## Universal Import/Export API

### Description

Currently every developer must create their own import/export functionality to manage Custom Device configurations. If a universal format were developed, this would mean that this functionality was “free” to the developer. It would also make it much easier to script out the creation of a Custom Device. Dan Eaton has developed a working tool for this already, so integrating his tool in to this new scope would be all that’s required.

### Time Estimate

|  |  |
| --- | --- |
| Task Name | Hours |
| Code Review of Existing Solution | 4 |
| API Documentation / MOdification | 8 |
| Testing | 8 |
| VIPB Build | 2 |
|  |  |
| Total | **22** |

### Risks

#### API Version Management

We would need to ensure the API supports files created in older versions for the API.

LEVEL: MEDIUM

Mitigation: Extra care should be spent when designing the API to include good version support

#### Custom Device Version Management

The API should also support Custom Device versioning, so that configurations exported by previous versions of the Custom Device can be read by the most recent Custom Device version

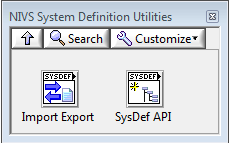
LEVEL: MEDIUM

### Code Review of Existing Solution

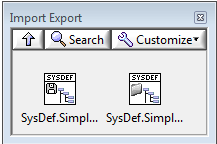
Existing solution is based in [VeriStand Addon System Definition Utilities](https://github.com/NIVeriStandAdd-Ons/VeriStand-Addon-System-Definition-Utilities) (Dan Eaton).

This tool uses VeriStand .NET API to import and export Custom Device configurations. This is currently just an API that can be used in different parts of the code to add this functionality to your Custom Device.

When built and installed, this API will be added to your functions palette as NIVS System Definition Utilities.

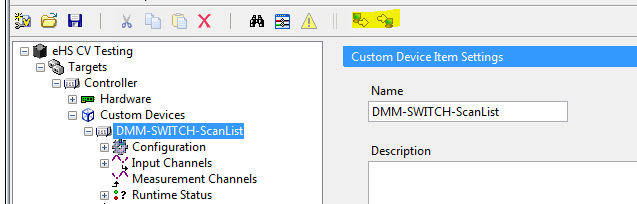


The most relevant VIs are the ones in the Import Export subpalette.



The ones in the SysDef API palette are used by these 2 VIs and can be used to access partial features separately.

First tried to add the Import Export VIs as buttons when selecting the Custom Device (Main Page VI).



When pressing the Import button all properties are loaded correctly from the configuration file for all the sections and created channels, but won’t add any dynamically created channels. After closing and opening again the system definition file the dynamically created channels appear.

After discussing the issue with Dan, he told me he had only used his API from the Initialization VI. I tried it from the Initialization VI and it works perfectly fine, all the dynamically created channels appear immediately.

Changes have to be done to the API to add the functionality to call it from the Main Page VI.

\*It is important to keep in mind that calling the Import VI from the Initialization VI means creating a new Custom Device with the loaded configuration. Calling the Import VI from the Main Page VI means merging the loaded configuration with the existing Custom Device.

The options to add this functionality to the Custom Device Template are the following:

Export

* Toolbar button when selecting the Custom Device (Main Page VI)
* Right-click when selecting the Custom Device (Main Page VI)

Import

* Adding a Pop Up asking the user if he wants to load a configuration or create a new Custom Device (Initialization VI)



* \*\*Toolbar button when selecting the Custom Device (Main Page VI)
* \*\*Right-click when selecting the Custom Device (Main Page VI)

\*\*These 2 options imply modifying the API.

### API Modification

These are the changes that need to be done to the API to allow the “merging” functionality to call the Import function from the Main Page VI.

A conditional structure needs to be added. When OFFLINE == TRUE the Import API uses original Dan’s code. When OFFLINE == FALSE, native VeriStand have to be used instead of VeriStand .NET API.

