Catch Up From the Last 2 Days

• U-E-F-I versus YOU-FEE?
• Agent of SHIELD provides not help
  https://www.youtube.com/watch?v=9lc95nXK WMM (0:8)
• And if YOU-FEE, then
  ACK-PEE?
  
  – https://www.youtube.com/watch?v=A0seUVAC09 E (1:40)
‘Speaker’ Bios
Vincent Zimmer

Vincent Zimmer is a Senior Principal Engineer with Intel. He joined Intel in 1997 and has been working on EFI / Framework / UEFI / PI along with touching TCG and other stds groups since 1999. He lives in the Seattle area and can be reached via various paths –

vincent.zimmer@intel.com or 425-881-4874
or https://twitter.com/vincentzimmer?lang=en
or https://www.linkedin.com/in/vzimmer or ....
Gahan Saraiya

Gahan is a skilled Platform Lead at Intel, working within the Software and Advanced Technology Group. With over four years of experience, he has been dedicated to the development and implementation of firmware configurations that support the expansion of automation infrastructure. Gahan's contributions have consistently made a positive and lasting impact on the organization during his tenure.

Know more about Gahan
Christine Chen

Christine is the software development engineer at Intel, has been working on the Edk2 BaseTools related development and support works since 2020. She has designed and developed BaseTools FMMT python tool, and has been participated in BaseTools incremental Build project, BaseTools C tool python conversion work. She is focus on BIOS configuration enhancement in BaseTools build system currently.
Agenda

- Introduction
- Evolution
- Present Practice
- Next Generation of Firmware Configuration
- Questions
Introduction
Introduction

• Growing requirements of various application demands different firmware flows
• Drilled down to classify with conditional flow
Evolution
Evolution

• Boot Flow Control with Firmware Configuration
• HII Implementation
Boot Flow Control with Firmware Configuration

• EFI Variables act as control variable for firmware boot flow process

• Fulfills need of simpler to complex boot flow context, i.e.
  – boot priority ordering
  – security configuration
  – overclocking of device
HII Implementation

HII establishes Infrastructure between external entity to firmware configuration

Control variable (EFI Variable) mapped through various component of HII

vfr/hfr – customized syntactical language to construct user interface
uni – string representation of Identifier mapped in vfr/hfr
Header file *.h – variable structure and GUID
Source file *.c – HII Protocols

www.uefi.org
Present Practice
Present Practice

• Non-Recommended Practices
• Runtime population of form representation
• Tools and Tech
Present Practice – Non-Recommended Practices

• Action callbacks associated with form representation
## Present Practice - Tools and Tech

<table>
<thead>
<tr>
<th>TOOL/TECH</th>
<th>REFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RedFish</td>
<td>DMTF’s Redfish®</td>
</tr>
<tr>
<td>UFFAF</td>
<td>UEFI Firmware Foundational Automation Framework</td>
</tr>
<tr>
<td>USF YAML</td>
<td>yaml_boot_configuration</td>
</tr>
<tr>
<td>FDT</td>
<td>Flattened device tree</td>
</tr>
<tr>
<td>FSP UPD</td>
<td>FSP UPD</td>
</tr>
<tr>
<td>DFCI</td>
<td>Mu DFCI</td>
</tr>
</tbody>
</table>
## Present Practice - Tools and Tech

<table>
<thead>
<tr>
<th>Tool</th>
<th>User Visible Data Format</th>
<th>Firmware Implementation</th>
<th>End-User Programming Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>RedFish</td>
<td>json</td>
<td>Edk2 Advanced Feature Package</td>
<td>Python</td>
</tr>
<tr>
<td>UFFAF</td>
<td>Xml, json</td>
<td>Intel Advanced Feature Package</td>
<td>Python</td>
</tr>
<tr>
<td>USF YAML</td>
<td>yaml</td>
<td>BaseTools</td>
<td>Python</td>
</tr>
<tr>
<td>FDT</td>
<td>dts</td>
<td>Device tree compiler (DTC)</td>
<td>C</td>
</tr>
<tr>
<td>FSP UPD</td>
<td>Bsf, yaml</td>
<td>Configuration editor</td>
<td>Python</td>
</tr>
</tbody>
</table>
UFFAF Execution Flow

- **Action Trigger**
- **Write Request to Physical Memory Address**
- **Trigger Software SMI**
- **Memory Map of Request-Response Cycle**
- **Interrupt**
- **Read and Validate Request Buffer**
- **Is valid Request**
- **Process Request and write response status to Physical memory**
**UFFAF**

**UEFI Firmware Foundational Automation Framework**

---

**UFFAF Execution Flow**

- **Build**
  - Build Start
  - Source Compilation (compilation of form representation files with VfrCompiler)
  - Trigger post build process [after source compilation]
  - Firmware Binary

- **Post Build**
  - Driver script to parse IFR record intermediate files

- **Records**
  - IFR intermediate record (*.i)

- **User Action**
  - Action Trigger
  - Configuration Modify
  - New Firmware Binary

---

www.uefi.org
YAML and USF

7. YAML Format Boot Configuration

7.1. Introduction

This document describes the format of the YAML based boot setting file used to specify features, settings, and tool display information to the Intel Firmware Support Package (FSP) and tools like Config Editor. It further describes the structure and content of YAML format files, which can simplify the configuration of a static binary and during dynamic boot process.

7.2. Target Audience

This document is intended for person creating and using firmware images. It is most likely of interest if a developer needs to create customized feature and expose binary settings for delivery to customers, or that a new SOC device is being enabled.

7.3. Configuration Description (YAML) Explained

The declarations required to build the configuration data blobs and the header files are provided in a configuration description file. This file uses the YAML syntax.

YAML (https://yaml.org/) is a data serialization language designed to be human-friendly and work well with modern programming languages. A quick syntax reference can be found here - https://yaml.org/reftcard.html

Configuration YAML files will be processed by configuration tools like GenCfgData, CfgDataTool, CfgDataStitch in order to generate the configuration header files and binary blobs.

The main platform configuration file is specified in CfgDataDef.yaml. Please note that you may find many YAML files. However, only CfgDataDef.yaml is the primary file used for the platform configuration, and other sub YAML files will be included by the primary YAML file to provide component specific configuration.

An example configuration file in YAML syntax is provided in Figure 15 below.
USF YAML focus on the VFR and StructurePCD BIOS Configuration method, expect to unified these source files into YAML type. Based on this expectation, the evolution process in shown as left diagram. For the final flow, all the Bios Configuration related information are from YAML file. YamlCompiler tool and YamlToVPD tool will be used to generate the binary which will be built into the BIOS Image and be consumed during boot time.
YAML Config Tool for Intel® FSP UPD

YAML UPD Editor Features:
• Read FSP binary information
• Allow patching any BIOS/IFWI image containing FSP UPDs
• Read YAML config format while Boot Setting File (BSF) backward compatible
• Bit format FSP support instead of bytes
• Modifying BSF parameters and export loadable delta files
• FSP 1.x and 2.x format backward compatible
• Search function

Next Generation of Firmware Configuration
Next Generation of Firmware Configuration

• Uniting Configuration syntax usage to YAML configuration

• Industry wide standard language syntax with advantageous integration across tech-stack
Other events

Flattened devicetree (FDT)

- devicetree.org

See more on FDT at https://2023ocpglobal.fnvirtual.app/a/schedule/
Thanks for attending the UEFI Fall 2023 Developers Conference & Plugfest

For more information on UEFI Forum and UEFI Specifications, visit http://www.uefi.org