APPIO Documentation

Release 0.1

talsen team GmbH

Dec 17, 2019

Contents:

CHAPTER 1

appio

1.1 appio build

Builds an opcuaapp from an APPIO project

1.1.1 Usage

```
appio build <options>
```

1.1.2 Options

Options	Description
-h | –help	prints the command help
-n | –name	APPIO project name

1.1.3 Parent command

appio

1.1.4 Detailed Description

APPIO build buildes a chosen APPIO project, including all generated information-models and the user specific application code. The binaries are created in the build folder inside of the APPIO project. For building, Meson and Ninja are used.

Note: After a successfull build you can publish the opcuaapp with *appio publish*. This copies the binaries into the publish folder for further use.

1.2 appio clean

Cleans the output of an APPIO project.

1.2.1 Usage

```
appio clean <options>
```

1.2.2 Options

Options	Description
-h | –help	prints the command help
-n | –name	APPIO project name

1.2.3 Parent command

appio

1.2.4 Detailed Description

APPIO clean deletes the build folder and its contents of the chosen APPIO project.

Note: You can use the *appio build* command to rebuild the temporary files.

1.3 appio deploy

Create a deb-package containing the opcuaapp.

1.3.1 Usage

```
appio deploy <options>
```

1.3.2 Options

Options	Description
-h | –help	prints the command help
-n | –name	appio project name

1.3.3 Parent command

appio

1.3.4 Detailed Description

This wraps the opcuaapp into a debian package with the name 'appio-opcuaapp.deb' inside the now generated subdirectory 'deploy'.

Note: Before the deployment you have to execute the command *appio publish*.

1.4 appio generate

Generate informationmodels or certificates in the APPIO project.

1.4.1 Usage

```
appio generate <argument> <options>
```

1.4.2 Arguments

Argument	Description
information-model	Generate an informationmodel
certificate	Generate a certificate

1.4.3 Options

Options	Description
-h | –help	prints the command help
-n | –name	appio project name

Options(certificate only)	Description
–days	duration of the certificate in days(default: 365 days)
–keysize	size of the key(default : 1024 bit)
–organization	Organization name used for certification (default: MyOrg)

1.4.4 Parent command

appio

1.4.5 Detailed Description

Generates either an OPC UA certificate or the source code of the imported information-model.

After the generation of the informationmodel, *appio build* needs to be called.

Note: You have to import an informationmodel with *appio import*.

1.5 appio help

Displays the command line APPIO help.

1.5.1 Usage

```
#1
appio --help

#2
appio -h

#3
appio help
```

1.5.2 Options / Commands

Options / Commands	Description
–help	prints the help
-h	
help	

1.5.3 Parent command

appio

1.5.4 Detailed Description

APPIO help opens the general command line help.

Note: For a more detailed help about a command, type 'appio [command] -h-help'.

1.6 appio import

Imports external resources.

1.6.1 Usage

```
appio import <argument> <options>
```

1.6.2 Arguments

Argument	Description
information-model	import an informationmodel
certificate	import a certificate

1.6.3 Options

Options	Description
-h | –help	prints the command help
-n | –name	appio project name

Options(certificate only)	Description
-c | –certificate	path to the certificate
–client or –server	if the APPIO project is both a server and a client, this is used to determine in which the certificate should be imported
-k | –key	path to the keyfile

Options(informationmodel only)	Description
-p | –path	path of the information model certificate
-s | –sample	choose a sample provided by APPIO

1.6.4 Parent command

appio

1.6.5 Detailed Description

APPIO import imports an OPC UA conform information model or an existing X509 certificate and and corresponding private key.S

Alternatively a sample information model provided by APPIO can be used. The certificate can also be generated with appio *appio generate*.

Available samples are: DI-Informationmodel.

Note: The imported information-model needs to generated with *appio generate*.

1.7 appio new

Creates a new APPIO project.

1.7.1 Usage

```
appio new <argument> <options>
```

1.7.2 Arguments

Argument	Description
opcuaapp	will create an opcuaapp
sln	will create a solution

1.7.3 Options

Options	Description
-h | –help	prints the command help
-n | –name	opcuaapp/sln name
-t | –type	Type of opcuapp
–nocert	no certifcate creation

Types of opcuaapps	Description
Client	opcuaapp will be a client
Server	opcuaapp will be a server
ClientServer	opcuaapp will be both

Options(opcuaapp server only)	Description
-u | –url	URL of the server
-p | –port	port of the server

1.7.4 Parent command

appio

1.7.5 Detailed Description

APPIO new creates a new APPIO project. There are two possible Options: creating a solution and creating an opcuaapp.

The result is a folder with the chosen name and a basic opcuaapp/solution setup.

The solution is used for managing opcuaapps.

1.8 appio reference

Add or remove a reference of a server to/from an Client.

1.8.1 Usage

```
appio reference <argument> <options>
```

1.8.2 Arguments

Argument	Description
add	add an OPCU UA server
remove	remove an OPCU UA server

1.8.3 Options

Options	Description
-c | –client	target client
-s | –server	target server
-h | –help	opens the command help

1.8.4 Parent command

appio

1.8.5 Detailed Description

Add or remove a reference of an APPIO OPC UA server to/from an APPIO OPC UA Client.

Note: If you want to access the data nodes from the target server, check out the source code from this example *Pump example*.

1.9 appio publish

Publish an APPIO project for deployment or usage. This copies the binaries into the publish folder.

1.9.1 Usage

```
appio publish <options>
```

1.9.2 Options

Options	Description
-h | –help	prints the command help
-n | –name	appio project name

1.9.3 Parent command

appio

1.9.4 Detailed Description

APPIO publish copies the binaries generated by APPIO build into the publish folder.

Note: You can use the *appio deploy* command to deploy the opcuaapp into a deb package

Note: There is no need to deploy the opcuaapp if you don't need a debian package. You can use the binaries right away.

1.10 sln

1.10.1 sln add

Adds an opcuaapp to an APPIO solution.

Usage

```
appio sln add <options>
```

Options

Options	Description
-h | –help	prints the command help
-s | –solution	APPIO solution name
-p | –project	opcuaapp name

Parent command

sln

Detailed Description

Sln add adds a chosen opcuaapp to the defined APPIO solution.

Note: If you use multiple opcuaapps, a solution makes the handling of these much easier. Example: You can build all the added opcuaapps with appio *sln build*.

1.10.2 sln build

Builds an APPIO solution, which means all added opcuaapps are built.

Usage

```
appio sln build <options>
```

Options

Options	Description
-h | –help	prints the command help
-s | –solution	APPIO solution name

Parent command

sln

Detailed Description

APPIO sln build executes appio build for each APPIO project contained in the solution.

Note: After a successfull build you can publish the opcuaapp with appio *sln publish*.

1.10.3 sln deploy

Deploys all added opcuaapps of a solution into deb-packages.

Usage

```
appio deploy <options>
```

Options

Options	Description
-h | –help	prints the command help
-s | –solution	solution name

Parent command

sln

Detailed Description

APPIO sln build executes appio deploy for each APPIO project contained in the solution.

Note: Before the deployment you have to execute the command *sln publish*.

1.10.4 sln publish

Prepares the added opcuaapps for usage.

Usage

```
appio sln publish <options>
```

Options

Options	Description
-h | –help	prints the command help
-s | –solution	appio solution name

Parent command

sln

Detailed Description

APPIO sln publish executes appio publish for each APPIO project contained in the solution.

Note: You can use the *appio deploy* command to deploy the opcuaapps into debian packages.

Note: There is no need to deploy the opcuaapps. You can use the binaries right away.

1.10.5 sln remove

Removes an added opcuaapp froma an APPIO solution.

Usage

```
appio sln remove <options>
```

Options

Options	Description
-h | –help	prints the command help
-s | –solution	appio solution name
-p | –project	opcuaapp name

Parent command

sln

Detailed Description

APPIO sln remove removes a chosen opcuaapp of the defined APPIO solution, that was added with the *sln add* command.

1.11 appio version

Displays the versions of the appio dlls.

1.11.1 Usage

```
appio --version
```

1.11.2 Options

Options	Description
–version	prints the dll-versions

1.11.3 Parent command

appio

1.11.4 Detailed Description

APPIO version displays the version of the APPIO dlls.

CHAPTER 2

APPIO manuals

2.1 How to use an information-model

In this tutorial you will learn how to extend the opcuaapp with an information-model of your choice. Before we begin, you should have created an APPIO project. For testing the results of this tutorial a opc ua front end client like UA Expert by Unified Automation is needed.

2.1.1 Import the information-model

Get the information-model of your choice and place it your APPIO project folder. You can also use the example information-model 'DiNodeset'.

Execute APPIO import.

```
#custom information-model

appio import information-model -n appiotutorial -p myNS.xml -t DiTypes.bsd

#or the sample

appio import information-model -n appiotutorial -s
```

The information-model is now ready for generating.

2.1.2 Generate the opcuaapp

Next step: generating. You have to choose the Informationmodel.

```
appio generate information-model -n appiotutorial
```

APPIO uses the information-model parser from the open62541 stack to import the Informationmodel.

```
The opcuaapp with the name 'appiotutorial' was succesfully updated with the␣
↪Information model 'myNS.xml'.
```

2.1.3 Build and deploy

Now it is time to build and publish the opcuaapp.

Execute:

```
appio build -n appiotutorial && \
appio publish -n appiotutorial
```

Result:

```
Build 'appiotutorial' success!
publish 'appiotutorial' success!
```

2.1.4 Starting the OPC UA server

To start the OPC UA server generated by APPIO, execute:

```
cd  appiotutorial/publish/ && \
./server-app
```

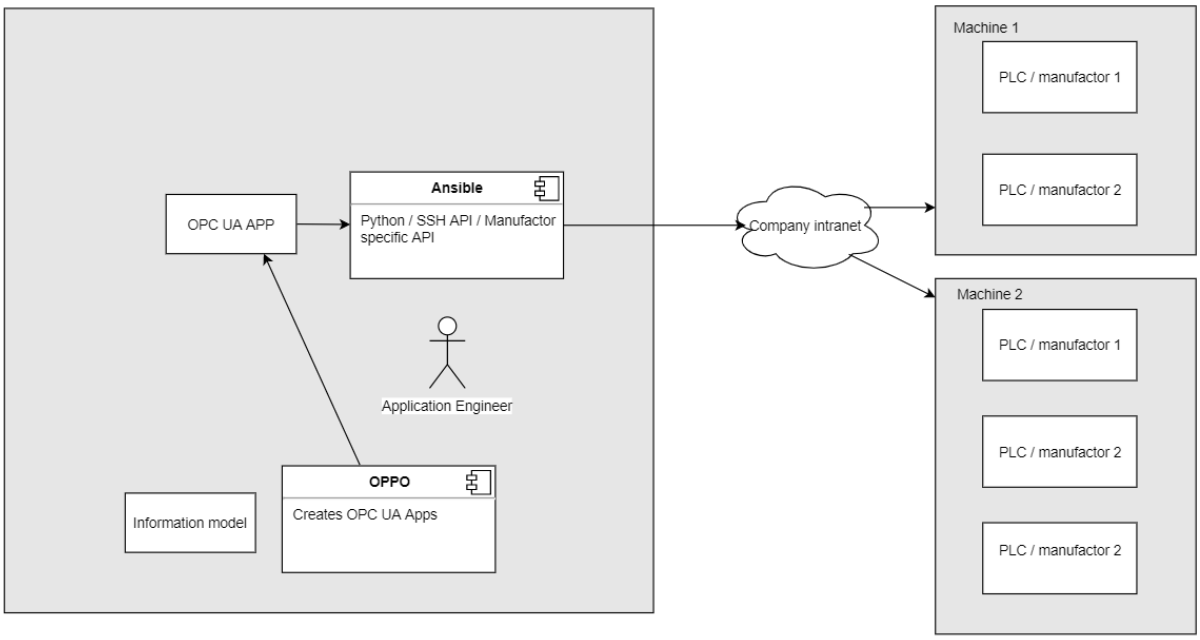
You can read your information-model with the frontend OPC UA client.

2.2 APPIO and infrastructure as code

APPIO was as an infrastructure tool designed, in that sense that it creates and orchestrates OPC UA clients and servers. After the creation of the Debian package, the package should be delieverd and installed on the target system. This can be done by infrastructure as code tools like Ansible. The idea is to create the opcuaapps on the local system and to make the delivery with Ansible. This should be easy on Linux systems, and it might be also possible to add PLC support of certain manufacturs to Ansible. Using infrastructure as code tools also introduces the opportunity to use APPIO in test driven way. In this example Molecule is used to create an Ansible role that creates an opcuaapp.

The advantage of this is, to have automatic tests that ensure the correct creation of the opcuaapps. In a production environment that demands a high changeability of its machines and their communciation, a quick and secure way to automaticly generate and delviever the communciation instances is invalueble. In conjunction wtih an infrastructure as code tool APPIO is a good step in the right direction.

Here is the component diagramm:



Please note that this example is only a prove of concept. In the future, more examples will be added. If you see possible improvements please let us know.

This example contains:

- an Ansible role that installs APPIO on the local system
- an Ansible role that creates an APPIO opcuaapp on the local system
- an Ansible role that delievers the APPIO opcuaapp to an Ubuntu server

You can find the example in the example folder of the APPIO repository.

2.3 Installation

Please clone the APPIO repository, install all the dependencies and execute the following command at the project root:

```
bash-developer/run-reinstall--appio-terminal--local.sh
```

Two packages will be installed:

- open62541–v1.0.0
- appio-terminal

open62541–v1.0.0 is the used OPC UA Stack.

2.3.1 List of dependencies

The dependencies of APPIO are:

- up to date linux
- .net Core 2.1 sdk

- gcc compiler
- meson
- python3
- dpkg

If you use Ubuntu the dependencies can be easily installed with these commands:

```
apt update
apt upgrade
apt install gcc meson python3 dpkg
```

2.3.2 Using package manager

APPIO can be easily installed with the package manager. First you need to install the open62541 stack. Continue with installing APPIO.

```
cd /home/appiodev/Downloads/artifacts/installer/
dpkg --install open62541--v1.0.0.deb
dpkg --install appio-terminal.deb
```

2.3.3 Using grahpical Installer

It is also possible to install APPIO with the graphical installer. Simply double click on the packages to install them. Keep in mind that you have to install the open62541 package first.

2.3.4 Testing

After installing you can test the succesfull installation with:

```
APPIO --help
```

This should open the APPIO commandline help.

After that you can continue with the *Tutorial*.

2.4 Pump example

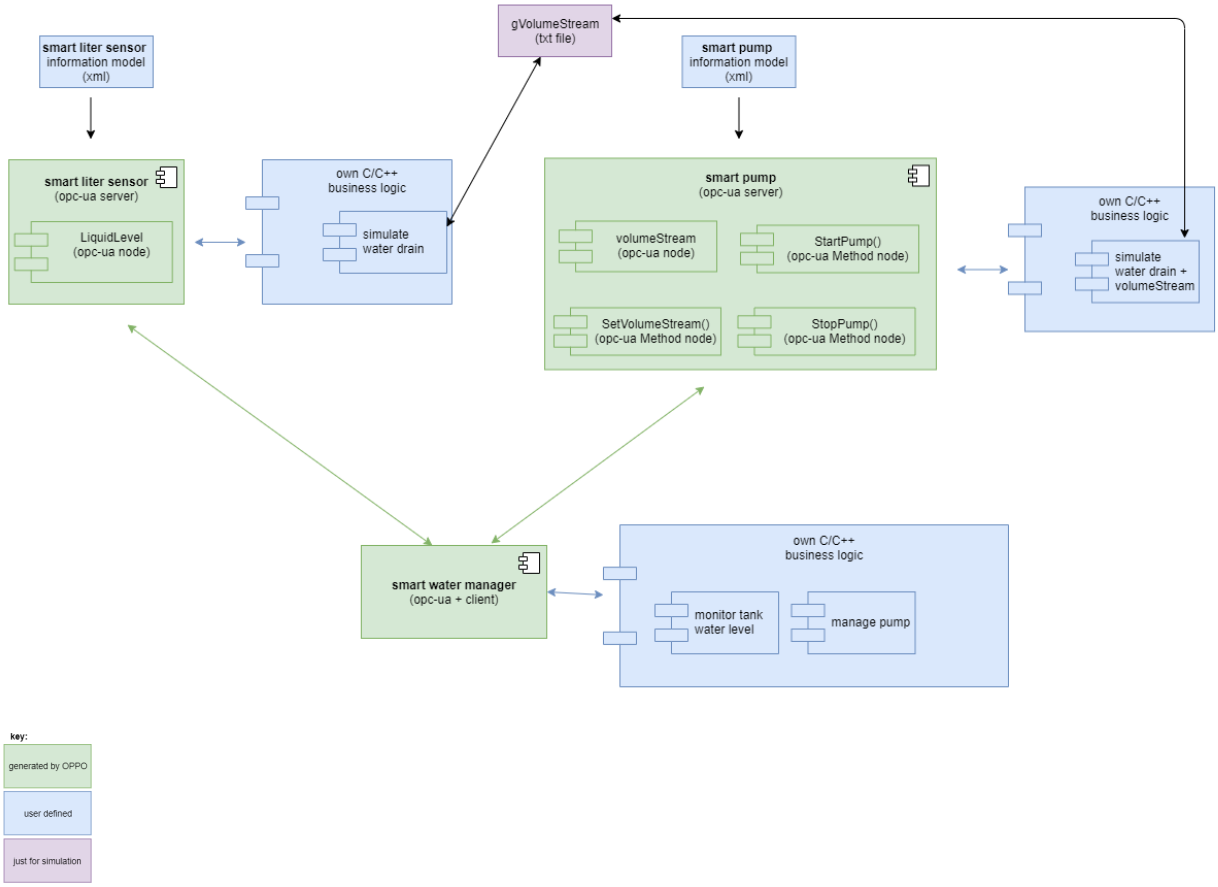
In this tutorial we will see a more practical example. We will create an APPIO solution which contains three opcuaapps. Two servers and one client who simulate a simple pump control. Before we begin, you need to have installed APPIO on a fully updated Linux. For checking the results of this tutorial an OPC UA front end client like UA Expert by Unified Automation is recommended.

The files needed for this tutorial can be found in the example folder of the github repository.

2.4.1 Overview

This example contains two OPC UA Servers, which simulate a smart pump and a smart sensor. The smart sensor measures the actual water level of a container, who leaks water. The smart pump increases this level when activated.

The client called smart water manager, gets the actual water level from the smart sensor via an OPC UA Method, and activates/deacivates the smart pump (again over OPC UA methods) accordingly to user defined limit.



2.4.2 Start: Creating the opcuaapps

First we will create the smart liter sensor app:

```
appio new opcuaapp --name SmartLiter --type Server --url 127.0.0.1 --port 3000 --
↪nocert
```

This will create the OPC UA server for the pump:

```
appio new opcuaapp --name SmartPump --type Server --url 127.0.0.1 --port 4000 --nocert
```

This will create the OPC UA client for the water manager:

```
appio new opcuaapp --name SmartWaterManager --type Client --nocert
```

Now all needed opcuaapps were created.

2.4.3 Importing the information model files

The information-models for the SmartPump and the SmartLiter are provided by us. They use the DiNodeset. Execute the following commands:

```
appio import information-model -n SmartLiter -p path/to/DiNodeset.xml && \
appio import information-model -n SmartPump -p path/to/DiNodeset.xml
```

Finally we need to import the real information models:

```
appio import information-model -n SmartLiter -p path/to/slsNodeset.xml && \
appio import information-model -n SmartPump -p path/to/spNodeset.xml
```

Now we need to generate them:

```
appio generate information-model -n SmartLiter && \
appio generate information-model -n SmartPump
```

2.4.4 Adding References to the client

After the configuration of the OPC UA servers we need to add server references to the client:

```
appio reference add --client SmartWaterManager --server SmartLiter   && \
appio reference add --client SmartWaterManager --server SmartPump
```

2.4.5 Creating a solution

For conviences we will create a solution and add the opcuaapps:

```
appio new sln -n PumpSolution                                  && \
appio sln add -s PumpSolution --project SmartPump              && \
appio sln add -s PumpSolution --project SmartLiter             && \
appio sln add -s PumpSolution --project SmartWaterManager
```

2.4.6 Implementing the business logic code

In the generated file you will find placeholders for making the connection between your business logic and the opcuaapp. This File is called mainCallbacks.c

Example:

```
// callback of Method
UA_MethodCallback MethodCallback(UA_Server *server, const UA_NodeId *sessionId, void␣
↪*sessionContext, const UA_NodeId *methodId, void *methodContext, const UA_NodeId␣
↪*objectId, void *objectContext, size_t inputSize, const UA_Variant *input, size_t␣
↪outputSize, UA_Variant *output)
{
    // Your Method call of your business code goes here
    return UA_STATUSCODE_GOOD;
}
```

For the convience of this tutorial we prepared all files. You will find them in the examples folder we provided.

Here for the SmartPump:

```
cp pathTo/APPIOframework/examples/smart_pump_example/source/smart_pump/mainCallbacks.
↪c SmartPump/src/server/mainCallbacks.c  &&
cp pathTo/APPIOframework/examples/smart_pump_example/source/smart_pump/pumpSimulation.
↪c SmartPump/src/server/pumpSimulation.c
```

(continues on next page)

(continued from previous page)

Here for the smart liter sensor:

```
cp pathTo/APPIOframework/examples/smart_pump_example/source/smart_liter_sensor/
↪tankSimulation.c SmartLiter/src/server/tankSimulation.c &&
cp pathTo/APPIOframework/examples/smart_pump_example/source/smart_liter_sensor/
↪mainCallbacks.c SmartLiterSensor/src/server/mainCallbacks.c
```

And now for the client:

```
cp pathTo/APPIOframework/examples/smart_pump_example/source/smart_water_manager/main.
↪c SmartWaterManager/src/client/main.c &&
cp pathTol/APPIOframework/examples/smart_pump_example/source/smart_water_manager/
↪manager.c SmartWaterManager/src/client/manager.c
```

2.4.7 building and testing

Now it's time to build and test this solution:

```
appio sln build -s PumpSolution  && \
appio sln publish -s PumpSolution
```

Now start all the opcuaapps and test it with the front end client. The pump should be turned on / off when the value provide by the sensor crosses the limit.

2.5 Terminology

Term	Description
opcuaapp	This is an OCP UA Project from APPIO.
sln	The solution is for the orchestration of opcuaapps. Example: Joint deployment of opcuaapps.
OPC UA Instance	Can be a OPC UA Client, a OPC UA Server or both. Will be generated from an opcuaapp by APPIO.
Information-model	Is a set of OPC UA Nodes, that are defining the server address space.

2.6 Tutorial

This tutorial guides you trough the first steps with APPIO. Before we begin, you need to install APPIO on a fully updated Debian-Linux. In this example we use Ubuntu. An OPC UA front end client like UA Expert by Unified Automation is recommended for testing purposes.

2.6.1 Creating a new APPIO project

The First Step is to create a new APPIO project. Please create a new folder somewhere on your harddisk and open the terminal inside of the folder an execute:

```
appio new opcuaapp -n appiotutorial -t Server -u 127.0.0.1 -p 3000 --nocert
```

If this was succesfull, appio responds with:

```
An opcuaapp with name 'appiotutorial' was successfully created!
```

When APPIO executes the new command, it creates everything necessary for the opcuaapp. In this case this means the basic source code for an OPC UA Server on the local network listenning on port 3000. If no user defined information-model is chosen, APPIO will use a default one with a temperature node for testing purposes.

Note: APPIO is stateless. This means it is possible to execute the commands without any certain order.

2.6.2 Building the APPIO project

It is time to build the project. Use the APPIO build command like this.

```
appio build -n appiotutorial
```

APPIO builds the project now. You can see the build process inside the terminal.

After a succesfull build, APPIO responds with:

```
Build 'appiotutorial' success!
```

2.6.3 Publishing the APPIO project

Next step: publishing the opcuaapp. The publish command copies the binaries of the opcuaapp into a newly created subdirectroy named publish.

```
appio publish -n appiotutorial
```

If the publishing was succesfull, APPIO responds the usual way:

```
Publish 'appiotutorial' success!
```

2.6.4 Deploying the APPIO project

The deploying command puts the opcuaapp inside a debian package.

Execute:

```
appio deploy -n appiotutorial
```

Result:

```
Deploy 'appiotutorial' success!
```

Please install the debian package now. You will find it inside the deploy folder of your APPIO project.

2.6.5 Starting the OPC UA server

To start the OPC UA server generated by APPIO, execute:

```
server-app
```

The will start the server and you can read the default node on the server with OPC UA client.

Note: You can close the server any time with 'ctrl + c'

2.7 Uninstallation

2.7.1 With package manager

To uninstall APPIO simply execute the following commands:

```
dpkg -r appio-terminal
dpkg -r open62541--v1.0.0
```

2.7.2 Testing

After installing you can test the succesfull uninstallation with:

```
appio --help
```

The APPIO help should not be displayed.