APEI output format

APEI uses printk as hardware error reporting interface, the output format is as follow:

```
<error record> :=
APEI generic hardware error status
severity: <integer>, <severity string>
section: <integer>, severity: <integer>, <severity string>
flags: <integer>
<section flags strings>
fru_id: <uuid string>
fru_text: <string>
section_type: <section type string>
<section data>

<severity string>* := recoverable | fatal | corrected | info

<section flags strings># :=
[primary][, containment warning][, reset][, threshold exceeded]\
[, resource not accessible][, latent error]

<section type string> := generic processor error | memory error | \
PCIe error | unknown, <uuid string>

<section data> :=
<generic processor section data> | <memory section data> | \
<pcie section data> | <null>

<generic processor section data> :=
[processor_type: <integer>, <proc type string>]
[processor_isa: <integer>, <proc isa string>]
[error_type: <integer>
<proc error type strings>]
[operation: <integer>, <proc operation string>]
[flags: <integer>
<proc flags strings>]
[level: <integer>]
[version_info: <integer>]
[processor_id: <integer>]
[target_address: <integer>]
[requestor_id: <integer>]
[responder_id: <integer>]
[IP: <integer>]

<proc type string>* := IA32/X64 | IA64

<proc isa string>* := IA32 | IA64 | X64

<processor error type strings># :=
[cache error][, TLB error][, bus error][, micro-architectural error]

<proc operation string>* := unknown or generic | data read | data write | \
instruction execution

<proc flags strings># :=
[restartable][, precise IP][, overflow][, corrected]

<memory section data> :=
[error_status: <integer>]
[physical_address: <integer>]
[physical_address_mask: <integer>]
[node: <integer>]
[card: <integer>]
[module: <integer>]
[bank: <integer>]
[device: <integer>]
[row: <integer>]
[column: <integer>]
[bit_position: <integer>]
[requestor_id: <integer>]
[responder_id: <integer>]
[target_id: <integer>]
[error_type: <integer>, <mem error type string>]

<mem error type string>* :=
unknown | no error | single-bit ECC | multi-bit ECC | \
single-symbol chipkill ECC | multi-symbol chipkill ECC | master abort | \
target abort | parity error | watchdog timeout | invalid address | \
```

```
mirror Broken | memory sparing | scrub corrected error | \
scrub uncorrected error

<pcie section data> :=
[port_type: <integer>, <pcie port type string>]
[version: <integer>.<integer>]
[command: <integer>, status: <integer>]
[device_id: <integer>:<integer>:<integer>.<integer>
slot: <integer>
secondary_bus: <integer>
vendor_id: <integer>, device_id: <integer>
class_code: <integer>]
[serial number: <integer>, <integer>]
[bridge: secondary_status: <integer>, control: <integer>]
[aer_status: <integer>, aer_mask: <integer>
<aer status string>
[aer_uncor_severity: <integer>]
aer_layer=<aer layer string>, aer_agent=<aer agent string>
aer_tlp_header: <integer> <integer> <integer> <integer>]

<pcie port type string>* := PCIe end point | legacy PCI end point | \
unknown | unknown | root port | upstream switch port | \
downstream switch port | PCIe to PCI/PCI-X bridge | \
PCI/PCI-X to PCIe bridge | root complex integrated endpoint device | \
root complex event collector

if section severity is fatal or recoverable
<aer status string># :=
unknown | unknown | unknown | unknown | Data Link Protocol | \
unknown | unknown | unknown | unknown | unknown | unknown | unknown | \
Poisoned TLP | Flow Control Protocol | Completion Timeout | \
Completer Abort | Unexpected Completion | Receiver Overflow | \
Malformed TLP | ECRC | Unsupported Request
else
<aer status string># :=
Receiver Error | unknown | unknown | unknown | unknown | unknown | \
Bad TLP | Bad DLLP | RELAY_NUM Rollover | unknown | unknown | unknown | \
Replay Timer Timeout | Advisory Non-Fatal
fi

<aer layer string> :=
Physical Layer | Data Link Layer | Transaction Layer

<aer agent string> :=
Receiver ID | Requester ID | Completer ID | Transmitter ID
```

Where, [] designate corresponding content is optional

All <field string> description with * has the following format:

```
field: <integer>, <field string>
```

Where value of <integer> should be the position of "string" in <field string> description. Otherwise, <field string> will be "unknown".

All <field strings> description with # has the following format:

```
field: <integer>
<field strings>
```

Where each string in <fields strings> corresponding to one set bit of <integer>. The bit position is the position of "string" in <field strings> description.

For more detailed explanation of every field, please refer to UEFI specification version 2.3 or later, section Appendix N: Common Platform Error Record.