ZeeVee Zyper Management Platform Plugin Review

**Recommendations**

* ~~The PluginInfo Header currently has a ‘ShowDebug’ property which is not a supported property and it doesn’t look like it’s being used anywhere else so I would go ahead and remove it. Here is a list of what can/should go in the PluginInfo Header:~~

| **Name** | **Type** | **Required?** | **Description** |
| --- | --- | --- | --- |
| Name | String | Yes | Name of the plugin |
| Version | String | Yes | Current plugin version |
| Id | String | Yes | Unique identifier for the plugin. Must not conflict with other installed plugins |
| Description | String | No | A brief description of the plugin |
| BuildVersion | String | No | Current code iteration tracking |
| Author | String | No | Name or Contact Info of the author |
| Manufacturer | String | No | Name of manufacturer the product the plugin integrates |
| Model | String | No | Model name of the product the plugin integrates |
| IsManaged | Boolean | No | Add the plugin to the managed inventory of the design |
| Type | Reflect Type | No | Reflect reporting type |

~~Note that typically Version will be of the form x.x whereas BuildVersion will be of the form x.x.x.x. Your Version is currently of the form x.x.x.x. Our VS Code compiler increments the BuildVersion.~~

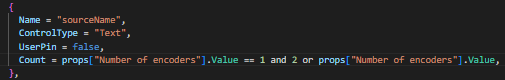
~~For a Certified plugin, the PluginInfo Header needs IsManaged and Type where IsManaged is equal to true and Type is equal to ‘Reflect and Reflect.Types.Other or 0’ like this:~~



* ~~Currently, almost all of the controls (besides the Device Status and Current\_Input controls) have only an input control pin. Many of these, if not all, should probably also have an output pin to give users more freedom in how they use the plugin (for example, someone may want to drive some external logic with the CEC ON and CEC OFF buttons). Setting the PinStyle to “Both” will enable both an input and output pin.~~
* ~~There are a few functions (such as ZEEVEE.PULSE() and ZEEVEE.SEND()) that are defined but never used. Consider removing these.~~
* ~~I would suggest any “setup” of the plugin including IP Address, Port, Username, and Password controls (see bullet point 1 of ‘Requirements’) be put on the first page that opens when double-clicking the plugin.~~
* While it may be difficult to fix this given how the plugins actually get drawn under the hood, just be aware that if the number of encoders and the number of decoders are maxed out at 100 each, it becomes almost impossible to navigate the pages as it tries to draw everything. You may consider coming up with different methods to get all of this information displayed.

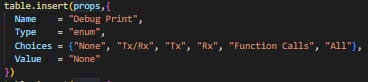
**Requirements**

* IP Address, Port, Username, and Password are currently set in the properties but in general, these should be set in runtime via text controls on the plugin. We like to avoid having a ‘Connect’ button and instead, when these controls are changed, a connection attempt should be made.
* ~~Currently at the start of the ZEEVEE.INIT() function, the status is being set to "INITIALIZING". In a certified plugin, our status component should never be set to “INITIALIZING” or “NOTPRESENT” as these states are controlled by the Core.~~
* ~~If you have the “Number of decoders” and “Number of encoders” properties both set to 1 (default) and you try to use the controls on the ‘One-Way Commands’ page, you get a “Property ‘1’ does not exist on Control” error in the debug output. This happens because in the GetControls() function, the Count of these controls is determined by these properties and when the count is 1, you no longer have a table of controls but instead a single control. So for example, if “Number of decoders” is set to 1, then Controls.displayName[1] doesn’t actually exist (and would have to be reference by Controls.displayName instead) which is why you get these errors. There are a few ways around this:~~
  + ~~Hardcode the count of these controls to the max value for the “Number of decoders” and “Number of encoders”. Since GetControlLayout() is what actually places these controls on the plugin, this won’t cause any change in appearance.~~
  + ~~Implement logic in runtime to determine whether to use Controls.<control name> or Controls.<control name>[index] depending on if there’s one or more controls.~~
  + ~~Implement logic in GetControls() where if the value of these properties is 1, ensure more than one control exists, so you can index them properly in runtime. Maybe something like this (if the value is 1, simply set it to 2, otherwise leave it as what’s set in the properties):~~



~~I’ll also note that currently if you set the “Number of encoders” property to 2 and the “Number of decoders” to 1, the plugin immediately goes into this error on initialization and all controls are greyed out. Addressing the above should fix this.~~

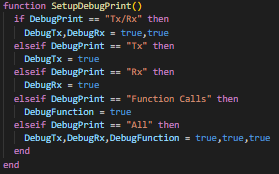
* ~~If the number of encoders and decoders properties are maxed out, on initialization of the plugin, you get a ‘Max execution limits exceeded’ error and all controls are greyed out. Additionally, as mentioned in the recommendations above, maxing out these controls almost makes the plugin unusable anyways as it tries to draw all of the controls.~~
* ~~Another requirement of certified plugins is a Debug Print property that is visible when the Show Debug property is set to Yes. For QSC-authored plugins, we typically add a property called “Debug Print” with the following choices: "None", "Tx/Rx", "Tx", "Rx", "Function Calls", "All".~~

~~~~

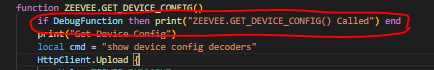
~~Then in the RectifyProperties() function found in the basic plugin framework, you’d include some logic to have it only show when the Show Debug is set to Yes.~~

~~~~

~~We then include a function in runtime that looks like this (as well as defining the DebugTx, DebugRx, and DebugFunction variables to be false):~~

~~~~

~~Lastly, you can use the global variables set in the above function to print useful debugging information. For example, at the top of each function, you could include a line like this:~~

~~~~

~~This way, if Show Debug is set to ‘Yes’ and Debug Print is then set to ‘Functions’, it will only print these function call statements. You can/should do the same with incoming Rx data as well as outgoing Tx data.~~

~~For certified plugins, we require this debug paradigm.~~