

UEFI Overview

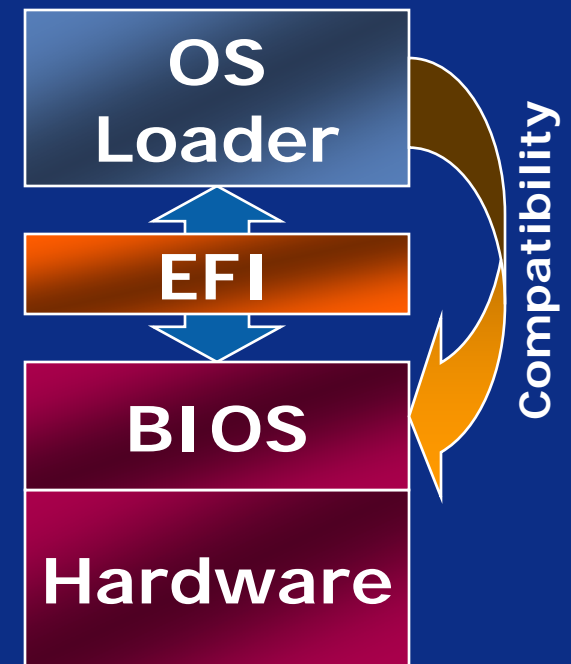
May 16th, 2007

Michael A. Rothman

Intel / Chair of UEFI Configuration Sub-team

A look at EFI

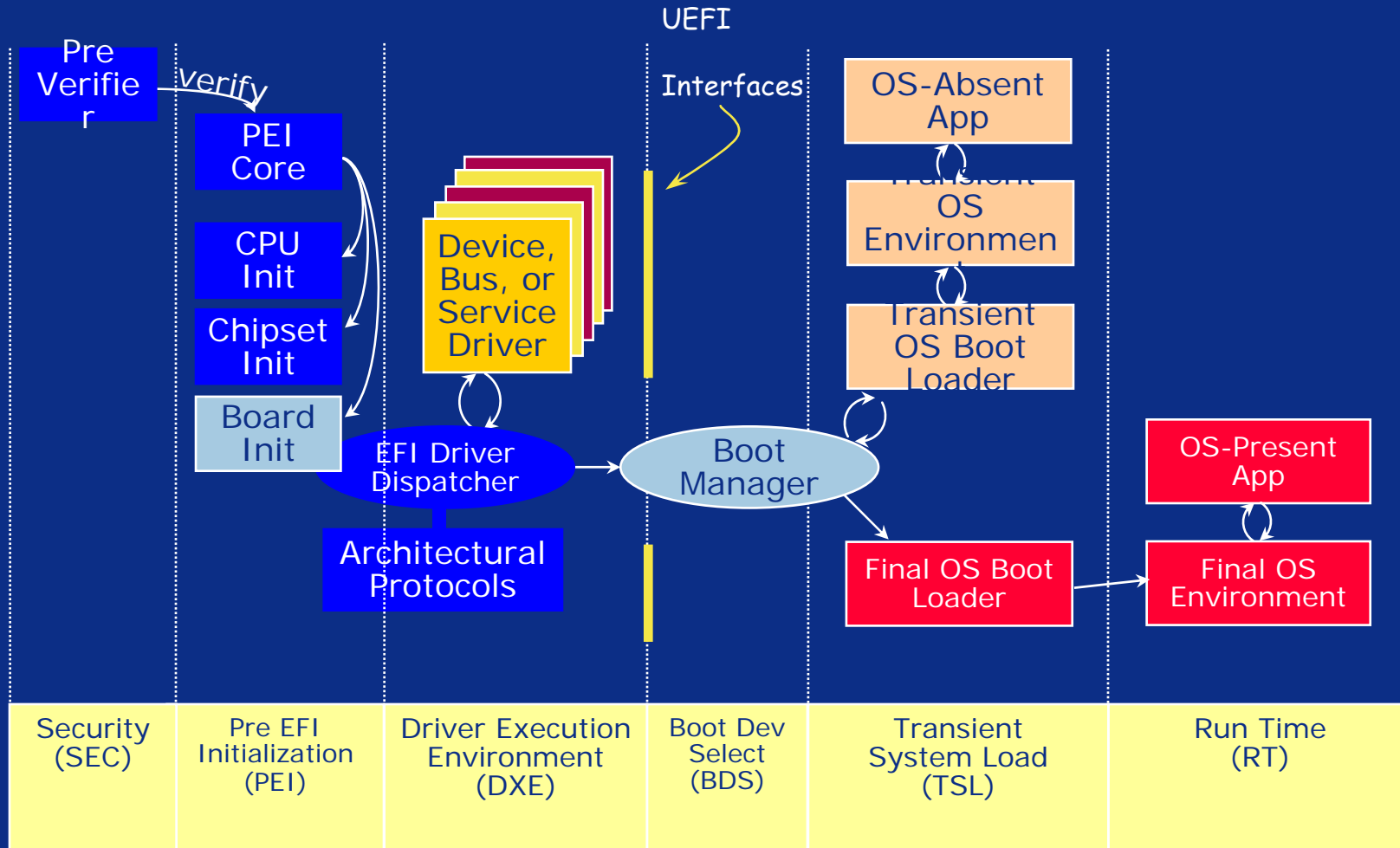
- Interface specification
 - Implementation agnostic
- Abstracts BIOS from OS
 - Decouples development
- Compatible by design
 - Evolution, not revolution
- Modular and extensible
 - OS-Neutral value add
- Provide efficient Option ROM Replacement
 - Common source for multiple CPU architectures



*Other names and brands may be claimed as the property of others



A couple steps forward



Power on → [. . Platform initialization . .] → [. . . . OS boot] → Shutdown



*Other names and brands may be claimed as the property of others



Unified EFI Forum, Inc.

A Firmware Standards Organization Gets Created

A Washington non-profit Corporation

- Develops, promotes and manages evolution of Unified EFI Specification
- Continue to drive low barrier for adoption

Promoter members:

- AMD, AMI, Apple, Dell, HP, IBM, Insyde, Intel, Lenovo, Microsoft, Phoenix

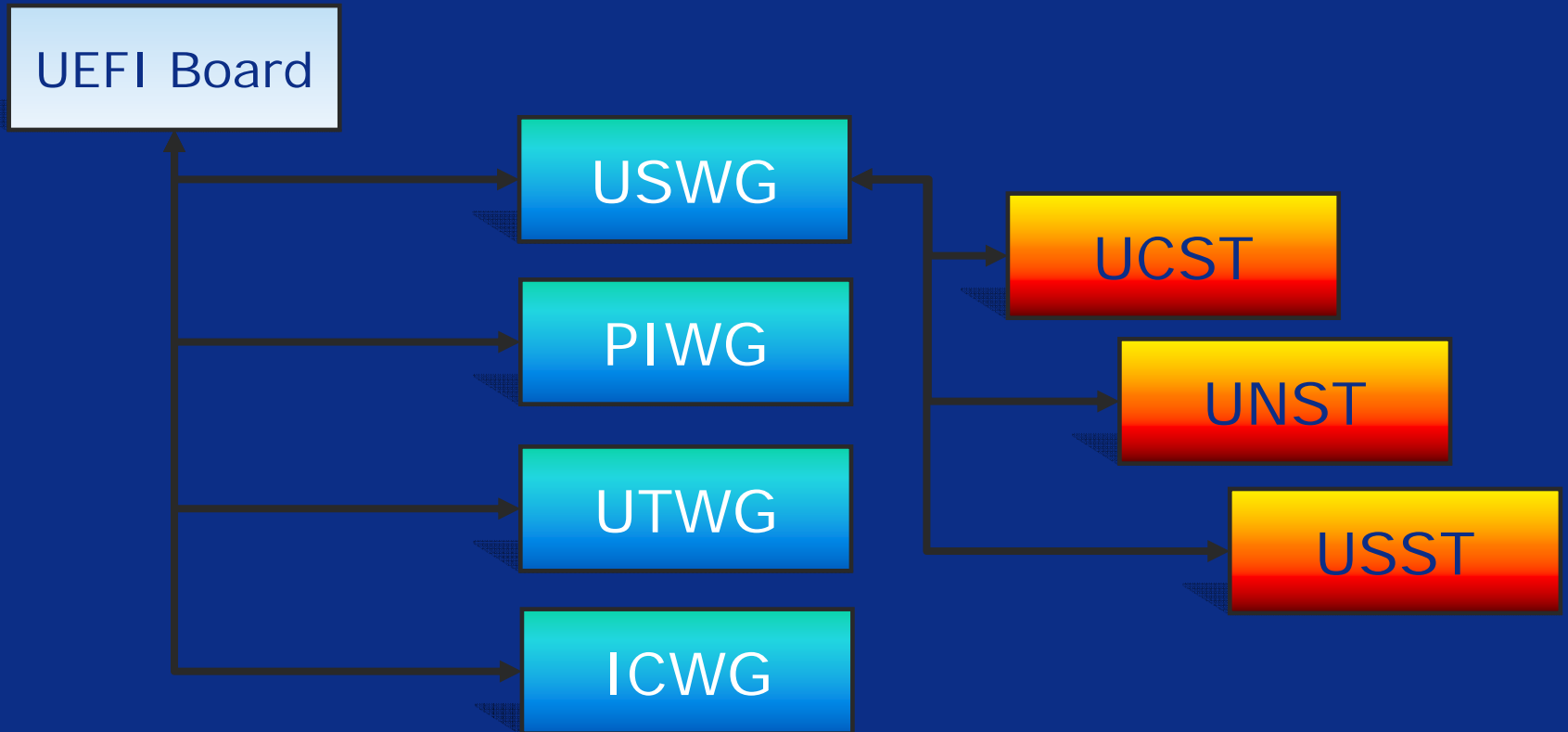
More information: www.uefi.org



*Other names and brands may be claimed as the property of others



How the Forum Works



Publications/Decisions ratified by the board

Each work group approves/delivers different content to the public.

Each sub-team focuses on specific topics and contributes material to the work group.

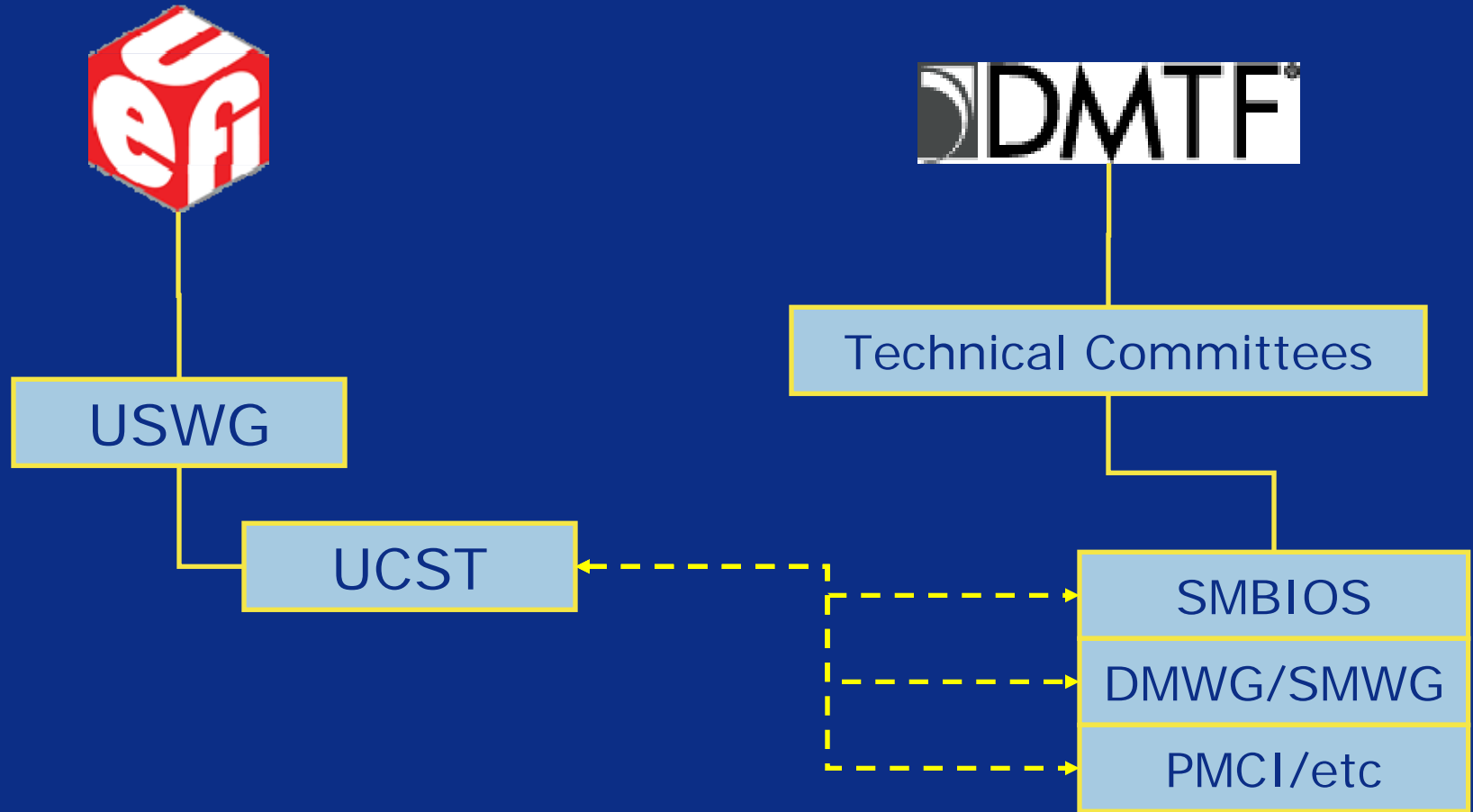


*Other names and brands may be claimed as the property of others



UEFI Relationships In Industry

Relationship with DMTF



*Other names and brands may be claimed as the property of others



Concept Review - Protocols

Read Keystroke Example

Legacy

UEFI/Framework

INT 16h	AH = 10h	Input
	AH = Scan code	Output
	AL = ASCII character	Output

Simple Text Input Protocol	ReadKeyStroke	Protocol
	Reset	
	WaitForKey	
	This, &Key	Input
	Key = EFI_INPUT_KEY	Output

Caller Sample Code

```
mov ax, 1000h
int 16h
```

Handler Sample Code

```
cmp ah, 10h
jz HandleExtReadKey
cmp ah, 11h
jz CheckForKey
;; Do more checking

HandleExtReadKey:
;; Do real work here
mov ax
ret
```

Caller Sample Code

```
TextIn->ReadKeystroke (TextIn, &Key);
```

Handler Sample Code

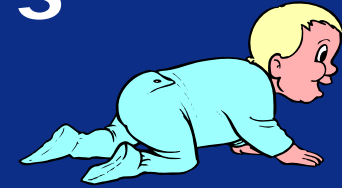
```
ReadKeystrokeHandler (
    IN EFI_SIMPLE_TEXT_INPUT_PROTOCOL *This,
    OUT EFI_INPUT_KEY *Key
)
{
    // Do real work here
}
```



*Other names and brands may be claimed as the property of others



Moving Data from pre-boot into O/S runtime



EFI System Table

EFI Configuration Table

GUID	Pointer
Table A GUID	Address A
Table B GUID	Address B
⋮	
Table Y GUID	Address Y
Table Z GUID	Address Z



Local Configuration/Manageability



*Other names and brands may be claimed as the property of others

Problem Statement

- No standard/interoperable mechanism to address pre-boot based issues like:
 - Localization
 - Standard delivery of string packages
 - Fonts
 - Create standard glyph support along with optional font styles
 - Shared Configuration Infrastructure
 - Alleviate the burden for many configuration engines in a system (e.g. add-in device no longer needs to delay boot or poll for hot-keys, etc)
- Should be able to also address:
 - Human -> Machine system configuration
 - Think Setup
 - Machine -> Machine system configuration
 - Think Automation

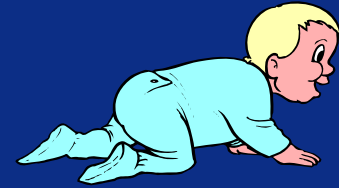


Platform Configuration/Manageability

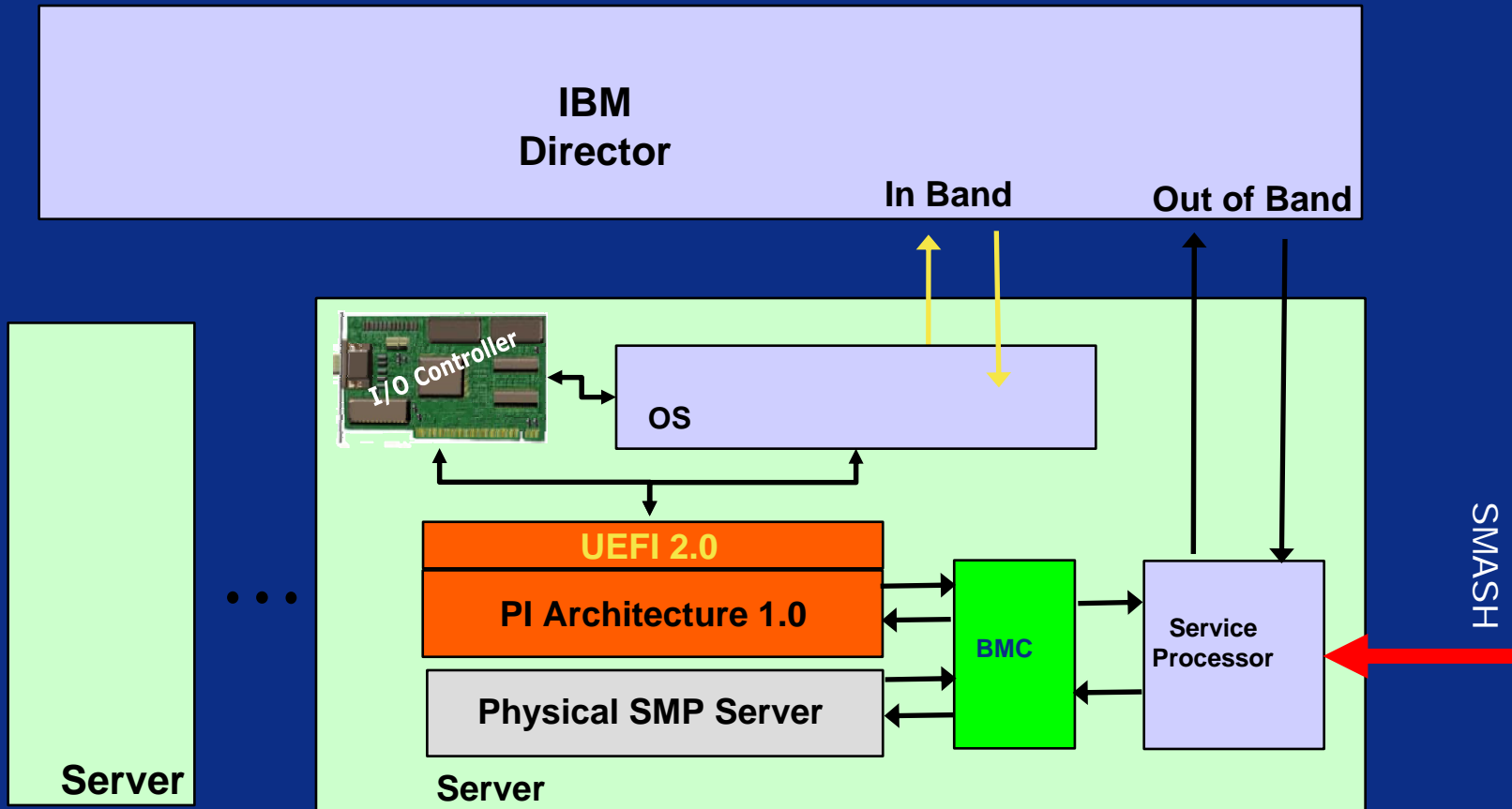
*Other names and brands may be claimed as the property of others



UEFI Interactions



- Interaction Component View



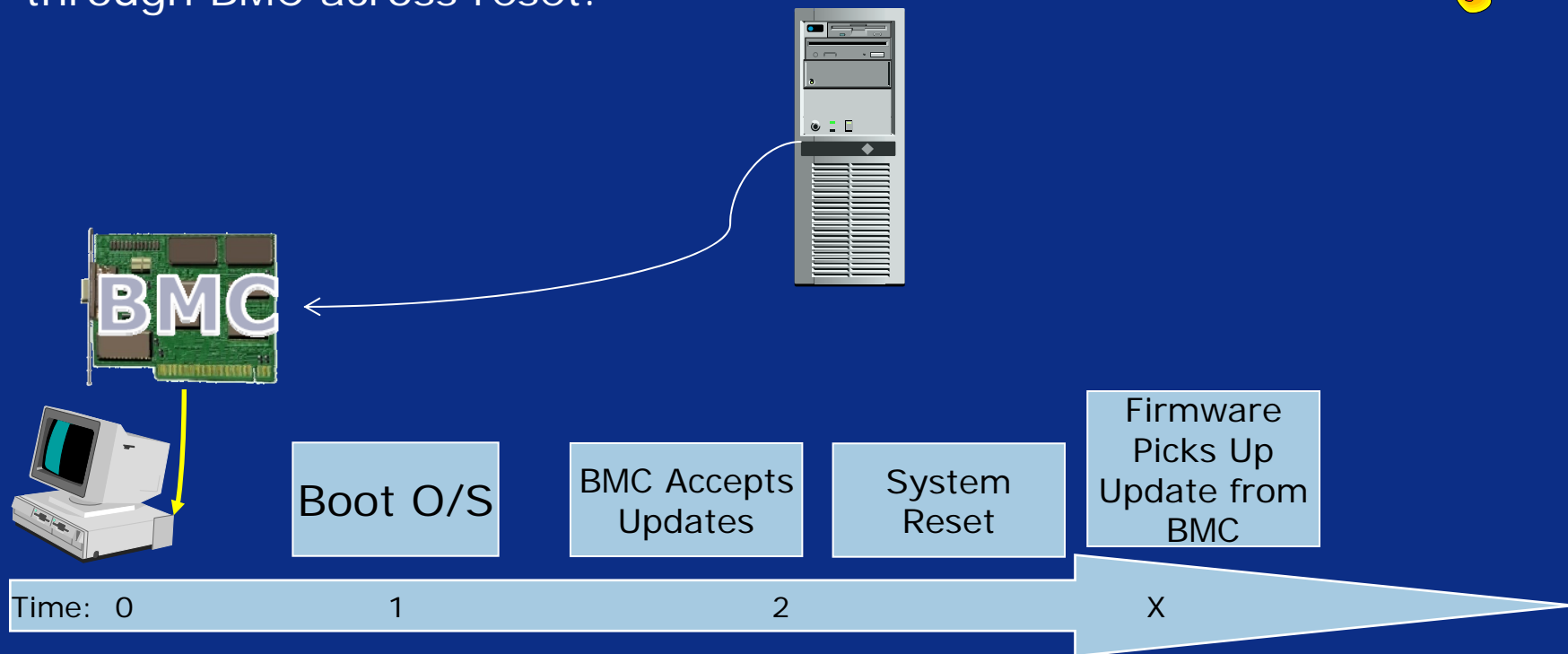
Local Configuration/Manageability



*Other names and brands may be claimed as the property of others

Interactions with BMC

- Local Interactions / with data being handed to firmware through BMC across reset.

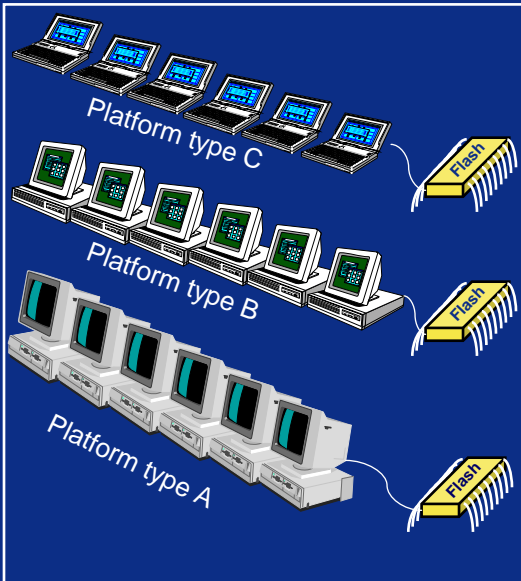


Local Configuration/Manageability



*Other names and brands may be claimed as the property of others

Configure Heterogeneous Targets



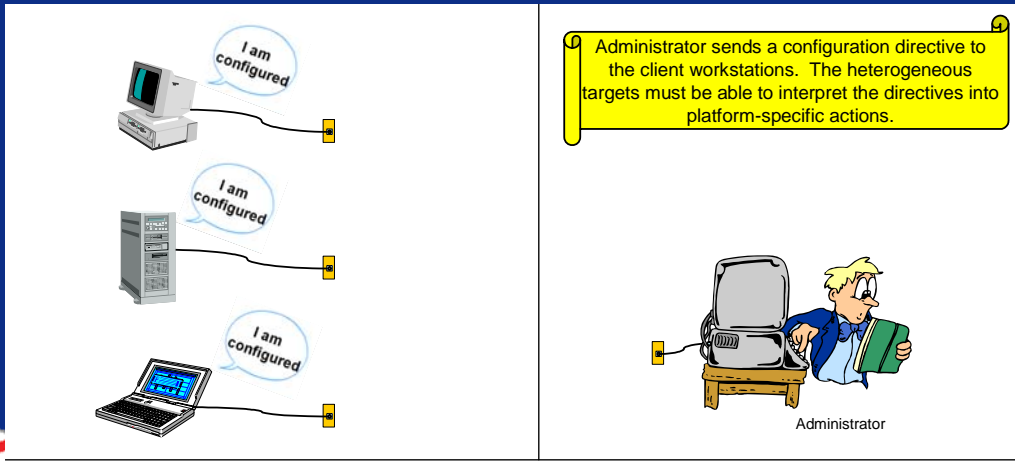
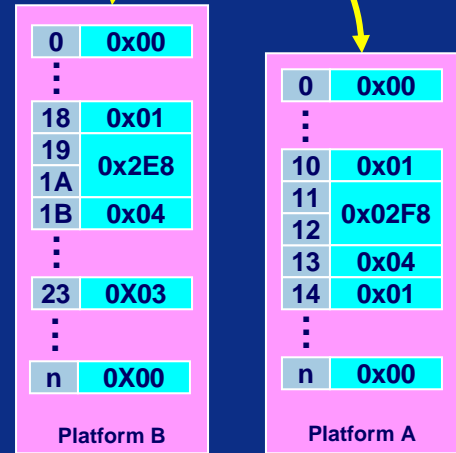
Three classes of platforms each with different configuration maps in their FLASH

Settings Keyword	Option Keyword/Value Pairs	FLASH Map Offset
HT_ENABLE	Enable=1, Disable=0	0x00
COM1_ENABLE	Enable=1, Disable=0	0x01
COM1_ADDRESS	0x2e8, 0x2f8, 0x3e8, 0x3f8	0x02
COM1_IRQ	0x03, 0x04	0x04
⋮		
Platform C Tag Definitions		

Settings Keyword	Option Keyword/Value Pairs	FLASH Map Offset
HT_ENABLE	Enable=1, Disable=0	0x23
COM1_ENABLE	Enable=1, Disable=0	0x18
COM1_ADDRESS	0x2e8, 0x2f8, 0x3e8, 0x3f8	0x19
COM1_IRQ	0x03, 0x04	0x1B
⋮		
Platform B Tag Definitions		

Settings Keyword	Option Keyword/Value Pairs	FLASH Map Offset
HT_ENABLE	Enable=1, Disable=0	0x14
COM1_ENABLE	Enable=1, Disable=0	0x10
COM1_ADDRESS	0x2e8, 0x2f8, 0x3e8, 0x3f8	0x11
COM1_IRQ	0x03, 0x04	0x13
⋮		
Platform A Tag Definitions		

Large Corporate Customer Platform Schema Definition

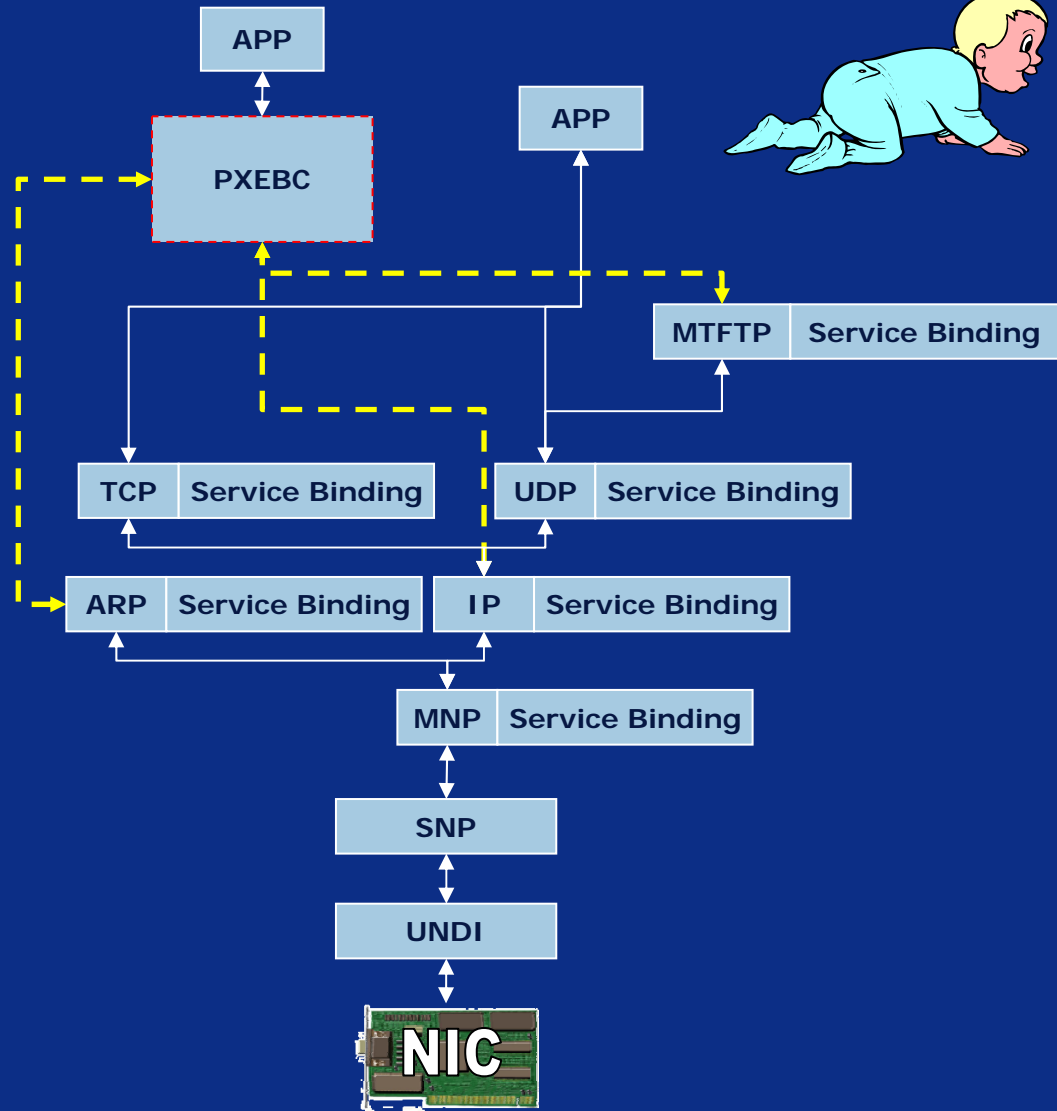


Platform Configuration/Management



*Other names and brands may be claimed as the property of others

Network Infrastructure



Remote Configuration/Management



*Other names and brands may be claimed as the property of others

UEFI Configuration/Manageability Infrastructure

- Keyboard Localization



Spanish



US English



French

- String/Text Localization

String ID #4	String Representation	H	E	L	L	O		W	O	R	L	D	
	Unicode Encoding	0x0048	0x0045	0x004C	0x004C	0x004F	0x0020	0x0057	0x004F	0x0052	0x004C	0x0044	0x0000
String ID #4	String Representation	H	O	L	A		M	U	N	D	O		
	Unicode Encoding	0x0048	0x004F	0x004C	0x0041	0x0020	0x004D	0x0055	0x004E	0x0044	0x004F	0x0000	
String ID #4	String Representation	你	好	世	界								
	Unicode Encoding	0x4F60	0x597D	0x4E16	0x754C	0x0000							



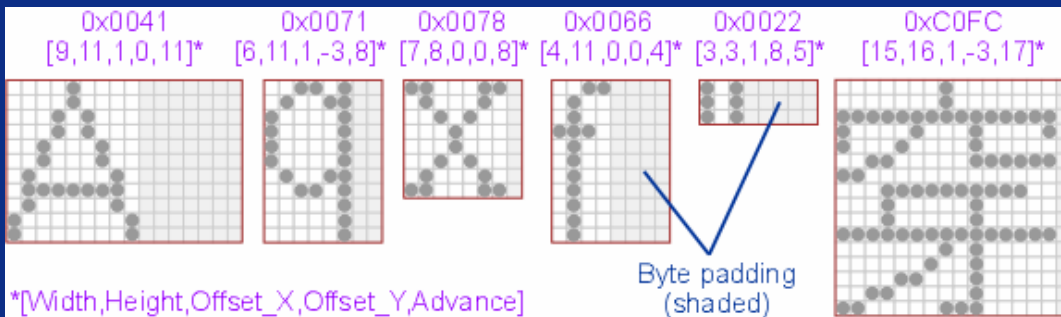
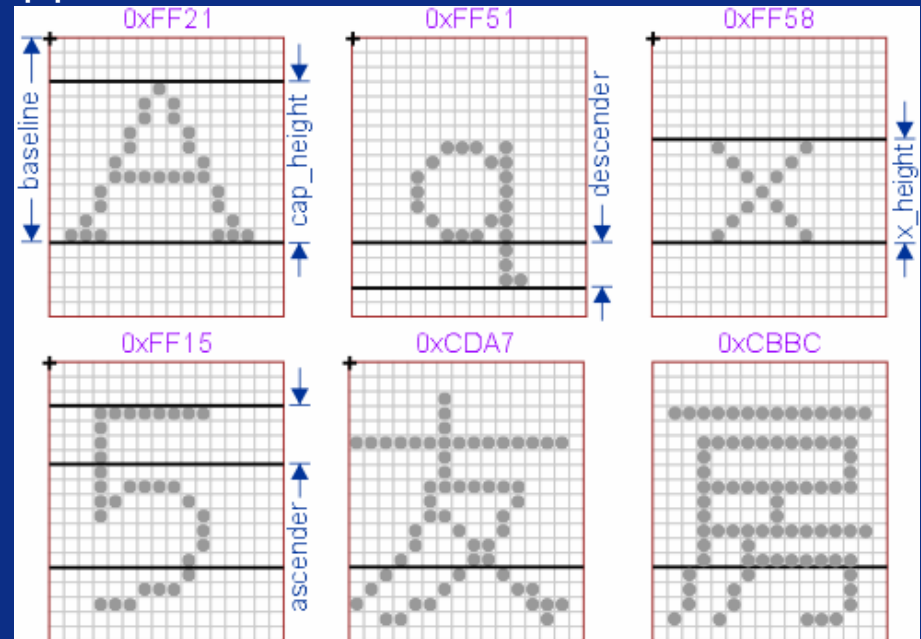
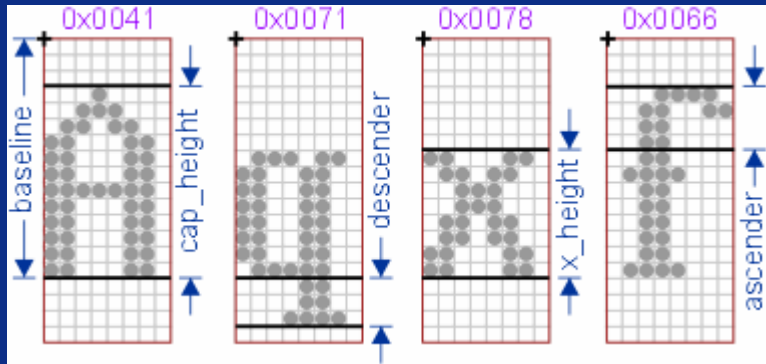
Configuration/Manageability Infrastructure



*Other names and brands may be claimed as the property of others

Glyph Support

- Dependent on Int10h character support? No....



*[Width,Height,Offset_X,Offset_Y,Advance]



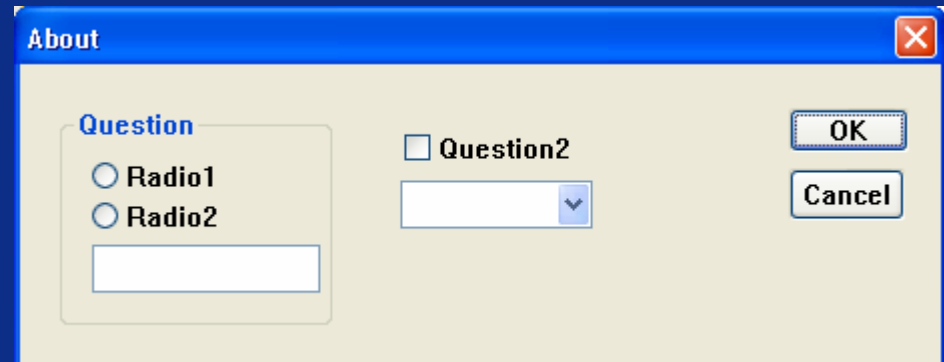
Configuration/Manageability Infrastr



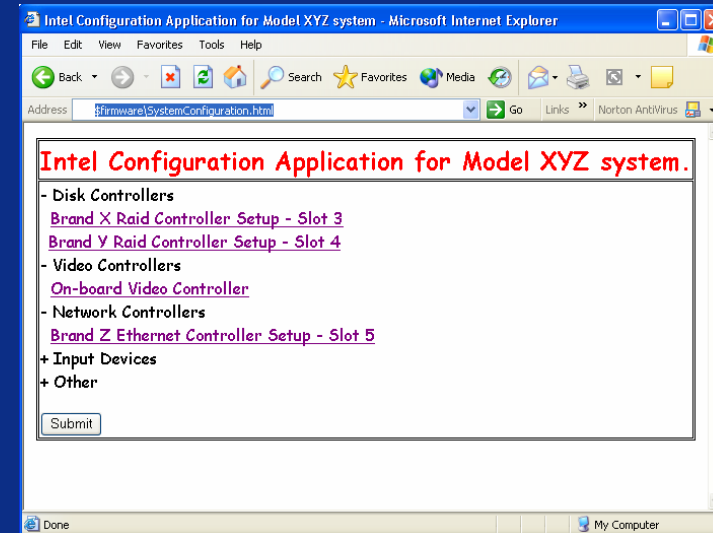
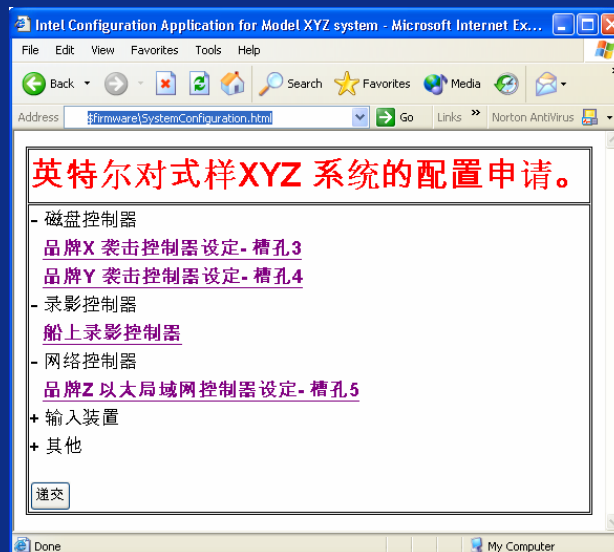
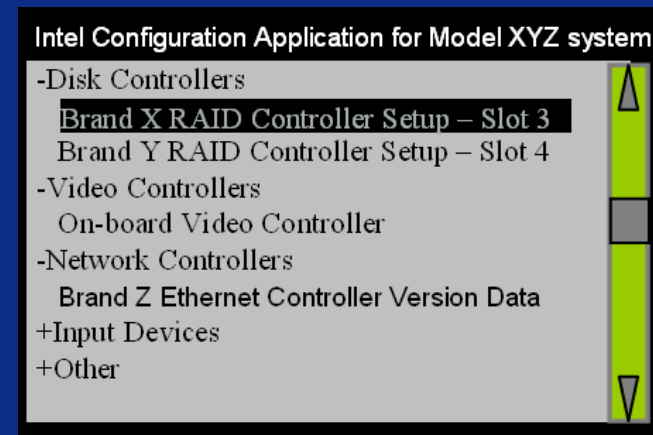
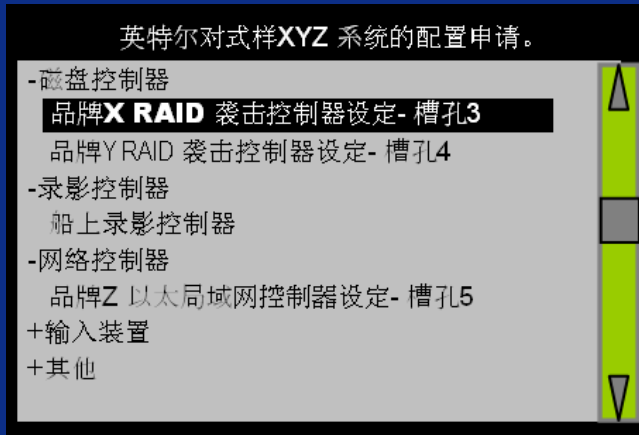
*Other names and brands may be claimed as the property of others

Using Forms

- Forms-based model for setup question descriptions
 - Must meet BIOS requirements
 - Scalable UI display support (Server Front Panel to local high resolution monitor).
 - Small encoding size
 - Encoding that is Self Describing
 - Position Independent
 - Can support scripting
 - Extensible syntax
- Exact look and feel defined by the browser and not defined in UEFI 2.1.
 - Developer/OEM/IHV defines questions to ask and what strings to display
 - Browser determines “how” to display the questions



Defining a User Interface – leave to OEM/IBV

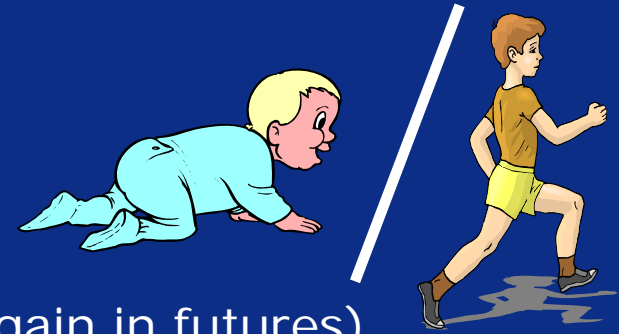


Configuration/Manageability Infrastructure



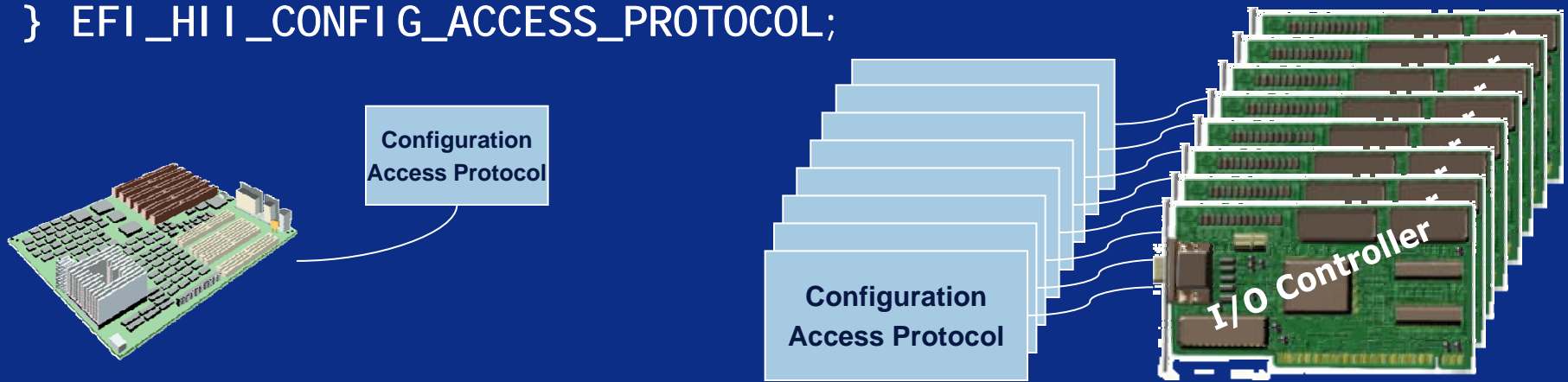
*Other names and brands may be claimed as the property of others

UEFI Interactions



- Device Access APIs (usable today – talk about again in futures)...

```
typedef struct {
  EFI_HII_EXTRACT_CONFIG      ExtractConfig;
  EFI_HII_ROUTE_CONFIG       RouteConfig;
  EFI_HII_FORM_CALLBACK      Callback;
} EFI_HII_CONFIG_ACCESS_PROTOCOL;
```



Local Configuration/Manageability



*Other names and brands may be claimed as the property of others

Backup



*Other names and brands may be claimed as the property of others



UEFI Interactions

Way it has worked



System Copyright Legacy Firmware Version 1.0 Status OK

64MB Lower System RAM initialized
 64GB System RAM initialization started
 PCI enumeration started
 Keyboard Found
 Mouse Found
 Brand X RAID controller initialized
 Press Alt-F2 to enter setup
 Boot Device Z initialized

⋮
 PCI enumeration started
 Keyboard Found
 Mouse Found
 Brand X RAID controller initialized
 Boot Device Z initialized
 Add-in card Y initialized
 64GB System RAM initialization complete
 USB Host Controller initialized
 USB Boot Device A initialized



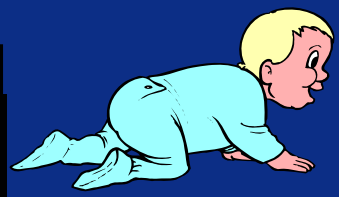
System Copyright Firmware Version 2.0.3 Status OK

Continue
 Set Language
 Device Manager
 Boot Selection
 Progress Data

⋮
 Brand X RAID controller initialized
 Brand Y RAID controller initialized
 Brand Z Ethernet controller initialized
 ⋮

Device Manager

-Disk Controllers
 Brand X RAID Controller Setup – Slot 3
 Brand Y RAID Controller Setup – Slot 4
 -Video Controllers
 On-board VGA Controller
 -Network Controllers
 Brand Z Ethernet Controller Version Data
 +USB Controllers
 +Input Devices
 +Other



Way it can work



Local Configuration/Manageability

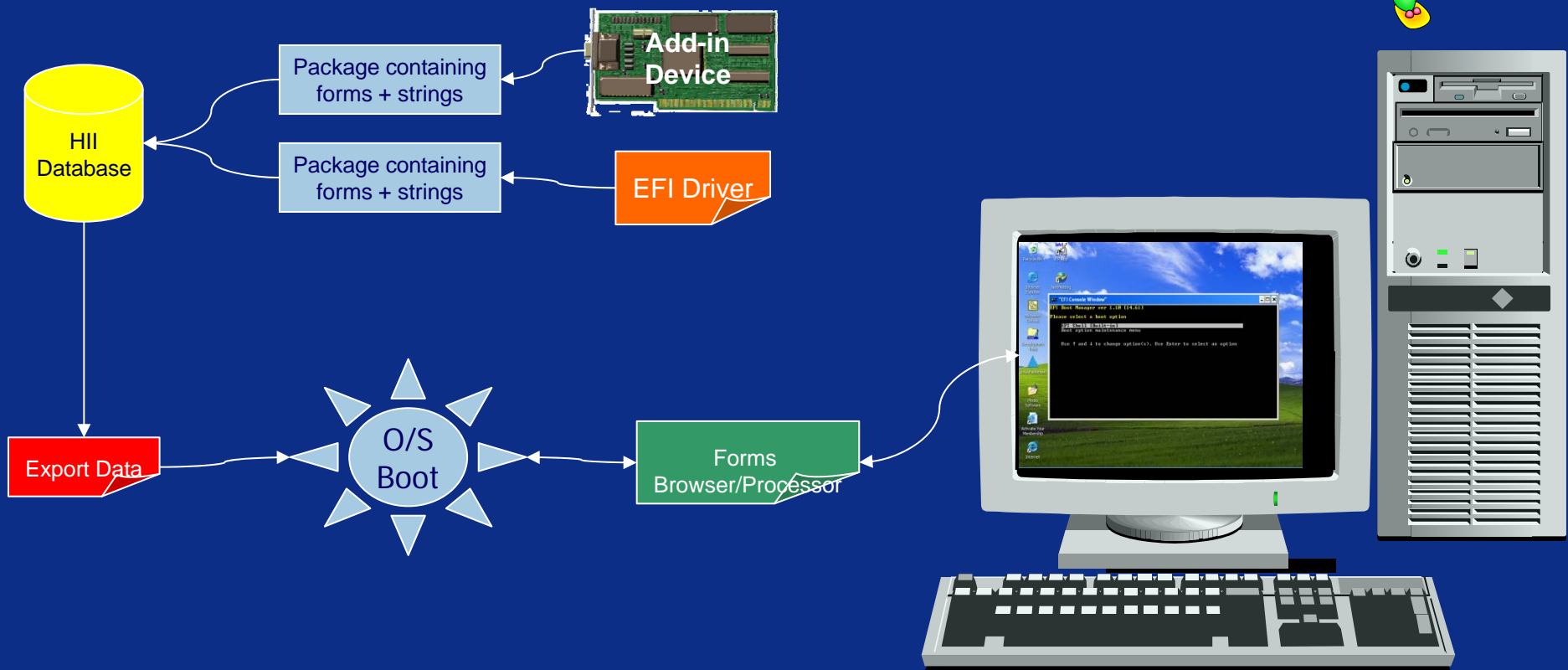
*Other names and brands may be claimed as the property of others



UEFI Interactions



- Local Interactions – Exporting up to the O/S

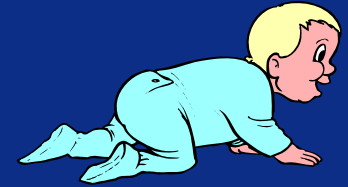


Local Configuration/Manageability



*Other names and brands may be claimed as the property of others

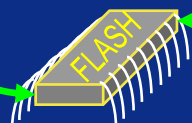
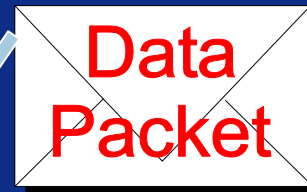
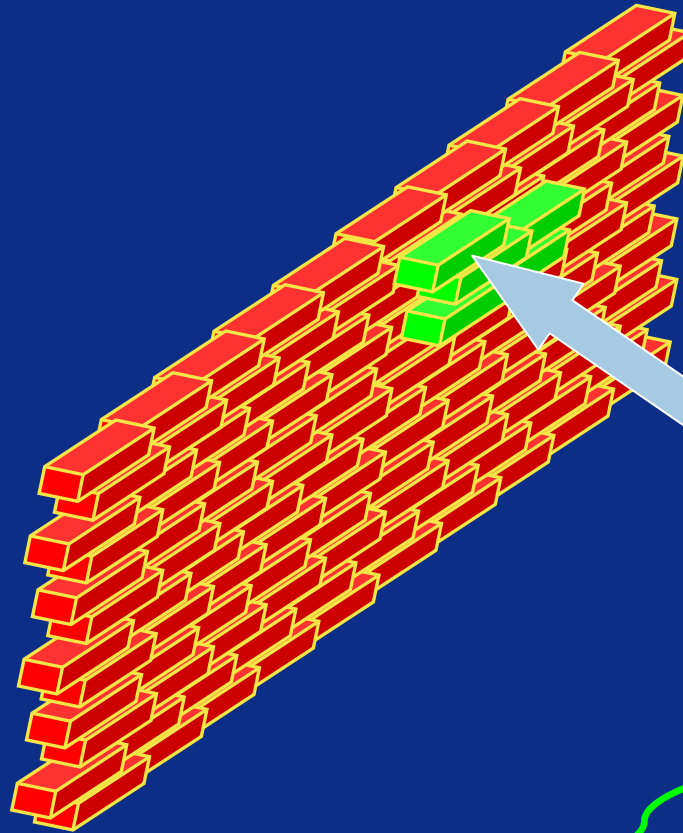
Moving Data through Capsules



Introduce new APIs for complex data communication.

Pre-boot

Runtime

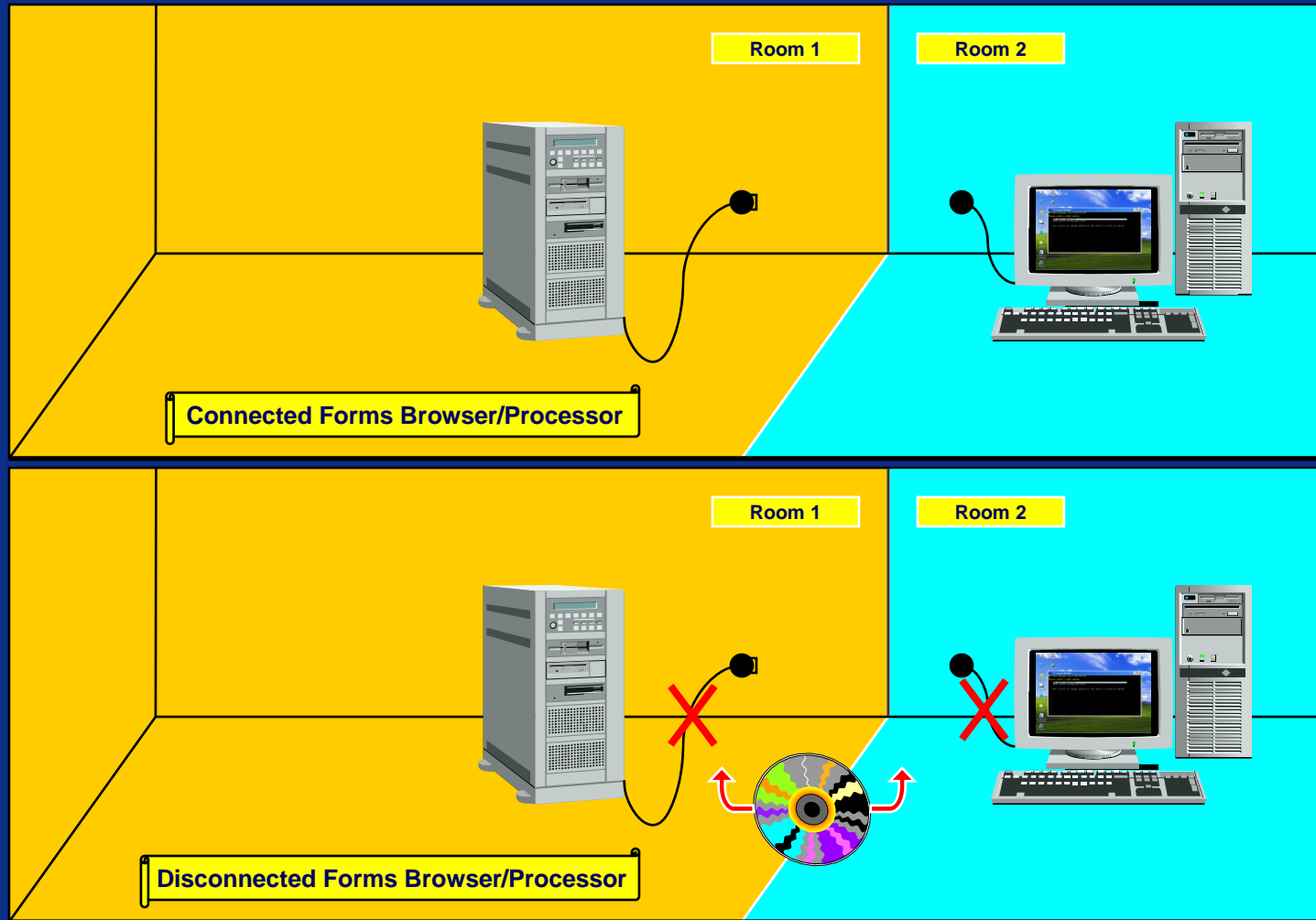


Local Configuration/Manageability



*Other names and brands may be claimed as the property of others

Configuration Portability



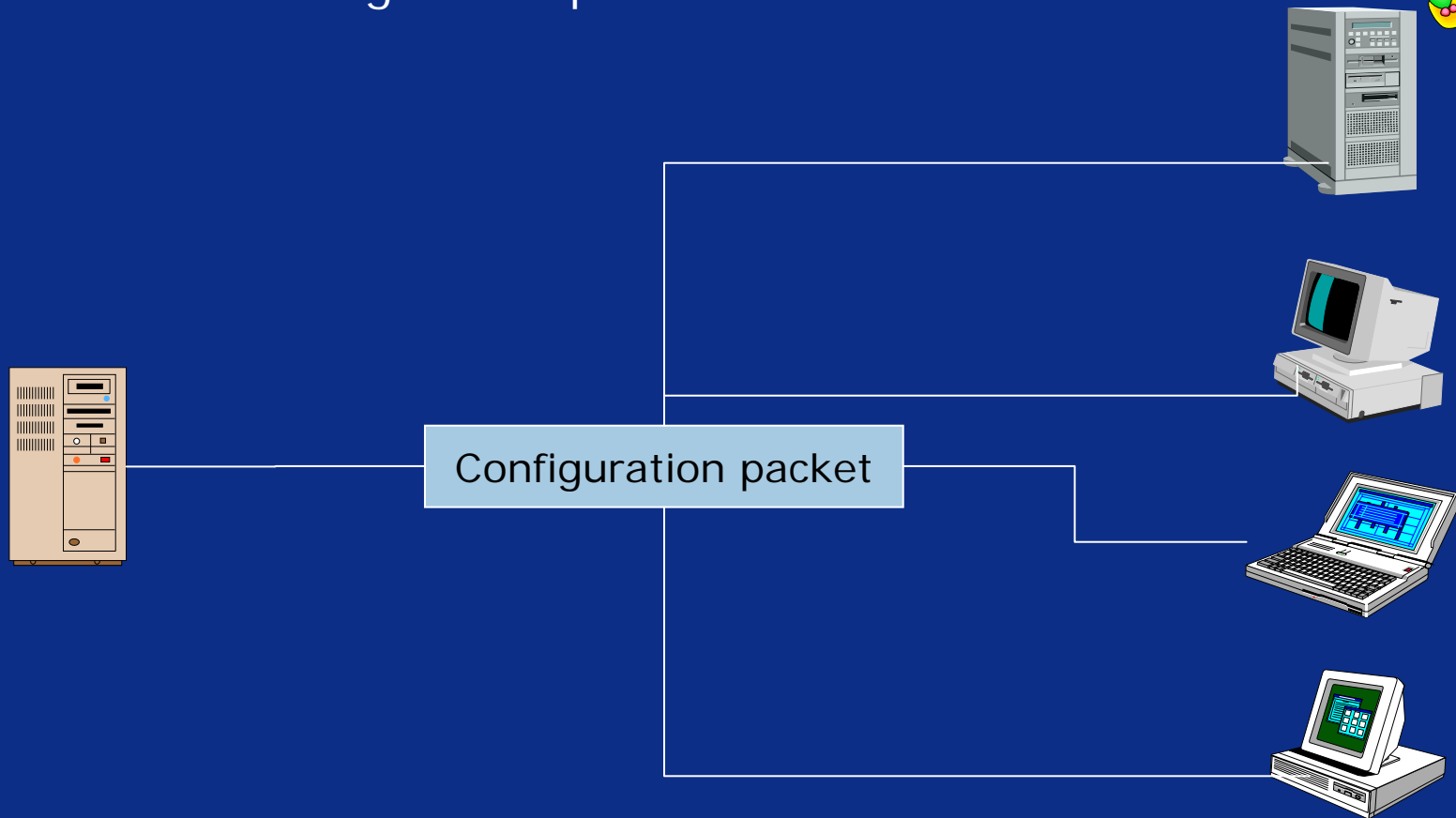
Platform Configuration/Manageability



*Other names and brands may be claimed as the property of others

Network-based Configuration Interactions

- Construction of configuration packets

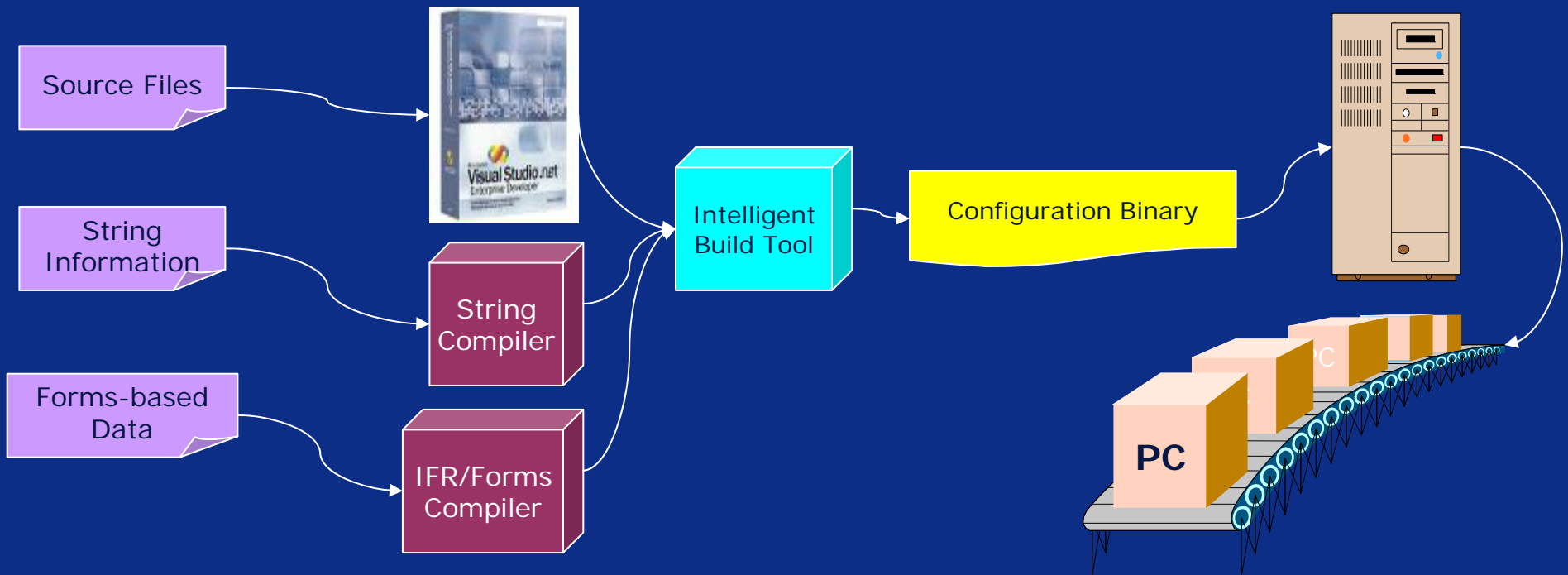


Remote Configuration/Manageability

*Other names and brands may be claimed as the property of others



UEFI Interactions



Remote Configuration/Manageability

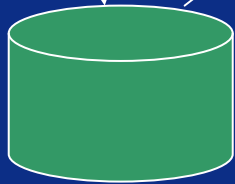


*Other names and brands may be claimed as the property of others

Use in Manufacturing

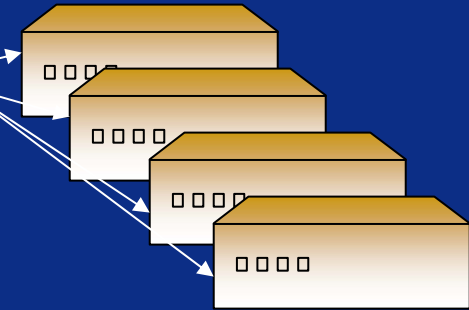
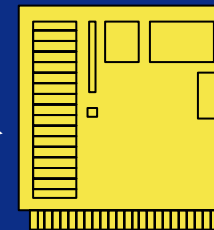


0. Vendor Build generates blob. Put on vendor's server.



Step 2: Integrator configures cards for his application

Step 3: When done, data is downloaded to manufacturing server.



Step 4: Later, systems start manufacture and are configured during integration.

Step 1: Integrator downloads data blobs describing cards and mobo to his notebook

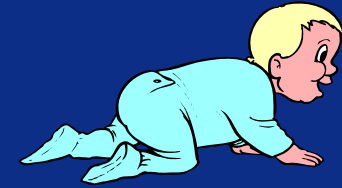


Remote Configuration/Manageability



*Other names and brands may be claimed as the property of others

UEFI Overview



In – UEFI 2.1 target

- Produce configuration infrastructure specification (What everyone is reviewing now)
- Purpose:
 - Enable IHVs to have a standard to write against for purposes of HW config.
 - Enable Industry (OEM/IBV/etc) for **proprietary** platform configuration and display mechanisms within the pre-boot and extend this into O/S runtime enablement.
 - Other functional extensions are possible....



Current Directions / UEFI & DMTF

*Other names and brands may be claimed as the property of others



UEFI Configuration Sub-team (UCST)



Stage 2 – to be worked on (UEFI 2.2 or whitepaper or both)

- Using the efforts in Stage 1, further describe how to move from a proprietary platform config mechanism to an environment which mixes multiple proprietary and standard namespace-based ones.
 - Issues to work on:
 - Refining our description of how exactly to determine the config keyword from a pre-existing namespace (e.g. CLP) and apply it to the platform.
 - Challenge will be if existing namespaces are insufficient in describing a “keyword”, get it so that they do, or establish some whitepaper material for guiding people on how to do it.
 - There is an open challenge to avoid creating our own namespace for many reasons – but it is always an option, just not one I relish the thought of doing.

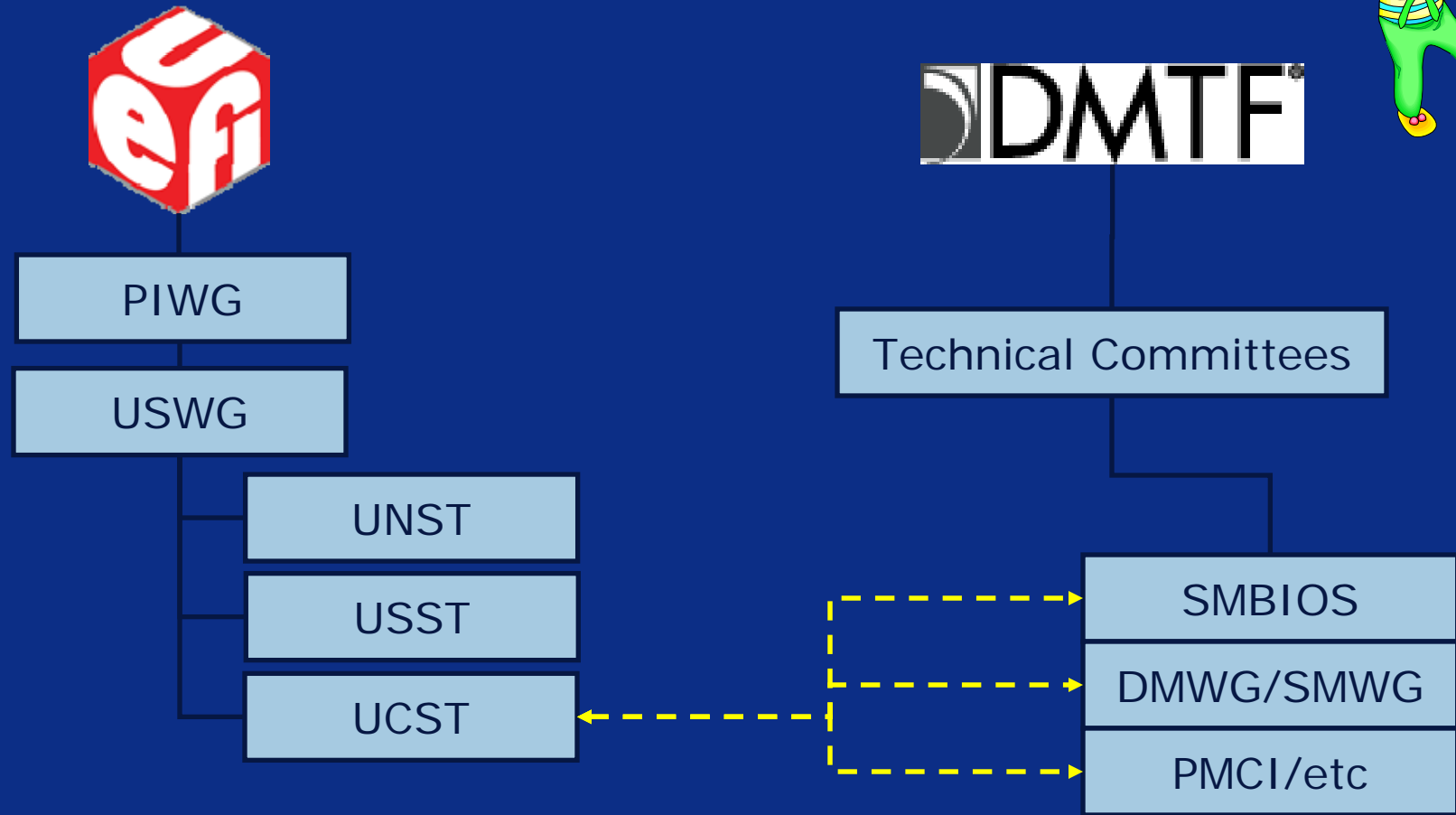


Futures

*Other names and brands may be claimed as the property of others



UEFI & DMTF Work Register -Now Approved-



Current Directions / UEFI & DMTF



*Other names and brands may be claimed as the property of others

DMTF Register Actions

- Drive UEFI representation through schema, profile(s) and mapping specification(s) to ensure that the proper support (such as configuration capabilities and namespace requirements) exists for both traditional BIOS as well as UEFI standards.
 - CIM Schema 2.16 (3Q2007)
 - BIOS Profile 1.0 (3Q2007)
 - BIOS SM CLP Mapping Specification 1.0 (3Q2007)
 - Investigate and Contribute to BIOS Profile(s).
- Promote relevant DMTF material back into the UEFI 2.2 specification. (1Q2008)
- Inclusion of updates as appropriate to the upcoming DASH Management Initiative updates.
 - Inclusion of BIOS Profile in DASH 1.1 (4Q2007)



Futures

*Other names and brands may be claimed as the property of others



DMTF Register Actions - II

- Collaboration on appropriate UEFI requirements on specifications developed within DMTF Working Groups. This includes:
 - System and Option ROM Identifiers
 - Command/Response strings
- This would result in the following specifications that would need to be shared with UEFI:
 - SM CLP Specification Updates & Work in Progress Drafts
 - SM CLP Mapping Specification Updates & Work in Progress Drafts
 - CIM Schema Work in Progress Drafts
 - DMTF Profile Work in Progress Drafts
- This would result in the following specifications that would need to be shared with DMTF:
 - UEFI Specification Updates & Work in Progress Drafts

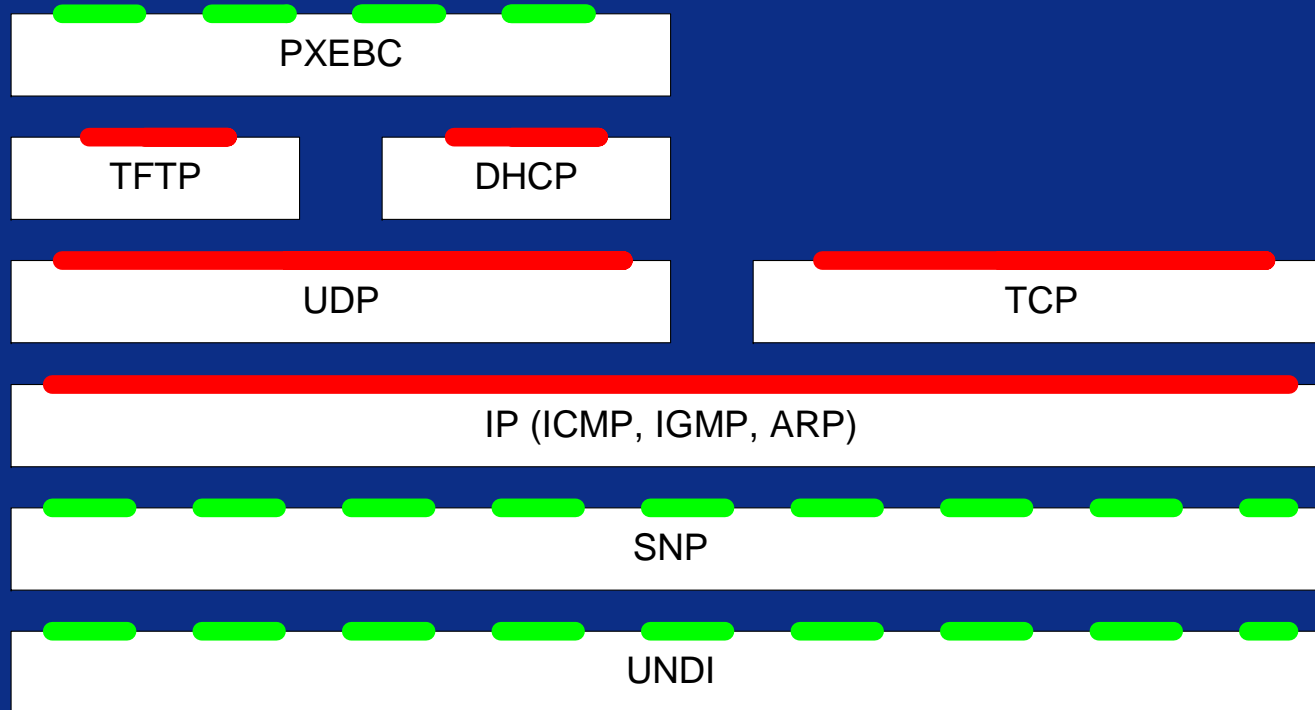


Futures

*Other names and brands may be claimed as the property of others



Protocol Diagram



Red (solid) lines are new (for UEFI 2.0) network protocol APIs that can be accessed by multiple applications and drivers at the same time.

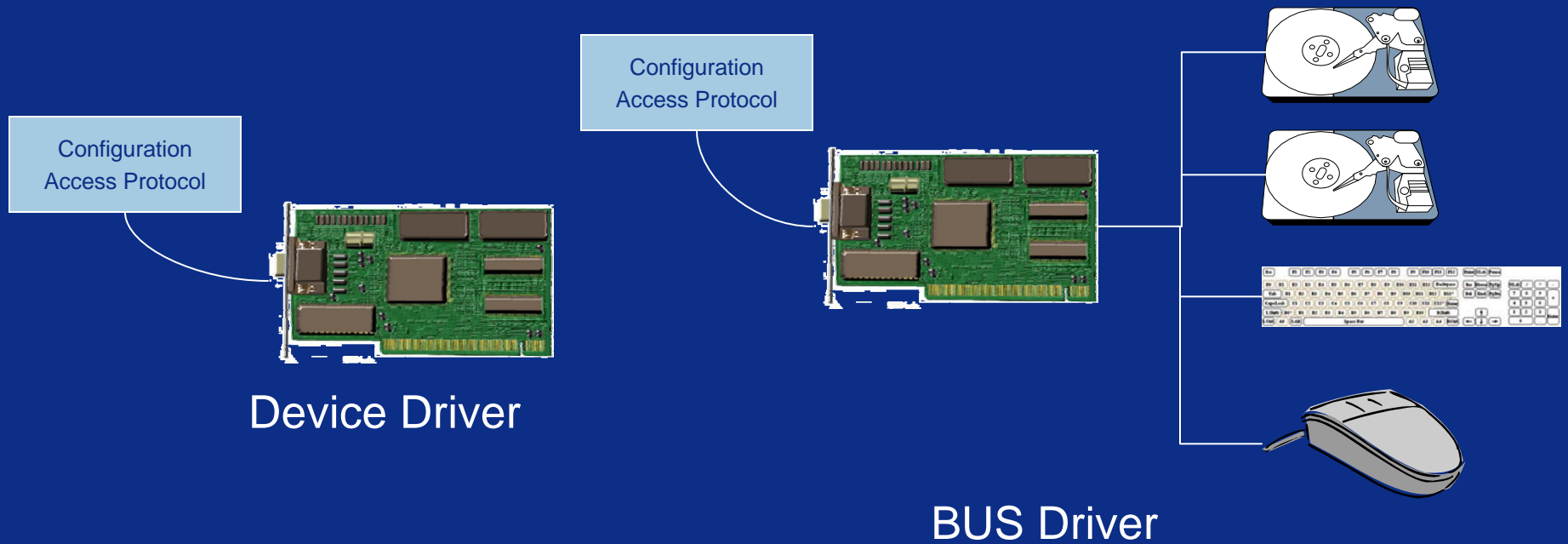
Green (broken) lines are existing network protocol APIs (from EFI 1.1 or earlier) that can only be accessed by one application or driver at a time.



*Other names and brands may be claimed as the property of others



Driver Model Relationship



*Other names and brands may be claimed as the property of others

