HSPI Documentation

Release 1.0.0

Alex Dresko

Nov 08, 2018

Contents:

1 Summary **1**

2 Other Helpful Links **3**

2.1 What is it? . . . 3
2.1.1 Project templates for Visual Studio 2015 and Visual Studio 2017 . . . 3
2.1.2 The HSPI framework and nuget package . . . 3
2.1.3 HomeSeerNuget . . . 4
2.2 Features . . . 4
2.2.1 The templates . . . 4
2.2.1.1 C# . . . 4
2.2.1.2 VB.NET . . . 4
2.2.2 Code Quality . . . 5
2.2.3 Framework . . . 5
2.3 Using HSPI . . . 5
2.3.1 Getting Started . . . 5
2.3.2 Adding an action to your plugin . . . 6
2.3.2.1 General functions that you will use . . . 10
2.3.3 FAQ . . . 10
2.3.3.1 Using a different IP address and/or port . . . 10
2.4 How to contribute . . . 11
2.4.1 Developer Workflow . . . 11
2.4.1.1 Always start with an issue . . . 11
2.4.1.2 Creating issues . . . 11
2.4.1.3 Beginners and n00bs . . . 11
2.4.1.4 Git branching strategy . . . 12
2.4.1.5 Creating pull requests . . . 12
2.4.1.6 ZenHub . . . 13
2.4.2 Documentation . . . 13
2.4.2.1 Summary . . . 13
2.4.2.2 Installing the software . . . 14
2.4.2.3 Side by Side . . . 14
2.4.2.4 Common Substitutions . . . 14
2.4.3 Contributor Covenant Code of Conduct . . . 15
2.4.3.1 Our Pledge . . . 15
2.4.3.2 Our Standards . . . 15
2.4.3.3 Our Responsibilities . . . 15

2.4.3.4 Scope 16
2.4.3.5 Enforcement 16
2.4.3.6 Attribution 16
2.5 Roadmap 16
2.6 Change Log 16
2.6.1 vNext 16
2.6.1.1 Contributers 16
2.6.1.2 Templates 17
2.6.2 3/10/2017 17
2.6.2.1 Extension 17
2.6.2.2 Contributers 18
2.7 Meta 18

CHAPTER 1

Summary

HSPI aims to make extending HomeSeer significantly easier by empowering developers with awesome tools and a collaborative community. Learn more at https://github.com/alexdresko/HSPI

- **For all the latest HSPI news, status updates, and release information** https://hspiweb.wordpress.com/
- **New to HSPI and looking for documentation? We've got that.** http://hspi.readthedocs.io/en/latest/
- **Got a question about HSPI? Ask on Stack Overflow and tag with "HSPI". We're watching and waiting for your questions**
  http://stackoverflow.com/questions/ask?tags=HSPI

  Using Stack Overflow, as opposed to the HomeSeer forums, allow us to more easily track questions related to HSPI and their status.
- **Found a bug or want to submit a feature request? We're actively improving this thing.** https://github.com/alexdresko/HSPI/issues

  Using the Github issues, as opposed to the HomeSeer forums, allows us to more easily track issues and their status.
- **Want to chat in real time with other HSPI developers or view a real-time feed of everything happening in HSPI? The live f**
  https://gitter.im/HSPI/Lobby

CHAPTER 2

Other Helpful Links

- **HomeSeer's HS3 developer forum:** https://forums.homeseer.com/forumdisplay.php?f=1138
- **HomeSeer's original plugin samples:** https://forums.homeseer.com/showthread.php?t=160064
- **HomeSeer's SDK documentation:** http://homeseer.com/support/homeseer/HS3/SDK/default.htm

2.1 What is it?

HSPI consists of the following:

2.1.1 Project templates for Visual Studio 2015 and Visual Studio 2017

With HSPI, getting started with HomeSeer development is as simple as:

1. Locate and install the HSPI extension in Visual Studio via "Tools > Extensions and Updates" or download it from https://marketplace.visualstudio.com/items?itemName=thealexdresko.HomeSeerTemplates-18379
2. File > New Project > HomeSeer > Select a template
3. F5 to launch the project and verify connectivity to HomeSeer.

HSPI comes with a growing list of project templates you can use to accelerate your HomeSeer development.

2.1.2 The HSPI framework and nuget package

I imagine most HomeSeer development efforts end in failure. Mine did. Out of the box, it's just painful.

So, I scoured the HomeSeer developer forums, collecting all the code and tips possible. I then crammed everything into a nuget package (https://www.nuget.org/packages/HSPI) and used my coder nerd wizardry to bring it all together.

With HSPI, we can continue to refine this framework into something that makes HomeSeer plugin development mind-numbingly simple.

2.1.3 HomeSeerNuget

A nuget package containing the three assemblies required for HomeSeer development:

- HomeSeerAPI.dll
- HSCF.dll
- Scheduler.dll

Many first time HomeSeer developers get stumped working with the HomeSeer-developed plugin samples (https://forums.homeseer.com/showthread.php?t=160064) due to these assemblies not being included with the samples. **This nuget package is not specific to HSPI and can be used in any HomeSeer plugin, including the HomeSeer-developed plugin samples.** Because of its generic nature, HomeSeerNuget (https://www.nuget.org/packages/Homeseer/) is maintained in a separate repository (https://github.com/alexdresko/HomeSeerNuget).

2.2 Features

2.2.1 The templates

2.2.1.1 C#

Until we have awesome names for the templates, you'll have to use this key to determine which template is most appropriate for your project.

- **HSPIPluginA**

  This template, based on the code at https://forums.homeseer.com/showthread.php?t=178122, is as close as you can get to inheriting directly from HomeSeer's `IPluginAPI` interface while still retaining the benefits of HSPI.

  The plugin created in this template inherits from `Hspi.HspiBase`.

  It's a venerable sea of code – intimidating for new HomeSeer developers.

- **HSPIPluginB**

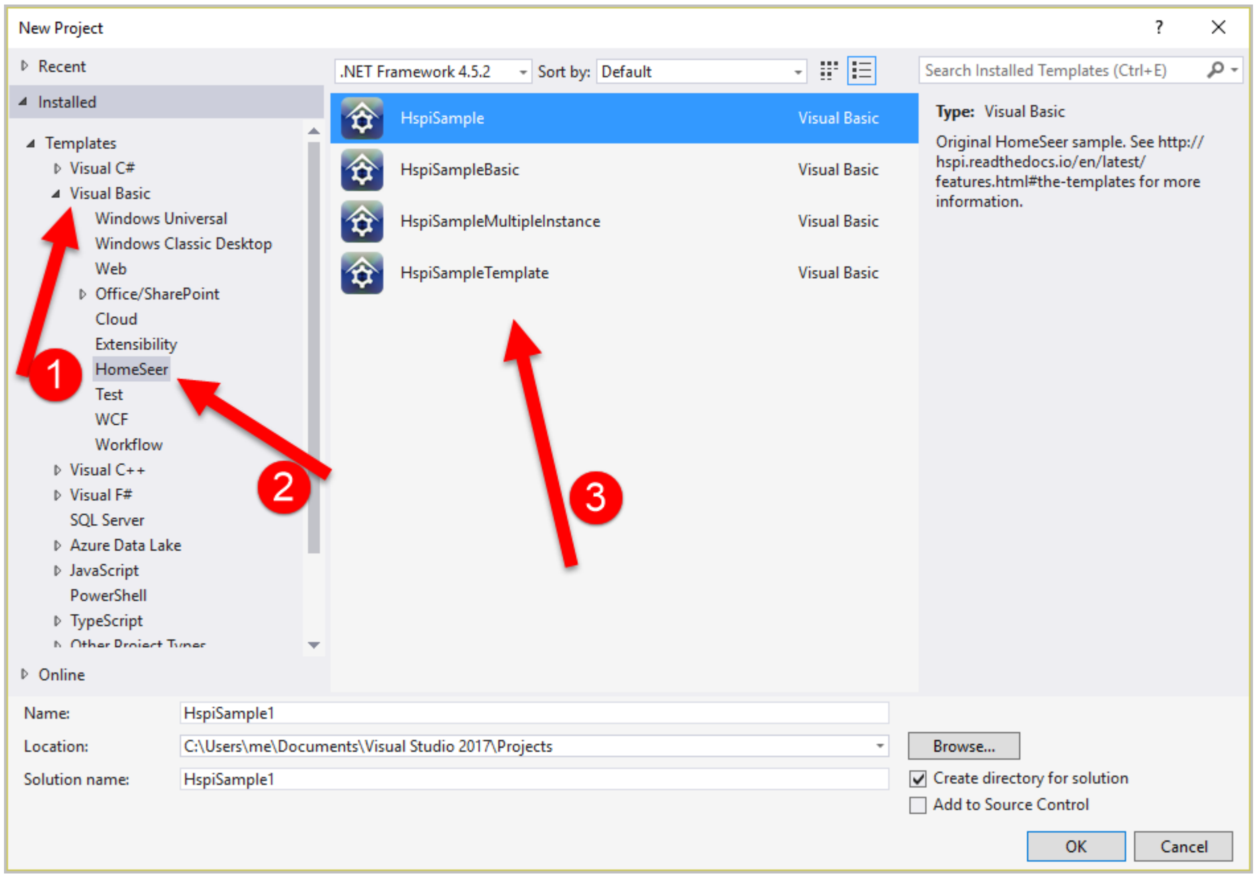
  This template requires the smallest amount of code out of the box. It's based on the code at https://forums.homeseer.com/showthread.php?t=184086.

  The plugin created in this template inherits from `Hspi.HspiBase2` which inherits from `Hspi.HspiBase`.

  You get all of the benefits of HSPI, without having to implement everything in HomeSeer's `IPluginAPI` interface. The base class implements suitable defaults for basic plugins, preventing you from having to understand everything in `IPluginAPI`. The flip side to such minimalistic code is that it's not obvious which methods you *can* override.

2.2.1.2 VB.NET

Yes, the HSPI extension includes the original HomeSeer VB.NET project samples and templates in the extension. They're mostly changed from the download at https://forums.homeseer.com/showthread.php?t=160064, except the project uses the HomeSeerNuget package to make getting started a little easier. You'll find everything under `Visual Basic > HomeSeer`.



Type: Visual Basic

Original HomeSeer sample. See http://hspi.readthedocs.io/en/latest/features.html#the-templates for more information.

Name: HspiSample1

Location: C:\Users\me\Documents\Visual Studio 2017\Projects

Solution name: HspiSample1

[✓] Create directory for solution

[ ] Add to Source Control

2.2.2 Code Quality

- All C# builds in HSPI are validated against a fairly strict set of static code analysis rules in an effort to automate code quality checks.
- A Resharper "Code Cleanup" setting called "HSPI" helps to ensure all code follows the same standards.

2.2.3 Framework

- `HspiBase` tries to eliminate many of HS's poor naming conventions from `IPluginAPI`. It's quite a bit easier to understand.

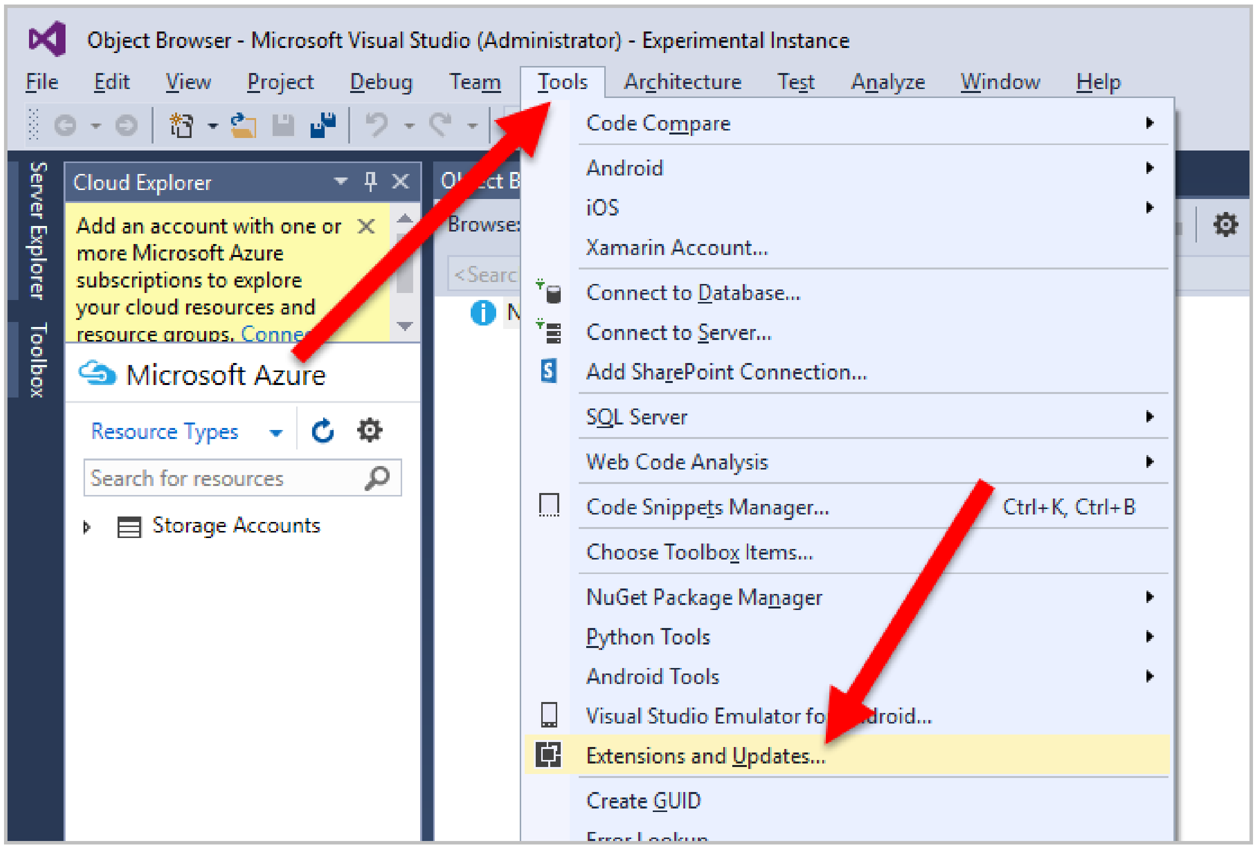
2.3 Using HSPI

2.3.1 Getting Started

If you are looking to contribute to the HSPI project, please see *How to contribute*

These instructions assume you have a default HomeSeer installation on the same computer that you'll be developing on. If that is not the case, you can still follow these instructions, and then refer to the remaining HSPI documentation to adapt the project for your environment.

1. In Visual Studio, open "Tools > Extensions and Updates"


Object Browser - Microsoft Visual Studio (Administrator) - Experimental Instance
File Edit View Project Debug Team Tools Architecture Test Analyze Window Help
Cloud Explorer
Add an account with one or more Microsoft Azure subscriptions to explore your cloud resources and resource groups. Conne
Microsoft Azure
Resource Types
Search for resources
Storage Accounts
Code Compare
Android
iOS
Xamarin Account...
Connect to Database...
Connect to Server...
Add SharePoint Connection...
SQL Server
Web Code Analysis
Code Snippets Manager... Ctrl+K, Ctrl+B
Choose Toolbox Items...
NuGet Package Manager
Python Tools
Android Tools
Visual Studio Emulator for Android...
Extensions and Updates...
Create GUID
Server Explorer
Toolbox

2. Click the "Online" tab, search for "HomeSeer", and click "Download".

   Alternatively, download it from https://marketplace.visualstudio.com/items?itemName=thealexdresko.HomeSeerTemplates-18379

3. Once installed, you can "File > New Project" to create a HomeSeer plugin. The templates can be found under "Templates > Visual C# > HomeSeer".

   Refer to the *templates list* to determine which template is most appropriate.

4. F5 to launch the project and verify connectivity to HomeSeer.

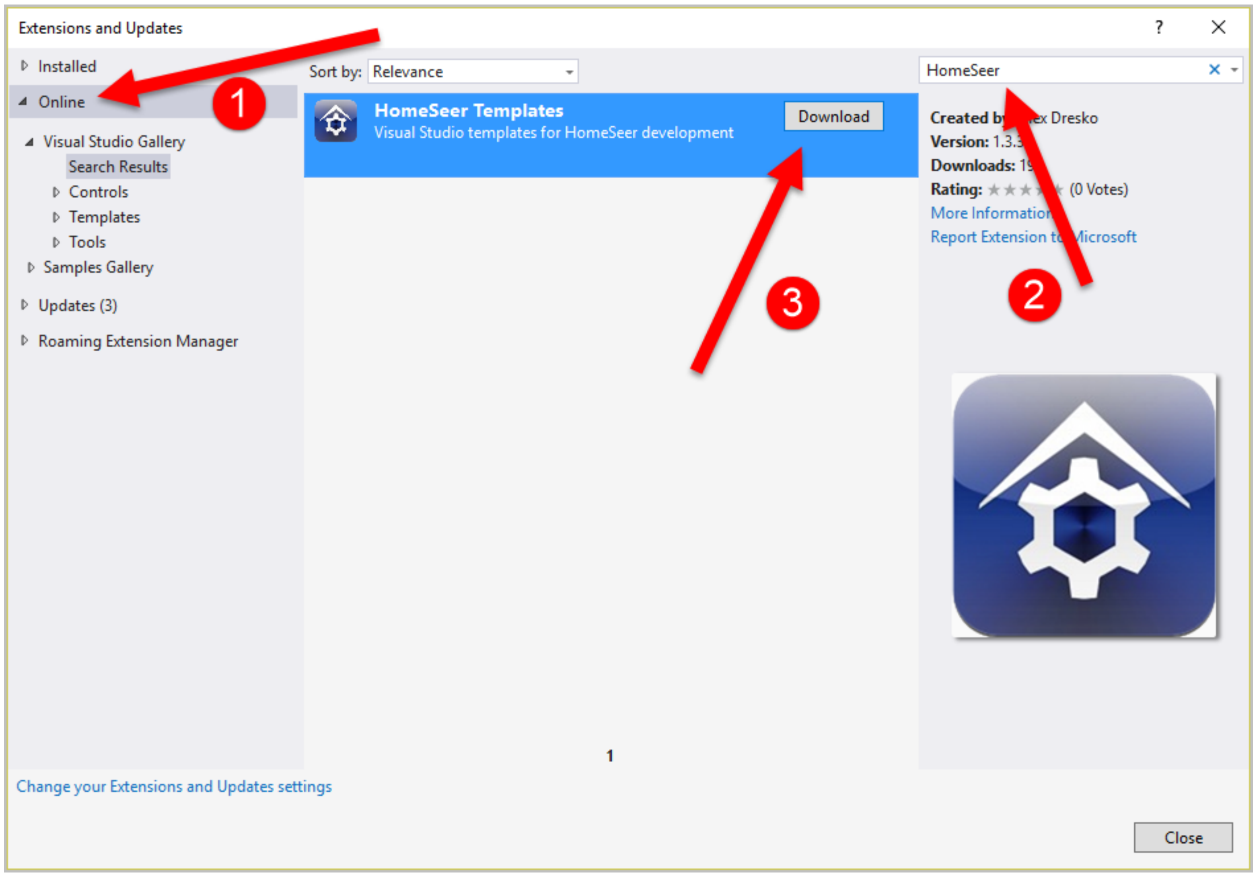
   If all goes well, you should see a success message in the console window:

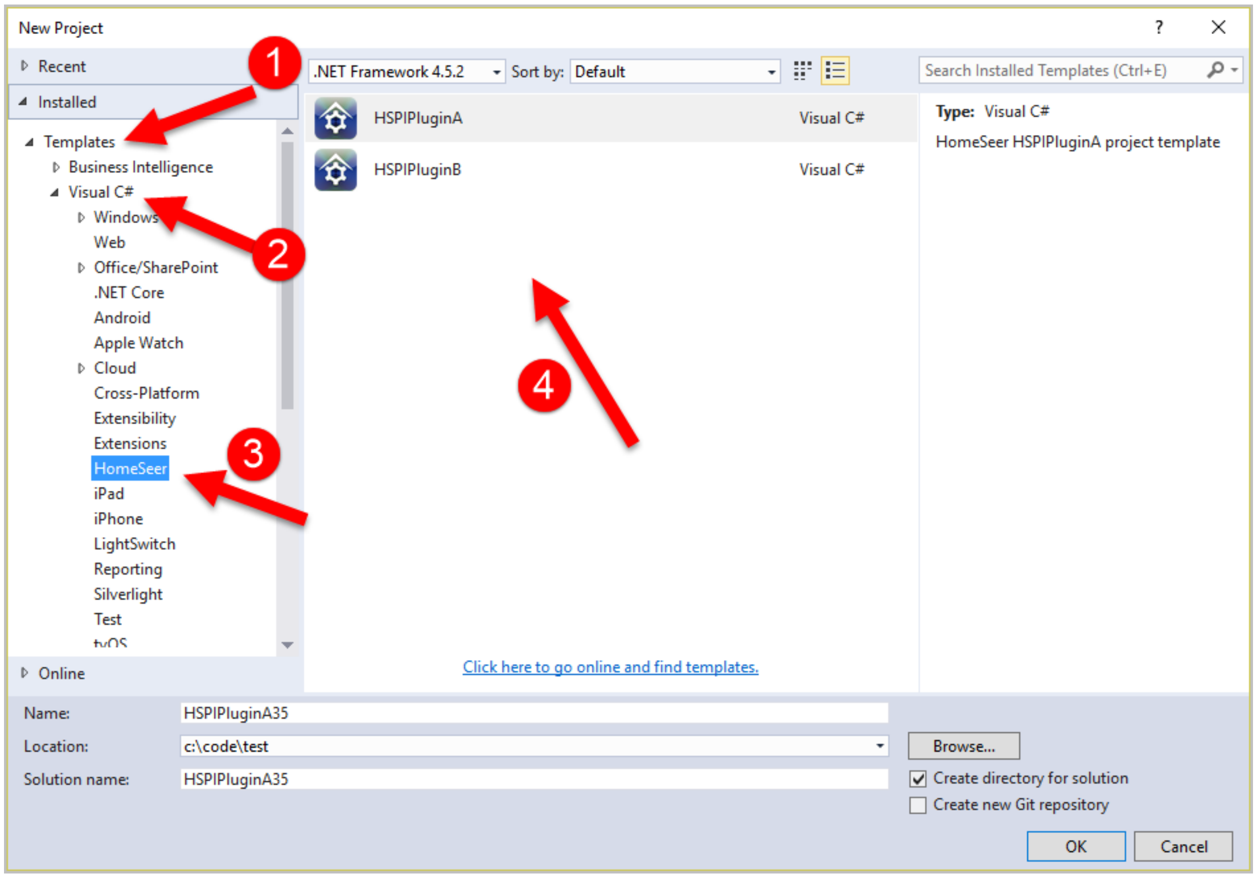
   Further, you should see your plugin listed under "PLUG-INS > MANAGE > Remote Plug-Ins":

2.3.2 Adding an action to your plugin

The following functions are used in creating an action:

- **public override int ActionCount()** This function returns the number of actions found in your plugin.
- **public override bool ActionConfigured(IPlugInAPI.strTrigActInfo actionInfo)** A verification if an action (given by actionInfo) is configured correctly (all relevant data is set up correctly). ActionInfo is a serilalized object which must be deserialized to a HsCollection (Dictionary of string and object).
- **public override string ActionFormatUI(IPlugInAPI.strTrigActInfo actionInfo)** This function returns the text/UI shown to the user when the action is formatted correctly (ActionConfigured returns true) and the


Extensions and Updates
Installed
Online
1
Visual Studio Gallery
Search Results
Controls
Templates
Tools
Samples Gallery
Updates (3)
Roaming Extension Manager
Sort by: Relevance
HomeSeer
HomeSeer Templates
Visual Studio templates for HomeSeer development
Download
3
2
Created by: Alex Dresko
Version: 1.3.3
Downloads: 19
Rating: (0 Votes)
More Information
Report Extension to Microsoft
1
Change your Extensions and Updates settings
Close



.NET Framework 4.5.2 Sort by: Default

Type: Visual C#

HomeSeer HSPIPluginA project template

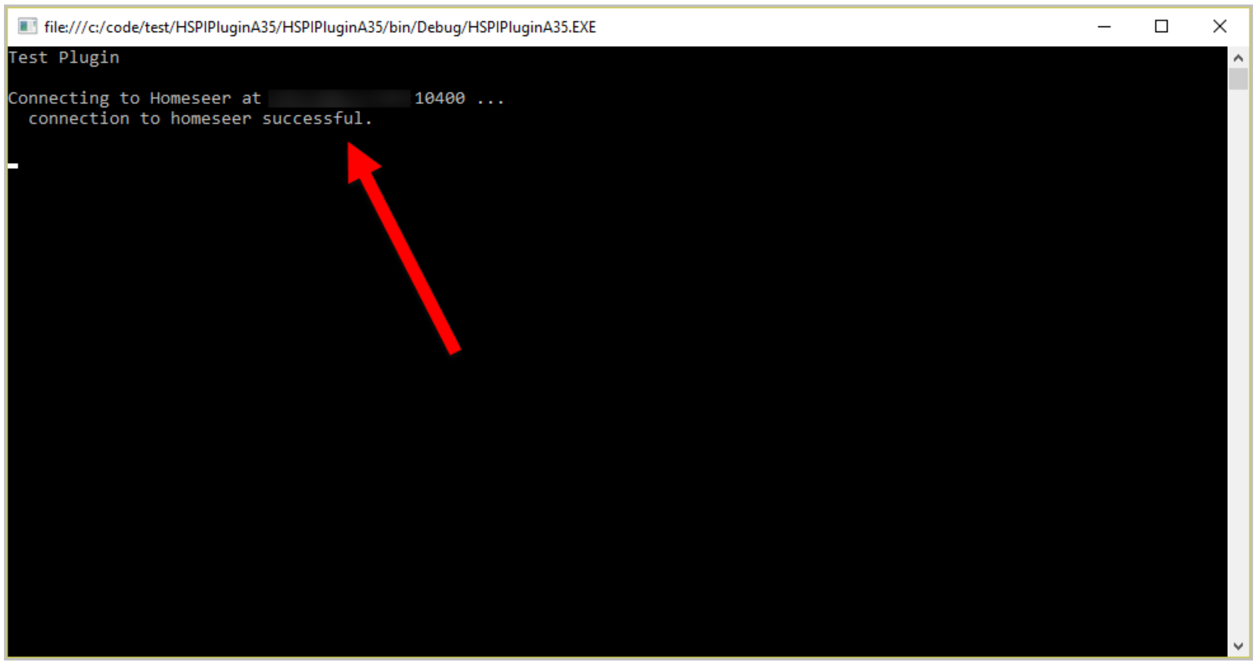
Name: HSPIPluginA35

Location: c:\code\test

Solution name: HSPIPluginA35

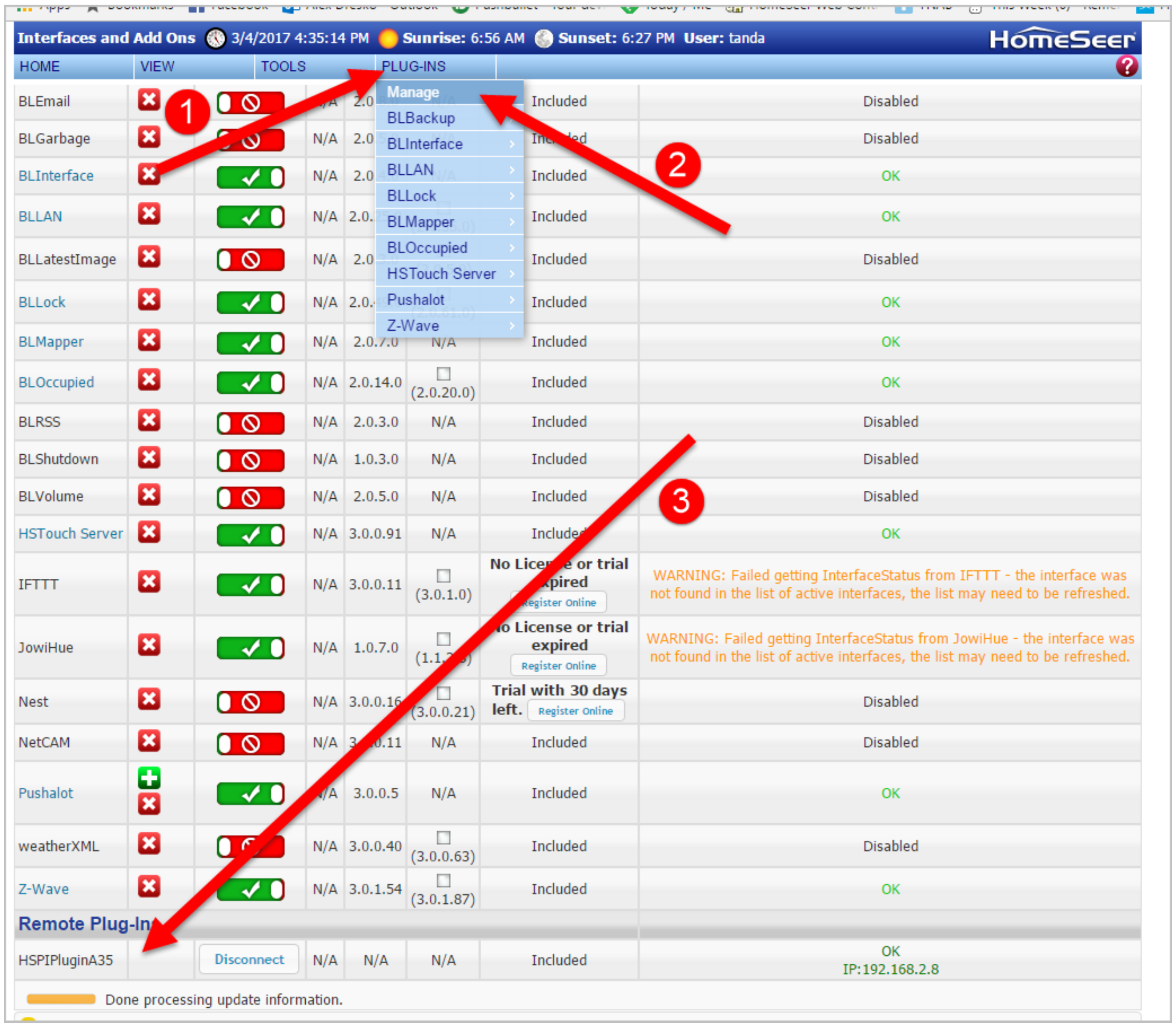
[✓] Create directory for solution

[ ] Create new Git repository


file:///c:/code/test/HSPIPluginA35/HSPIPluginA35/bin/Debug/HSPIPluginA35.EXE

```
Test Plugin

Connecting to Homeseer at [redacted] 10400 ...
  connection to homeseer successful.
```



BLEmail	✖	🟥 ⃠	/A	2.0		Included	Disabled
BLGarbage	✖	🟥 ⃠	N/A	2.0		Inc ed	Disabled
BLInterface	✖	🟩 ✓	N/A	2.0		Included	OK
BLLAN	✖	🟩 ✓	N/A	2.0.		Included	OK
BLLatestImage	✖	🟥 ⃠	N/A	2.0		Included	Disabled
BLLock	✖	🟩 ✓	N/A	2.0.		Included	OK
BLMapper	✖	🟩 ✓	N/A	2.0.7.0	N/A	Included	OK
BLOccupied	✖	🟩 ✓	N/A	2.0.14.0	☐ (2.0.20.0)	Included	OK
BLRSS	✖	🟥 ⃠	N/A	2.0.3.0	N/A	Included	Disabled
BLShutdown	✖	🟥 ⃠	N/A	1.0.3.0	N/A	Included	Disabled
BLVolume	✖	🟥 ⃠	N/A	2.0.5.0	N/A	Included	Disabled
HSTouch Server	✖	🟩 ✓	N/A	3.0.0.91	N/A	Include	OK
IFTTT	✖	🟩 ✓	N/A	3.0.0.11	☐ (3.0.1.0)	No Licen e or trial xpired Register Online	WARNING: Failed getting InterfaceStatus from IFTTT - the interface was not found in the list of active interfaces, the list may need to be refreshed.
JowiHue	✖	🟩 ✓	N/A	1.0.7.0	☐ (1.1.)	o License or trial expired Register Online	WARNING: Failed getting InterfaceStatus from JowiHue - the interface was not found in the list of active interfaces, the list may need to be refreshed.
Nest	✖	🟥 ⃠	N/A	3.0.0.16	☐ (3.0.0.21)	Trial with 30 days left. Register Online	Disabled
NetCAM	✖	🟥 ⃠	N/A	3 .0.11	N/A	Included	Disabled
Pushalot	✚ ✖	🟩 ✓	/A	3.0.0.5	N/A	Included	OK
weatherXML	✖	🟥	N/A	3.0.0.40	☐ (3.0.0.63)	Included	Disabled
Z-Wave	✖	🟩 ✓	N/A	3.0.1.54	☐ (3.0.1.87)	Included	OK
Remote Plug-In							
HSPIPluginA35		Disconnect	N/A	N/A	N/A	Included	OK IP:192.168.2.8

Done processing update information.

action is collapsed.

- **public override string ActionBuildUI(string uniqueControlId, IPlugInAPI.strTrigActInfo actionInfo)**
  This function builds the html UI where the user can set up the action in Events.
- **public override IPlugInAPI.strMultiReturn ActionProcessPostUI(NameValueCollection postData, IPlugInAPI.strTrigAct**
  Processing of any configuring done in the UI which is sent back from Homeseer.
- **public override string get_ActionName(int actionNumber)** The name of your action for the actionnumber given. Homeseer will use 1 as the first action.
- **public override bool HandleAction(IPlugInAPI.strTrigActInfo actionInfo)** This is the function handling the action when it is triggered in Homeseer.

2.3.2.1 General functions that you will use

- **public override string InitIO(string port)** This is where the initialization of the plugin happens. Keep it quick since Homeseer only waits a couple of seconds at the most during init. Any slower and the plugin is listed as not functioning.
- **public override void ShutdownIO()** This is where actions before shutdown and disconnect are done. Make sure all your stuff is ended correctly so noting is left "haning" during shutdown.

Also you might want to add a config page which involves adding a webpage. More of this in later documentation.

2.3.3 FAQ

2.3.3.1 Using a different IP address and/or port

By default, the console will always connect to 127.0.0.1 port 10400. There is no need to specify any command line arguments for that to work.

However, you can specify a custom server and port in "Debug > Start Options > Command line arguments"…

`_static-properties-ip-port.png`

And, of course, you can do the same thing at the command line:

`_static-ip-port.png`

2.4 How to contribute

2.4.1 Developer Workflow

2.4.1.1 Always start with an issue

If you just want to help, find something in the current milestone or any open issue with the Help Wanted tag.

Otherwise, please don't start any new work without first checking to see if there's an existing issue describing what you want to do. If there is not an existing issue, please create an issue first, and wait for feedback from the core team.

2.4.1.2 Creating issues

Use this information to determine if creating an issue is the correct thing to do:

- **Have a question about how HSPI works?** Ask on Stack Overflow and tag with "HSPI". This is for questions about how HSPI works, not for reporting bugs or feature requests. If you have an opinion, or want someone else's opinion, it's probably better to use the HSPI chat link.

  http://stackoverflow.com/questions/ask?tags=HSPI

  > *Using Stack Overflow, as opposed to the HomeSeer forums, allow us to more easily track questions related to HSPI and their status.*

- **Github issues** This is not for asking questions, providing an opinion, or requesting an opinion. For questions about how HSPI works, use the Stack Overflow link. For opinion stuff, use the HSPI chat link.

  https://github.com/alexdresko/HSPI/issues

  > *Using the Github issues, as opposed to the HomeSeer forums, allows us to more easily track issues and their status.*

- **HSPI chat** This channel is great for giving and requesting feedback. Use Stack Overflow for questions about how HSPI works, and Github issues for bugs and feature requests.

  https://gitter.im/HSPI/Lobby

When you *do* need to create an issue, it should be easy now. Just create an issueas your normally would in Github and you'll be prompted for any required information.

2.4.1.3 Beginners and n00bs

Please read the entire *How to contribute* section of this site.

Also, we love n00bs. If you need help getting started, head on over to our chat and we'll try to help.

If you've never contributed to a github project, you might find the following links helpful:

How to Contribute to Open Source | Open Source Guides https://opensource.guide/how-to-contribute/

Understanding the GitHub Flow · GitHub Guides https://guides.github.com/introduction/flow/

Hello World · GitHub Guides https://guides.github.com/activities/hello-world/

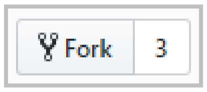
Forking Projects · GitHub Guides https://guides.github.com/activities/forking/

Getting your project on GitHub · GitHub Guides https://guides.github.com/introduction/getting-your-project-on-github/

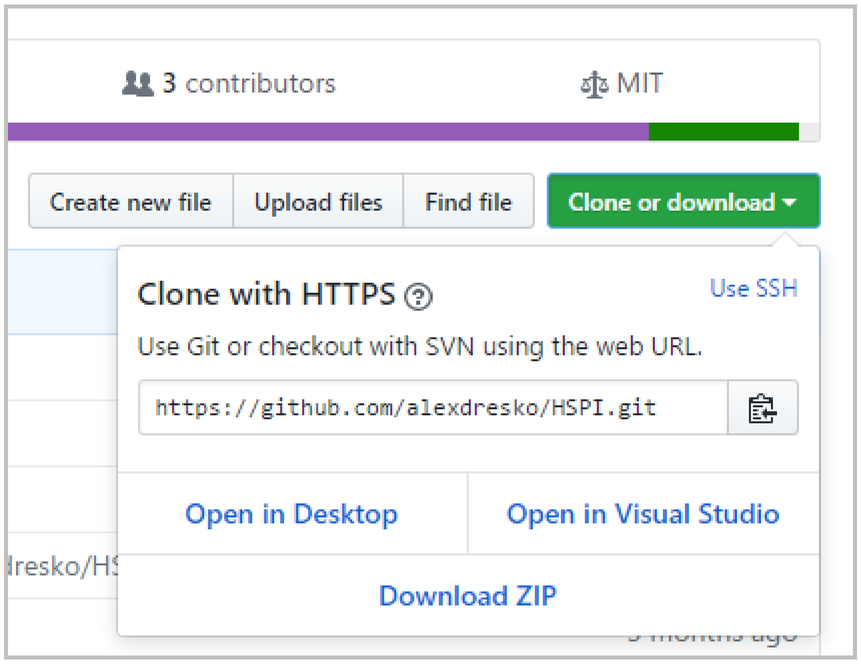
Mastering Markdown · GitHub Guides https://guides.github.com/features/mastering-markdown/

Otherwise, here are the basic steps:

1. Fork the HSPI repo using the `Fork` button.



1. Clone the repo. You might find the `Clone` button useful.



Use Git or checkout with SVN using the web URL.

```
https://github.com/alexdresko/HSPI.git
```

1. Switch to the `Dev` branch. How you do this and the remaining steps will depend on your git stack.
2. Create a new branch for your changes.
3. Commit your changes.
4. Push your changes.
5. Create a pull request against the *Dev* branch.

Note that it's entirely possible to perform all of the above steps without ever leaving Github, though we recommend only doing that for simple changes to HSPI.

2.4.1.4 Git branching strategy

We use a sloppy rendition of Gitflow. Basically, all new work should generally branch from *Dev*. When we feel like we've got enough work to consitute a release, we'll merge *Dev* into *Master*. The primary reason for using this workflow is to prevent frequent builds on the master branch. Every time we build on the master branch, the public nuget package and Visual Studio template are updated. It would be very annoying to HSPI users if they were constantly asked to upgrade HSPI.

2.4.1.5 Creating pull requests

When you're ready to create a pull request, it should be simple. Just create a pull request as your normally would in Github and you'll be prompted for any required information.

2.4.1.6 ZenHub

We use ZenHub to enhance the Github issue management process. It's free for public Github repos, so grab the extension if you want to help maintain HSPI.

Our ZenHub configuration uses the default pipeline:

1. **New Issues** This is where new issues go.
2. **Icebox** This is for issues that need a little more time to simmer.
3. **Backlog** This is for approved issues. Generally, I try to only work on issues that are in the Backlog _and_ assigned to the current milestone. Any issues in the Backlog with the "up for grabs" tag can be tackled by any community contributer at any time.
4. **Blocked** This is for issues that can't be worked on for whatever reason. The reason should be detailed in the issue description.
5. **In Progress** This is for issues that are in progress.
6. **Review/Q** This is for issues associated to a PR that is being reviewed.
7. **Done** This is for issues where the corresponding code has been merged into the Dev branch.
8. **Closed** This is for issues where the corresponding code has been merged into Master.

The following link might prove helpful in understanding how we use ZenHub:

ZenHub - Agile GitHub Project Management: https://www.zenhub.com/guides/agile-concepts-in-github

I also super highly recommend reading this book if you have the time – It's not at all necessary – It's just a really good book, and it further explains how ZenHub integrates with HSPI.

ZenHub - GitHub Project Management Book - Download Free: https://www.zenhub.com/book/github-project-management

Obivously, the HSPI project isn't Agile in the truest sense, but it's easier for me if we stick with the terminology as much as possible.

2.4.2 Documentation

2.4.2.1 Summary

Please help write HSPI's documentation! This project started with the vision of combining and simplifying a lot of work that was already out there. None of that existing work had documentation, so we started with nothing. Any time HSPI confuses you is a good opportunity for documentation. If you can't contribute to HSPI's documentation, at least voice your confusion in our chat room so someone else can clarify things.

This document explains how to set up your environment to make changes to HSPI's documentation. If you're a new contributer to HSPI, be sure to read everything in *How to contribute*.

HSPI's documentation is hosted on https://readthedocs.org/ (RTD). The documentation is written using Sphinx, so you'll need Python. You'll also want gulp, and therefore Node if you want to take advantage of some of the utilities I wrote to make writing documentation for HSPI easier.

If you're starting with a completely new development environment, here are the steps I'd go through to get things set up. If you already have some of these tools, or prefer to do things differently, you're on your own. :)

2.4.2.2 Installing the software

1. Install Chocolatey. At any point in the following steps, you may need to restart your console for the newly installed tool to become available in your console session.
2. Install git via `choco install git -y`
3. Install python via `choco install python -y`
4. Install VS Code via `choco install visualstudiocode -y`. I prefer to write the documentation in VS Code simply because I like VS Code.
5. Install Nodejs via `choco install nodejs -y`
6. Install Sphinx via `pip install sphinx`
7. Install Sphinx autobuild via `pip install sphinx sphinx-autobuild`
8. Install gulp via `npm install -g gulp`
9. `cd` to the directory where you cloned HSPI from git
10. Install the npm dependencies via `npm install`
11. Install the RTD Sphinx template via `pip install sphinx_rtd_theme`
12. Fire up the live preview by running `gulp watch` from the root of the repo. This will launch a web browser on http://localhost:3000/. Any changes you make to any of the rst files in HSPI will usually automatically refresh the browser, allowing you to see the changes you made as they'll appear on http://hspi.readthedocs.io. You might sometimes have to run `gulp build` to perform a full build of all RST files if it seems like the files aren't updating properly. Also, keep an eye on this console window between saves to make sure you don't have any errors in your RST syntax.

That's it. I just tested those instructions on my wife's laptop, in the car while she's in Walmart, tethered to my phone, while writing this very page. Clearly it works.

To learn more about Sphinx and restructuredText, go to http://www.sphinx-doc.org/en/stable/contents.html. It takes a while to get used to rst, but it's much more powerful than Markdown.

To learn more about ReadTheDocs, go to http://docs.readthedocs.io/en/latest/

2.4.2.3 Side by Side

HSPI documentation is stored alongside the HSPI source code. That's great because it allows the documentation and code to stay up to date more easily. If you make a change to code, it only takes a few moments to update the documentation.

Another advantage here is that you can directly include source code from HSPI into the documentation. Here's `HSPI\Option.cs` for example.

That's neat.

Check out the code for this page in the repository to see how that's done. And then go here for more information: http://www.sphinx-doc.org/en/stable/markup/code.html

Note that Sphinx has trouble applying syntax highlighting to some C# files.

2.4.2.4 Common Substitutions

The `rst_epilog` variable in `conf.py` contains a list of common subsitutions. At the time of writing, it looks like the list below, but it's expected to grow quickly.

```
rst_epilog = """
.. _Python: https://www.python.org/
.. _Sphinx: http://sphinx-doc.org/latest/install.html
.. _Chocolatey: https://chocolatey.org/install
.. _chat room: https://gitter.im/HSPI/Lobby
.. |ad| replace:: Alex Dresko
.. _ad: http://www.alexdresko.com
.. _`Contributor Covenant`: http://contributor-covenant.org
"""
```

2.4.3 Contributor Covenant Code of Conduct

2.4.3.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

2.4.3.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

2.4.3.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

2.4.3.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

2.4.3.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at Alex Dresko. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

2.4.3.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at http://contributor-covenant.org/version/1/4

2.5 Roadmap

I love when a software project has a good roadmap.

Unfortunately, I've never built a cool roadmap. :)

In no particular order, and without attaching a date to anything (estimates suck), here are some long term goals for HSPI:

- It would be cool if HomeSeer would officially declared HSPI to be the preferred HomeSeer developer toolkit. Even better if they contributed to HSPI instead of their current process which basically involves updating a zip file of VB.NET examples.

2.6 Change Log

2.6.1 vNext

2.6.1.1 Contributers

- 3/11/2017 10:27:59 AM by AD: Switched to more of a gitflow (http://nvie.com/posts/a-successful-git-branching-model/) based branching strategy. The biggest reason for the switch is because I'm trying to make better use of the build system – Basically, I don't want to continuously publish an updated nuget package and VSIX extension for every arbitrary change we make. Having a `dev` branch allows us to stage blocks of changes before merging `dev` into `master`. It's the merging of `dev` into `master` that triggers the nuget & VSIX publishing, so you can see why merging directly into `master` often would be problematic. Maybe it's just me, but I think it's annoying with a nuget package or VS extension is being updated multiple times a day.

- 3/11/2017 10:28:05 AM by AD: I closed out the "Good enough to use" milestone (https://github.com/alexdresko/HSPI/milestone/1) (because it is), and started "Definitive" https://github.com/alexdresko/HSPI/milestone/2
- 5/24/2017 1:44:31 PM by AD: Created a code of conduct in the documentation.
- 5/27/2017 by AD: Documentation documentation! Hopefully this will allow others to help with the HSPI documentation!

2.6.1.2 Templates

- 3/11/2017 7:35:13 PM by AD: The original HomeSeer provided VB.NET templates are included in the VS extension! For the most part, they are completely unmodified. I simply included them in the extension as a reference. For most cases, you'll want to start with the custom C# HSPI project templates that come with the extension.
- 3/11/2017 7:37:30 PM by AD: The binary files created by the templates automatically conform to the standard HS plugin structure wherein the plugin's dependencies are in a subdirectory called `bin/<plugin name>`. If that sounds confusing, here's what it looks like:

  ```
  HSPI_YourPlugIn.exe
  HSPI_YourPlugIn.exe.config
  bin\YourPlugIn\CommandLine.dll
  bin\YourPlugIn\CommandLine.xml
  bin\YourPlugIn\HomeSeerAPI.dll
  bin\YourPlugIn\HSCF.dll
  bin\YourPlugIn\Hspi.dll
  bin\YourPlugIn\Hspi.dll.config
  bin\YourPlugIn\Hspi.pdb
  bin\YourPlugIn\Scheduler.dll
  bin\YourPlugIn\SomeOtherAssembly.dll
  bin\YourPlugIn\YetAnotherAssembly.dll
  ```

  This directory structure enables plugins to avoid version conflicts with other plugins that depend on different versions of the same dependencies. Installing your plugin into HomeSeer is as simple as copying the entire output directory (`/bin/(debug|release)`) into your HomeSeer directory (typically `C:\Program Files (x86)\HomeSeer HS3`).

2.6.2 3/10/2017

2.6.2.1 Extension

- Long story short: **YOU MAY HAVE TO UNINSTALL THE OLD PLUGIN AND INSTALL THE NEW PLUGIN TO CONTINUE GETTING UPDATES** I thought I configured the original extension incorrectly in the Visual Studio marketplace, and Microsoft doesn't provide a way to change the setting I thought I messed up. So I removed the old extension from the gallery. That's why you get a 404 when you to go https://marketplace.visualstudio.com/items?itemName=thealexdresko.HomeSeerTemplates. The new location's forever home is at https://marketplace.visualstudio.com/items?itemName=thealexdresko.HomeSeerTemplates-18379 (arrrggg). I'm pretty frustrated by the whole experience and – worse – it doesn't even look like the setting makes a difference. I'm trying to automate publishing of the VSIX to the marketplace from the CI build, but I just don't think it's possible anymore.

2.6.2.2 Contributers

- Contributing to the HSPI .sln now requires Visual Studio 2017 (You can install install the templates and consume the HSPI nuget package on VS 2015)
- All C# builds in HSPI are validated against a fairly strict set of static code analysis rules in an effort to automate code quality checks.
- A Resharper "Code Cleanup" setting called "HSPI" helps to ensure all code follows the same standards.
- `HspiBase` was updated to eliminate many of HS's poor naming conventions from `IPluginAPI`. It's quite a bit easier to understand the interface now.
- HSPI: Converted all string concatenation to interpolation. All the manually constructed HTML is much easier to read now.
- HSPI: massive Resharper cleanup on the entire HSPI codebase. The code is so much easier to read.
- HSPI: removed some `ref` code that simply wasn't necessary.
- HSPI: First external contribution everrrrr! https://github.com/alexdresko/HSPI/pull/43 Thanks @dmurphy
- Contributers: Testing the extension is now much easier. A super fancy Powershell script does all the work of copying the ".Dev" code into the project template, and makes sure requisite nuget packages are updated in the correct location. You just press F5 and an experimental instance of VS will start. From there, you can "File > New Project" and verify everything works.

2.7 Meta

HSPI stands for HomeSeer Plugin Interface, and also happens to be the mandatory class name that must be used when creating a HomeSeer plugin. In hindsight, it's probably not the best project name I ever came up with.

test edit