Redfish LogEntry and Event support for diagnostic data

WORK IN PROGRESS
DMTF Redfish Forum
September 2022
Disclaimer

• The information in this presentation represents a snapshot of work in progress within the DMTF.

• This information is subject to change without notice. The standard specifications remain the normative reference for all information.

• For additional information, see the DMTF website: http://www.dmtf.org
Getting involved in Redfish

- Redfish Standards page
  - Schemas, Specs, Mockups, White Papers & more
  - [http://www.dmtf.org/standards/redfish](http://www.dmtf.org/standards/redfish)

- Redfish Developer Portal
  - Redfish Interactive Resource Explorer
  - Educational material, documentation & other links
  - [http://redfish.dmtf.org](http://redfish.dmtf.org)

- Redfish User Forum
  - User forum for questions, suggestions and discussion
  - [http://www.redfishforum.com](http://www.redfishforum.com)

- DMTF Feedback Portal
  - Provide feedback or submit proposals for Redfish standards
  - [https://www.dmtf.org/standards/feedback](https://www.dmtf.org/standards/feedback)

- DMTF Redfish Forum
  - Join the DMTF to get involved in future work
  - [http://www.dmtf.org/standards/spmf](http://www.dmtf.org/standards/spmf)
Introduction

- Redfish provides both human-facing and programmatic support for parsing logs and events
- Core component is a MessageId (key) which is defined in a “message registry” – a schema / dictionary for messages and their parameters
- Events sent from a Redfish service share property-level definitions with entries in a Redfish log (a collection of LogEntry resources)
- Some log entries and events require additional data associated with the occurrence to allow clients to analyze or debug the condition
  - A LogEntry resource allows for an AdditionalDataURI that enables the client to separately retrieve large crash dumps or other associated data
  - But for many use cases, the amount of additional data is “small” and could be provided within the LogEntry resource to avoid the need to separately retrieve data
- The UEFI Specification defines a Common Platform Error Record (CPER) format for recording error information and related data for further diagnosis
  - Redfish supports retrieval of CPER records via AdditionalDataURI in a LogEntry
Goals

- Create a Platform message registry to define standard messages for OS crashes, core dumps, and system faults that provide a UEFI-defined CPER error record.
- Provide a means to include small-to-moderate amounts of diagnostic data within a LogEntry resource.
  - Also provide support in Event for transporting this data to subscribers.
“Platform v1.0” message registry

• Define messages for faults originating in the “CPU / memory complex”
  • Transmit raw diagnostic data for client-side analysis or decode
  • Manager is likely a “pass through” of this data from platform or OS
• Message IDs:
  • `UnhandledExceptionDetectedAfterReset`
    • Indicates that an unhandled exception caused the platform to reset
    • "An unhandled exception caused a platform reset."
  • `OperatingSystemCrash`
    • Indicates the operating system was halted due to a catastrophic error
    • "An operating system crash has occurred."
  • `PlatformError`
    • Indicates that a platform error has occurred
    • “A platform error has occurred.”
  • `PlatformErrorAtLocation`
    • Indicates that a platform error has occurred, with device location info available
    • "A platform error has occurred at location `%1`."

© 2022 DMTF
LogEntry and Event enhancements

- Add CPER object to hold NotificationType and SectionType
  - Can add other decoded information from a CPER record or section
- Add DiagnosticData property to allow inclusion of a small-to-moderate amount of binary data within the LogEntry or Event resource
  - Value is a Base64-encoded string of data
  - Type of data follows value of existing DiagnosticDataType
  - Provide guidance for maximum size of this data
    - Perhaps this is reported by the LogService (configurable?)
    - Create a “include diagnostic data” subscription option for EventDestination (allow client to specify maximum data size?)
- If DiagnosticData received (from an external data provider) is too large given payload guidance, service will provide a URI for retrieval
  - Use the existing AdditionalDataURI and AdditionalDataSizeBytes
LogEntry example - CPER with large diagnostic data

```
{  
    "@odata.type": "#LogEntry.v1_14_0.LogEntry",  
    "Id": "3",  
    "Name": "CPER Log Entry with large additional data",  
    "EntryType": "Event",  
    "Severity": "Critical",  
    "Created": "2022-03-07T14:45:00Z",  
    "Message": "A platform error has occurred.",  
    "MessageId": "Platform.1.0.PlatformError",  
    "Links": {  
      "OriginOfCondition": {  
        "@odata.id": "/redfish/v1/Systems/1"  
      }  
    },  
    "CPER": {  
      "NotificationType": "902834BC-AD67-0BAD-BEEF-123456789012"  
    },  
    "DiagnosticDataType": "CPER",  
    "AdditionalDataSizeBytes": 2834000,  
    "AdditionalDataURI": "/dumpster/log3_cper.bin",  
    "@odata.id": "/redfish/v1/Systems/1/LogServices/Log1/Entries/3"  
}
```

Message from Platform message registry

New CPER object provides specific data needed by client to route to appropriate decoding routines or analysis application

DiagnosticDataType describes the format of the data stored at the AdditionalDataURI without having to rely on filenames or file extensions
LogEntry example - CPER with inline diagnostic data

```json
{
    "@odata.type": "#LogEntry.v1_14_0.LogEntry",
    "Id": "3",
    "Name": "CPER Log Entry with large additional data",
    "EntryType": "Event",
    "Severity": "Critical",
    "Created": "2022-03-07T14:45:00Z",
    "Message": "A platform error has occurred.",
    "MessageId": "Platform.1.0.PlatformError",
    "Links": {
        "OriginOfCondition": {
            "@odata.id": "/redfish/v1/Systems/1"
        }
    },
    "CPER": {
        "NotificationType": "902834BC-AD67-0BAD-BEEF-123456789012"
    },
    "DiagnosticDataType": "CPER",
    "DiagnosticData": "VGhlIGNha2UgaXMgYSBsaWUhCg==ASDEWIhnqn55Qe924MFAFHDF0IAFHDANHV4582bAIYQN",
    "@odata.id": "/redfish/v1/Systems/1/LogServices/Log1/Entries/4"
}
```

**DiagnosticDataType** describes the format of the **DiagnosticData**

Small amount of **DiagnosticData** can be included in the payload, removing need to retrieve separately.
Event example with proposed inline diagnostic data

```json
{
    "@odata.type": "#Event.v1_8_0.Event",
    "Id": "1",
    "Name": "Event with DiagnosticData included in payload",
    "Context": "ContosoFaultAnalysisEngine",
    "Events": [
        {
            "EventType": "Other",
            "EventId": "8675309",
            "Severity": "Critical",
            "MessageSeverity": "Critical",
            "Message": "A platform error has occurred at location `CPU #1`.",
            "MessageId": "Platform.1.0.PlatformErrorAtLocation",
            "MessageArgs": [
                "CPU #1"
            ],
            "OriginOfCondition": {
                "@odata.id": "/redfish/v1/Systems/1/Processors/1"
            },
            "CPER": {
                "NotificationType": "902834BC-AD67-0BAD-BEEF-123456789012"
            },
            "DiagnosticDataType": "CPER",
            "DiagnosticData": "VGhlIGNha2UgaXMgYSBsaWU="
        }
    ]
}
```