Hewlett Packard Enterprise

Redfish API implementation on iLO RESTful API for HPE iLO 4

Technical white paper

Contents

Introduction

When HPE pioneered the original Integrated Lights-Out (iLO) management processor, the industry responded with commodity baseboard management controllers, typically based on the Intelligent Platform Management Interface (IPMI) protocol. With the iLO 4 Management Engine, HPE innovated again by introducing an interface based on the modern REpresentational State Transfer (REST) approach being favored for systems and software. This so-called RESTful API provides robust control that benefits traditional IT architectures as well as administrators with cloud and Web-based data center infrastructures which achieve scalability by deploying large numbers of basic servers. Managing these scale-out architectures poses particular challenges for data center planners and administrators. Traditional interfaces like IPMI are not able to address the scale-out and cloud-based requirements for simplicity and security available in modern programming interfaces.

Redfish takes the work initially implemented as the iLO RESTful API and uses it as a foundation for an open, industry-standard specification and schema for today's cloud and Web-based data center infrastructures sponsored and controlled by the Distributed Management Task Force, Inc. (DMTF), a peer-review standards body recognized throughout the industry. Redfish establishes a new management standard for system control that is scalable, easy to use, and secure. HPE is a founding member of the Redfish specification and most major vendors are now participating in the effort. Redfish defines the industry standard for the Software Defined Data Center (SDDC) and the effort to modernize heterogeneous data centers. In addition, HPE ProLiant servers expose iLO RESTful API extensions, allowing customers to experience the full range of value-add features available from a programmable interface.

Redfish and Software Defined Compute (SDC) provide the control plane for IT infrastructure enabling customers to program simple configuration and maintenance tasks. At scale, these tasks have been extremely complex, fragile, and time consuming. Redfish also enables development of new higher-level automation and orchestration features, previously only possible in large-scale highly customized service provider environments.

Redfish is based on current software architecture, methods, tools, and scripting environments used in most IT environments. HPE customers can now benefit from adopting the Redfish standard into their HPE ProLiant Gen8[1] and Gen9 installed base, as well as on other non-HPE server products which implement to the standard.

This document will help you understand the benefits of using Redfish, how it's connected to the iLO RESTful API, and why the industry is transitioning to Redfish. You will also gain an understanding of how the technology works and integrates with multiple IT vendors. You will learn how you can become less dependent on existing protocols such as IPMI. Finally, the document provides information on Redfish implementation in iLO RESTful API on the iLO 4 v2.30 or later environment.

Some of the information in this document is based on DMTF documentation. Links to the DMTF Redfish specification and other DMTF Redfish-related documentation can be found in the resources section of this document.

[1] ProLiant Gen8 servers will experience reduced Redfish functionality

Transitioning from the HPE initial RESTful implementation to the Redfish API

Since the introduction of the REpresentational State Transfer (REST) architectural style, it has quickly started replacing the Simple Object Access Protocol (SOAP) as the most widely used method in cloud ecosystems and the Web API community. For systems management, the REST API is simpler and easier to learn than traditional tools like IPMI.

HPE ProLiant Gen9 servers with the iLO RESTful API offer many benefits over traditional server management technologies (see table 1).

Table 1. iLO RESTful API features and benefits

	Features	Benefits
Provision	• Retrieve server information details • UEFI configuration • Manage UEFI Secure Boot • Boot Order Management • iLO 4 configuration • Find iLO NIC MAC/Networking details • iLO User Account Management • Virtual Media Mount/Unmount	• Easy access and configuration for remote management • Manage server settings and deploy at scale • Software Development Kit (SDK) provide Python sample code available on GitHub to get started with your server deployment
Monitor	• Diagnostics • System Memory Details • Smart Array and NIC Inventory • Get/Clear iLO and IML Logs • Get Power Detail	• Monitor servers remotely using commonly available tools or custom software
Health	• Active Health System download	• Engage with HPE support using AHS logs
Manage	• Reset Server • Reset iLO	• Secure your data and remove critical information prior to server retirement

Customers asked for a modern interface in which they expect APIs to use the protocols, structures, and security models that are common in emerging cloud interfaces. Specifically, customers asked for RESTful protocols that express data in JavaScript Object Notation (JSON) formats.

Systems managers find themselves in heterogeneous environments where they are unable to use a common language to monitor cloud and Web-based infrastructures. Data center administrators, programmers, and software development/IT operations personnel (DevOps) must deal with a multitude of devices that didn't exist when IPMI was created, and struggle to integrate these same devices into their environments. RESTful protocols addressed these issues with an interface that could adapt across a variety of server applications, enhance interoperability across multiple server environments, simplify management, and give systems administrators a single common language. The iLO RESTful API on HPE ProLiant Gen9 servers provided these capabilities and became a precursor to Redfish.

HPE understood that the iLO RESTful API has tremendous potential for our customers, and that long-term vision requires an industry standard for use across heterogeneous data centers. That's why HPE, through the DMTF, acted as one of the drivers in the effort to make Redfish the common standard for managing systems and devices in scale-out environments. The effort includes industry stakeholders such as Intel®, Dell, Emerson, Lenovo, Microsoft®, and others to reach consensus on the Redfish Standard.

The Redfish specification

The Redfish specification, a standard for data center and systems management delivers improved performance, functionality, scalability, and security. Designed to meet the expectations of end users for simple and interoperable management of modern scalable platform hardware, Redfish takes advantage of widely used technologies to speed implementation and help system administrators be more effective.

DMTF president Jeff Hilland describes Redfish as "based on the tools and scripting environments most users already have, enabling feature-rich remote management while being compatible with the existing tool set. Redfish was built from the ground up to scale to the modern multiple server environments encountered in today's enterprise, hyper-scale, and cloud infrastructures".

The Redfish specification uses a hypermedia data model represented within a RESTful interface. The definitions for Redfish resources are defined to conform to the OData schema, and are delivered in a JSON-based payload. The specification includes methods for discovery,

event handling, and data center resource management. Redfish is capable of numerous implementations on a wide variety of system architectures, while maintaining a single interface definition.

You can explore the complete DMTF Redfish specification at dmtf.org/standards/redfish.

HPE extensions to Redfish

The Redfish data model enables many common server management tasks but also provides a well-defined extension mechanism for implementers to define additional capabilities. HPE uses the extension mechanism to enable a large additional set of status, health, and configuration data. For example, the HPE extensions to Redfish include UEFI system configuration, Smart Storage status, plus additional actions that can be performed on HPE servers, and extended iLO 4 management. HPE extensions are available to the API in both Redfish and legacy iLO RESTful modes.

HPE timetable for Redfish implementation

The iLO RESTful API provides a modern programmable interface and a lightweight data model specification that is simple, remote, secure, and extensible. In the autumn of 2014, the iLO RESTful API introduced this architectural style for HPE ProLiant Gen9 servers with HPE iLO 4 2.0.

HPE now introduces the iLO RESTful API with Redfish conformance. This industry standard Software Defined Compute (SDC) infrastructure management API is being implemented into ProLiant Gen9 servers and will function across heterogeneous environments. The release and HPE implementation timeline for the iLO RESTful API and Redfish Standard follows here:

- August 2014—HPE implemented iLO RESTful API on all ProLiant Gen9 systems as the next generation of server management interface. Enabling iLO 4 and UEFI configurations.
- August 2015—DMTF announced the release of Redfish 1.0, a standard for data center and systems management. Redfish documentation and schema detail available for download at dmtf.org/standards/redfish.
- September 2015—HPE server customers can upgrade their ProLiant install base to gain Redfish "conformance" through the iLO 4 Firmware update at hpe.com/info/restfulapi.

Redfish conformance

HPE was the first to provide an iLO RESTful API interface for embedded management. And now HPE is first to implement Redfish conformance[2] in HPE ProLiant Gen9 servers already embedded with iLO RESTful APIs. The Redfish specification is ideal for enterprise environments and service providers that self-provision and automate their data centers, and for programmers or DevOps looking to enhance data center automations and adapt faster to different workloads.

iLO RESTful API is available in HPE ProLiant Gen9 servers with iLO 4 FW v2.00 and later, and Moonshot Chassis Manager 1.30 and later enterprise ProLiant servers. HPE iLO 4 conformance for Redfish (with firmware v2.30 or later) includes:

- iLO 4 preserves legacy iLO RESTful API compatibility for ProLiant servers through Gen9. A future major iteration of the iLO product will remove the legacy support.
- iLO 4 implements Redfish by creating a "dual-mode" API, where the mode is selected by the client based upon the existence or absence of the Redfish-specified "OData-Version" HTTP header.
- The Redfish-defined resource types are Redfish conformant (such as "Chassis" and "ComputerSystem").
- OData v4 properties as required in the Redfish specification to enable compatibility with OData-enabled client software.
- Many HPE value-add resource types implemented by iLO 4 are Redfish-conformant in that they include OData properties and Redfish-specified property patterns (such as "Links") even though the schema are not found in the Redfish standard. Some resources like BIOS configuration are not yet OData v4 conformant.

[2] For further conformance details on Redfish standard 1.0, see the DMTF Specification 1.0 release at dmtf.org/standards/redfish.

Note
The HPE value-add resource types implemented by external providers (UEFI for example) are not yet Redfish conformant and continue to behave exactly as existing iLO RESTful resources. HPE plans for these resources to become Redfish conformant in the future.

Redfish mode

The iLO RESTful API 1.x expresses the Uniform Resource Identifier (URI) protocol version as "/rest/v1". Redfish 1.0 expresses the starting URI as "/redfish/v1/". Only version 1 is available now, and it complies with the version 1 Redfish specification. iLO 4 makes the same data available at both URIs.

The behavior of a resource (to behave in Redfish or legacy mode) is not dependent upon the URI used to access the resource, but upon the presence or absence of the HTTP OData-Version header. Since, it is based on OData v4, Redfish implementation requires the starting metadata OData header to be version 4 (OData v4). This means that In HPE ProLiant Gen9 servers, the presence or absence of this OData header is the switch iLO uses to enter Redfish mode or remain in legacy HPE RESTful mode. The URIs do not determine mode.

Global data model differences

These global properties and configurable settings are different from the iLO RESTful API and define behavior within the Redfish model.

Navigational links

Redfish changes the linking between resources. Where the HPE RESTful URIs places all links as "href" inside a "links" sub-object, Redfish makes the following basic changes:

- "href" renamed to "@odata.id"
- "links" sub-object renamed to "Links"
- Child-resource links are removed from the links sub-object and placed directly into the root of the main object (promoted out of the links sub-object). Associated resource links (unlike links to child resources) remain in the "Links" sub-object. This structure facilitates the OData "expand" operation which can in-line child resources. "Expand" is not yet supported by iLO.

iLO 4 implements Redfish linking by:

- Adding "@odata.id" style links to root-level child resources in both legacy and Redfish modes
- Rendering the links sub-object as "links" or "Links" depending upon mode (OData-Version header):
  - Redfish mode: "links" sub-object is hidden, "Links" is shown. All Links are "@odata.id" style with trailing slash. Additionally, some Links ("self", for example) are hidden.
  - Legacy mode: "links" shown, "Links" hidden. All links are "href" style.

Table 2. Navigational links

Mode	"links"	"Links"	Root @odata.id links	Link property name	Self link
Redfish	Hidden	Included	Included	@odata.id	/@odata.id
legacy	Included	Hidden	Included	href	/links/self/href

General schema changes

Table 3 lists of schema changes that apply to all Redfish conformant resources.

Table 3. Schema changes

iLO RESTful API property (JSON pointer)	Redfish property
/links	/Links
/links/self/href	/@odata.id
/Type	/@odata.type
/Modified	<removed>
/AvailableActions	/Actions (alternative schema)

Redfish data-type

In Redfish, the "@odata.type" properties in table 4 must be added to the specific Redfish resource types (where "x.y.z" is replaced with the appropriate schema version information).

Table 4. Schema changes

iLO RESTful API type	iLO RESTful API collection Membertype	Redfish @odata.type
Collection	<any member type>	<membertype>Collection
Collection	SchemaFile	JsonSchemaFileCollection.JsonSchemaFileCollection
SchemaFile		JsonSchemaFile.1.0.0.JsonSchemaFile
PowerMetrics		Power.1.0.0.Power
ThermalMetrics		Thermal.1.0.0.Thermal
EthernetNetworkInterface		EthernetInterface.1.0.0.EthernetInterface
Collection	EthernetNetworkInterface	EthernetInterfaceCollection.EthernetInterfaceCollection

Specific resource differences

In this section you will find specific Redfish property and setting changes, compared to the iLO RESTful API.

Collection changes

Collections are different in Redfish, but retain the same overall architectural style. For example, iLO RESTful API has a generic collection type with a Membertype specifying the type of resources collected, Redfish breaks collections into type-specific collections. For example, RESTful Collection of "MemberType=ComputerSystem" become Redfish type "ComputerSystemCollection". You can see more of these changes in table 5.

Table 5. Collection changes

iLO RESTful property (JSON pointer)	Redfish property
/MemberType	Removed—the membertype is now part of the @odata.type (no generic collection type)
/Total	/Members@odata.count
/Items	Removed—see Redfish specifications for alternate method of inlining members using "expand".
/links/Member	/Members

Status block changes

Many iLO RESTful API and Redfish resources contain a property called "Status" that is a sub-object containing several status properties. Minor changes were made in Redfish (table 6), but these will affect many resources throughout the model.

Table 6. Status block changes

iLO RESTful property (JSON pointer)	Redfish property
/HealthRollUp	/HealthRollup (note case-change)

Extended error response changes

The extended error response, now called ExtendedInfo in Redfish, is significantly different and table 7 maps the old to the new responses. If the OData header is present on a request, the legacy properties will be hidden.

Table 7. Extended error/ExtendedInfo response changes

iLO RESTful property (JSON pointer)	Redfish property
/Type	
/Name	
/Messages	/@Message.ExtendedInfo
/Messages/MessageID	/@Message.ExtendedInfo/MessageId
	/code
	/message

For a complete list of specific Redfish 1.0 property changes, see Table A1 in the Appendix.

Custom POST actions

Table 8 shows that POSTing to an action returned in the "Actions" property of a resource results in the execution of the custom action.

Table 8. Custom POST actions

iLO RESTful property (JSON pointer)	Redfish property
/AvailableActions	/Actions (Redfish-defined actions)
/Oem/Hp/AvailableActions	/Oem/Hp/Actions (HPE-specific actions)
/Actions/Oem (specified by Redfish)	
/AvailableActions[*]/Action	/Actions/#<typename>.<actionname> (e.g. #ComputerSystem.Reset)
	/Actions/#<typename>.<actionname>/target
/Actions/#<typename>.<actionname>/target	Combined with allowable values (next)

HPE OneView

HPE OneView offers a uniform way of interacting with resources by providing a iLO RESTful API for infrastructure management, and as a building block of the composable infrastructure in a SDDC architecture. OneView is the orchestration layer that allows you to use the iLO RESTful API to automate and effectively operate various components of the infrastructure from one location.

Summary

Redfish is a new industry standard specification and schema that meets the expectations of end users for simple, modern, and secure management of scalable platform hardware. One of the significant benefits Redfish offers to adopters is the potential for a secure, multi-node capable replacement for IPMI-over-LAN.

Redfish and HPE Software Defined Compute provides the control plane for IT infrastructure enabling customers to program simple configuration and maintenance tasks that at scale have been extremely complex, fragile, and time consuming. Redfish also allows customers to develop higher-level automation and orchestration features, previously only possible in large scale highly customized service provider environments. Redfish lets server administrators, development, and operations staff spend less time on the day-to-day maintenance and simplifies tools to develop advanced next generation capabilities required for their business. Redfish reduces vendor lock-in, increases productivity and security, lowers operating costs, and plays a crucial role in enabling next-generation infrastructures.

HPE is the first major system manufacturer to adopt the DMTF Redfish standard. The Redfish Standard enjoys broad vendor endorsement and defines the industry standard for Software Defined Compute.

Appendix A: Specific property changes

As of iLO 4 firmware 2.30, the following properties are renamed and/or moved in Redfish (remember the legacy properties still exist when not using the OData header). In some cases there is not yet (or may not be) a replacement Redfish property.

Table A1. Specific property changes

Resource type (@odata.type)	Legacy RESTful property	Redfish replacement property
Chassis.1	Version	none
Chassis.1	links/PowerMetrics	/Power
Chassis.1	links/ThermalMetrics	/Thermal
ComputerSystem.1	BIOSPOSTCode	none
ComputerSystem.1	Power	/PowerState
ComputerSystem.1	Version	none
ComputerSystem.1	VirtualSerialNumber	none
ComputerSystem.1	links/Logs	/LogServices
EthernetInterface.1	Autosense	/AutoNeg
EthernetInterface.1	FactoryMacAddress	/PermanentMACAddress
EthernetInterface.1	LinkTechnology	none
EthernetInterface.1	MacAddress	/MACAddress
Event.1	Events[]/MessageID	/Events[]/MessageId
EventDestination.1	HttpHeaders	none
EventDestination.1	TTLCount	none
EventDestination.1	TTLUnits	none
EventService.1	DeliveryRetryIntervalInSeconds	/DeliveryRetryIntervalSeconds
EventService.1	SubscriptionRemovalAction	none
EventService.1	SubscriptionRemovalTimeIntervalInMinutes	none
ExtendedInfo.1	definitions/Messages[]/MessageID	/definitions/Messages[]/MessageId
ExtendedInfo.1	Messages	/@Message.ExtendedInfo
LogEntry.1	Number	none
LogEntry.1	RecordId	/EventNumber
Manager.1	CommandShell/Enabled	/CommandShell/ServiceEnabled
Manager.1	CommandShell/ServiceEnabled	/CommandShell/ServiceEnabled

Table A1. Specific property changes (continued)

Resource type (@odata.type)	Legacy RESTful property	Redfish replacement property
Manager.1	Firmware	/FirmwareVersion
Manager.1	GraphicalConsole/Enabled	/GraphicalConsole/ServiceEnabled
Manager.1	GraphicalConsole/ServiceEnabled	/GraphicalConsole/ServiceEnabled
Manager.1	SerialConsole/Enabled	/SerialConsole/ServiceEnabled
Manager.1	SerialConsole/ServiceEnabled	/SerialConsole/ServiceEnabled
Manager.1	links/EthernetNICs	/EthernetInterfaces
Manager.1	links/NetworkService	/NetworkProtocol
ManagerNetworkProtocol.1	HTTP/Enabled	/HTTP/ProtocolEnabled
ManagerNetworkProtocol.1	HTTPS/Enabled	/HTTPS/ProtocolEnabled
ManagerNetworkProtocol.1	IPMI/Enabled	/IPMI/ProtocolEnabled
ManagerNetworkProtocol.1	KVMIP/Enabled	/KVMIP/ProtocolEnabled
ManagerNetworkProtocol.1	SNMP/Enabled	/SNMP/ProtocolEnabled
ManagerNetworkProtocol.1	SSDP/Enabled	none
ManagerNetworkProtocol.1	SSDP/Port	none
ManagerNetworkProtocol.1	SSH/Enabled	/SSH/ProtocolEnabled
ManagerNetworkProtocol.1	SessionTimeoutMinutes	none
ManagerNetworkProtocol.1	VirtualMedia/Enabled	/VirtualMedia/ProtocolEnabled
ManagerNetworkProtocol.1	links/SNMPService	Being added to the OEM section. Expect: /Oem/Hp/Links/SNMPService
MessageRegistry.1	Version	/RegistryVersion
Power.1	PowerAllocatedWatts	/PowerControl/PowerAllocatedWatts
Power.1	PowerAvailableWatts	/PowerControl/PowerAvailableWatts
Power.1	PowerCapacityWatts	/PowerControl/PowerCapacityWatts
Power.1	PowerConsumedWatts	/PowerControl/PowerConsumedWatts
Power.1	PowerRequestedWatts	/PowerControl/PowerRequestedWatts
ServiceRoot.1	Time	none
ServiceRoot.1	links/Schemas	/JSONSchema
ServiceRoot.1	links/SessionService	/SessionService
ServiceRoot.1	links/Sessions	/SessionService
Thermal.1	Fans[]/Context	/Fans[]/PhysicalContext (enum values are different)
Thermal.1	Fans[]/CurrentReading	/Fans[]/ReadingRPM
Thermal.1	Fans[]/Units	/Fans[]/ReadingRPM
Thermal.1	Temperatures[]/Context	/Temperatures[]/PhysicalContext
Thermal.1	Temperatures[]/CurrentReading	/Temperatures[]/ReadingCelsius
Thermal.1	Temperatures[]/Number	none
Thermal.1	Temperatures[]/Units	/Temperatures[]/ReadingCelsius

Appendix B: Terms and definitions

Most of these Redfish-related terms and definitions come from the DMTF Redfish specification 1.0 found at dmtf.org/standards/redfish.

Table B1. Terms and definitions

Baseboard Management Controller	**BMC**—An embedded device or service, typically an independent microprocessor or System-on-Chip with associated firmware, within a Computer System used to perform systems monitoring and management-related tasks, which are commonly performed out-of-band.
Collection	A Collection is a resource that acts as a container of other Resources. The members of a collection usually have similar characteristics. The container processes messages sent to the container. The members of the container process messages sent only to that member without affecting other members of the container.
CRUD	Create, Read, Update and Delete. Basic intrinsic operations used by any interface.
Event	A record that corresponds to an individual alert.
Managed System	In the context of this specification, a managed system is a system that provides information or status, or is controllable, via a Redfish-defined interface.
Message	A complete request or response, formatted in HTTP/HTTPS. The protocol, based on REST, is a request/response protocol where every Request should result in a Response.
Operation	The HTTP request methods which map generic CRUD operations. These are POST, GET, PUT/PATCH, HEAD, and DELETE.
OData	The Open Data Protocol, as defined in OData-Protocol.
OData Protocol	The Open Data Protocol (OData) is a Web protocol for querying and updating data which provides a way to unlock your data and free it from silos that exist in applications today. OData does this by applying and building upon Web technologies such as HTTP, Atom Publishing Protocol (AtomPub) and JSON to provide access to information from a variety of applications, services, and stores.
OData Service Document	The name for a resource that provides information about the Service Root. The Service Document provides a standard format for enumerating the resources exposed by the service that enables generic hypermedia-driven OData clients to navigate to the resources of the Redfish Service.
Redfish Alert Receiver	The name for the functionality that receives alerts from a Redfish Service. This functionality is typically software running on a remote system that is separate from the managed system.
Redfish Client	Redfish Client Name for the functionality that communicates with a Redfish Service and accesses one or more resources or functions of the Service.
Redfish Protocol	The set of protocols that are used to discover, connect to, and inter-communicate with a Redfish Service.
Redfish Schema	The Schema definitions for Redfish resources. It is defined according to OData Schema representation that can be directly translated to a JSON Schema representation.
Redfish Service	Also referred to as "Service". The collection of functionality that implements the protocols, resources, and functions that deliver the interface defined by this specification and its associated behaviors for one or more managed systems.
Redfish Service Entry Point	Also referred to as "Service Entry Point". The interface through which a particular instance of a Redfish Service is accessed. A Redfish Service may have more than one Service Entry Point.
Request	A message from a Client to a Server. It consists of a request line (which includes the Operation), request headers, an empty line and an optional message body.
Resource	A Resource is addressable by a URI and is able to receive and process messages. A Resource can be either an individual entity, or a collection that acts as a container for several other entities.
Resource Tree	A Resource Tree is a tree structure of JSON encoded resources accessible via a well-known starting URI. A client may discover the resources available on a Redfish Service by following the resource links from the base of the tree. **Note:** For Redfish client implementation: Although the resources are a tree, the references between resources may result in graph instead of a tree. Clients traversing the resource tree must contain logic to avoid infinite loops.
Response	A message from a Server to a Client in response to a request message. It consists of a status line, response headers, an empty line and an optional message body.
Service Root	The term Service Root is used to refer to a particular resource that is directly accessed via the service entry point. This resource serves as the starting point for locating and accessing the other resources and associated metadata that together make up an instance of a Redfish Service.
Subscription	The act of registering a destination for the reception of events.
Uniform Resource Identifier	A URI is a compact string of characters for identifying an abstract or physical resource. This document defines the generic syntax of URI, including both absolute and relative forms, and guidelines for their use; it revises and replaces the generic definitions in RFC 1738 and RFC 1808.

Resources

Managing HPE Servers Using the RESTful API
h20564.www2.hpe.com/hpsc/doc/public/display?docId=c04423967

HPE Library entries for RESTful Interface Tool
hpe.com/support/restfulinterface/docs

DMTF Redfish documents, presentations, tutorials, and 1.0 specification
dmtf.org/standards/redfish

HPE Library
hpe.com/info/enterprise/docs

HPE ProLiant Technology Papers
hpe.com/servers/technology

Learn more at
hpe.com/info/restfulapi

Sign up for updates

4AA6-1727ENW, August 2016, Rev. 1

Hewlett Packard Enterprise