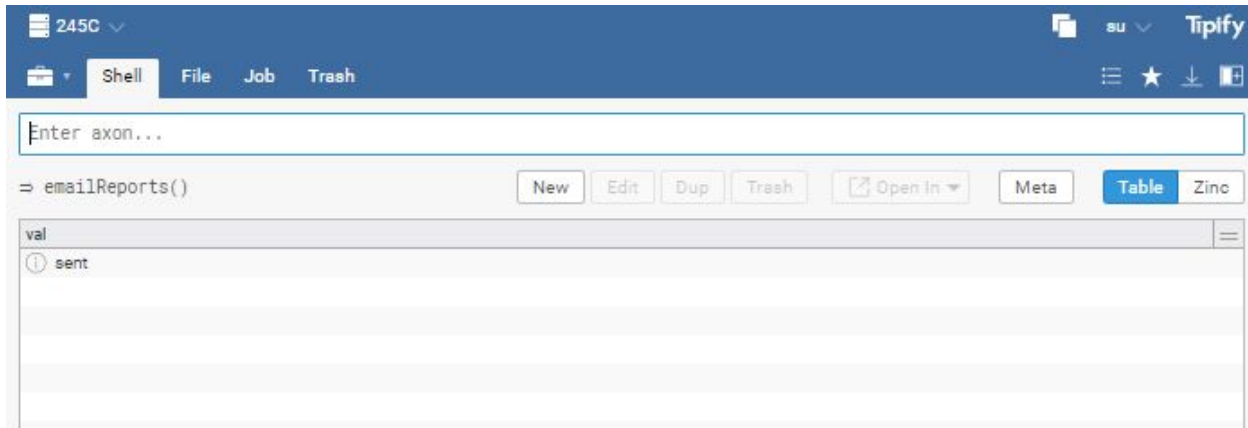
1 // Example code for ahuCoolAndHeat found in standard hvac library
2 (ahu, dates) => do
3
4 // get periods when cooling and heating
5 cool: ahuCoolPeriods(ahu, dates)
6 heat: ahuHeatPeriods(ahu, dates)
7
8 // compute intersection of those periods
9 hisPeriodIntersection([cool, heat])
10 end
  
```

## 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.



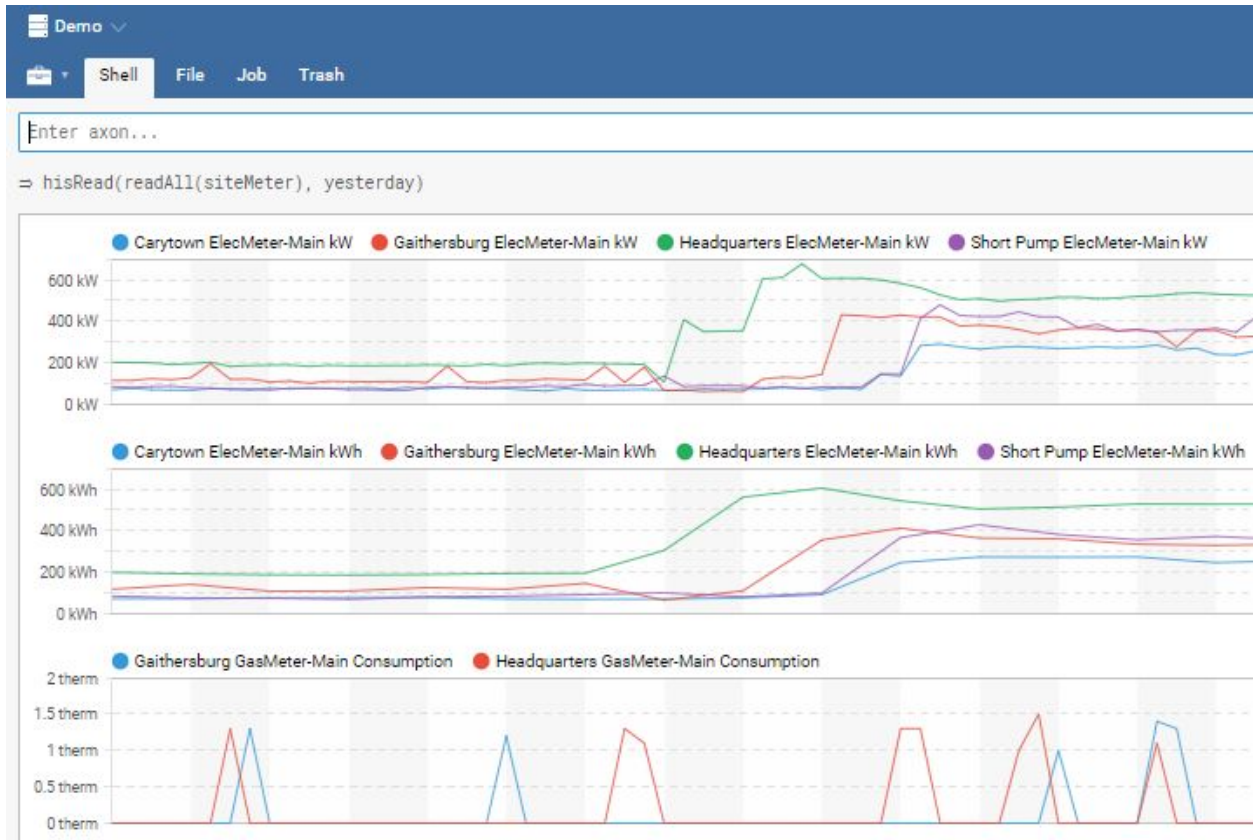
## Dot Calls

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)
```



## 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>).