presented by

Building Better Firmware Experience

An OEM Perspective

UEFI Summer Summit – July 16-20, 2012

Presented by

Samer El-Haj-Mahmoud

(Hewlett-Packard Company)

Agenda

- Introduction
- HII Configuration Forms
- Controller Name
- Driver Health Protocol
- Driver Diagnostics Protocol
- Firmware Management Protocol
- More Information
- Summary / Q&A

Introduction

Introduction

- The UEFI Specification offers a rich set of Protocols and features that can be used to provide a better pre-boot user experience
  - Standard APIs that enable common tools
  - Centralized console management
  - Localization of almost every user visible text

Introduction

- Many of these interfaces are “optional”
  - There seems to be a trend in focusing on “required” interfaces
- Providing the best user experience requires collaboration among all stake holders
  - UEFI drivers and applications (from IHVs, ISVs, OSVs) directly impact the user-visible pre-boot experience
  - Inconsistencies in firmware implementation across vendors can yield an overall poor user experience

HII Configuration Forms

HII Configuration Forms

- Drivers should include carefully designed HII Configuration Forms
  - Main touch point for user interaction with the device configuration
  - Multiple devices, as well as the System Firmware settings, present in the same browser UI
    - Uniformity is essential for a better user experience

HII Configuration Forms

- HII Forms should be comprehensive, offering all user visible configuration settings
  - Complete replacement for legacy Option ROM configuration utilities
  - HII configuration must apply to both UEFI and CSM modes of operations (as necessary)
    - Settings that are applicable only to CSM boot (such as PXE Boot enable/disable) must be clearly indicated as so (in help text or a separate Form)

HII Configuration Forms

- All user interaction through HII Forms
- Let the HII browser handle all input/output
  - Do not directly wait for user input
  - Do not display any content directly to the console
  - Pop-up messages must be avoided
    - Use EFI_IFR_INCONSISTENT_IF and EFI_IFR_NO_SUBMIT_IF op-codes or Modal Forms (EFI_IFR_MODAL_TAG) instead

HII Configuration Forms

- Implement EFI_HII_CONFIG_ACCESS_PROTOCOL
  - Even if the HII Forms are limited to read-only settings
    - This enables extracting a snapshot of the device configuration settings, for offline viewing or remote system management

HII Configuration Forms

- Localize HII Forms strings
  - All strings must be available in all supported languages
  - Work with OEMs for list of languages
    - English (en-US) is a must
    - Japanese (jp-JP) and Simplified Chinese (zh-Hans) are commonly required

HII Configuration Forms

- Minimize callbacks
  - Callbacks are not available for runtime/offline/remote configuration utilities
  - Use callbacks only when absolutely required
    - To update real-time data (fan speeds, voltages, etc..),
  - Do not use callbacks to:
    - Modify how items are displayed
    - Validate question data
      - Use IFR calculation and comparison op-codes, and question minimum / maximum values instead

HII Configuration Forms

- Do not make assumptions on how the Forms will be displayed
  - HII browsers and console displays vary considerably in capabilities
    - Text UI, rich graphics UI, CLI, etc…
  - Do not format tables or horizontal/vertical spacing
    - Different fonts and screen resolutions will yield different results
  - Let the HII Browser engine worry about the formatting. Focus on content

HII Configuration Forms

- Use EFI_IFR_FLAG_RESET_REQUIRED to indicate a reboot is required
  - Never reboot the system directly

HII Configuration Forms

- Support default values for all settings
  - At a minimum, support EFI_HII_DEFAULT_CLASS_STANDARD
  - This enables the Setup Browser “Reset to Defaults” operation to work for any device
  - It also enables resetting “all device and platform settings” to default

HII Configuration Forms

- Support UEFI Configuration Language for mapping of HII settings
  - Very useful for programmatically parsing HII Forms and identifying individual settings
  - Easy to support by adding a special language to the HII String packages

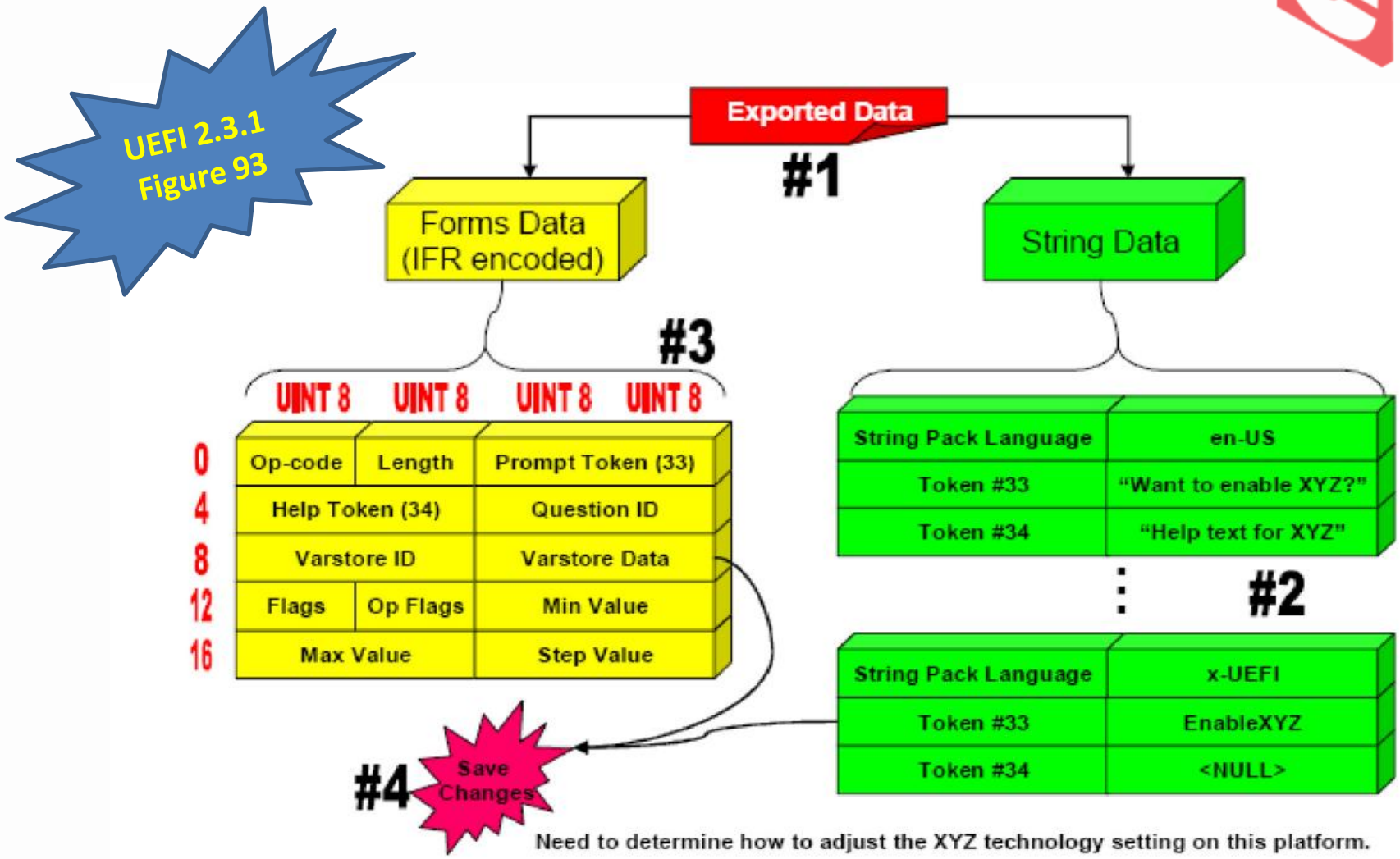
```
#define UEFI_CONFIG_LANG ”x-UEFI”
```

  - Defined in the UEFI 2.3.1 specification
    - Sections 28.2.6.1, 28.3.6.1 and 28.2.11.2.

HII Configuration Forms

- Assign unique “keyword” strings in x-UEFI language for each setting
  - Each EFI_IFR_QUESTION *Prompt* string must be unique within the HII FormSet
  - Each EFI_IFR_ONE_OF_OPTION *Option* string must be unique within the list of options for that question
  - Platform firmware and tools can identify individual settings in the HII FormSet using their known “keyword” strings

HII Configuration Forms



HII Configuration Forms

Forms Package (VFR)

```
oneof varid     = IfrData.TechnologyXyz,
    prompt      = 33,
    help        = 34,
    option text = 35, value = 2, flags = RESET_REQUIRED;
    option text = 36, value = 0, flags = RESET_REQUIRED;
    option text = 37, value = 1, flags = RESET_REQUIRED | DEFAULT;
endoneof;
```

String Package (en-US)

```
33  =  “Technology XYZ Setting”
34  =  “This setting allows the user to ..”
35  =  “Always Enabled”
36  =  “Always Disabled”
37  =  “Auto Configuration”
```

String Package (x-UEFI)

```
33  =  “TechnologyXyz”
34  =  “”
35  =  “Enabled”
36  =  “Disabled”
37  =  “Auto”
```

Shell> Set “TechnologyXyz” = “Enabled”

Controller Name

Controller Name

- Controller Name strings must be chosen carefully
  - This is directly visible to end users
    - Setup, boot menus, pre-boot applications, etc..
  - Users should be able to easily identify the device using this string alone
  - Present your brand!

Controller Name

- Devices can provide their Controller Name using:
  1. EFI_COMPONENT_NAME2_PROTOCOL
  2. HII FormSet Title
- Both strings should be identical
  - This ensures consistent user experience in different applications/views

Controller Name

- Use marketing/branding names
  - Include manufacturer and model number
  - Adjust the strings for OEM rebranded devices
  - The strings should be descriptive, yet not very lengthy (to avoid wrapping on multiple lines)
    - 75 characters or less seem reasonable

Controller Name

- Describe representative device characteristics
  - Device type (Storage, Ethernet, FCoE, etc..)
  - 1Gb vs. 10Gb network adapters
  - Number of ports
  - Etc…

Controller Name

- For multi-port devices, include the port number
  - Match the device hardware labeling


Port 1 : HP Ethernet 1Gb 4-port FLR Adapter


Port 2 : HP Ethernet 1Gb 4-port FLR Adapter


Port 3 : HP Ethernet 1Gb 4-port FLR Adapter


Port 4 : HP Ethernet 1Gb 4-port FLR Adapter

Controller Name

- Examples:

```
Port 3 : HP NC1234 PCIe Dual Port 1Gb Adapter
```

```
Port 6 : HP NC1234 Flex 10 BL-x Adapter
```

```
Company X Storage Controller XYZ
```

Controller Name

- Avoid cryptic information, such as
  - Hard-to-decode Product IDs
  - Lengthy serial numbers
  - PCI Bus:Dev:Fun addresses

```
MM0300GBLFF 7VF25NHD
```

```
Mass Storage Device 20100311000000010FA
```

```
HP Ethernet 1Gb 4-port Adapter (2:0:0)
```

Controller Name

- Avoid MAC addresses
  - Not user friendly
  - Not very useful in identifying the device
  - Users can always get the MAC address from the device configuration info (HII, etc…)

```
HP Ethernet 1Gb 4-port Adapter – 1C:42:6A:B2:5A:2B
```

Controller Name

- Avoid platform specific information such as PCI/PCIe Slot #, “Embedded”, “Onboard”, etc…
  - These are platform specific strings that should be provided by the system firmware, not the device firmware

`HP Ethernet 1Gb 4-port Adapter (Slot 3)`

`Embedded HP Ethernet 1Gb 4-port Adapter`

Controller Name

- Provide localized strings
  - Include all languages supported by the device's HII forms to ensure best user experience when localization is enabled
- At a minimum, provide English strings
  - Match both "en" or "en-US" language codes to ensure interoperability
  - A future UEFI spec update will provide clarification on Best Matching Language algorithm

Driver Health Protocol

Driver Health Protocol

- Drivers should report their health status using EFI_DRIVER_HEALTH_PROTOCOL
  - Implement only if the device may require repair from a failed state
  - Used as an alternative method to outputting status/warning/error messages directly to the console

Driver Health Protocol

- Enables Status and Repair operations in a single UI
  - Allows the repair operation to be deferred until there is a UI, instead of repairing during boot-up (blank screen)
  - Good example is rebuilding a RAID set that could take a very long time

Driver Health Protocol

- Localize Health Protocol strings
  - Strings returned in `MessageList` and `FormHiiHandle` should be available in all languages supported in the driver's HII Forms
    - Ensures best user experience when localization is enabled

Driver Health Protocol

- Use `EfiDriverHealthStatusRebootRequired` value in the HealthStatus parameter to indicate a reboot is required
  - Never reboot the system directly

Driver Diagnostics Protocol

Driver Diagnostics Protocol

- Drivers should implement EFI_DRIVER_DIAGNOSTICS2_PROTOCOL
  - Enables the platform firmware to build a common GUI and/or CLI tool for pre-boot firmware diagnostics of different devices
    - Better user experience than IHV specific tools/UIs
    - Improves the overall system diagnostics capabilities

Driver Diagnostics Protocol

- Localize Diagnostics Protocol strings
  - Strings returned from the driver's `RunDiagnostics()` function should be localized to all languages supported in the driver's HII Forms to ensure best user experience when localization is enabled

Firmware Management Protocol

Firmware Management Protocol

- If a device supports firmware updates, its driver should implement EFI_FIRMWARE_MANAGEMENT_PROTOCOL
  - Enables the platform to build a common GUI and/or CLI tool for pre-boot firmware update of different devices
    - Better user experience than using multiple IHV specific tools/UIs

Firmware Management Protocol

- If SetImage() returns AbortReason, the string must be user friendly
  - Provide the string in the currently selected PlatformLang, or in English if the driver does not support that language
  - Support all languages supported by the driver's HII configuration Forms to ensure best user experience when localization is enabled

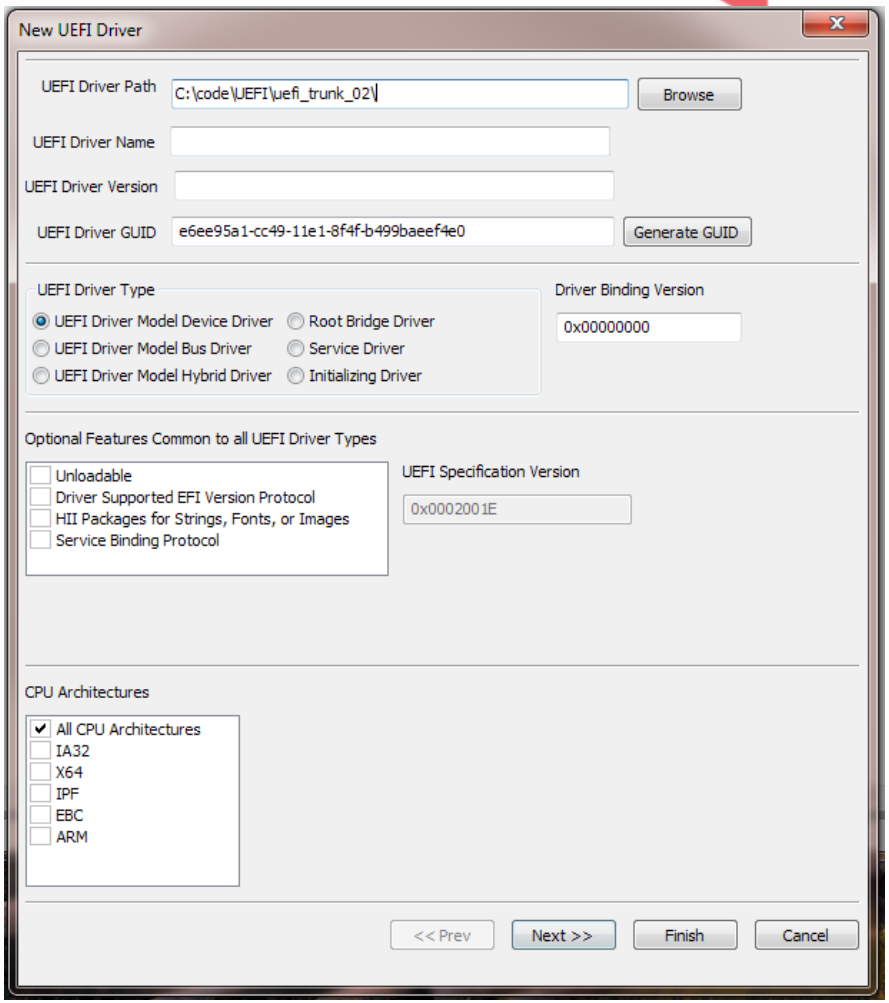
More information

- UEFI Driver Writer's Guide
  - Includes UEFI 2.3.1 updates from Intel
  - Good developer reference
  - Includes tips and sample code for easier implementations of the protocols discussed

http://sourceforge.net/projects/edk2/files/EDK%20II%20Releases/Driver_Developer/

More information

- UEFI Driver Wizard
  - GUI to simplify UEFI Driver Development
  - Open source project contributed by Intel
  - Automates the basic implementation of many protocols
  - IHVs are highly encouraged to leverage the wizard



Summary / Q&A

- Building a better firmware user experience is a collaborative effort
- Many Protocols and features that are “optional” in the UEFI Specification are required for a better user experience
- IHVs are highly encouraged to work closely with OEMs to design more usable firmware

Thanks for attending the
UEFI Summer Summit 2012

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

presented by