presented by

Using Capsules for Firmware Configuration Update

Spring 2019 UEFI Plugfest

April 8-12, 2019

Presented by Zachary Bobroff (AMI)

Agenda

- Introduction
- History of BIOS/UEFI FW Setup
- UEFI Capsules
- Putting it Together
- Security Considerations
- Call to Action

Introduction

New Capsule Type in UEFI 2.8

- Version 2.8 of the UEFI allows for the exchange of configuration data between the operating system (OS) and the UEFI firmware
- OsIndications flags were extended so that the OS can request the export of configuration data
  - Please review EFI_OS_INDICATIONS_JSON_CONFIG_DATA_REFRESH flag from section 8.5.4
  - Note: OsIndications has already provided a method for the user to inform the firmware to enter the setup browser on the next reboot
- The OS can also provide a capsule back to the firmware to update configuration data of the UEFI firmware

What is Configuration Data?

- The UEFI spec definition for configuration data is very abstract, that allows for many types of configuration data to be processed
- The most obvious use case allows the OS to read the current UEFI Human Interface Infrastructure (HII) settings and provide updates when needed
- OS can provide visual display or other built in methods to read and update the settings on the next reboot
  - Allows for clean integration of OS settings
  - Can be very useful for headless systems that do not have a standard method to enter a setup browser
- With any new powerful feature, certain security measures must be implemented!

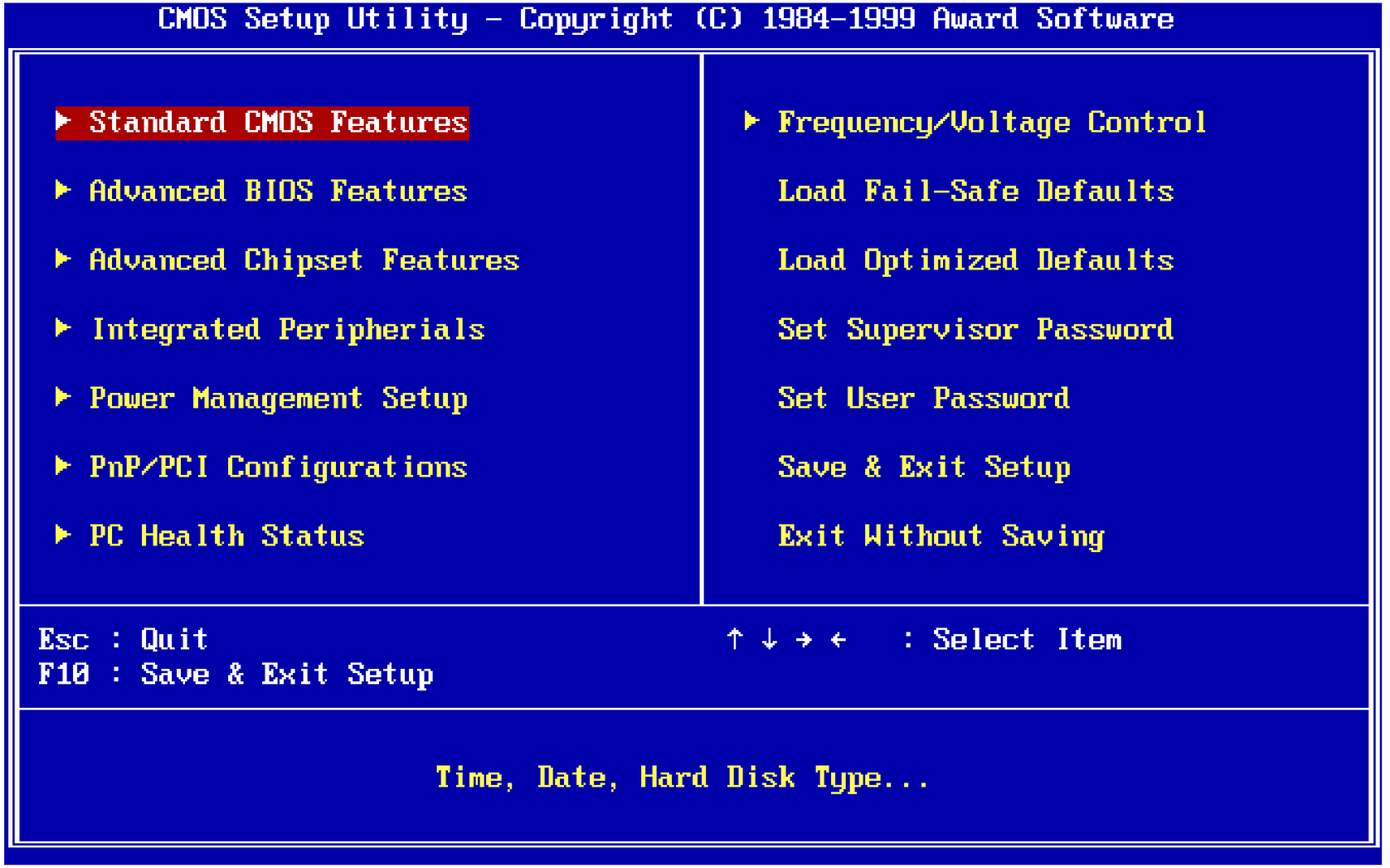
History of BIOS/UEFI Firmware Setup

Legacy BIOS Setup

- BIOS configuration data has existed since the early days of x86 firmware
- Every implementation has been proprietary, with different look and feel, and provided different levels of features and functionality
- Plug-in cards even provided their own pre-boot configuration interface that added further differentiation in the configuration realm

Pictures of Old BIOS Setup

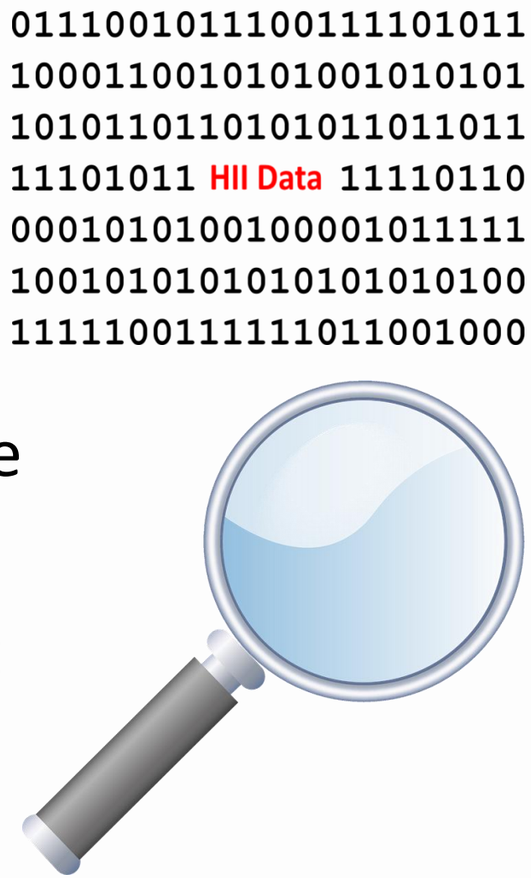
Pictures of Old BIOS Setup



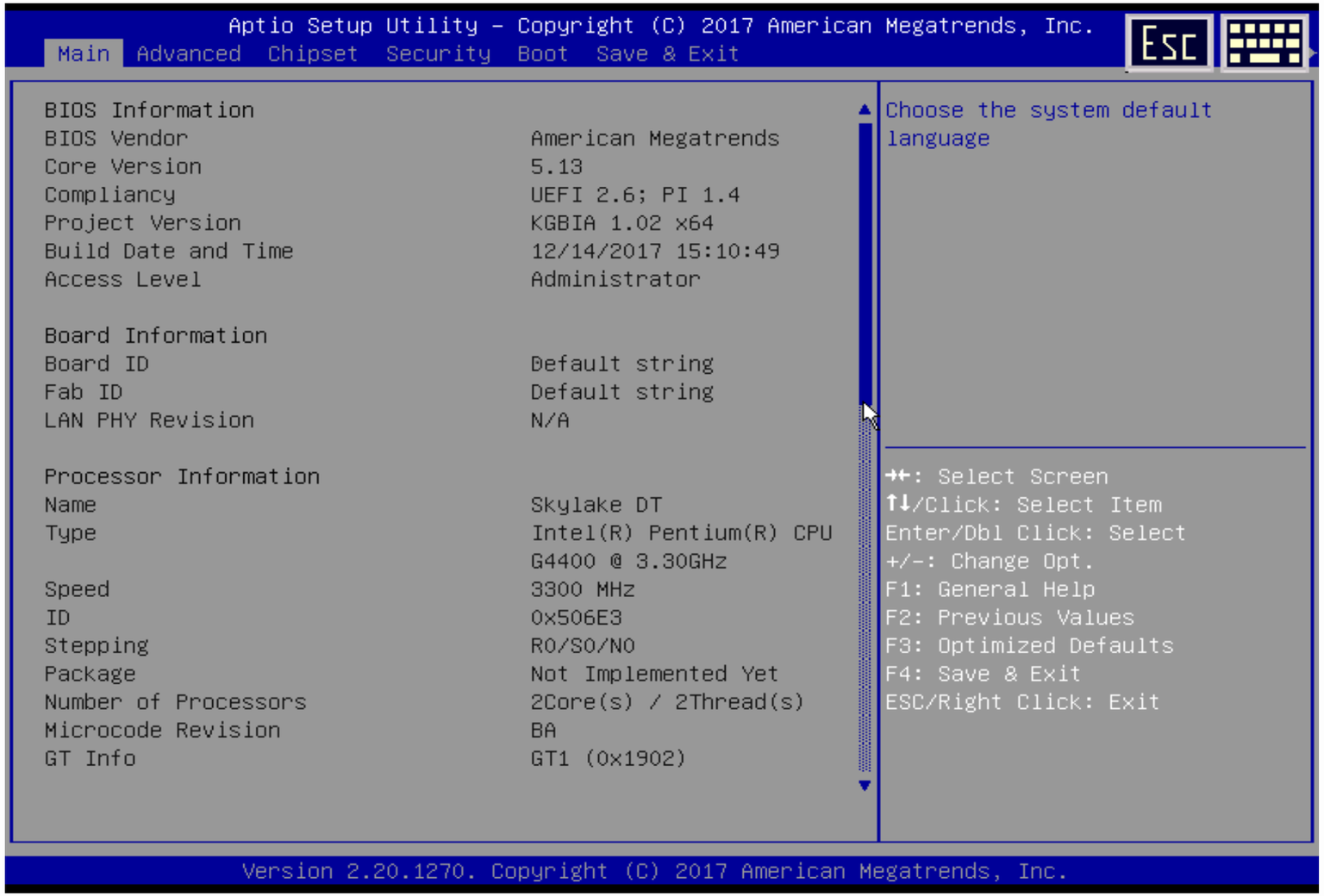
Citation: https://en.wikipedia.org/wiki/BIOS#/media/File:Award_BIOS_setup_utility.png

UEFI Firmware Configuration

- When UEFI was introduced, the specification architects wanted to provide a common set of features for configuration
- Data from firmware is posted into the HII in packs
- Firmware provides a browser that reads the HII data and provides a method to interactively configure these settings
- Still allows for a highly customizable look and feel, but common set of features and functionality



Modern Picture of UEFI FW Setup Browser



HII Extensibility

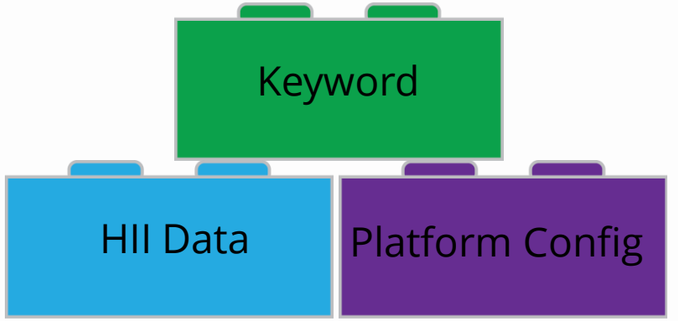
- HII is highly extensible and allows plug-in card vendors or 3rd party binary providers to add content into the HII database

- One HII database allows one setup browser to provide a unified look and feel for all platform settings

- HII still allows for many of the demanding needs of firmware configuration, but pushes the industry to use a common set of standard methods

UEFI Adds High Level Abstractions

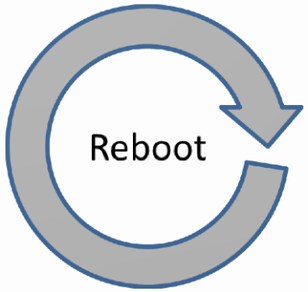
- The UEFI specification has evolved through the years and added several high level abstractions
- One key addition was in UEFI 2.5 and is the EFI_CONFIG_KEYWORD_HANDLER_PROTOCOL in section 35.3 which abstracts the platform configuration data
  - Can allow for a script based configuration of a platform
  - You no longer need to worry about what driver/code publishes a configuration knob
  - You can just use the generic protocol that talks to a variety of platform level APIs to configure anything on the platform



UEFI Capsules

UEFI Capsules

- UEFI Introduced Capsules in UEFI 2.0
- Capsules were mainly used for passing binary blobs of data between the OS and the firmware
  - Capsules could be used to pass information from other sources too
  - UpdateCapsule is a runtime service, but most capsule processing is typically available only prior to ExitBootServices()
- Capsule use remained limited in the early days of UEFI. They were mainly used for some proprietary data passing

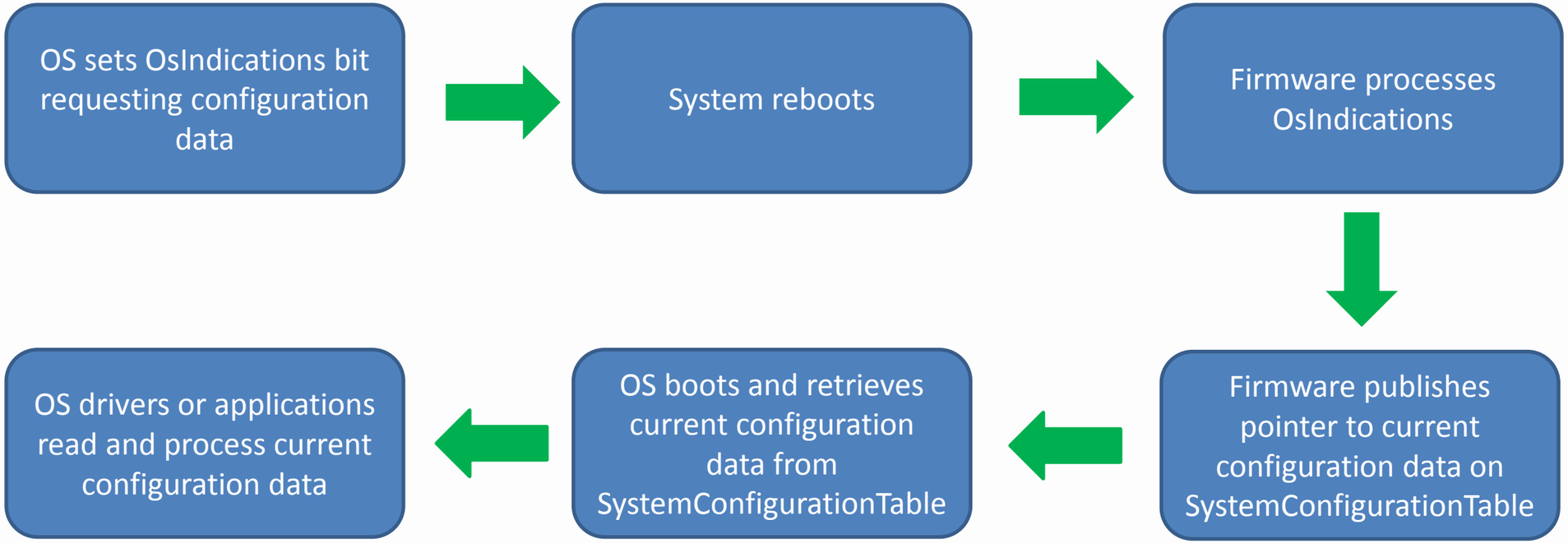


Capsule Usage Expands

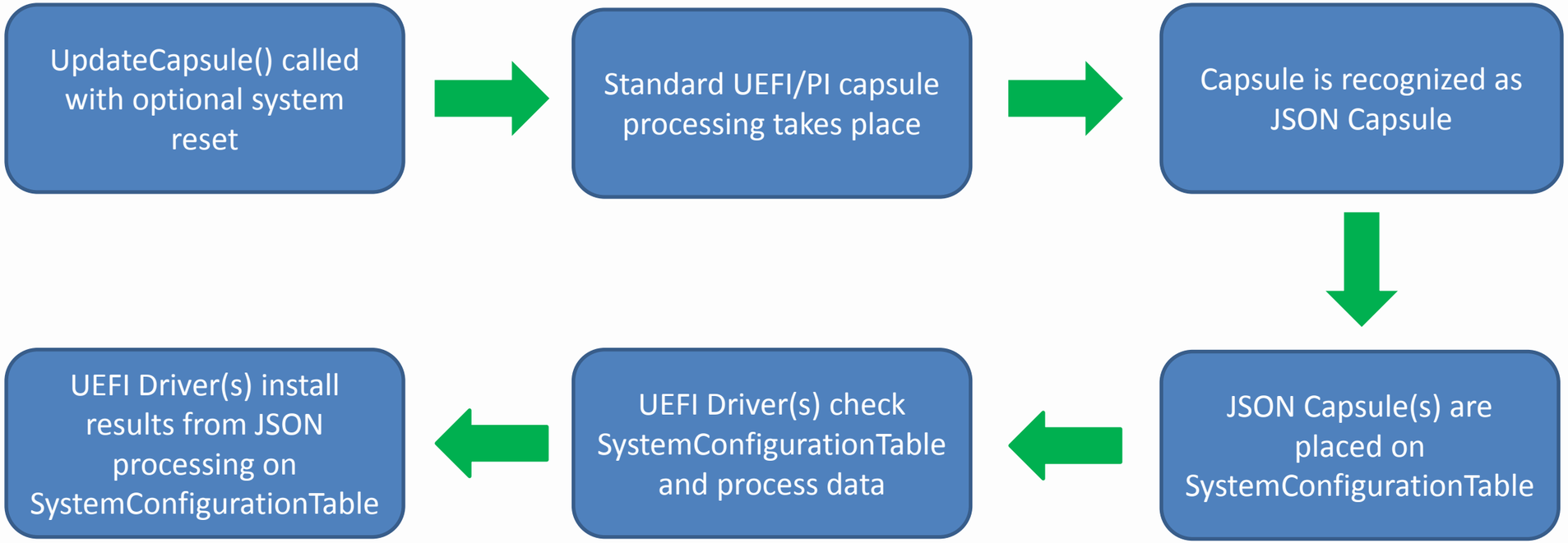
- Capsule usage started to expand greatly with the advent of SP NIST 800-147
- NIST 800-147 mandated that the most secure method for updating of the platform firmware was on a reboot
- Best method to pass the firmware image over a reboot was a capsule!
- Capsules are commonly used for firmware update methods such as ESRT or using FMP for plug-in card firmware updates

Putting it Together

Current Config Data Export Flow



JSON Config Data Capsule Flow



JSON Config Data Capsule Considerations

- If multiple JSON config capsules are provided, an array of pointers is installed on the SystemConfigurationTable
- JSON schema is not defined which allows firmware vendors and plug-in card vendors to provide solutions to meet whatever their needs may be
  - "It is expected that particular drivers have the specific knowledge of the JSON schema used in the payload so that they can describe system configuration data in JSON then install to the EFI System Configuration Table", UEFI Spec section 23.5.2
  - Allows for more flexibility than just HII configuration data

```
#pragma pack(1)
typedef struct {
UINT32 Version;
UINT32 TotalLength;
// EFI_JSON_CONFIG_DATA_ITEM ConfigDataList[];
} EFI_JSON_CAPSULE_CONFIG_DATA;
#pragma pack ()
```

```
typedef struct {
UINT32 ConfigDataLength;
UINT8 ConfigData[ConfigDataLength];
} EFI_JSON_CONFIG_DATA_ITEM;
```

UEFI Driver/OEM Considerations

- Drivers can use the CreateEventEx using the GUID EFI_JSON_CAPSULE_DATA_TABLE_GUID to be notified when a JSON capsule is installed on the SystemConfigurationTabale
- Driver writers should use standardized names for their keywords and work with the UEFI forum and DMTF to leverage industry standard keywords when possible
  - https://uefi.org/configNamespace
- Driver writers are encouraged to use the keyword handler protocol to simplify their code for processing JSON configuration capsules and use similar keywords in their JSON data structures
- OEMs should also make use of industry standard keywords for setup configuration to allow end customers to manage non-homogenous hardware simply and effectively

Security Considerations

Capsule Trust

- Most common use of a UEFI Capsule today is for firmware upgrade
- According to NIST 800-147 all firmware upgrades must be digitally signed and verified
- Signature offers a safe level of security to verify the capsule is authentic and unmodified from the origin
- If a capsule is created within the OS, how can it be trusted?

Malicious Configuration Capsule Possibilities

- Malicious capsules may try to:
  - Change firmware settings to put the system in an unbootable state
  - Change firmware settings related to security components
  - Change firmware settings to slow down or delay booting
  - Change firmware settings that remove or add a boot option

How do you mitigate these issues for configuration capsules?

Mitigating Malicious Issues

- First step in mitigating security issues is to limit your attack vector
- Only expose configuration settings valuable to this type of service
  - UEFI SecureBoot, TPM and other security device enable/disable options can still be exported, but without capsule's ability to change their settings
- Can these capsules be signed and trusted?

Config Data Capsule Signing

- Signature database (DB) can be extended with additional public keys
- The private key pair can be used to sign configuration capsules which could then be trusted
- In a corporate environment this is fine because Network Admin should be the central control for capsule production and signing
- If in non-corporate environment, can users be trusted not to leave the private key on their computer?
- A plug-in card trying to use this service via the OS driver or application adds further layers of complexity

Call to Action

Call to Action

- Firmware configuration capsules open up exciting new possible features
- These features will be powerful so they must be secure
- All UEFI stakeholders should see how to best use firmware configuration capsules and start a conversation on best methods to secure their usage

Thanks for attending the 2019 Spring UEFI Plugfest

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

presented by