Intel® Device Discovery

Enhancing the Endpoint Discovery Capabilities of Device Management Tools

Software Developer’s Guide

Revision 1.01

October 2023

Intel Confidential

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis. You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

All product plans and roadmaps are subject to change without notice.

The products described may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

© Intel Corporation. Intel, the Intel logo, vPro, and other Intel marks are trademarks of Intel Corporation or its subsidiaries.

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

Copyright© 2023, Intel Corporation. All rights reserved.

Contents

[1 Introduction 5](#_Toc144456634)

[1.1 Intel® Device Discovery 5](#_Toc144456635)

[1.2 Intel® Manageability and Security IPF Extension Providers 5](#_Toc144456636)

[1.3 References 6](#_Toc144456637)

[2 System Requirements 7](#_Toc144456638)

[2.1 Visual C++ Runtime Installation 7](#_Toc144456639)

[2.2 Intel® IPF Framework Installation 7](#_Toc144456640)

[2.3 Intel® IPF SDK Installation 7](#_Toc144456641)

[2.4 Intel® MEI Driver Installation 7](#_Toc144456642)

[2.5 Intel® Manageability and Security IPF Extension Providers Installation 8](#_Toc144456643)

[3 Capabilities 9](#_Toc144456644)

[3.1 Intel® PSR IPF Provider 9](#_Toc144456645)

[3.2 Intel® UPID IPF Provider 12](#_Toc144456646)

[3.3 Intel® AMT IPF Provider 13](#_Toc144456647)

[3.4 Intel® PBI IPF Provider 13](#_Toc144456648)

[3.5 Schemas 17](#_Toc144456649)

[4 Sample Code 19](#_Toc144456650)

[4.1 Intel® IPF SDK 19](#_Toc144456651)

[4.2 C++ Sample Code 19](#_Toc144456652)

[4.2.1 JSON C++ Library 19](#_Toc144456653)

[4.2.2 Intel® PSR Example 19](#_Toc144456654)

[4.2.3 Intel® UPID Example 20](#_Toc144456655)

[4.2.4 Intel® AMT Example 22](#_Toc144456656)

[4.2.5 Intel® PBI Example 23](#_Toc144456657)

[4.3 C# Sample Codes 25](#_Toc144456658)

[4.3.1 JSON.NET Framework 25](#_Toc144456659)

[4.3.2 Intel® PSR Example 25](#_Toc144456660)

[4.3.3 Intel® UPID Example 26](#_Toc144456661)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Target Platform | Intel® Manageability and Security IPF Extension Providers Software Package Revision | Revision Date |
| 1.0 | Meteor Lake mobile systems | 18.0.0017.0 | September, 2023 |
| 1.01 | Meteor Lake mobile systems | 18.0.0017.1 | October, 2023 |

Intended Audience

The target audience for this guide is software engineers creating local and remote management applications, tools, and services.

Customer Support

Please contact your Intel technical support representative or assigned customer enabling engineer.

§ §

# Introduction

## Intel® Device Discovery

Intel® Device Discovery is a new method for collecting the features and capabilities of endpoints deployed in the public or private networks. It enhances device management software with improved endpoint visibility to track, configure, and manage commercial clients.

The new discovery method utilizes the *Intel® Innovation Platform Framework (Intel® IPF),* which provides an abstraction layer that allows applications to interact with the platform using a simple namespace-based API. Objects in the namespace are implemented by a set of *Intel® IPF Providers* supported by a given platform. By using the abstraction layer, developers can rely on a consistent interface that isolates them from generational changes to hardware interfaces or low-level SDKs.

Intel IPF enables basic remote discovery on Intel-based PCs and comprehensive remote discovery for Intel vPro platforms, as directed by a larger set of Intel IPF Extension Providers supported for Intel vPro®. Comprehensive device discovery is expected to include platform type, device identity, device history, plus a list of all key manageability and security features on a given endpoint that may impact configuration, provisioning, and maintenance decisions. The set of discovery data that can be collected on a given device is expected to grow with each platform generation. The first iteration is expected to launch with Meteor Lake notebooks, with the set of *Intel® Manageability and Security IPF Extension Providers* described below.

## Intel® Manageability and Security IPF Extension Providers

*Intel® Manageability and Security IPF Extension Providers* are IPF Extension Providers enabling applications to read the platform data, set platform properties and take certain actions in the manageability domain.

The Intel® Manageability and Security IPF Extension Providers package is comprised of the following:

* Intel® Active Management Technology (Intel® AMT) IPF Provider
* Intel® Unique Platform ID (Intel® UPID) IPF Provider
* Intel® Platform Service Record (Intel® PSR) IPF Provider
* Intel® Platform Brand Identity (Intel® PBI) IPF Provider

Intel® AMT uses a hardware base out-of-band communication channel for remote device management regardless of the presence of a working operation system. More information about Intel® AMT can be found [here](https://www.intel.com/content/www/us/en/developer/articles/guide/getting-started-with-active-management-technology.html?wapkw=AMT).

Intel UPID is a globally unique and persistent identity for identifying the platform through its lifecycle. The platform UPID is immutable and attestable. More information about Intel UPID can be found [here](https://cdrdv2.intel.com/v1/dl/getContent/768498).

Intel PSR records platform history, make and model details, platform events, and other info to inform management and replacement decisions on a device. Intel PSR offers tamper-resistant ledgers with attestable authenticity. More information about Intel PSR can be found [here](https://www.intel.com/content/www/us/en/secure/developer/articles/technical/platform-service-record-2-0-introduction.html).

Intel Platform Brand Identity enables the system to report its platform type on query (Example: Intel vPro® Enterprise) with additional capabilities expected over time. More information about Intel Platform Brand Identity can be found [here](https://www.intel.com/content/www/us/en/secure/content-details/739179/meteor-lake-platform-brand-identity-overview-oem-enablement-technical-training-material.html?wapkw=brand%20identity).

## References

|  |  |  |
| --- | --- | --- |
| **Document** | Link | Notes |
| IPF Framework |  | Ver 2.1.10060 |
| IPF Windows SDK |  | Ver 2.1.10060 |
| Intel® Innovation Platform Framework Namespace User Guide | [732035](https://cdrdv2.intel.com/v1/dl/getContent/732035) |  |
| Intel® Innovation Platform Framework (Intel® IPF) Client Development Guide User Guide | [685592](https://cdrdv2.intel.com/v1/dl/getContent/685592) |  |
| Intel® Innovation Platform Framework (Intel® IPF) Provider Development Guide User Guide | [685487](https://cdrdv2.intel.com/v1/dl/getContent/685487) |  |

1. Contact Intel customer support for the latest version of these documents.

§ §

# System Requirements

The following system configurations are required to use and develop applications based on Intel® Manageability and Security IPF Extension Providers. Refer to Intel® IPF Client Development Guide for more information:

* Microsoft\* Windows 10 or 11 64-bit OS
* Intel vPro platform with Meteor Lake processor
* Intel® Management Engine Interface (Intel® MEI) driver
* [Visual C++ Redistributable](https://aka.ms/vs/17/release/vc_redist.x64.exe)
* Intel® IPF Framework Package
* Intel® IPF SDK Package

Intel UPID, Intel PSR, Intel AMT, and Intel PBI must be supported on a given platform to use the associated IPF Providers. Contact your sales representative for information on how to acquire supporting hardware and configure these capabilities on the platform.

## Visual C++ Runtime Installation

Visual C++ runtime redistributable can be downloaded from:

<https://aka.ms/vs/17/release/vc_redist.x64.exe>

## Intel® IPF Framework Installation

Refer section 1.3 for details on where to download Intel® IPF framework 2.1 package.

Run the installer in the package to install Intel® IPF framework package.

To verify if Intel® IPF framework is installed successfully, open Windows “Device Manager”, and check if “Intel® Innovation Platform Framework Extensible Framework Component” is listed under “Software Component”.

## Intel® IPF SDK Installation

Refer section 1.3 for details on where to download Intel® IPF SDK. Refer to the manual in the SDK on the pre-requisites and installation instructions.

## Intel® MEI Driver Installation

Intel® MEI Driver should have been pre-installed on the system as part of BKC (Best-Known Configuration) image. To verify if the driver has been installed, open Windows “Device Manager” and check if “Intel® Management Engine Interface” is listed under “System Devices”.

Intel® MEI Driver can be downloaded from:

<https://www.intel.com/content/www/us/en/download/682431/intel-management-engine-drivers-for-windows-10-and-windows-11.html>

## Intel® Manageability and Security IPF Extension Providers Installation

The Intel Manageability and Security IPF Extension Providers package contains the following providers:

* Intel AMT IPF Provider – AmtProvider.dll
* Intel UPID IPF Provider – UpidProvider.dll
* Intel PSR IPF Provider – PsrProvider.dll
* Intel PBI IPF Provider – PbiProvider.dll

Running the ‘setup.cmd install’ through a command prompt as administrator will install all these providers to the platform. “setup.cmd” script is included in the package.  
  
Alternatively, these providers can be individually installed on the device by installing two INF files provided in the package. Right click each INF file and click ’install’ to install the provider:

• IPF\_XXX\_provider\_ext.inf (Install this file first)  
 • IPF\_XXX\_provider\_sw.inf

where ‘XXX’ is either AMT, UPID, PSR or PBI.

To verify if a provider is installed successfully, open Windows “Device Manager” and check if the component name of the provider is listed under “Software components”.

The following table describes the component name for each Manageability IPF provider.

|  |  |
| --- | --- |
| **Provider Name** | Software Component Name |
| Intel® AMT IPF Provider | Intel® Manageability AMT Provider |
| Intel® UPID IPF Provider | Intel® Manageability UPID Provider |
| Intel® PSR IPF Provider | Intel® Manageability PSR Provider |
| Intel® PBI IPF Provider | Intel® Manageability PBI Provider |

§ §

# Capabilities

Each provider communicates its capabilities to applications using JSON schemas. See the IPF Provider Development Guide document for more information.

* The data schema declares the properties or objects exposed by a provider.
* The command schema defines the supported commands and their arguments.
* The result schema defines the return values supported by the commands.

Schemas of Intel Manageability and Security IPF Extension Providers are included in section 3.5 for reference.

The capabilities are organized into nodes that are populated by providers. Each node is represented using a hierarchical node path. All node paths in the manageability providers start with “Platform” root node, followed by a predefined “Manageability” node. The third node in the node path represents the logical device of a provider.

The following table describe the node path prefix for each provider.

|  |  |  |
| --- | --- | --- |
| **Provider** | Logical Device | Node Path Prefix |
| AMT IPF Provider | AMT | Platform.Manageability.AMT |
| UPID IPF Provider | UPID | Platform.Manageability.UPID |
| PSR IPF Provider | PSR | Platform.Manageability.PSR |
| PBI IPF Provider | PBI | Platform.Manageability.PBI |

An absolute node path always starts with a node path prefix, followed by a dot and child node path defined in the following sections.

For example, the “Major” capability in the PSR IPF provider has the following node path:

Platform.Manageability.PSR.Configuration.FWVersion.Major

A square bracket “[]” is appended to a capability child node path if the property is an array. For example, the “Event” capability in the first element of CriticalEvents array in the PSR IPF provider has he following node path:

Platform.Manageability.PSR. Events. CriticalEvents[0]. Event

The following sections describe the capabilities of each provider.

## Intel® PSR IPF Provider

| **Capability Name** | Data Type | Child Node Path | Capability Type |
| --- | --- | --- | --- |
| Major | integer | Configuration.FWVersion | Get |
| Minor | integer | Configuration.FWVersion | Get |
| Hotfix | integer | Configuration.FWVersion | Get |
| Build | integer | Configuration.FWVersion | Get |
| Major | integer | Configuration.PSRVersion | Get |
| Minor | integer | Configuration.PSRVersion | Get |
| RecordStatus | enum | Configuration | Get |
| UserNonce | string | Configuration | Get |
| CSMENonce | string | Configuration | Get |
| PSRHash | string | Configuration | Get |
| Signature | string | Configuration | Get |
| SignatureMechanism | enum | Configuration | Get |
| CertChain | array | Configuration | Get |
| UPID | string | Genesis | Get |
| PSRID | string | Genesis | Get |
| GenesisDate | string | Genesis | Get |
| OEMInfo | string | Genesis | Get |
| OEMMakeInfo | string | Genesis | Get |
| OEMModelInfo | string | Genesis | Get |
| CountryofOrigin | string | Genesis | Get |
| OEMDataStorage | string | Genesis | Get |
| CountryofOrigin | string | Genesis | Get |
| OEMDataStorage | string | Genesis | Get |
| SupportedPlatformCapabilities | array | Capabilities | Get |
| S0Runtime | integer | Counters | Get |
| S3Count | integer | Counters | Get |
| S4Count | integer | Counters | Get |
| S5Count | integer | Counters | Get |
| WarmResetCount | integer | Counters | Get |
| ISHConnectionCount | integer | Counters | Get |
| CSMEResetCount | integer | Counters | Get |
| PRTCResetCount | integer | Counters | Get |
| CSMERecoveryCount | integer | Counters | Get |
| CSMEDAMCount | integer | Counters | Get |
| CSMEUnlockedCount | integer | Counters | Get |
| PSRSVNIncrementedCount | integer | Counters | Get |
| ExcessiveShockCount | integer | Counters | Get |
| ExcessiveOperationalTemperatureCount | integer | Counters | Get |
| NumberofCriticalEvents | integer | Events | Get |
| EventID | integer | Events.CriticalEvents[] | Get |
| Source | enum | Events.CriticalEvents[].EventDetails | Get |
| Action | string | Events.CriticalEvents[].EventDetails | Get |
| Status | enum | Events.CriticalEvents[].EventDetails | Get |
| EventName | string | Events.CriticalEvents[] | Get |
| EventTimestamp | string | Events.CriticalEvents[] | Get |
| PSRBlob | string | BinaryBlob | Get |
| SetUserNonce |  | Configuration | Command |

The following table describes the value of the RecordStatus capability under the Platform.Manageability.PSR.Configuration node path.

|  |  |  |
| --- | --- | --- |
| **Enum Value** | Numeric Value | Comments |
| “not started” | 0 | PSR logging was not started yet |
| “started” | 1 | PSR logging was started and is currently running |
| “stopped” | 2 | PSR logging was stopped due to some critical event |

The following table describes the value of the EventName, EventID capabilities under the Platform.Manageability.PSR.Events.CriticalEvents[] node path.

| **EventName** | EventID | Comments |
| --- | --- | --- |
| Reserved | 0-7 |  |
| Log Started | 8 | First event in the log. Generate when the log is initiated. |
| Log Ended | 9 | Event added add the end of the log. This is to have the timestamp when the log is retrieved. |
| Log Full | 10 | Indicates log that there is not more free storage to room in the log to add more events to the log |
| Reserved | 11-15 |  |
| Replay Protection Infrastructure Failure | 16 | The event is logged when a Replay Protection infrastructure failure is detected. E.g., loss of RTC power if AR is RTC based. |
| Log Missing | 17 | Event is logged if one of the files that are used to store the PSR is found missing by the PSR FW |
| Log Integrity Compromised | 18 | Event is logged if PSR is present but either its integrity or replay protection is found to be compromised by the PSR FW |
| PRTC Reset | 19 | Event is logged if PRTC failure is detected |
| Log in Recovery State | 20 | CSME FW has entered a state/condition where the PSR cannot operate or is not recording. When normal operation is resumed, an event indicating that such a state happened is added. (and the recoding continues as before) |
| DAM State Entered | 21 | CSME has entered DAM state |
| Unlocked State Entered | 22 | CSME FW has been unlocked for debugging. During that time, PSR is not available and is not recording. When normal operation is resumed. an event indication of that state is added. (and the recoding continues as before) |
| PSR SVN Incremented | 23 | There is an increase of the Firmware's SVN (via a TCB recovery) or the SVN of the PSR process (via a FW updated that increases the SVN |
| Reserved | 24-31 |  |
| Chassis Intrusion Detected | 32 | Platform has had a chassis intrusion as determined by the BIOS |
| Excessive Shock | 33 | Platform has experienced an excessive shocked as detected by the ISH |
| Excessive Operational Temperature | 34 | Platform has reached the specified excessive operational temperature as managed by the EC |
| Erase | 35 | Platform Erase Action result |
| Reserved | 36-255 |  |

## Intel® UPID IPF Provider

|  |  |  |  |
| --- | --- | --- | --- |
| **Capability Name** | Data Type | Child Node Path | Capability Type |
| FeatureSupported | boolean | Configuration | Get |
| AttestationSupported | Boolean | Configuration | Get |
| FeatureState | boolean | Configuration | Get |
| FeatureOSControlState | boolean | Configuration | Get |
| OEMPlatformID | string | Configuration | Get |
| CSMEPlatformID | string | Configuration | Get |
| UPID | string | Configuration | Get |
| RefurbishCounter | integer | Configuration | Get |
| HWGen | integer | Configuration | Get |
| OEMID | integer | Configuration | Get |
| GetSignature |  | Configuration | Command |
| GetCertChain |  | Configuration | Command |
| SetFeatureState |  | Configuration | Command |

## Intel® AMT IPF Provider

|  |  |  |  |
| --- | --- | --- | --- |
| **Capability Name** | Data Type | Child Node Path | Capability Type |
| AMTVersion | string | Configuration | Get |
| AMTMode | enum | Configuration | Get |
| ProvisioningState | enum | Configuration | Get |
| ConnectionStatus | enum | Configuration | Get |
| CIRAConnectionStatus | enum | Configuration | Get |
| KVMSessionActive | boolean | Configuration | Get |
| IderSessionActive | boolean | Configuration | Get |
| SoLSessionActive | boolean | Configuration | Get |
| FQDN | string | Configuration | Get |
| PkiFQDNSuffix | string | Configuration | Get |
| WiredNetworkEnabled | boolean | Configuration | Get |
| WiredNetworkDHCPEnabled | string | Configuration | Get |
| WiredNetworkIPAddress | string | Configuration | Get |
| WiredNetworkMACAddress | string | Configuration | Get |
| WiredNetworkLinkStatus | string | Configuration | Get |
| WirelessNetworkEnabled | boolean | Configuration | Get |
| WirelessNetworkDHCPEnabled | boolean | Configuration | Get |
| WirelessNetworkIPAddress | string | Configuration | Get |
| WirelessNetworkMACAddress | string | Configuration | Get |
| WirelessNetworkLinkStatus | string | Configuration | Get |
| NumberofCertificateHashes | integer | Configuration | Get |
| Active | boolean | Configuration. CertificateHashes[] | Get |
| Hash | string | Configuration. CertificateHashes[] | Get |
| Name | string | Configuration. CertificateHashes[] | Get |
| HashAlgorithm | enum | Configuration. CertificateHashes[] | Get |
| Unprovision |  | Configuration | Command |

## Intel® PBI IPF Provider

The Intel PBI IPF Provider enables applications to collect system configuration data based on the Intel Platform Brand Identity of a given system. The data set is refreshed by the Intel chipset on every boot to confirm platform-defining characteristics such as Intel® processor SKU and Intel® ME firmware version.

The table below shows the discoverable “info” data set for Meteor Lake notebooks. The “info” data set is intended to be used by ISVs to discover the brand, version, and feature information.

|  |  |  |  |
| --- | --- | --- | --- |
| **Capability Name** | Data Type | Child Node Path | Capability Type |
| Brand | enum | Info | Get |
| Version | string | Info | Get |
| FeatureName | enum | Info.Features[] | Get |
| FeaturePresence | enum | Info.Features[].FeatureInfo | Get |
| FeatureActive | enum | Info.Features[].FeatureInfo | Get |

The Brand capability has one of the following 3 enumerated values:

* Intel vPro® Enterprise
* Intel vPro® Essentials
* Intel® Based PC

The Version indicates the CSME FW version number.

FeaturePresence can be “Present”, “Absent” or “Unknown”. A “Present” feature can be either active or inactive.

FeatureActive has one of the following 4 enumerated values:

* “Active”: indicating feature is present, enabled, configured and ready for servicing
* “Inactive”: Indicating feature is present but is either disabled, or not configured, as such it is not ready for servicing
* “N/A”: indicating feature is absent and feature state is not applicable
* “Unknown”: indicating feature readiness is unknown

FeatureName value is defined as below.

* Intel(R) Unique Platform ID
* Intel(R) Platform Service Record
* Intel(R) Standard Manageability
* Intel(R) AMT
* Out of band KVM Remote Control
* Intel(R) One Click Recovery
* Intel(R) Remote Platform Erase
* Out of band over TBT Dock
* Intel(R) Runtime BIOS Resilience
* Intel(R) System Resources Defense
* Intel(R) System Security Report
* NVMe based FW Recovery
* Intel(R) ICPS

The table below shows the discoverable “configuration” data set for Meteor Lake notebooks. The “configuration” data set is intended to be used by ISVs or OEMs to attest the authenticity of data exposed by this IPF provider.

|  |  |  |  |
| --- | --- | --- | --- |
| **Capability Name** | Data Type | Child Node Path | Capability Type |
| Major | integer | Configuration. FeatureMapTableRevision | Get |
| Minor | integer | Configuration. FeatureMapTableRevision | Get |
| Hotfix | integer | Configuration. FeatureMapTableRevision | Get |
| Build | integer | Configuration. FeatureMapTableRevision | Get |
| RequestedBrand | enum | Configuration | Get |
| RequestedBrandEntitlements | integer | Configuration | Get |
| QualifiedBrand | enum | Configuration | Get |
| QualifiedBrandEntitlements | integer | Configuration | Get |
| QualifiedFeatures | integer | Configuration | Get |
| QualifiedFeaturesPolicy | integer | Configuration | Get |
| FeaturePolicyName | enum | Configuration.FeaturesConfiguration[] | Get |
| FeatureClass | enum | Configuration.FeaturesConfiguration[].FeatureConfig | Get |
| FeatureSupported | enum | Configuration.FeaturesConfiguration[].FeatureConfig | Get |
| FeatureEnabled | enum | Configuration.FeaturesConfiguration[].FeatureConfig | Get |

The RequestedBrand and QualifiedBrand capabilities have one of the following 3 enumerated values:

* Intel vPro® Enterprise
* Intel vPro® Essentials
* Intel® Based PC

The 32-bit value of ReuqestedBrandEntitlements and QualifiedBrandEntitlements is defined as:

| **Bit** | Entitlement |
| --- | --- |
| 0 | Intel® Connectivity Performance Suite (ICPS) is entitled if set |
| 1 | Intel vPro® Enterprise if set |
| 2 | Intel vPro® Essentials if set |
| 4 | Intel® Based PC if set |
| 31 | Intel® ICPS ICM (Intel Connectivity Management) entitlement is enabled if set |
| All other bits | Reserved for future use |

A list of valid feature policy names is defined below:

* Intel(R) Platform Service Record Policy
* Intel(R) Standard Manageability Policy
* Intel(R) AMT Policy
* Out of band KVM Remote Control Policy
* Intel(R) One Click Recovery Policy
* Intel(R) Remote Platform Erase Policy
* Out of band over TBT Dock Policy
* Intel(R) Runtime BIOS Resilience Policy
* Intel(R) System Resources Defense Policy
* Intel(R) System Security Report Policy
* NVMe based FW Recovery Policy
* Intel(R) ICPS Policy

FeatureClass has one of the following 3 enumerated values:

* “Mandatory”: must be supported or present on the platform
* “Optional”: can be either supported or unsupported on the platform
* “Not allowed”: shall not be supported or must be absent on the platform

FeatureSupported has one of the following 3 enumerated values:

* “Supported”: feature is supported on the platform
* “Unsupported”: feature is not supported on the platform
* “Unknown”: feature support state on the platform is unknown

FeatureSupported and FeaturePresence are equivalent in this release.

FeatureEnabled has one of the following 4 enumerated values:

* “Enabled”: indicating a particular feature is enabled on the platform
* “Disabled”: indicating a particular feature is disabled on the platform
* “N/A”: indicating a particular feature is not supported and state is not applicable
* “Unknown”: indicating enabled state of a particular feature is unknown

The following table defines bitmap of the 64-bit *QualifiedFeatures* and QualifiedFeaturePolicy capabilities.

|  |  |
| --- | --- |
| **Bit** | Feature |
| 1 | Intel® Standard Manageability |
| 2 | Intel® AMT |
| 3 | Intel® One-Click Recovery |
| 4 | Intel® Remote Platform Erase |
| 5 | Intel® Platform Service Record |
| 18 | Out-of-band KVM Remote Control |
| 27 | Out-of-band over Thunderbolt™ dock |
| 28 | Intel® Unique Platform ID |
| 32 | Intel® System Security Report |
| 33 | Intel® Runtime BIOS Resilience |
| 34 | Intel® System Resource Defense |
| 35 | NVMe-based firmware recovery |
| 63 | Intel® Connectivity Performance Suite |
| All other bits | Reserved for future use |

For “*QualifiedFeatures*” capability, value of each bit indicates whether a specific feature is allowed or not:

0 – Not allowed

1 – Allowed

For “*QualifiedFeaturesPolicy*” capability, value of each bit indicates whether an allowed feature is mandatory or optional:

0 – Optional

1 – Mandatory

## Schemas

Schemas of each provider are embedded in the following table for reference.

| **Provider** | Data Schema | Command Schema | Result Schema |
| --- | --- | --- | --- |
| Intel® AMT IPF Provider |  |  |  |
| Intel® UPID IPF Provider |  |  |  |
| Intel® PSR IPF Provider |  |  |  |
| Intel® PBI IPF Provider |  | N/A | N/A |

§ §

# Sample Code

Intel® IPF SDK and JSON library are required to build the following sample code.

## Intel® IPF SDK

Application developers can follow the IPF Client Development Guide on how to develop IPF client applications using either C# or C++.

Refer section 1.3 for details on where to download Intel® IPF 2.1 SDK and IPF Client Development Guide.

## C++ Sample Code

### JSON C++ Library

Get the header file “nlohmann/json.hpp” from the following github project:

<https://github.com/nlohmann/json>

### Intel® PSR Example

#include <iostream>

#include <fstream>

#include <stdio.h>

#include <string.h>

#include "IpfClient.h"

#include "nlohmann/json.hpp"

using Json = nlohmann::json;

static void ConsolePrint(const std::string& field, const std::string& value)

{

std::cout << field << std::endl << value << std::endl << std::endl;

}

int main(int argc, char \*argv[])

{

Ipf::ClientApiJson ipf;

auto jsonStream = ipf.GetNode("Platform.Manageability.PSR");

auto jsonSchema = ipf.GetSchema("Platform. Manageability.PSR", Ipf::SchemaType::DATA);

ConsolePrint("PSR data schema:", jsonSchema.dump(4));

ConsolePrint("PSR data:", jsonStream.dump(4));

// Set User Nonce

Json cmdobj = { {"SetUserNonce", {{"Nonce", “aGVsbG8=”}} }};

auto result = ipf.ExecuteCommand("Platform.Manageability.PSR.Configuration", cmdobj);

jsonStream = ipf.GetNode("Platform.Manageability.PSR");

ConsolePrint("PSR data:", jsonStream.dump(4));

int ver = ipf.GetValue("Platform.Manageability.PSR.Configuration.FWVersion.Major");

ConsolePrint("Major FW Version:", std::to\_string(ver));

}

### Intel® UPID Example

#include <iostream>

#include <fstream>

#include <stdio.h>

#include <string.h>

#include "IpfClient.h"

#include "nlohmann/json.hpp"

using Json = nlohmann::json;

static void ConsolePrint(const std::string& field, const std::string& value)

{

std::cout << field << std::endl << value << std::endl << std::endl;

}

int main(int argc, char \*argv[])

{

Ipf::ClientApiJson ipf;

auto jsonSchema = ipf.GetSchema("Platform. Manageability.UPID ", Ipf::SchemaType::DATA);

auto jsonStream\_upid = ipf.GetNode("Platform. Manageability.UPID ");

ConsolePrint("UPID data schema:", jsonSchema.dump(4));

ConsolePrint("UPID data:", jsonStream\_upid.dump(4));

bool res = ipf.GetValue("Platform. Manageability.UPID.Configuration.FeatureSupported ");

ConsolePrint("UPID feature supported:", std::to\_string(res));

Json cmdobj = {{"GetSignature", {{"DataToSign", “aGVsbG8=”}} }};

auto result = ipf.ExecuteCommand("Platform.Manageability.UPID.Configuration", cmdobj);

ConsolePrint("UPID signature:", result.dump(4));

cmdobj = {{"GetCertChain", “null” }};

result = ipf.ExecuteCommand("Platform.Manageability.UPID.Configuration", cmdobj);

ConsolePrint("UPID cert chain:", result.dump(4));

return 0;

}

### Intel® AMT Example

#include <iostream>

#include <fstream>

#include <stdio.h>

#include <string.h>

#include "IpfClient.h"

#include "nlohmann/json.hpp"

using Json = nlohmann::json;

static void ConsolePrint(const std::string& field, const std::string& value)

{

std::cout << field << std::endl << value << std::endl << std::endl;

}

int main(int argc, char \*argv[])

{

Ipf::ClientApiJson ipf;

auto jsonSchema = ipf.GetSchema("Platform.Manageability.AMT", Ipf::SchemaType::DATA);

auto jsonStream = ipf.GetNode("Platform.Manageability.AMT");

ConsolePrint("AMT data schema:", jsonSchema.dump(4));

ConsolePrint("AMT data:", jsonStream.dump(4));

jsonSchema = ipf.GetSchema("Platform.Manageability.AMT", Ipf::SchemaType::COMMAND);

ConsolePrint("AMT command schema:", jsonSchema.dump(4));

jsonSchema = ipf.GetSchema("Platform.Manageability.AMT", Ipf::SchemaType::RESULT);

ConsolePrint("AMT result schema:", jsonSchema.dump(4));

Json cmdobj = { {"Unprovision", "null"} };

auto result = ipf.ExecuteCommand("Platform.Manageability.AMT.Configuration", cmdobj);

ConsolePrint("Unprovisioning result:”, result.dump(4));

return 0;

}

### Intel® PBI Example

#include <iostream>

#include <fstream>

#include <stdio.h>

#include <string.h>

#include "IpfClient.h"

#include "nlohmann/json.hpp"

using Json = nlohmann::json;

static void ConsolePrint(const std::string& field, const std::string& value)

{

std::cout << field << std::endl << value << std::endl << std::endl;

}

int main(int argc, char \*argv[])

{

Ipf::ClientApiJson ipf;

auto jsonSchema = ipf.GetSchema("Platform.Manageability.PBI", Ipf::SchemaType::DATA);

auto jsonStream = ipf.GetNode("Platform.Manageability.PBI");

ConsolePrint("PBI data schema:", jsonSchema.dump(4));

ConsolePrint("PBI data:", jsonStream.dump(4));

int res = ipf.GetValue("Platform.Manageability.PBI.Configuration.QualifiedBrandEntitlements ");

ConsolePrint("Qualified brand entitlements:", std::to\_string(res));

std::string brand = ipf.GetValue("Platform.Manageability.PBI.Configuration.QualifiedBrand");

ConsolePrint("Qualified brand:", brand);

return 0;

}

## C# Sample Codes

### JSON.NET Framework

The JSON.NET framework from <https://www.newtonsoft.com/json> is required to build the following sample codes.

This framework is included in the IPF SDK.

### Intel® PSR Example

using Ipf;

using Ipf.CLI;

using Newtonsoft.Json.Linq;

using System;

namespace ConsoleApp1

{

internal class Program

{

private static void ConsolePrint(String field, String value)

{

Console.WriteLine(field);

Console.WriteLine(value);

Console.WriteLine();

}

private static void Main(string[] args)

{

try

{

ClientApiJson ipf\_client = new ClientApiJson;

JToken psr\_node = ipf\_client.GetNode("Platform.Manageability.PSR");

ConsolePrint("PSR node:", psr\_node.ToString());

JObject psr\_schema = ipf\_client.GetSchema("Platform.Manageability.PSR", SchemaType.DATA);

ConsolePrint("PSR data schema:", psr\_schema.ToString());

JToken firmVer = ipf\_client.GetValue("Platform.Manageability.PSR.Configuration.FWVersion.Major");

ConsolePrint("PSR firmware version major:", firmVer.ToString());

}

// Catch IPF Exceptions

catch (IpfManagedException.IpfException e)

{

Console.WriteLine("Got IpfException when accessing the namespace:");

Console.WriteLine(e.Message);

}

}

}

}

### Intel® UPID Example

using Ipf;

using Ipf.CLI;

using Newtonsoft.Json.Linq;

using System;

namespace ConsoleApp1

{

internal class Program

{

private static void ConsolePrint(String field, String value)

{

Console.WriteLine(field);

Console.WriteLine(value);

Console.WriteLine();

}

private static void Main(string[] args)

{

try

{

ClientApiJson ipf\_client = new ClientApiJson;

JToken upid\_node = ipf\_client.GetNode("Platform.Manageability.UPID");

ConsolePrint("UPID node:", upid\_node.ToString());

JObject upid\_schema = ipf\_client.GetSchema("Platform.Manageability.UPID", SchemaType.DATA);

ConsolePrint("UPID data schema:", upid\_schema.ToString());

upid\_schema = ipf\_client.GetSchema("Platform.Manageability.UPID", SchemaType.COMMAND);

ConsolePrint("UPID command schema:", upid\_schema.ToString());

upid\_schema = ipf\_client.GetSchema("Platform.Manageability.UPID", SchemaType.RESULT);

ConsolePrint("UPID result schema:", upid\_schema.ToString());

JToken platf\_id = ipf\_client.GetValue("Platform.Manageability.UPID.Configuration.CSMEPlatformID");

ConsolePrint("CSME Platform ID:", platf\_id.ToString());

string cmd = @"

{

""GetSignature"" :{

""DataToSign"": "" aGVsbG8= "" }

}";

JObject cmdobj = JObject.Parse(cmd);

JObject result = ipf\_client.ExecuteCommand("Platform.Manageability.UPID.Configuration", cmdobj);

ConsolePrint("UPID signature: “, result.ToString());

}

// Catch IPF Exceptions

catch (IpfManagedException.IpfException e)

{

Console.WriteLine("Got IpfException when accessing the namespace:");

Console.WriteLine(e.Message);

}

}

}

}

§ §