Hewlett Packard Enterprise

Gestión de servidores Hewlett Packard Enterprise mediante la API de RESTful

Resumen

En esta guía se describen los procedimientos iniciales para utilizar la API de RESTful para iLO (iLO 4) y la API de RESTful para el módulo Moonshot iLO Chassis Management para gestionar servidores Hewlett Packard Enterprise.

Nº de referencia: 795538-076
Publicado: agosto de 2016
Edición: 1

La información que incluye este documento está sujeta a cambios sin previo aviso. Las únicas garantías de los productos y servicios de Hewlett Packard Enterprise están establecidas en las declaraciones expresas de garantía que acompañan a dichos productos y servicios. No se podrá interpretar nada de lo aquí incluido como parte de una garantía adicional. Hewlett Packard Enterprise no se hace responsable de los errores u omisiones de carácter técnico o editorial que puedan figurar en este documento.

Software confidencial. Para la posesión, uso o copia de su software es necesaria una licencia válida de Hewlett Packard Enterprise. En cumplimiento con la normativa FAR 12.211 y 12.212, la licencia del Software Informático Comercial, la Documentación del Software Informático y los Datos Técnicos sobre Elementos Comerciales se han concedido al Gobierno de EE. UU. bajo la licencia comercial estándar del proveedor.

Los enlaces a páginas web de otros fabricantes le llevan fuera de la página web de Hewlett Packard Enterprise. Hewlett Packard Enterprise no tiene ningún control sobre la información ajena a la página web de Hewlett Packard Enterprise y no se hace responsable de ella.

Reconocimientos

Microsoft® y Windows® son marcas comerciales o registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

Google Inc.© Todos los derechos reservados. Google™ es una marca comercial de Google Inc.

Linux® es una marca registrada de Linus Torvalds en Estados Unidos y otros países.

Contenido

1 Introducción a la API de RESTful

La RESTful API para iLO (iLO 4) y la API de RESTful para el módulo Moonshot iLO Chassis Management es una interfaz de programación que permite una gestión de servidores de vanguardia. Este documento contiene información útil acerca de cómo interactuar con la RESTful API. La RESTful API utiliza las operaciones básicas de HTTP (`GET`, `PUT`, `POST`, `DELETE` y `PATCH`) para enviar un recurso con formato JSON a un URI en iLO 4 o en el módulo Moonshot iLO Chassis Management, o bien devolverlo desde estos.

En los lenguajes de secuencias de comandos modernos, resulta sencillo escribir clientes REST simples para las API de RESTful. La mayoría de ellos, como Python, pueden transformar JSON en estructuras de datos internos, como diccionarios, lo que permite obtener un fácil acceso a los datos. Esto permite escribir código personalizado directamente en la RESTful API, en lugar de utilizar herramientas intermedias de HPE como HPQLOCFG o CONREP.

Este documento se ha actualizado con ejemplos de la versión 2.30 del firmware de iLO 4.

2 Conformidad con Redfish 1.0

La RESTful API se publicó por primera vez con iLO 4 2.00 en los servidores HPE Gen9. La RESTful API también servía de punto de partida para el nuevo estándar Redfish 1.0 DMTF (consulte **http://www.dmtf.org/standards/redfish**).

Desde el lanzamiento de la RESTful API, los miembros del SPMF de DMTF han introducido varios cambios en el estándar Redfish. En un nivel alto, los cambios incluyen:

- Uso de anotaciones OData en los datos para transmitir metadatos
- Revisión de las estructuras de error
- Eliminación o cambio de nombre de ciertas propiedades de JSON
- Uso de un esquema de colección distinto
- Uso de un conjunto de URI de punto de entrada distinto (`/redfish/v1/` frente a `/rest/v1`)

iLO 4 2.30 es compatible con Redfish 1.0 y conserva la compatibilidad con versiones anteriores de la RESTful API existente. En el futuro, la RESTful API se convertirá en la implementación mejorada de Redfish de iLO 4. Aunque es compatible con el estándar Redfish 1.0, la RESTful API incluye características específicas de Hewlett Packard Enterprise, como la configuración del BIOS. Es recomendable que actualice el código de cliente para confirmar la compatibilidad con el estándar Redfish. iLO eliminará las propiedades que no correspondan con las definiciones del esquema de Redfish (aunque conservará las extensiones OEM).

iLO 4 2.30 consigue la compatibilidad con Redfish 1.0 y versiones anteriores:

1. Duplicando el modelo de recursos en `/redfish/v1/` y en `/rest/v1`.
2. Devolviendo las propiedades de Redfish y de compatibilidad de forma predeterminada.
3. Devolviendo solo las propiedades de compatibilidad con Redfish (con extensiones Hewlett Packard Enterprise) si el encabezado OData requerido por Redfish está incluido en la solicitud (OData-Version: 4.0).

3 API de REST con arquitectura HATEOAS

Representational State Transfer (REST) es un servicio web que utiliza operaciones CRUD básicas (Create (Crear), Read (Leer), Update (Actualizar), Delete (Eliminar) y Patch (Parchear)) realizadas en recursos utilizando comandos HTTP como `POST` (Publicar), `GET` (Obtener), `PUT` (Colocar), `PATCH` (Parchear) y `DELETE` (Eliminar). La RESTful API ha sido diseñada utilizando una arquitectura de REST denominada HATEOAS (Hypermedia as the Engine of Application State). Esta arquitectura permite al cliente interactuar con iLO a través de una URL fija simple (`rest/v1`) y otros muchos URI de alto nivel documentados en el modelo de datos de iLO. El resto de los modelos de datos se pueden detectar siguiendo los “vínculos” claramente identificables en los datos. Esto presenta la ventaja de que el cliente no necesita conocer un conjunto de URL fijas. Cuando se crea una secuencia de comandos para automatizar tareas mediante la RESTful API, únicamente se necesita codificar esta dirección URL simple y diseñar la secuencia de comandos para detectar las URL de la API de REST necesarias para completar una tarea. Para obtener más información sobre los conceptos de REST y HATEOAS, consulte:

- **http://en.wikipedia.org/wiki/Representational_state_transfer**
- **http://en.wikipedia.org/wiki/HATEOAS**

Ventajas principales de la API de RESTful

La RESTful API se está convirtiendo en la interfaz de gestión principal para iLO 4 y los servidores Hewlett Packard Enterprise basados en el módulo Moonshot iLO Chassis Management. Su conjunto de características será mayor que las de la API XML de iLO (RIBCL) y las interfaces IPMI existentes. Mediante la RESTful API, es posible realizar un inventario completo del servidor, controlar el encendido y el reinicio, configurar los valores de configuración de iLO y el BIOS, y buscar registros de eventos, así como muchas otras funciones.

La RESTful API sigue la tendencia de Internet en la transición a un modelo común para las nuevas interfaces de software. Muchos servicios web en distintos sectores utilizan las API de REST debido a su facilidad de implementación y consumo, y porque ofrecen más ventajas de escala que las tecnologías anteriores. HPE OneView, OpenStack y muchas otras API de gestión de servidores son ahora API de REST. La mayoría de las ofertas de software de Hewlett Packard Enterprise Management, así como toda la infraestructura definida por software, se basan en las API de REST.

La RESTful API tiene la ventaja adicional de la coherencia entre todas las arquitecturas de servidores actuales y previstas. El mismo modelo de datos funciona en servidores tradicionales de montaje en bastidor, blades y nuevos tipos de sistemas como Moonshot. Esta ventaja se debe a que el modelo de datos está diseñado para describir automáticamente las funciones del servicio al cliente y tiene espacio para flexibilidad diseñada desde un principio.

Operaciones con recursos

Operación	Verbo HTTP	Descripción
Crear	`POST (Publicar)` URI del recurso (carga útil = datos del recurso)	Crea un nuevo recurso o invoca una acción personalizada. Un `POST (Publicar)` sincrónico devuelve el recurso recién creado.
Leer	`GET (Obtener)` URI del recurso	Devuelve la representación del recurso solicitado.
Actualizar	`PATCH` (Parchear) o `PUT` (Colocar) URI del recurso (carga útil = datos del recurso)	Actualiza un recurso existente. Únicamente puede utilizar una operación `PATCH` para las propiedades marcadas como `readonly = false` en el esquema.
Eliminar	`DELETE (Eliminar)` URI del recurso	Elimina el recurso especificado.

Códigos de retorno

Código de retorno	Descripción
2*xx*	Operación correcta.
4*xx*	Error del lado cliente con mensaje de error devuelto.
5*xx*	Error de iLO con mensaje de error devuelto.

NOTA: Si se produce un error, que se indica con un código de retorno 4*xx* o 5*xx*, se devuelve una respuesta JSON `ExtendedError`. No se devuelve el recurso esperado.

4 Sugerencias para utilizar la API de RESTful

La RESTful API para iLO está disponible en los servidores ProLiant Gen9 que ejecutan iLO 4 2.00 o versiones posteriores con una licencia estándar de iLO, aunque es posible que algunas características de los datos no estén disponibles sin una licencia Advanced. La RESTful API para el módulo Moonshot iLO Chassis Management está disponible en los servidores Moonshot que ejecutan iLO Chassis Manager 1.30 o versiones posteriores y no requiere una licencia.

Para acceder a la RESTful API, es necesario un cliente compatible con HTTPS, como un explorador web con la extensión del complemento de cliente REST Postman o CURL (una popular utilidad de línea de comandos de HTTP).

Ejemplos de RESTful Interface Tool y Python

Aunque no es un requisito, puede utilizar RESTful Interface Tool con la RESTful API. Esta herramienta de línea de comandos proporciona un nivel de abstracción y comodidad superior al acceso directo a la RESTful API. Para obtener más información, consulte: **http://www.hpe.com/info/restfulapi**.

Asimismo, Hewlett Packard Enterprise ha publicado código de ejemplo de Python que implementa cierto número de operaciones comunes en un cliente de la RESTful API. Este código puede descargarse de **https://github.com/HewlettPackard/python-proliant-sdk**. En algunos casos, los ejemplos de este documento pueden hacer referencia a ejemplos del código Python con esta notación:

Python: Consulte `ex1_functionname()` en el código de ejemplo de Python. Esto significa que se debe buscar el nombre de función especificado en el código de ejemplo de python.

Si prefiere no implementar un cliente en Python, este es un buen ejemplo de seudocódigo que implementa la lógica necesaria para llevar a cabo una operación.

Ejemplo de operación de la API de REST

Llevemos a cabo nuestra primera operación `GET` (Obtener) utilizando la RESTful API. Realizaremos una operación `GET` (Obtener) de HTTP en el puerto HTTPS de iLO, generalmente el puerto 443 (aunque puede ser diferente si ha configurado previamente iLO para que utilice otro puerto). El cliente debe estar preparado para superar el reto del certificado HTTPS. La interfaz no está disponible en HTTP abierto (puerto 80), por lo que debe utilizar HTTPS.

La operación `GET` (Obtener) se realizará con respecto a un recurso en `/rest/v1` (sin una barra al final):

- Correcto: `GET` **`https://myilo/rest/v1`**
- Incorrecto: `GET` **`https://myilo/rest/v1/`**

La mejor opción para llevar a cabo esta operación `GET` inicial es realizarla con una herramienta como CURL o el cliente REST Postman mencionado anteriormente. Más adelante querrá hacer esto con su propio código de secuencia de comandos, pero de momento, es útil ver el intercambio de información de encabezado HTTP mediante un explorador.

NOTA: Este tutorial contiene seudocódigo para acceder a los recursos de la RESTful API de iLO. Para seguir mejor este código, se recomienda encarecidamente que realice las operaciones en el cliente REST Postman o en CURL, para que pueda ver fácilmente cómo las estructuras de datos coinciden con el seudocódigo. Los mismos principios ilustrados aquí también se aplican al módulo Moonshot iLO Chassis Management.

Redfish: Redfish cambia el URI de punto de entrada a `/redfish/v1/`.

Se debe incluir la barra al final e iLO devuelve un estado de redireccionamiento HTTP (308) al cliente si este tiene acceso a los URI siguientes:

- `/redfish`
- `/redfish/`
- `/redfish/v1`

En el futuro, iLO 4 responderá a todos los URI de estilo `/rest/v1/` con el redireccionamiento 308 a los URI equivalentes de `/redfish/v1/`.

Ejemplo de CURL

CURL es una utilidad de línea de comandos disponible para muchos sistemas operativos que permite un fácil acceso a la RESTful API. CURL está disponible en **http://curl.haxx.se/**. Tenga en cuenta que todos los ejemplos de CURL utilizarán un indicador `–insecure`. Esto hace que CURL omita la validación del certificado HTTPS. En un caso real, iLO debería estar configurado para utilizar un certificado proporcionado por el usuario y esta opción no es necesaria. Tenga en cuenta también que se utiliza la opción `–L` para obligar a CURL a seguir las respuestas de redireccionamiento HTTP. Si iLO cambia las ubicaciones de URI para varios elementos, puede indicar al cliente dónde se encuentra la nueva ubicación y seguir automáticamente el nuevo vínculo.

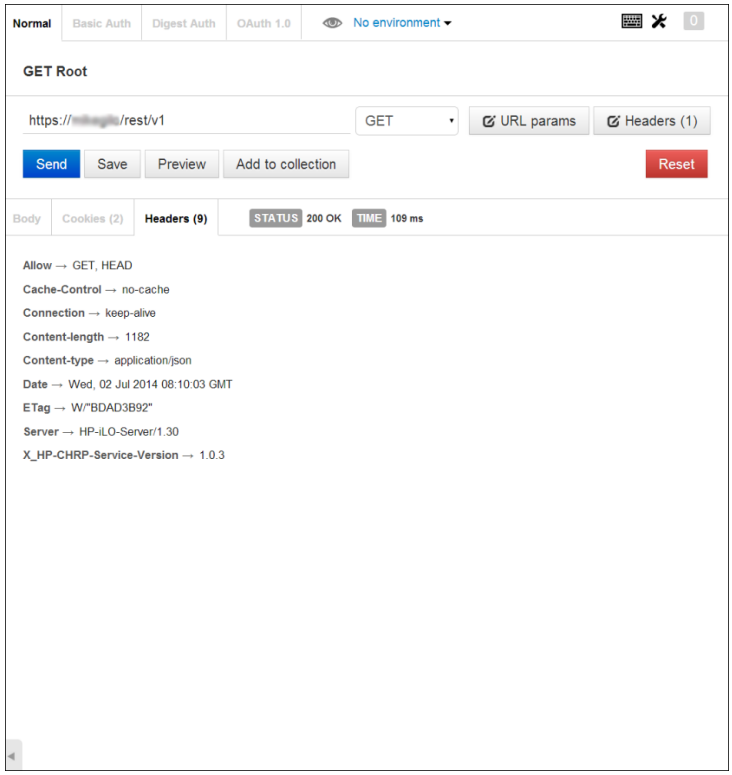
```
> curl https://myilo/rest/v1 -i --insecure -L
-i devuelve encabezados de respuesta HTTP, --insecure omite la comprobación de certificados TLS/SSL, -L sigue los
redireccionamientos
```

```
{"@odata.context":"/redfish/v1/$metadata#ServiceRoot","@odata.id":"/redfish/v1/","@odata.type":"#ServiceRoot.1.0.0.ServiceRoot",

 "Id":"v1","Name":"HP RESTful Root
Service","Oem":{},"ServiceVersion":"0.9.5","Time":"2014-08-05T13:12:02Z","Type":"ServiceRoot.1.0.0",
"UUID":"627490fe-bded-5d10-8176-0af0927c690e","links":{"AccountService":{"href":"/rest/v1/AccountService"},"Chassis":{"href":"/rest/v1/Chassis"},
"EventService":{"href":"/rest/v1/EventService"},"JSONSchema":{"href":"/rest/v1/Schemas"},"Managers":{"href":"/rest/v1/Managers"},
"Registries":{"href":"/rest/v1/Registries"},"Schemas":{"href":"/rest/v1/Schemas"},"SessionService":{"href":"/rest/v1/SessionService"},
"Sessions":{"href":"/rest/v1/SessionService/Sessions"},"Systems":{"href":"/rest/v1/Systems"},"self":{"href":"/rest/v1"}}}
```

Ejemplos de encabezados de respuesta HTTP

Cuando se realiza una operación GET, se recibe un objeto JSON de respuesta. Eche un vistazo a este objeto JSON y los encabezados HTTP devueltos más arriba.

Las primeras líneas son los encabezados HTTP de respuesta que identifican el contenido. Les sigue la respuesta JSON en formato comprimido. Aunque parezca confuso, JSON está altamente estructurado, y puede dar lugar a documentos de fácil lectura.



Lo primero que debe tener en cuenta es un par de encabezados de respuesta. El encabezado `ETag` es un resumen del contenido JSON. La manipulación de `ETag` es un tema avanzado, así que se omitirá por el momento, pero sí que vale la pena señalar que el valor de cadena de un `ETag` no se ha concebido para que el cliente lo analice. Es una cadena opaca, e iLO puede cambiar su algoritmo de generación de `ETag` en una versión futura.

El segundo encabezado de interés es el encabezado `Allow` (Permitir). Allow (Permitir) está disponible en todas las operaciones `GET` (Obtener) de la RESTful API e indica qué operaciones HTTP están permitidas en este recurso. Observará que solo se permite utilizar `GET` (Obtener) con `/rest/v1`. Esto significa que es un objeto de solo lectura debido a que no se pueden utilizar las operaciones `PATCH` (Parchear), `PUT` (Colocar) o `DELETE` (Eliminar) (las operaciones de modificación). En muchos recursos, se encontrará con que la interfaz es compatible con `PATCH`. `PATCH` (Parchear) está diseñada específicamente para actualizar un objeto existente en lugar de sustituirlo (como haría `PUT` [Colocar]). Esto le permite `PATCH` (Parchear) un objeto parcial pero sin preocuparse de que está eliminando las propiedades omitidas.

Ejemplo de objeto JSON raíz

Esta es la versión completa de la respuesta JSON anterior (editada debido a su longitud):


```
{
  "@odata.context": "/redfish/v1/$metadata#ServiceRoot",
  "@odata.id": "/redfish/v1/",
  "@odata.type": "#ServiceRoot.1.0.0.ServiceRoot",
  "Id": "v1",
  "Name": "HP RESTful Root Service",
  "Oem": {},
  "ServiceVersion": "0.9.5",
  "Time": "2014-08-05T13:12:02Z",
  "Type": "ServiceRoot.1.0.0",
  "UUID": "627490fe-bded-5d10-8176-0af0927c690e",
  "links": {
    "AccountService": {
      "href": "/rest/v1/AccountService"
    },
```

```
    "Chassis": {
      "href": "/rest/v1/Chassis"
    },
    "EventService": {
      "href": "/rest/v1/EventService"
    },
    "JSONSchema": {
      "href": "/rest/v1/Schemas"
    },
    "Managers": {
      "href": "/rest/v1/Managers"
    },
    "Registries": {
      "href": "/rest/v1/Registries"
    },
    "Schemas": {
      "href": "/rest/v1/Schemas"
    },
    "SessionService": {
      "href": "/rest/v1/SessionService"
    },
    "Sessions": {
      "href": "/rest/v1/SessionService/Sessions"
    },
    "Systems": {
      "href": "/rest/v1/Systems"
    },
    "self": {
      "href": "/rest/v1"
    }
  }
}
```


En JSON, no hay ninguna ordenación fija de los nombres de las propiedades, por lo que iLO puede devolver propiedades JSON en cualquier orden. Igualmente, iLO no puede asumir el orden de las propiedades en los JSON enviados. Este es el motivo por el que la mejor estructura de datos de la secuencia de comandos para un cliente de RESTful es un diccionario: un conjunto sencillo de pares clave-valor desordenado. Esta falta de orden también es el motivo por el que se observará estructura integrada en los objetos (objetos dentro de objetos). Esto permite tener datos relacionados entre sí con una organización más lógica y una visualización más agradable, que evitará conflictos en los nombres de propiedades o nombres de propiedades extremadamente largos. También permite la utilización de bloques idénticos de JSON en muchos lugares del modelo de datos, como el estado.

Tipo y versión del recurso

Observe que la respuesta JSON tiene una propiedad denominada `Type` (Tipo) cuyo valor es `"ServiceRoot.1.0.0"`. El tipo es muy importante en la RESTful API. Es la clave para comprender el significado del resto de propiedades del objeto. Los recursos del mismo tipo siguen la misma *definición de esquema* (es decir, las propiedades con los mismos nombres significan lo mismo).

La propiedad Type incluye un nombre (`"ServiceRoot"`) e información de la versión (`"1.0.0"`). La información de la versión tiene el formato `principal.secundaria.errata`. En las futuras versiones de firmware de iLO ciertamente se ampliarán los datos incluidos en sus diversos recursos, por lo tanto, como cliente, debe estar preparado para ver diferentes versiones de un `Type` (Tipo). Cualquier versión que no sea principal es compatible con versiones anteriores, por lo que una nueva propiedad añadida a un objeto podría producir incrementos de `Type` (Tipo) en la versión secundaria (por ejemplo, `Chassis.1.0.0` se convierte en `Chassis.1.1.0`). Un cambio de versión principal (por ejemplo, `Chassis.2.0.0`) es un cambio brusco sin garantía de compatibilidad con versiones anteriores.

Las versiones iniciales de la RESTful API contenían muchos recursos de versión de tipo 0.9.5, cuya versión principal es 0. No obstante, a partir de iLO 4 2.30 muchos tipos están en proceso de transición a la versión 1.0.0. Hemos mantenido la compatibilidad con versiones anteriores de iLO 4, por lo que los cambios de este tipo de 0 a 1 no indican cambios bruscos en este caso, sino la madurez del modelo de datos de la RESTful API.

Redfish: En Redfish, `Type` se ha sustituido por `@odata.type`. En unos pocos casos, la cadena del nombre de tipo también ha cambiado (por ejemplo, `PowerMetrics` ahora es simplemente `Power`). Tenga en cuenta que si incluye el encabezado OData-Version especificado en la especificación de Redfish, no aparecerá una propiedad Type en la respuesta, solo el `@odata.type`. Si no incluye este encabezado, aparecerán ambos. En el futuro, cuando iLO elimine el contenido que no sea de Redfish, `Type` se eliminará y se conservará `@odata.type` como el indicador de tipo.

Información sobre OneView

Si el servidor se gestiona con OneView, se incluye información sobre la instancia de OneView en este recurso raíz. Consulte la propiedad en la ruta del indicador JSON `/rest/v1#/Oem/Hp/Manager/0/IPManager`. El contenido de este subobjeto incluye una referencia al servidor de OneView.

NOTA: Si el servidor se gestiona con OneView, muchos de los valores de configuración se configuran específicamente para utilizarlos con OneView. Actúe con precaución al modificar las propiedades en la RESTful API directamente cuando el servidor se gestione con OneView, ya que podría provocar que el servidor deje de estar sincronizado con la vista del sistema de OneView durante un tiempo.

Vínculos

Por último, echemos un vistazo a la propiedad links (vínculos). En la RESTful API, se incluye la lista de navegación de los datos en los datos en sí, no en la especificación, y por lo tanto, puede adaptarse según sea necesario a diferentes servidores con el tiempo sin necesidad de introducir cambios en la especificación. Esto se denomina una API de hipermedia, en la que la lista de navegación está integrada en los datos (como en las páginas web). La propiedad links (vínculos) contiene los indicadores a otros recursos relacionados. En el interior de los objetos links (vínculos) hay definidos distintos tipos de relación, como Systems (Sistemas), Managers (Gestores), Chassis (Chasis) o Sessions (Sesiones). Cada una de ellas contiene una propiedad denominada `href` que es el URI del recurso relacionado. Un tipo de relación (por ejemplo, `Systems` [Sistemas]) distingue entre los distintos vínculos en función de a dónde se dirijan. Por ejemplo, Systems (Sistemas) va a una colección de recursos `ComputerSystem`.

Navegación por el modelo de datos

La RESTful API no define todos los URI a los distintos elementos, como las temperaturas o las fuentes de alimentación de un servidor. No puede asumir que la información de versión del BIOS se encuentra siempre en un URI concreto. Esto es más complejo para el cliente, pero es necesario para garantizar que el modelo de datos puede cambiar para acomodar diferentes arquitecturas de servidor futuras sin necesidad de cambios en la especificación. Por ejemplo, si la versión del BIOS se encuentra en `/rest/v1/Systems/1` y un cliente da por sentado que siempre está allí, se producirá un error cuando la interfaz se implemente en un sistema Moonshot con 180 nodos informáticos, cada uno con su propia versión del BIOS. La definición de indicadores a características específicas de la especificación hace que la RESTful API sea demasiado rígida. Por este motivo, solo se publican y garantizan unos pocos URI seleccionados como indicadores de inicio estables. El código de cliente no debe dar por sentado nada sobre los URI que se muestran en el modelo de datos. Debe tratarlos como cadenas opacas, o el cliente no interoperará con otras implementaciones de la RESTful API.

No codifique plantillas de análisis del URI en el cliente. En su lugar, debe empezar en /rest/v1 o en uno de los URI de punto de entrada fijo mostrados a continuación y rastrear el modelo de

datos utilizando los tipos de relación definidos y encontrar las instancias de los tipos que le interesen.

Redfish: Redfish utiliza la propiedad `@odata.id` para indicar un vínculo a otro recurso. Estos vínculos pueden estar en una sección `"Links"` (para recursos relacionados) o en el propio objeto raíz (para recursos secundarios). Tenga en cuenta que Redfish utiliza `"Links"` (L mayúscula) frente a `"links"` para mantener la compatibilidad con la RESTful API. iLO mostrará `Links` o `links` en función de la presencia o ausencia del encabezado HTTP OData-Version en la solicitud.

URI de puntos de entrada fijos en el modelo de datos

Los siguientes URI son fijos en el modelo de datos y el software de cliente puede comenzar por cualquiera de ellos cuando accede a la RESTful API.

- `/rest/v1`

  El recurso raíz.

- `/rest/v1/Systems`

  La colección de nodos informáticos.

- `/rest/v1/Chassis`

  La colección de chasis.

- `/rest/v1/Managers`

  La colección de procesadores de gestión (iLO).

- `/rest/v1/Sessions`

  La API de gestión de sesiones.

- `/rest/v1/Registries`

  La colección de registros.

- `/rest/v1/Schemas`

  La colección de esquemas.

Veamos algunos de los vínculos:

Systems

Este es un vínculo a la colección de nodos del sistema. Los sistemas son nodos informáticos (la terminología proviene de DMTF) con, por ejemplo, CPU, memoria, ranuras de expansión, gestión de energía y la versión del BIOS. Un servidor DL o BL tiene un único nodo informático en la colección de sistemas mientras que Moonshot puede tener hasta 180 elementos en la colección.

Debe pensar en el vínculo del sistema como su portal a la vista lógica del servidor o servidores. La colección de sistemas está documentada para que sea `/rest/v1/Systems`.

Chassis

Ésta es la vista física. Un `Chassis` (Chasis) debe considerarse como un contenedor físico. El tipo de relación `Chassis` (Chasis) es un vínculo a la colección de `Chassis` (Chasis). En un servidor ProLiant DL/ML/BL, este valor será una colección de uno de los chasis. Una colección de `Chassis` (Chasis) de Moonshot puede incluir los cartuchos, así como el chasis (ya que los cartuchos son tipos de chasis).

Los `Chassis` (Chasis) son contenedores que a menudo tienen fuentes de alimentación, sensores de temperatura y una ubicación física. Los `Chassis` (Chasis) contienen nodos informáticos lógicos (sistemas), pero también pueden contener otro chasis (tenga en cuenta que un chasis

Moonshot contiene cartuchos, que son otro tipo de chasis, cada uno con contenido de nodos). Asimismo, no hay ninguna relación uno a uno entre chasis y sistemas para dar cabida a sistemas de varios nodos y sistemas que se distribuyen en más de un chasis.

La colección de chasis está documentada para que sea `/rest/v1/Chassis`.

Managers

Este es un vínculo a iLO 4 o al propio módulo Moonshot iLO Chassis Management para las tareas de gestión de iLO realizadas tradicionalmente con RIBCL. Por ejemplo, la configuración de red, la administración de licencias y la gestión del firmware de iLO.

La colección de procesadores de gestión está documentada para que sea `/rest/v1/Managers`.

Ejemplo de autenticación básica

A continuación se muestra el error que aparece en `GET /rest/v1/Systems` cuando no se intenta la autenticación:

Respuestas ExtendedError/ExtendedInfo

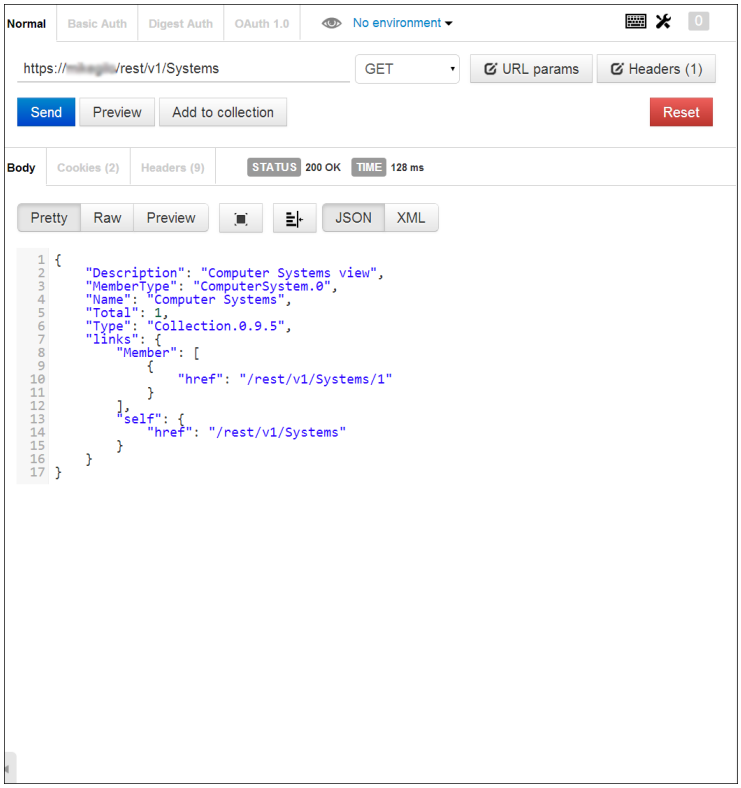
```
{
    "@odata.type": "#ExtendedInfo.ExtendedInfo",

    "Messages": [
        {
            "MessageID": "Base.0.10.NoValidSession",
        }
    ],
    "Type": "ExtendedError.1.0.0",
    "error": {
        "@Message.ExtendedInfo": [
            {
                "MessageId": "Base.0.10.NoValidSession"
            }
        ],
        "code": "iLO.0.10.GeneralError",
        "message": "A general error has occurred. See ExtendedInfo for more information."
    }
}
```

Redfish: La respuesta anterior es la combinación de compatibilidad y respuesta de Redfish. Las propiedades de Redfish están **resaltadas**. Si la solicitud incluye el encabezado OData-Version, las propiedades que no sean de Redfish se ocultarán. Asimismo, la respuesta tiene el tipo de compatibilidad `"ExtendedError"`, pero es el tipo Redfish `"ExtendedInfo"`.

Adición de autenticación básica con Postman

1. Haga clic en la pestaña **Autenticación básica**.
2. Introduzca el **Nombre de usuario** y **Contraseña**.
3. Haga clic en **Actualizar encabezados**.

   Esto generará el hash de Base64 de la autenticación básica.
4. Haga clic en **Enviar**.



Adición de autenticación básica a una línea de comandos de CURL

- Agregue el parámetro `-u username: password`.

Ejemplo

```
> curl https://myilo/rest/v1/Systems -i --insecure -u username:password -L
```

```
HTTP/1.1 200 OK
Allow: GET, HEAD
Cache-Control: no-cache
Content-length: 437
Content-type: application/json
Date: Tue, 05 Aug 2014 14:39:56 GMT
ETag: W/"9349EA93"
Link: </rest/v1/SchemaStore/en/ComputerSystemCollection.json>; rel=describedby
Server: HP-iLO-Server/1.30
X-Frame-Options: sameorigin
X_HP-CHRP-Service-Version: 1.0.3


{"@odata.context":"/redfish/v1/$metadata#Systems","@odata.id":"/redfish/v1/Systems/"
,"@odata.type":"#ComputerSystemCollection.ComputerSystemCollection","Description":"Computer Systems view",
"MemberType":"ComputerSystem.1","Members":[{"@odata.id":"/redfish/v1/Systems/1/"}],"Members@odata.count":1,
"Name":"Computer Systems","Total":1,"Type":"Collection.1.0.0","links":{"Member":[{"href":"/rest/v1/Syst
ems/1"}],"self":"/rest/v1/Systems"}}
```

Gestión de sesiones

Si realiza una operación `GET` (Obtener) en cualquier otro recurso que no sea el recurso raíz `/rest/v1`, obtendrá un mensaje de error de HTTP 401 (Prohibido) que indica que no tiene la autenticación necesaria para acceder al recurso.

La RESTful API le permite utilizar la autenticación básica de HTTP si lo desea para las operaciones de un solo uso. Es por este motivo que el complemento de Postman o CURL resultan tan útiles. Estas herramientas pueden insertar fácilmente la codificación de base64 correcta para las credenciales de autenticación básica en las solicitudes.

Inicio y fin de sesión

Para operaciones de varios recursos más complejas, debe iniciar sesión y establecer sesión. Para iniciar sesión, iLO cuenta con un objeto de gestor de sesiones en el URI documentado

/rest/v1/Sessions. Para crear una sesión, necesitamos realizar una operación POST (Publicar) de un objeto JSON en el gestor de sesiones:

```
POST /rest/v1/Sessions con los encabezados HTTP necesarios y un cuerpo que contenga


   {
     "UserName": "username",
     "Password": "password"
   }
```

IMPORTANTE: Debe incluir el encabezado HTTP `Content-Type: application/json` para todas las operaciones de la API de RESTful que incluyan un cuerpo de solicitud con formato JSON.

Si la sesión se crea correctamente, recibirá una respuesta HTTP 201 (Creado) desde iLO. También habrá dos encabezados HTTP adicionales.

Encabezado	Valor	Descripción
X-Auth-Token	El token de sesión (cadena).	Se trata de una cadena exclusiva para iniciar sesión. Debe incluirse como un encabezado en todas las operaciones HTTP subsiguientes en la sesión.
Ubicación	El URI del recurso de sesión recién creado.	iLO asigna un nuevo recurso de sesión que describa su sesión. Este es el URI que debe `DELETE (Eliminar)` para cerrar la sesión. Si pierde este URI de ubicación, puede encontrarlo *rastreando* los vínculos HREF de la colección Sessions (Sesiones). Guarde este URI para facilitar el cierre de la sesión.

Navegación por el modelo de datos

Una vez que pueda iniciar sesión, podrá recorrer el modelo de datos completo. Puesto que el modelo está vinculado entre sí mediante vínculos href, es fácil crear un cliente que puede rastrear el modelo completo simplemente recorriendo en iteración cada objeto (con recursión para los subobjetos y matrices) buscando hrefs dentro de los objetos de vínculos.

Tenga en cuenta que aun cuando comienza en un URI conocido, el modelo de datos no es un árbol en sentido estricto, puesto que puede haber referencias combinadas en todo el modelo de datos (por ejemplo, desde el chasis al sistema y viceversa), por lo que necesitará crear un diccionario de URI visitados a medida que rastrea el modelo para evitar un recorrido infinito. Cuando realiza una operación `GET (Obtener)` con un URI, agréguelo a este diccionario y, a continuación, si ve dicho URI nuevamente, no lleve a cabo otra operación `GET (Obtener)`. Como recomendación adicional, debe mantener las claves URI del diccionario visitado en minúsculas, por ejemplo, `tolower()`. Los URI del modelo de datos deben utilizar convenciones coherentes de mayúsculas y minúsculas, pero los errores ocurren, y si hay una inconsistencia en la carcasa, mantener el diccionario con claves de cadenas URI en minúscula, protegerá el algoritmo transversal.

Al rastrear el modelo de datos, observará que una gran cantidad de objetos son del tipo `Collection (Colección)`.

Colecciones

Las colecciones son un tipo de objeto de uso frecuente en la RESTful API. Las colecciones siempre tienen un tipo genérico *Collection* (lo que significa que todas se ciñen al mismo esquema y comparten los mismos nombres de propiedad). Las colecciones son muy versátiles y se distinguen mediante una propiedad denominada `MemberType (Tipo de Miembro)` que indica el tipo de los miembros que las componen. `MemberType (Tipo de Miembro)` solo especifica hasta el nombre del tipo y versión principal, lo que permite la combinación de versiones secundarias del mismo tipo en la misma colección.

Las colecciones cuentan con una implementación muy flexible y pueden presentar a los miembros de varias formas diferentes. Las dos estructuras de datos básicos en una colección: los vínculos a los miembros de la colección (matriz `/links/Member`) y la matriz `Items (Elementos)`. Según cómo se implementa la colección, es posible que se omita la matriz de `Members (Miembros)` o `Items (Elementos)` en una colección:

- **Formato 1**

  Vínculos a recursos secundarios

  - Contiene la matriz `/links/Member`.
  - La matriz `/links/Member` contiene hrefs a los recursos secundarios.
  - Operaciones de colección (si se admiten):

    `GET`

    `POST` para crear

- **Formato 2**

  Vínculos a los recursos secundarios y elementos incrustados

  - Contiene la matriz `/links/Member`.
  - La matriz `/links/Member` contiene hrefs a los recursos secundarios.
  - Contiene la matriz `Items`.

    Cada uno de ellos es totalmente representativo del `MemberType` (*información del elemento*).
  - La matriz `/links/Member` contiene fragmentos a los elementos Items.
  - Operaciones de colección (si se admiten):

    `GET`

    `POST` para crear

- **Formato 3**

  Solo elementos incrustados (solo lectura)

  - La matriz `/links/Member` puede omitirse.
  - Contiene la matriz `Items`.

    Cada uno de ellos es totalmente representativo del `MemberType`.
  - La matriz `/links/Member` que contiene fragmentos a los elementos Items puede omitirse.
  - Operaciones de colección (si se admiten):

    `GET`

A continuación se describe cada uno de los formatos:

- El Formato 1 es una colección completamente modificable que requiere la iteración utilizando el comando `GET (Obtener)` para cada href en la matriz de Member (Miembro). Este es el formato más flexible y genérico, pero es menos eficaz debido a que requiere que más operaciones HTTP para hacer un recorrido de iteración completo. Las colecciones con este formato permitirían el comando `POST (Publicar)` para crear elementos y los recursos secundarios permitirían el comando `DELETE (Eliminar)` para eliminar elementos. Los

recursos secundarios permitirían el comando `PATCH (Parchear)` para modificar elementos. Este formato de colección se utiliza con Sessions (Sesiones) y Accounts (Cuentas) para las que se admite la creación y eliminación de los miembros y la iteración de la colección es menos importante.

- El Formato 2 es una variante del Formato 1 e incluye una matriz de `Items (Elementos)` que contiene un subconjunto o la representación completa del elemento en el mismo objeto de la colección, así como un vínculo a los recursos secundarios. Si el instalador ha elegido bien el subconjunto del miembro, podría obtener *información* sobre los miembros de la colección observando la información de nivel superior en la matriz de `Items (Elementos)` y decidir realizar una operación `GET (Obtener)` independiente para los elementos de interés (explorar en profundidad). Esto aumenta la escala al permitir que el cliente determine mejor a qué nodos secundarios aplicar `GET (Obtener)`. La representación de la matriz `Items (Elementos)` no tiene por qué ser totalmente compatible con el esquema `MemberType (Tipo de Miembro)` ya que puede omitir elementos para ahorrar espacio. La matriz `Items (Elementos)` no sustituye a los recursos secundarios, pero ofrece información de los datos de diferenciación más importantes.
- El Formato 3 es para colecciones de solo lectura (por ejemplo, los dispositivos PCI o ranuras) donde no es necesario crear, eliminar o modificar elementos. No hay recursos secundarios y es posible que no exista una sección de vínculos. La matriz `Items (Elementos)` contiene miembros compatibles con todo el esquema. Las ventajas son eficientes con el tamaño, no existe sección de vínculos y no es necesario iterar utilizando operaciones `GET (Obtener)`. Esta opción es perfecta para las cosas que se representan mejor como matrices de solo lectura.

Para iterar la colección en la RESTful API, consulte `collection()` en el código de ejemplo de Python; se trata de una función generadora de Python que devuelve cada miembro de la colección utilizando la palabra clave yield.

Ejemplo de colección

```
> curl https://myilo/rest/v1/Systems -i --insecure -u username:password -L
```

```
{
    "@odata.context": "/redfish/v1/$metadata#Systems",
    "@odata.id": "/redfish/v1/Systems/",
    "@odata.type": "#ComputerSystemCollection.ComputerSystemCollection",
    "Description": "Computer Systems view",
    "MemberType": "ComputerSystem.1",
    "Members": [
        {
            "@odata.id": "/redfish/v1/Systems/1/"
        }
    ],
    "Members@odata.count": 1,
    "Name": "Computer Systems",
    "Total": 1,
    "Type": "Collection.1.0.0",
    "links": {
        "Member": [
            {
                "href": "/rest/v1/Systems/1"
            }
        ]
    }
}
```

Redfish: Redfish ya no tiene un tipo "`Collection`" genérico. Las colecciones siguen teniendo el mismo aspecto, pero no hay ningún tipo común. En su lugar, hay un tipo `ComputerSystemCollection`, un tipo `ChassisCollection`, etc., lo que elimina la necesidad de la propiedad `MemberType`. El array Members ahora

está en el directorio raíz del objeto y Total se ha sustituido por `Members@odata.count`. Las propiedades específicas de Redfish están **resaltadas** y las propiedades que no son de Redfish están ocultas si el cliente proporciona el encabezado OData-Version. Las propiedades de Redfish transmiten la misma información que las propiedades compatibles, pero con un formato diferente.

Sistema informático

Los ComputerSystems representan los nodos informáticos del modelo de datos. Un servidor de montaje en bastidor ProLiant DL puede tener un solo nodo informático, mientras que un chasis Moonshot completamente cargado puede contener 180 nodos informáticos. Para ver un ejemplo de cómo iterar todos los recursos ComputerSystem en el modo de datos:

Python: Consulte `ex1_change_bios_setting()` en el código de ejemplo de Python. Allí se utiliza el bucle de nivel superior mediante la función de iteración de colección con `/rest/v1/Systems`.

Chassis

Los chasis representan contenedores físicos o virtuales de recursos informáticos. Por ejemplo, un servidor ProLiant DL de montaje en bastidor posee un solo chasis que representa el contenedor de metal del servidor. Los recursos de chasis describen los aspectos físicos del servidor. Para ver un ejemplo de cómo iterar todos los recursos de chasis del modelo de datos:

NOTA: **Python**: Consulte `ex29_get_PowerMetrics_Average()` en el código de ejemplo de Python.

Después de aplicar `GET (Obtener)` a un objeto del chasis, puede ver los atributos, incluidos `ChassisType (Tipo de Chasis)`, las propiedades compartidas del sistema (por ejemplo, `AssetTag (Pestaña de Activo)`) y la información sobre la dimensión.

El `ChassisType (Tipo de Chasis)` está sobrecargado para varias cosas, por lo que un `Chassis (Chasis)` puede estar dentro de otro `Chassis (Chasis)`. Por ejemplo, un chasis «blade» (hoja) está dentro de un chasis de un receptáculo «blade» (hoja), que se encuentra dentro de un bastidor y que también se considera un chasis. Las relaciones `contains`

`(contiene)` y `contained by (contenido por)` se encuentran en la sección de vínculos del objeto del chasis.

Gestor

El recurso Manager (Gestor) representa al propio iLO 4.

Después de aplicar `GET (Obtener)` al objeto iLO, incluido `Manager (Gestor)`, puede ver los atributos, incluida la información sobre el `Model (Modelo)` y el `Firmware` y vínculos a diversos servicios de iLO. Estos servicios incluyen el servicio de licencia, el servicio de red y la configuración de los recursos e información NIC de iLO.

Ejemplo de cómo iterar todos los recursos Manager del modelo de datos

Python: Consulte `ex7_find_iLO_MAC_address()` en el código de ejemplo de Python.

NOTA: El bucle de nivel superior utiliza la función de iteración de colección con `/rest/v1/Managers`.

Esquemas

Cada tipo de recurso tiene un archivo de esquema que define el formato del objeto, las propiedades permitidas, las propiedades necesarias, los tipos y otra información. El modelo de datos utiliza el borrador 4 de la norma del esquema JSON para realizar esta acción. Para obtener más información, consulte: **http://json-schema.org**. Para descargar un paquete para el esquema de validación de Python, consulte: **https://github.com/Julian/jsonschema**. Hay paquetes disponibles para otros idiomas con el fin de validar los recursos con el esquema correspondiente. Esto es parte de la estrategia destinada a aprovechar cadenas de herramientas comunes del sector.

El esquema define la *clase* de recurso y cada instancia de la clase podría implementar un subconjunto de las propiedades disponibles. Un cliente debe tener presente que no todas las propiedades del esquema se implementan en todas las instancias. Las omisiones podrían ser consecuencia de una o varias de las siguientes opciones:

- Es posible que la propiedad no se aplique a un modelo de servidor específico de Hewlett Packard Enterprise.

NOTA: Cada propiedad se identifica como `readonly = true` o `readonly = false`. Esto indica que se puede realizar una operación `PATCH (Parchear)` en la propiedad, pero solo si el recurso tiene un encabezado `Allow (Permitir)` que permita la operación `PATCH (Parchear)`. Es posible que dos recursos compartan el mismo `Type (Tipo)` pero que tengan diferentes encabezados `Allow (Permitir)` que permitan o impidan una operación `PATCH (Parchear)`, incluso para propiedades que sean `readonly = false`.

Métodos para la búsqueda de esquemas y registros de mensajes

Existen dos métodos, en línea y fuera de línea, para encontrar el esquema que define los datos en la RESTful API. En línea significa que la información de esquema está disponible al realizar una operación `GET` (Obtener) en iLO. Fuera de línea significa que los archivos de esquema se encuentran disponibles en el DVD de HPE ProLiant Support Pack.

Archivo .zip fuera de línea del Service Pack para ProLiant

El Service Pack para ProLiant (SPP) contiene una carpeta denominada `hp_restful_api` que contiene varios archivos ZIP con los esquemas y registros de la RESTful API.

Dentro de este archivo .zip hay una estructura idéntica a la estructura que se encuentra en la interfaz RESTful de iLO, salvo que la búsqueda comienza leyendo `/Schemas/index.json` o `/Registries/index.json`. El formato de estos dos archivos `index.json` coincide con el formato `SchemaFile.0.9.5` de los recursos iLO a los que `/rest/v1.Esquemas/href` hace referencia.

Además, el SPP contiene un archivo .zip de los esquemas y registros publicados para el BIOS de UEFI de cada servidor ProLiant compatible. Dichos archivos .zip se nombran como `hp-rest-classes-bios-P89-1.00_07_11_2014.zip`, donde `P89` es el nombre de familia ROM (para cada sistema ProLiant), y `1.00-07_011_2014` es la fecha y la versión de ROM.

Para descargar el esquema y los archivos .zip de registro fuera de línea, consulte: **http://www.hpe.com**.

Gestión de la versión de esquema: Compatibilidad con versiones anteriores y futuras

El tipo de los recursos de la RESTful API se especifica por nombre y versión (por ejemplo, `"ServiceRoot.1.0.0"`). Esto proporciona suficiente información para identificar no solo el esquema correcto, sino también la versión exacta. Hay casos donde la versión de esquema exacta puede no estar disponible, ya sea en línea o fuera de línea (por ejemplo, una actualización del BIOS puede utilizar un esquema más reciente del que iLO presenta en su repositorio de esquema). Por este motivo, es necesario que el código de cliente lleve a cabo búsquedas de esquemas del tipo *mejor coincidencia*. Por ejemplo, un recurso puede ser del tipo `"HpBaseConfigs.1.1.0"` y el único esquema disponible puede ser `"HpBaseConfigs.1.0.3"` o viceversa.

Para facilitar la mejor coincidencia, se ofrecen las directrices siguientes:

- Un archivo de esquema (fuera de línea en una carpeta) se denomina generalmente con una extensión .json para el esquema, pero este no es el nombre oficial del esquema. El nombre base oficial del esquema está incluido en la propiedad `title` del objeto de esquema. Además, debido a algunos cambios de nombre de los tipos de Redfish, en ocasiones el nombre antiguo del esquema está disponible en una propiedad `oldtitle`. Por ejemplo:

```
"title": "EthernetInterface.1.0.0",
"oldtitle": "EthernetNetworkInterface.0.92.0",
```

- La versión principal de un esquema, junto con el nombre, define una familia de esquemas compatibles. Por ejemplo, `EthernetInterface.2.x.y` no es necesariamente compatible con `EthernetInterface.1.x.y`. La excepción es que la versión 1 debe considerarse compatible con futuras versiones con respecto al esquema de la versión 0 debido a que la transición de la versión 0 a la 1 es un proceso de maduración de las versiones iniciales del esquema.
- Números de versiones secundarias y de errata más altos indican diferencias compatibles con versiones futuras. Por ejemplo, `EthernetInterface1.1.1` es un superconjunto de `EthernetInterface.1.0.0`, por lo que si un cliente solo utiliza definiciones de propiedades de 1.0.0, puede usar el esquema de 1.1.1 de la misma forma que el de 1.0.0. Sin embargo, si el cliente necesita utilizar propiedades nuevas de 1.1.1, los nuevos datos no estarán presentes en un recurso del tipo 1.0.0.

Un posible algoritmo de esquema del tipo *mejor coincidencia* podría intentar calcular la diferencia matemática entre la versión de un recurso y todos los esquemas candidatos, y elegir el esquema con la diferencia más pequeña.

5 Ejemplos de uso de la API de RESTful para iLO

Esta sección contiene ejemplos de cómo realizar una tarea específica mediante la API de REST para iLO.

Acceso a la API de RESTful para iLO mediante Python

Python es un lenguaje popular de desarrollo para las API de REST, e incluye potentes herramientas para las consultas de rest. A continuación se muestra un ejemplo sencillo de una consulta del recurso `ServiceRoot` mediante Python:

```
>>> from httplib import HTTPSConnection
>>> from base64 import b64encode
>>> userid='iloUserID'
>>> passwd='iloPassword'
>>> auth='BASIC ' + b64encode(userid + ":" + passwd)
>>> header = {'Authorization': auth}
>>> conn = HTTPSConnection(host='iloHostName', strict=True)
>>> conn.request('GET', '/rest/v1', headers=header)
>>> conn.getresponse().read()
```

Es un ejemplo sencillo de solicitud que muestra las bibliotecas básicas relacionadas y cómo formar correctamente la solicitud. Existen muchos otros parámetros que pueden utilizarse para aumentar o disminuir la seguridad relacionada con SSL y el proceso de comprobación de certificados. La documentación de httplib proporciona una imagen más sólida de las capacidades de las bibliotecas.

Además de las bibliotecas integradas, como httplib, existen muchas otras bibliotecas conocidas que permiten llevar a cabo operaciones REST con Python. La biblioteca requests es una de las bibliotecas HTTP de python más populares disponibles hoy en día, y también se adapta bien al desarrollo para la API de REST.

Para ver un conjunto de ejemplos más completo de cómo utilizar Python con la RESTful API, examine los ejemplos de **https://github.com/HewlettPackard/python-proliant-sdk**.

BIOS

Los ejemplos de esta sección no son aplicables a los servidores Moonshot que ejecutan iLO Chassis Manager.

Python: Consulte `ex1_change_bios_setting()` en el código de ejemplo de Python; allí se demuestra cómo encontrar la configuración del BIOS y parchear nuevos valores.

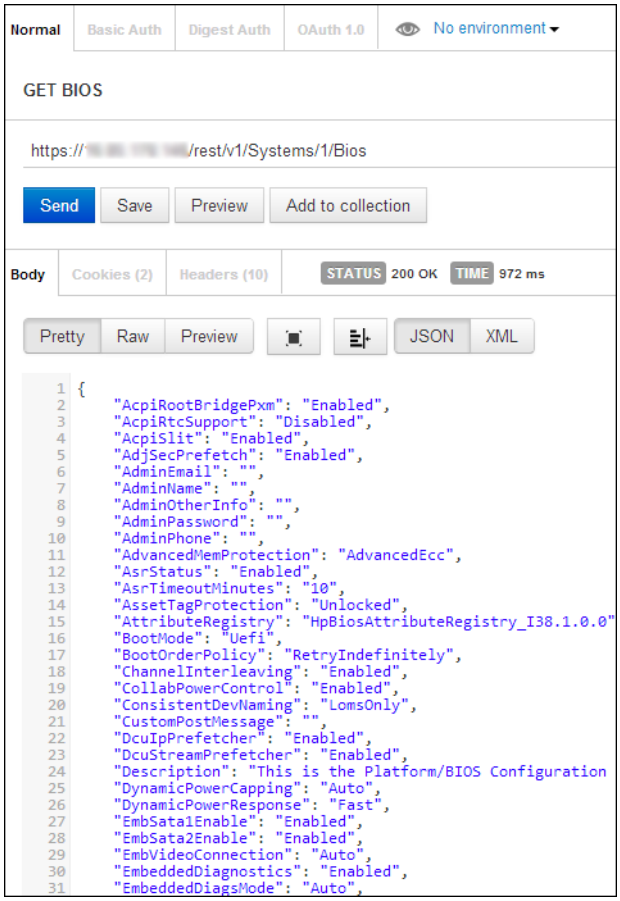
NOTA: La RESTful API para iLO se puede utilizar para cambiar la configuración del BIOS de los servidores ProLiant Gen9 o posteriores.

Acerca del BIOS

Para `GET (Obtener)` la configuración actual del BIOS:

La RESTful API para iLO permite cambiar la configuración del BIOS de UEFI. El BIOS es una entidad de nivel de sistema de las arquitecturas actuales. El vínculo a la configuración del BIOS se encuentra en el objeto nodo del sistema informático.

```
SystemCollection = GET /rest/v1/Systems
For each item in SystemCollection.links.Member # this is a collection
  System = GET item.href
  BIOS = GET System.Oem.Hp.links.BIOS.href
```



El resultado de la operación `GET (Obtener)` del BIOS se encuentra en un objeto plano. No puede utilizar `PATCH (Parchear)` para actualizar un valor de propiedad. Solo puede llevar a cabo operaciones `GET (obtener)` en este recurso.

Cambio de la configuración y conceptos básicos sobre `SettingsResults`

El objeto de configuración actual del BIOS es de solo lectura. Este objeto contiene un vínculo a un recurso `Settings (Configuración)` en el que puede llevar a cabo una operación `PATCH (Parchear)`, que es la configuración pendiente. Si realiza una operación `GET (obtener)` para obtener el recurso `Settings (Configuraciones)`, observará que le permite utilizar comandos `PATCH (Parchear)`. Puede cambiar las propiedades y, a continuación, utilizar un comando `PATCH (Parchear)` para volver al URI de `Settings (Configuraciones)`. Los cambios en la configuración no tienen efecto hasta que se reinicia el servidor. Antes de reiniciar el servidor, los valores de configuración pendientes y actuales están disponibles por separado. Después de reiniciar el servidor, se aplican los valores de configuración pendientes y se pueden ver los errores de la propiedad `SettingsResults (Resultados de las configuraciones)` en el objeto principal.

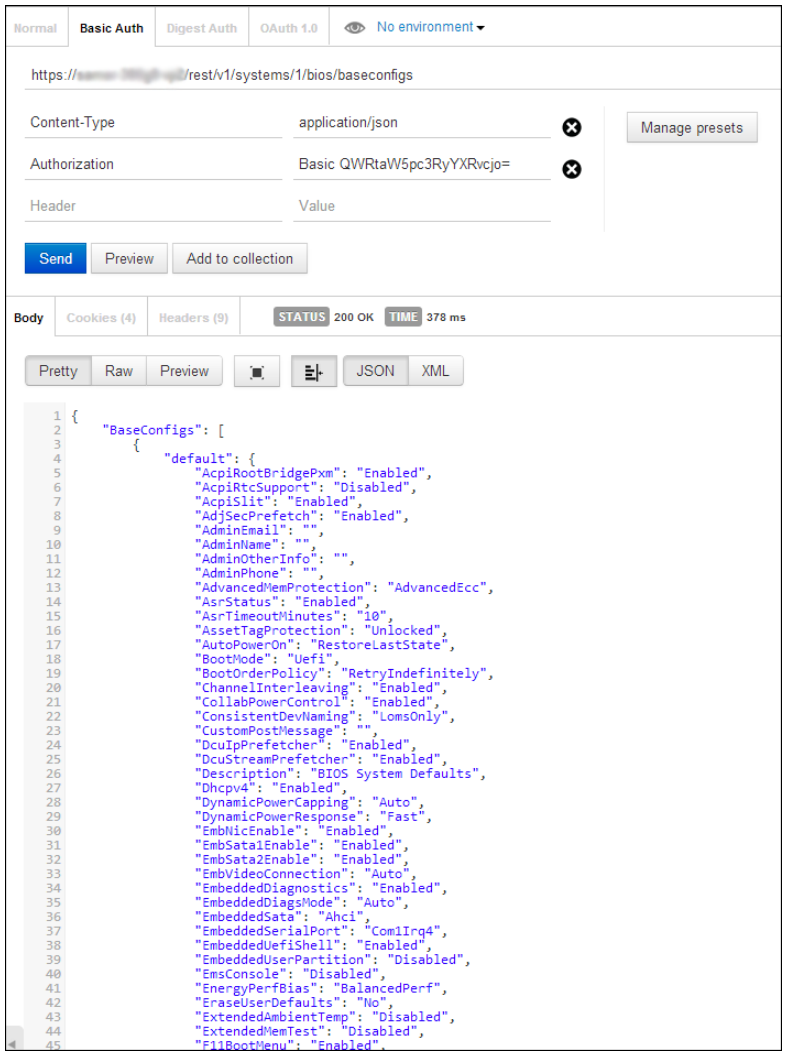
Ventajas de utilizar dos recursos independientes

- Permite que los componentes fuera de línea (por ejemplo, el BIOS) procesen cambios en la configuración de una manera aplazada.
- Permite que los valores actuales y pendientes permanezcan disponibles para su revisión hasta que el componente fuera de línea procese los valores de configuración pendientes.
- Evita la necesidad de colas de trabajo complejas.

Ejemplo de lectura de los valores predeterminados del BIOS

El objeto de configuración actual del BIOS contiene un vínculo a un objeto independiente de solo lectura, `BaseConfigs (Configuraciones de Base)`, que ofrece una lista de los valores por defecto del BIOS. Para `GET` (obtener) el recurso BaseConfigs del BIOS:

```
SystemCollection = GET /rest/v1/Systems
For each item in SystemCollection.links.Member # this is a collection
  System = GET item.href
  BIOS = GET System.Oem.Hp.links.BIOS.href
  BaseConfig = GET BIOS.links.BaseConfigs.href
```



Tenga en cuenta que `BaseConfigs (Configuraciones de Base)` contiene una matriz de conjuntos de valores predeterminados (o conjuntos de configuración básica). Cada conjunto de configuración base contiene una lista de las propiedades del BIOS y sus valores predeterminados. El conjunto de configuración base predeterminado contiene los valores de fabricación predeterminados del BIOS. `BaseConfigs (Configuraciones de Base)` puede contener otros conjuntos, como `default.user (usuario predeterminado)`, para valores predeterminados personalizados por el usuario.

Información general sobre los recursos del BIOS y el registro de atributos

Los recursos del BIOS tienen un formato diferente al de otros muchos recursos. Los recursos del BIOS se adhieren a un tipo de esquema, al igual que todos los objetos. Sin embargo, la configuración del BIOS varía enormemente entre los tipos de servidores y las revisiones del BIOS, por lo que es extremadamente difícil publicar un esquema estándar que defina todas las

propiedades de configuración posibles del BIOS. Además, no es posible comunicar algunas de las opciones avanzadas, como las dependencias entre configuraciones y la estructura de menú en el esquema json. Por lo tanto, el BIOS utiliza un *Registro de atributos*.

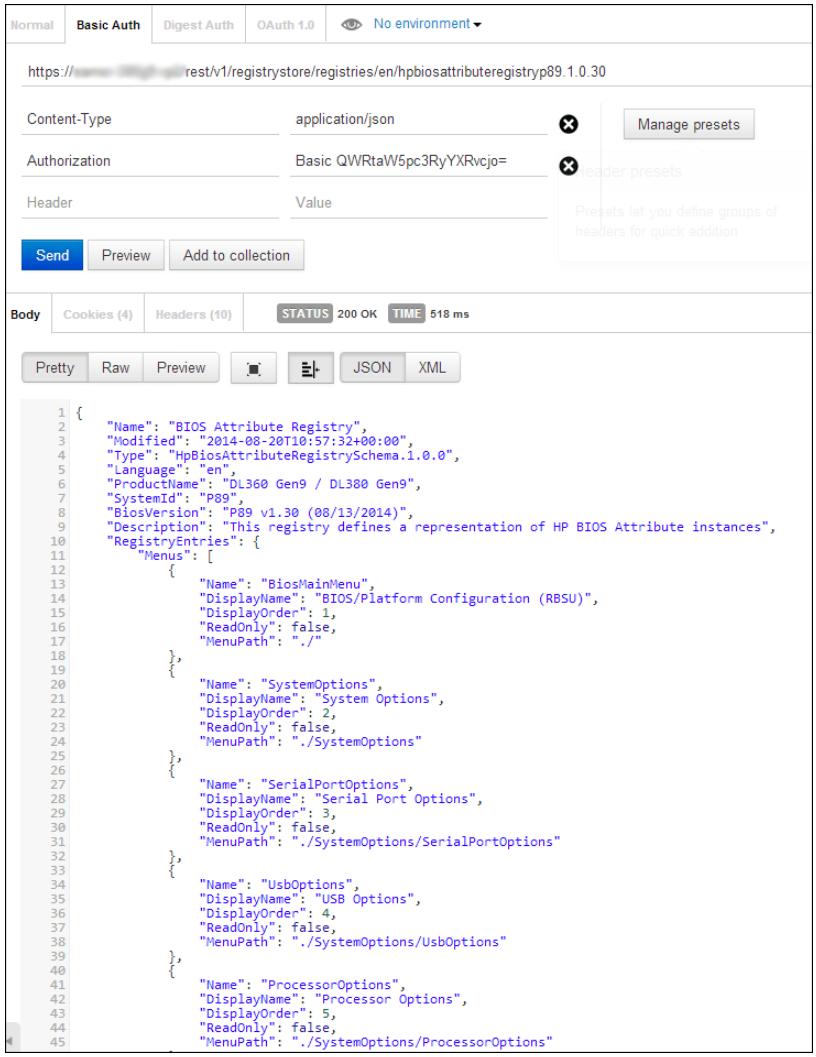
Ejemplo de registro de atributos

Cuando usted `GET (obtenga)` el objeto de configuración del BIOS, tenga en cuenta que todavía tiene la estructura básica e incluye un `Type (Tipo)`. No obstante, también tiene una propiedad denominada `AttributeRegistry (Registro de Atributos)`. Esta propiedad apunta a un archivo de registro. Este archivo contiene un conjunto de datos para cada configuración de BIOS (por ejemplo, `PowerProfile (Perfil de Poder)`), incluida la información de la estructura de menú, las dependencias con otros valores de configuración y otra información.

Para buscar el recurso de registro de atributos del BIOS:

```
SystemCollection = GET /rest/v1/Systems
For each item in SystemCollection.links.Member # this is a collection
  System = GET item.href
  BIOS = GET System.Oem.Hp.links.BIOS.href
  AttributeRegistryName = the value of the "AttributeRegistry" property in BIOS object


RegistryCollection = GET /rest/v1/Registries
For each item in RegistryCollection.links.Member # this is a collection
  Registry = GET item.href
  If Registry.Schema beginswith AttributeRegistryName
  For each language in Registry.Location
    If language.Language == "en"  # or choose another
      AttributeRegistry = GET language.Uri.extref
```

Debido a su tamaño, los registros de atributos del BIOS son recursos JSON comprimidos (gzip), por lo que los encabezados HTTP devueltos indican una codificación de contenido de `gzip`. El cliente REST deberá descomprimir el recurso. Esto se lleva a cabo automáticamente cuando se utiliza un cliente web (como el complemento Postman).



Atributos del BIOS

Lista de atributos del BIOS (configuración) y sus metadatos, que incluye:

- Tipo de cada atributo del BIOS (enumeración, cadena, numérico o boleano).
- Los posibles valores para los atributos de tipo enumeración.
- Muestra las cadenas (localizadas al idioma del registro) de los atributos y sus posibles valores.
- Texto de ayuda y texto de advertencia (localizado).
- Información de ubicación y orden de visualización, incluida la jerarquía de menú para un atributo.
- Límites de valores, como máximo, mínimo e incrementos de atributos numéricos, longitudes mínimas y máximas de caracteres y expresiones regulares para los atributos de cadena.
- Y otros metadatos.

Registros de atributos del BIOS

Los registros de atributos del BIOS contienen tres matrices de nivel superior:

- **Menús**: matriz que contiene los menús de los atributos del BIOS y su jerarquía. Esto se puede utilizar (por ejemplo) para crear una interfaz de usuario que se parezca a la configuración del BIOS local, o para agrupar propiedades que estén relacionadas, como

ProcessorOptions (Opciones de Procesador) y UsbOptions (Opciones de Usb).

- **Dependencias**: matriz que contiene una lista de dependencias de los atributos del BIOS en este servidor. Esto incluye también las dependencias entre configuraciones que pueden dar lugar a que una configuración del BIOS cambie su valor o su propiedad ReadOnly (Solo Lectura) basándose en el valor de otra configuración del BIOS.
- **BaseConfigs (Configuraciones de Base)**: matriz que contiene una lista de valores de fabricación predeterminados de los atributos del BIOS. Esto es equivalente a leer el recurso BaseConfigs (Configuraciones de Base) y analizar el objeto con el nombre default (predeterminado).

Ejemplo de actualización de la contraseña de administrador del BIOS

Para cambiar las contraseñas de administrador y de encendido, deberá cambiar dos propiedades para cada contraseña:

- Contraseña de administrador: establezca AdminPassword en la contraseña nueva y OldAdminPassword en la contraseña antigua.
- Contraseña de encendido: establezca PowerOnPassword en la contraseña nueva y OldPowerOnPassword en la contraseña antigua.

Si no se había definido la contraseña antigua, debe utilizar una cadena vacía ("") para la propiedad de contraseña antigua.

1. Itere en /rest/v1/Systems y elija un ComputerSystem (Sistema de Ordenador) miembro.

   Resultado = *{direcciónipdeilo}*/rest/v1/Systems/1/BIOS
2. Encuentre un vínculo en Oem/Hp/links denominado Bios y anote el BiosURI.
3. GET (obtener) BiosObj desde BiosURI y tenga en cuenta que solo permite el comando GET (obtener) (se trata de la configuración actual).
4. Busque un vínculo en BiosObj denominado Settings (Configuraciones) y tenga en cuenta este URI.
5. Cree un nuevo objeto JSON con las propiedades AdminPassword (Contraseña de administrador) y OldAdminPassword (Contraseña de administrador antigua) cambiadas a {"AdminPassword":"@Pa$$w0rd", "OldAdminPassword":""}.
6. Actualice la configuración del BIOS. Solo necesita enviar la propiedad AdminName (Nombre del Administrador) actualizada en el cuerpo de la solicitud.

   PATCH *{direcciónipdeilo}*/rest/v1/Systems/1/BIOS/Settings

   La configuración del BIOS se valida y adopta al reiniciar el servidor.

Ejemplo de actualización de la configuración del BIOS

Los privilegios de ID de sesión mínimos necesarios son Configure (Configurar).

1. Itere en /rest/v1/Systems y elija un ComputerSystem (Sistema de Ordenador) miembro.

   Resultado = *{direcciónipdeilo}*/rest/v1/Systems/1/BIOS
2. Encuentre un vínculo en Oem/Hp/links denominado Bios y tenga en cuenta el BiosURI.
3. GET (obtener) BiosObj desde BiosURI y tenga en cuenta que solo permite el comando GET (obtener) (se trata de la configuración actual).
4. Busque un vínculo en BiosObj denominado Settings (Configuraciones) y tenga en cuenta este URI.
5. Obtenga la configuración del BIOS mediante el URI en el paso 4.

   GET *{direcciónipdeilo}*/rest/v1/Systems/1/BIOS/Settings

6. Cree un nuevo objeto JSON con la propiedad `AdminName (Nombre del administrador)` cambiada a `{"AdminName":"Joe Smith"}`.
7. Actualice la configuración del BIOS. Solo necesita enviar la propiedad `AdminName (Nombre del Administrador)` actualizada en el cuerpo de la solicitud.

   `PATCH {direcciónipdeilo}/rest/v1/Systems/1/BIOS/Settings`
8. Obtenga la configuración del BIOS para asegurarse de que ha realizado el cambio a la propiedad `AdminName (Nombre del Administrador)`.

   `GET {direcciónipdeilo}/rest/v1/Systems/1/BIOS/Settings`

   La configuración del BIOS se valida y adopta al reiniciar el servidor.

Más información

Python: Consulte `ex1_change_bios_setting()` en el código de ejemplo de Python; allí se demuestra cómo encontrar la configuración del BIOS y parchear nuevos valores.

Ejemplo de activación del arranque seguro de UEFI del BIOS

Los privilegios de ID de sesión mínimos necesarios son `Configure` (Configurar).

1. Itere en /rest/v1/Systems y elija un ComputerSystem (Sistema de Ordenador) miembro. Busque un recurso secundario de tipo `HpSecureBoot` que permita operaciones `PATCH (Parchear)` (es posible que exista que más de uno, pero para este ejercicio, escoja el primero).

   `{direcciónipdeilo}/rest/v1/Systems/1/SecureBoot`
2. Obtenga la configuración de arranque seguro.

   `GET {direcciónipdeilo}/rest/v1/Systems/1/SecureBoot`
3. Cree un nuevo objeto JSON con la propiedad `SecureBootEnable` cambiada a `{"SecureBootEnable":true}`.
4. Actualice la configuración de arranque seguro. Solo necesita enviar la propiedad `SecureBootEnable` actualizada en el cuerpo de la solicitud.

   `PATCH {direcciónipdeilo}/rest/v1/Systems/1/SecureBoot`

   La configuración de arranque se valida y adopta al reiniciar el servidor.

Ejemplo de reversión de la configuración de UEFI del BIOS a los valores predeterminados

El recurso de la configuración del BIOS es compatible con una función especial que le permite restaurar la configuración predeterminada del BIOS para el recurso seleccionado. Esto se consigue llevando a cabo la operación `PATCH` (Parchear) o `PUT` (Colocar) en una propiedad especial en el objeto de configuración del BIOS: `{"BaseConfig": "default"}`. Esto se puede combinar con otros conjuntos de propiedades para, en primer lugar, definir los valores predeterminados y, luego, definir una configuración específica en una sola operación.

NOTA: Es posible que la propiedad `BaseConfig (Configuración de Base)` todavía no exista en los recursos BIOS o configuración del BIOS. Para determinar si el recurso BIOS admite la reversión de los valores de configuración a los valores predeterminados, realice una operación `GET (obtener)` con el recurso BaseConfigs (Configuraciones de base) del BIOS y observe la propiedad `Capabilities (Capacidades)`.

Los privilegios de ID de sesión mínimos necesarios son `Configure` (Configurar).

1. Itere en `/rest/v1/Systems` y elija un `ComputerSystem (Sistema de Ordenador)` miembro. Busque un recurso secundario de tipo `HpBios` que permita operaciones `PUT`

(Colocar) (es posible que exista que más de uno, pero para este ejercicio, escoja el primero).

{direcciónipdeilo}/rest/v1/Systems/1/BIOS/Settings

2. Obtenga los valores de configuración de BIOS UEFI.

   GET *{direcciónipdeilo}*/rest/v1/Systems/1/BIOS/Settings

3. Cambie o añada la propiedad BaseConfig (Configuración de Base) a {"BaseConfig": "default"} en el cuerpo de la respuesta.
4. Actualice la configuración UEFI del BIOS.

   PUT *{direcciónipdeilo}*/rest/v1/Systems/1/BIOS/Settings

   Los valores de configuración de BIOS UEFI se restauran a los valores predeterminados al reiniciar el servidor.

NOTA:

- También puede observar los valores predeterminados para la configuración del BIOS buscando el tipo de recurso HpBaseConfigs (Configuraciones de Base de Hp).

  {direcciónipdeilo}/rest/v1/Systems/1/BIOS/BaseConfigs

- BaseConfig (Configuración de Base) se puede combinar con otros valores de propiedad para restablecer los valores predeterminados y luego aplicar algunos valores de configuración específicos en una sola operación.

Ejemplo de configuración del iniciador de software iSCSI

El iniciador de software iSCSI le permite configurar un dispositivo de destino iSCSI para utilizarlo como origen de arranque. El objeto de configuración actual del BIOS contiene un vínculo a un recurso independiente de tipo HpiSCSISoftwareInitiator. El recurso de configuración actual del BIOS y los recursos de configuración actuales del iniciador de software iSCSI son de solo lectura. Para cambiar la configuración de iSCSI, debe seguir otro vínculo al recurso Settings, que permite realizar operaciones PUT (Colocar) y PATCH (Parchear).

Las configuraciones de destino de iSCSI se representan en una propiedad iSCSIBootSources, que es una matriz de objetos, cada uno de los cuales contiene la configuración de un solo destino. El tamaño de la matriz representa el número total de orígenes de arranque iSCSI que se pueden configurar al mismo tiempo. Existen muchas propiedades mutables, como iSCSIBootAttemptInstance, que se puede establecer en un número entero único en el intervalo [1, *N*], donde *N* es el tamaño de la matriz de orígenes de arranque. De manera predeterminada, este *número de instancia* es 0 para todos los objetos, lo que indica que debe ignorarse el objeto al configurar iSCSI.

Cada objeto también contiene dos propiedades de solo lectura: StructuredBootString y UEFIDevicePath, que solo adquieren un valor cuando se configura correctamente el destino como un origen de arranque. El esquema correspondiente contiene más información sobre cada propiedad.

El nombre del iniciador iSCSI se representa mediante la propiedad iSCSIInitiatorName.

Una propiedad de solo lectura adicional, iSCSINicSources, solo se muestra en el recurso de configuración actual iSCSI. Esta propiedad es una matriz de cadenas que representan las posibles instancias de NIC que pueden utilizarse como destinos para la configuración de arranque desde iSCSI. Para confirmar a qué dispositivo NIC corresponde cada cadena, se recomienda hacer referencias cruzadas a otros dos recursos:

- Un recurso de tipo HpBiosMapping que se puede encontrar mediante un vínculo Mappings en el recurso de configuraciones actual del BIOS. Su propiedad BiosPciSettingsMappings contiene una matriz de asignaciones entre las cadenas de dispositivo específicas del BIOS (como la cadena de origen NIC) y una cadena

`CorrelatableID` que puede utilizarse para hacer referencia al mismo dispositivo en contextos que no sean del BIOS.

- En el recurso `ComputerSystem`, se puede encontrar una colección de `HpServerPciDevices` mediante un vínculo PCIDevices. Buscando el `CorrelatableID` que suele coincidir con un `UEFIDevicePath` se encuentra el dispositivo PCI específico correspondiente a la instancia de la NIC. Una vez que encuentre el recurso `HpServerPciDevice`, tendrá acceso a todas las propiedades legibles por el usuario útiles para describir un origen de NIC.

La configuración de `iSCSIBootSources` e `iSCSIInitiatorName` puede cambiarse a través de operaciones `PATCH (Parchear)`, de forma muy parecida a como se modifican los valores de `HpBios`. Sin embargo, mientras que todos los valores del BIOS se encuentran en un solo objeto plano, los valores de iSCSI están anidados en matrices y subobjetos. Cuando realice una operación `PATCH (Parchear)`, utilice objetos vacíos (`{}`) en lugar de los objetos de origen de arranque que **no** desee modificar.

En el ejemplo siguiente se describe una situación en que tiene configurados dos orígenes de arranque desde iSCSI y le gustaría editar algunos valores existentes y agregar un tercer origen.

1. Itere en `/rest/v1/Systems` y elija un `ComputerSystem (Sistema de Ordenador)` miembro. Busque un recurso secundario de tipo `HpiSCSISoftwareInitiator` que permita operaciones `PATCH (Parchear)`.

   `{dirección-ip-de-ilo}/rest/v1/Systems/1/BIOS/iSCSI/Settings`

2. Examine la matriz `iSCSIBootSources` existente. Tendrá que inspeccionar la propiedad `iSCSIBootAttemptInstance` de cada objeto para encontrar los orígenes de arranque que va a cambiar.

   Recurso de ejemplo existente:

```
{
            "iSCSIBootSources": [
                {
                    "iSCSIBootAttemptInstance": 1,
                    ...
                },
                {
                    "iSCSIBootAttemptInstance": 2,
                    ...
                },
                {
                    "iSCSIBootAttemptInstance": 0,
                    ...
                },
                {
                    "iSCSIBootAttemptInstance": 0,
                    ...
                }
            ],
            ...
        }
```

3. Cree un nuevo objeto JSON con la propiedad `iSCSIBootSources`.

```
{
            "iSCSIBootSources": [
                {},
                {
                    "iSCSIConnectRetry": 2
                },
                {
                    "iSCSIBootAttemptInstance": 3,
                    "iSCSIBootAttemptName": "Name",
                    "iSCSINicSource": "NicBootX",
```

```
                    ...
                },
                {}
            ]
        }
```

Utilice un objeto vacío en la posición de la instancia `1` para indicar que no debe modificarse. Utilice un objeto en la posición de la instancia `2` que contenga las propiedades que deben modificarse (todas las propiedades omitidas permanecerán inalteradas).

Para agregar un nuevo origen de arranque, busque cualquier posición de la instancia `0` y reemplácela por un objeto que contenga todos los valores nuevos y, lo que es más importante, un nuevo valor único de `iSCSIBootAttemptInstance`.

4. Cambie la configuración del iniciador de software iSCSI.

   `PATCH {dirección-de-ilo}/rest/v1/Systems/1/BIOS/iSCSI/Settings`

Configuración de arranque

Cadena de nombre estructurado del arranque UEFI

Esta cadena de nombre estructurado del arranque UEFI es exclusiva y representa cada opción de arranque UEFI en el sistema. El software puede identificar y manipular dispositivos con el formato fijo de la cadena como se define en esta especificación. El software puede dar por hecho la cadena exclusiva para cada dispositivo de arranque en el orden de arranque UEFI.

La cadena de nombre estructurado del arranque UEFI está dividida en secciones separadas por caracteres '.', con el siguiente formato:

<Tipo de dispositivo>.<Ubicación>.<Instancia>.<Subinstancia>.<Calificador>

- **Tipo de dispositivo**: La primera sección describe el tipo de dispositivo (por ejemplo, `HD`, `CD`, `NIC` y `PCI`).
- **Ubicación**: Las sección segunda segunda y tercera describen la ubicación del dispositivo (por ejemplo, `Slot.7` o `Emb.4`).
- **Instancia**: La tercera sección se usa con la sección *`Ubicación`* para describir la ubicación del dispositivo (por ejemplo, el número de ranura o el número de instancia integrada.
- **Subinstancia**La cuarta sección es opcional y se utiliza como un número de subinstancia en caso de varias opciones de arranque con la misma instancia. Por ejemplo, este puede ser el número de puerto para una NIC de varios puertos.
- **Calificador**: La quinta sección es opcional y describe el protocolo lógico (por ejemplo, IPv4, IPv6 e iSCSI).

La información de Structured Boot String (Cadena de arranque estructurado) forma parte de `BootSources[]property` en el objeto `HpServerBootSettings` y la propiedad `StructuredName` en el objeto `HpServerPciDevice`.

Ejemplos de cadenas de nombre estructurado del arranque UEFI

Tabla 1 Ejemplos

HD.Emb.4.2	La segunda instancia de una unidad de disco duro en la bahía 4 de la controladora SA integrada.
NIC.Slot.7.2.IPv4	Puerto 2 de una NIC en la ranura PCIe 7 que se activa para PXE IPv4.
NIC.FlexLOM.1.1.IPv6	Puerto 1 de FlexLOM de NIC incrustado, que se activa para PXE IPv6.

Tabla 1 Ejemplos *(continuación)*

PCI.Slot.6.1	Tarjeta PCIe en la ranura 6.
HD.FrontUSB.2.2	Segunda partición de la unidad flash en el puerto 2 USB delantero.

Tabla 2 Ejemplos de cadenas de arranque estructurado compatibles actualmente

Tipo de dispositivo	Ubicación	Instancia	Subinstancia	Calificador	Ejemplos de cadena de arranque estructurado
Unidad de disco duro de Smart Array	Integrada	Número de compartimento	Incremental por LUN		HD.Emb.1.1
	Ranura	Número de ranura	Incremental por LUN		HD.Slot.1.1
Controladora Smart Array	Integrada	Instancia de la controladora	1		RAID.Emb.1.1
	Ranura	Número de ranura	1		RAID.Slot.1.1
Controladora Dynamic Smart Array (RAID de software)	Integrada	1	1		Storage.Emb.1.1
Controladora Dynamic Smart Array (RAID de software)	Ranura	Instancia de la controladora	1		Storage.Slot.1.1
Unidad de disco duro SATA	Integrada	Puerto SATA n.º 1			HD.Emb.1.1
Controladora SATA	Integrada	Instancia de la controladora	1		SATA.Emb.1.1
Todas las demás controladoras de almacenamiento (FC, SAS, etc.)	Integrada	1	1		Storage.Emb.1.1
	Ranura	N.º de ranura	1		Storage.Slot.1.1
Adaptador de red	LOM	Número de NIC 1 para la primera NIC 2 para la segunda NIC	Número de puerto	IPv4 o IPv6, o iSCSI o FCoE	NIC.LOM.1.2.IPv4 NIC.LOM.1.2.IPv6
	FlexibleLOM	Número de FlexibleLOM 1 para el primer FlexLOM 2 para el segundo FlexLOM	Número de puerto	IPv4 o IPv6, o iSCSI o FCoE	NIC.FlexLOM.2.1.IPv4 NIC.FlexLOM.2.1.IPv6
	Ranura	Número de ranura	Número de puerto	IPv4 o IPv6, o iSCSI o FCoE	NIC.Slot.3.2.Ipv4
Adaptador Fibre Channel	Ranura	Número de ranura	Número de puerto	IPv4 o IPv6, o iSCSI o FCoE	PCI.Slot.3.1

Tabla 2 Ejemplos de cadenas de arranque estructurado compatibles actualmente *(continuación)*

Tipo de dispositivo	Ubicación	Instancia	Subinstancia	Calificador	Ejemplos de cadena de arranque estructurado
Entrada de arranque del sistema operativo (como "Gestor de arranque de Windows")	Ranura Integrada		Incremental		HD.Emb.1.2 HD.Slot.1.2
Llave USB	USB delantero	N.º de puerto USB	Incremental por LUN		HD.FrontUSB.1.1
	USB trasero	N.º de puerto USB	Incremental por LUN		HD.RearUSB.1.1
	Internal USB	N.º de puerto USB			HD.InternalUSB.1.1
	Soporte virtual de iLO				HD.Virtual.1.1
ISO, imagen	Soporte virtual de iLO				CD.Virtual.2.1
Disco de instalación virtual (VID)	Almacén incorporado	N.º de puerto USB			HD.VirtualUSB.1.1
Partición de usuario integrada	Almacén incorporado	N.º de puerto USB			HD.VirtualUSB.2.1
CD/DVD USB	USB delantero	N.º de puerto USB			CD.FrontUSB.1.1
	USB trasero	N.º de puerto USB			CD.RearUSB.1.1
	Internal USB	N.º de puerto USB			CD.InternalUSB.1.1
Tarjeta SD	Ranura de la tarjeta SD	N.º de puerto USB			HD.SD.1.1
Disco	USB delantero USB trasero	N.º de puerto USB			FD.FrontUSB.1.1 FD.RearUSB.1.1
Embedded UEFI Shell	Integrada	1	1		Shell.Emb.1.1
Aplicaciones UEFI (integradas en el firmware de la ROM) (Diag, utilidad del sistema, etc.)	Integrada	1	Incremental		App.Emb.1.1 App.Emb.1.2 App.Emb.1.3
Archivo	URL	Dirección URL distinta Aumentada en 1	1		File.URL.1.1
Dispositivo de disco de RAM de HPE	Memoria RAM	1	Número de puerto		RAMDisk.Emb.1.1

Tabla 2 Ejemplos de cadenas de arranque estructurado compatibles actualmente *(continuación)*

Tipo de dispositivo	Ubicación	Instancia	Subinstancia	Calificador	Ejemplos de cadena de arranque estructurado
Clase de dispositivo USB especial con ruta de dispositivo: UsbClass(0xFFFF, 0xFFFF, 0xFF, 0xFF, 0xFF)	Cualquier dispositivo USB del sistema	1			Generic.USB.1.1
Ranura vacía, sin dispositivo	Ranura	Número de ranura	1		PCI.Slot.2.1
Dispositivo desconocido	Integrada Ranura Ubicación desconocida	Número de ranura o 1	Incremental		Unknown.Slot.1.1 Unknown.Unknown.1.1
NVMe	Ranura	Número de ranura	Número de unidad NVMe (el número se basa en la secuencia de enumeración del bus).		NVMe.Slot.1.1
NVMe	Integrada	Número de compartimento	1 (cada compartimiento de unidad contiene una unidad NVMe).		NVMe.Emb.1.1

Ejemplo de cambio del orden de arranque UEFI

El objeto de configuración actual del BIOS contiene un vínculo a un recurso independiente de solo lectura de tipo `HpServerBootSettings (Configuraciones de arranque de servidor Hp)` que enumera la configuración actual del orden de arranque UEFI. Este es el orden de arranque del sistema cuando el sistema está configurado en el modo de arranque UEFI. El recurso de configuración actual de orden de arranque UEFI contiene una propiedad `BootSources (Fuentes de arranque)`, que es una matriz de las fuentes de arranque UEFI. Cada objeto de esa matriz cuenta con una `StructuredBootString (Cadena de Arranque Estructurada)` única, entre otras propiedades que identifican esa fuente de arranque.

La misma lista de orden de arranque UEFI se representa en una propiedad `PersistentBootConfigOrder (Orden de configuración de arranque persistente)` independiente, que es una matriz ordenada de fuentes de arranque, con referencia a su `StructuredBootString (Cadena de arranque estructurada)`. Además, una propiedad `DesiredBootDevices` (Dispositivos de arranque deseados) enumera una lista ordenada independiente de fuentes de arranque que pueden no aparecer en la propiedad `BootSources` (Fuentes de arranque). Esto resulta útil para la configuración de arranque desde un destino iSCSI, FC LUN o SCSI específico que puede no haberse configurado (y detectado por el BIOS) todavía.

Al igual que el recurso de configuración actual del BIOS, el recurso de la configuración actual del orden de arranque UEFI es de solo lectura (como demuestra el encabezado Allow (Permitir),

que no incluye `PATCH (Parchear)` como operación permitida). Para cambiar el orden de arranque UEFI, debe seguir el vínculo a un recurso de configuración independiente que puede llevar a cabo una operación `PATCH (Parchear)` que incluya la configuración del orden de arranque UEFI pendiente y actualizar la propiedad `PersistentBootConfigOrder (Orden de configuración de arranque persistente)` y/o la propiedad `DesiredBootDevices (Dispositivos de arranque deseados)` en dicho recurso de configuración. Los valores de configuración siguen pendientes hasta el siguiente reinicio y los resultados se reflejan en la propiedad `SettingsResults (Resultados de las configuraciones)` en el recurso de configuración actual del orden de arranque UEFI.

Requisitos previos

- Privilegios de ID de sesión mínimos necesarios: Configure (Configurar)

Ejemplo de cambio del orden de arranque UEFI

1. Itere en `/rest/v1/Systems` y elija un `ComputerSystem (Sistema de Ordenador)` miembro. Busque un recurso secundario de tipo `HpSecureBoot (Arranque Seguro de Hp)` que permita operaciones `PATCH (Parchear)` (es posible que exista que más de uno, pero para este ejercicio, escoja el primero).

   `{direcciónipdeilo}/rest/v1/Systems/1/BIOS/Boot/Settings`
2. Obtenga el orden de arranque UEFI

   `GET {direcciónipdeilo}/rest/v1/Systems/1/BIOS/Boot/Settings`
3. Cree un nuevo objeto JSON con la propiedad `PersistentBootConfigOrder (Orden de configuración de arranque persistente)` y cambie el orden de arranque.
4. Cambie el orden de arranque UEFI. Solo necesita enviar la propiedad `PersistentBootConfigOrder (Orden de configuración de arranque persistente)` actualizada en el cuerpo de la solicitud.

   `PATCH {direcciónipdeilo}/rest/v1/Systems/1/BIOS/Boot/Settings`

El nuevo orden de arranque se valida y utiliza al reiniciar el servidor.

Consideraciones sobre la contraseña de administrador del BIOS

Si ha activado una contraseña de administrador del BIOS, debe suministrar información adicional tras realizar una operación `PATCH` (Parchear) o `PUT` (Colocar) de la configuración del BIOS. La operación `PATCH (Parchear)` o `PUT` debe incluir un encabezado HTTP extra:

Nombre de encabezado HTTP	Valor
X-HPRESTFULAPI-AuthToken	Una cadena que consta del resumen hexagonal SHA256 en mayúsculas de la contraseña del administrador. En Python es `hashlib.sha256(bios_password.encode()).hexdigest().upper()`.

Ejemplo de restablecimiento de toda la configuración del orden de arranque y del BIOS a los valores predeterminados de fábrica

Restablecimiento de toda la configuración del orden de arranque y del BIOS a los valores predeterminados de fábrica

1. Itere en `/rest/v1/Systems` y elija un `ComputerSystem (Sistema de Ordenador)` miembro. Busque un recurso secundario de tipo `HpBios` que permita operaciones `PATCH (Colocar)` (es posible que exista que más de uno, pero para este ejercicio, escoja el primero).

   `{direcciónipdeilo}/rest/v1/Systems/1/BIOS/Settings`
2. Obtenga la configuración de orden de arranque y el BIOS.

   `GET {direcciónipdeilo}/rest/v1/Systems/1/BIOS`
3. Cree un nuevo objeto JSON con la propiedad `RestoreManufacturingDefaults (Orden de configuración de arranque persistente)` y cambie el valor a yes (sí).

Restablecimiento de toda la configuración del orden de arranque y del BIOS a los valores predeterminados de fábrica

4. Restablezca la configuración de orden de arranque y el BIOS. Solo necesita enviar la propiedad `RestoreManufacturingDefaults (Orden de configuración de arranque persistente)` actualizada en el cuerpo de la solicitud.

   `PATCH {direcciónipdeilo}/rest/v1/Systems/1/BIOS`

Encendido

El control de encendido del servidor es una entidad de nivel de nodo del sistema, no un control de nivel de chasis. Por ejemplo, puede encender un nodo en un chasis de varios nodos. Puede controlar el encendido llevando a cabo una operación HTTP en un objeto del nodo del sistema informático. Para obtener más información sobre los objetos del nodo del sistema informático, consulte «Sistema informático».

Algunas operaciones de la interfaz no son verdaderamente RESTful `GET (obtener)`, `PUT (Colocar)`, `POST (Publicar)`, `DELETE (Eliminar)` o `PATCH (Parchear)`. Se denominan acciones personalizadas y se llevan a cabo con una operación `POST (Publicar)` de HTTP que contiene una carga de solicitud específica. Por lo general, las acciones se definen cuando la acción que se desea llevar a cabo no está representada adecuadamente por las propiedades disponibles en el tipo. Por ejemplo, un botón de encendido no es legible, por lo que no se puede `GET (obtener)` el estado del botón de encendido. En este caso, pulsar el botón de encendido es una acción.

Las acciones son operaciones `POST (Publicar)` con una propiedad `Action (Acción)` que nombra la acción a realizar y cero o más propiedades de parámetros.

Ejemplo de restablecimiento de un servidor

Python: Consulte `ex2_reset_server()` en el código de ejemplo de Python.

Requisitos previos

- Privilegios de ID de sesión mínimos necesarios: Configure (Configurar)

Ejemplo de restablecimiento de un servidor

1. Itere en `/rest/v1/Systems` y elija un `ComputerSystem (Sistema de Ordenador)` miembro que permita operaciones `POST (Publicar)`.

   `{direcciónipdeilo}/rest/v1/Systems/1`

2. Construya un objeto Action para enviar a iLO.

   `{"Action":"Reset","ResetType":"ForceRestart"}`

3. Cambie las propiedades `Action (Acción)` y `ResetType (Tipo de Reinicio)` a `{"Action":"Reset","ResetType":"ForceRestart"}`.
4. Reinicie el servidor.

   `POST {direcciónipdeilo}/rest/v1/Systems/1`

El servidor se restablecerá y reiniciará.

6 Mensajes de error y registros de la API de RESTful para iLO

Los mensajes de error aparecen en varios lugares de la RESTful API para iLO.

- Una respuesta inmediata a una operación de HTTP.
- Un `SettingsResult (Resultado de las configuraciones)` en el modelo de datos cuando otros proveedores (por ejemplo, BIOS) procesaron la configuración en algún momento y desean comunicar el estado en el modelo.

Todos los casos de error utilizan una estructura de error JSON básica denominada `ExtendedError` (Error extendido) (`ExtendedInfo` [Información extendida] en Redfish), que tiene un `Type` (Tipo) `ExtendedError.0.9.5`. La propiedad más importante en `ExtendedError (Error extendido)` es `MessageID (ID del mensaje)`, una cadena que contiene una clave de búsqueda en un registro de mensajes.

`MessageID (ID del mensaje)` ayuda a reducir el mantenimiento de iLO manteniendo gran parte del texto explicativo de un error fuera de código. En su lugar, iLO proporciona una respuesta `ExtendedError (Error extendido)`, donde `MessageID (ID del mensaje)` proporciona información suficiente para buscar más detalles desde otro archivo.

Por ejemplo, si se `POST (Publica)` al servicio de licencia de iLO instalar una licencia de iLO, pero se proporciona una cadena de clave incorrecta, iLO contesta con un mensaje de error parecido al siguiente:

```
HTTP response 400
{
  "Type": "ExtendedError.0.9.5",
  "MessageID": "iLO.1.0.InvalidLicenseKey"
}
```

La respuesta HTTP 400 es la respuesta estándar de la API de RESTful a un error. En el ejemplo anterior, el error se comprende fácilmente, pero algunos errores no son de fácil comprensión. Para mostrar un mensaje de error más significativo, analice la cadena `iLO.0.9.InvalidLicenseKey` en los siguientes componentes:

- `iLO.0.9`: este es el nombre base del registro de mensajes que debe consultar. Busque un archivo de registro que coincida.
- `InvalidLicenseKey (Clave de licencia inválida)`: se trata de la clave de búsqueda en el registro de mensajes.

La búsqueda devuelve un resultado similar al siguiente:

```
"InvalidLicenseKey":{
  "Description": "The supplied license key is not valid.",
  "Message": "The supplied license key is not valid.",
  "Severity": "Warning",
  "NumberofArgs": 0,
  "ParamTypes": [],
  "Resolution": "Provide a valid license key."
}
```

Muchos mensajes de error también pueden devolver parámetros. Estos parámetros pueden conectarse a las cadenas en el registro para formar mensajes detallados adaptados a la instancia del mensaje de error.

Redfish: Redfish tiene una respuesta de error alternativa. Consulte la especificación Redfish 1.0 y el esquema `ExtendedInfo`.

7 Prácticas recomendadas para la creación de clientes evitando suposiciones incorrectas

Al desarrollar un cliente para la RESTful API, asegúrese de no basarse en suposiciones incorrectas no garantizadas. El motivo por el que es tan importante evitar esas suposiciones es que las implementaciones pueden variar según los sistemas y las versiones de firmware, y queremos que su código funcione adecuadamente.

Arquitectura de la API

Por diseño, la RESTful API es una API de hipermedia. De esta forma, se evita crear suposiciones restrictivas en el modelo de datos que dificultarán la adaptación a implementaciones de hardware futuras. Una API de hipermedia evita estas suposiciones haciendo que el modelo de datos sea detectable a través de los vínculos entre recursos.

El cliente no debe interactuar con un URI, ya que permanecerá estático. Solo los URI de nivel superior específicos (cualquier URI en el código de ejemplo) pueden considerarse estáticos.

Todos los URI, a excepción de los URI de nivel superior conocidos, deben detectarse dinámicamente siguiendo los vínculos href en el modelo de datos. Los clientes no deben realizar suposiciones acerca de los URI para los miembros de recursos de una colección. Por ejemplo, el URI de un miembro de la colección NO siempre será /rest/v1/.../collection/1 o 2. En Moonshot, un miembro de la colección del sistema puede ser /rest/v1/Systems/C1N1.

Recorrido del modelo de datos para buscar URI

Aunque los recursos del modelo de datos están vinculados, debido a las referencias cruzadas de vínculos entre recursos, un cliente no debe suponer que el modelo del recurso es un árbol. En realidad, es un gráfico, así que cualquier rastreo del modelo de datos debe realizar un seguimiento de los recursos visitados para evitar un bucle transversal infinito.

Una referencia a otro recurso es cualquier propiedad denominada `href` (`@odata.id` en Redfish), independientemente de dónde aparezca en un recurso.

Una referencia externa a un recurso fuera del modelo de datos se establece mediante una propiedad llamada "extref". No se debe dar por hecho que cualquier recurso al que se haga referencia mediante extref sigue las convenciones de la API.

Nombres de tipos y versiones

Cada recurso tiene una propiedad `Type` (Tipo) cuyo valor tiene el formato `Typename.x.y.z`, donde:

- x = versión principal: Si se aumenta el valor, hay un cambio brusco en el esquema.
- y = versión secundaria: Si se aumenta el valor, hay un cambio adicional que no provoca una interrupción en el esquema.
- z = errata: No hay ningún cambio brusco. Por ejemplo, texto de descripción alternativa, soluciones a errores de escritura, etc.

Acerca de las operaciones POST de HTTP para crear nuevos recursos

Cuando se usa una operación POST para crear un recurso (por ejemplo, crear una cuenta o sesión), una respuesta correcta incluye un encabezado HTTP `Location` (Ubicación) que indica el URI de recurso del recurso recién creado. La operación POST también puede incluir una representación del objeto recién creado en un cuerpo de la respuesta JSON, pero también puede no incluirlo. No presuponga el cuerpo de la respuesta. Pruébelo. También puede ser un objeto `ExtendedError`.

Redireccionamiento HTTP

Todos los clientes deben gestionar correctamente el redireccionamiento HTTP (por ejemplo, 308, 301, etc.). iLO 4 utilizará el redireccionamiento para asignar nombre a partes del modelo de datos y para migrar este a los URI de Redfish especificados (por ejemplo, `/redfish/…`).

FUTURO: Tareas asíncronas

En el futuro, algunas operaciones pueden iniciar tareas asíncronas. En este caso, el cliente debe reconocer y gestionar HTTP 202 si fuese necesario y el encabezado `Location` (Ubicación) hará referencia a un recurso con la información de la tarea y el estado.

Esquema frente a implementación

El esquema JSON disponible en `/rest/v1/Schemas` determina el contenido de los recursos, pero tenga en cuenta lo siguiente:

- No todas las propiedades del esquema se implementan en cada implementación.
- Se establece el esquema de algunas propiedades para permitir un valor nulo y otro tipo como una cadena o un número entero.

El código de cliente sólido debe comprobar la existencia y el tipo de las propiedades que resulten de interés y provocar un fallo controlado cuando no se cumplan las expectativas.

Información de adicional sobre los clientes

Los clientes siempre deben estar preparados para:

- Propiedades no implementadas (por ejemplo, si una propiedad no se aplica en un caso determinado).
- Los valores nulos en algunos casos si el valor de una propiedad no se conoce actualmente debido a las condiciones del sistema.
- Códigos de estado de HTTP distintos a 200 OK. ¿Puede su código gestionar el error interno del servidor HTTP 500 sin otra información?
- Los URI no distinguen mayúsculas y minúsculas.
- Los nombres de encabezado HTTP no distinguen mayúsculas y minúsculas.
- Los valores Enum y las propiedades de JSON distinguen mayúsculas y minúsculas.
- Un cliente debería soportar cualquier conjunto de encabezados HTTP devueltos por el servidor.

8 Eventos de RESTful y el servicio de eventos

A partir de iLO 4 2.30, iLO ofrece un nuevo servicio de suscripción a eventos que permite suscribirse para recibir notificaciones cuando cambien los datos REST o cuando se produzcan ciertas alertas. Estas notificaciones se presentan como operaciones POST de HTTPS con destino a un URI de su elección.

El servicio de eventos se encuentra en el modelo de datos en /rest/v1/EventService. Este recurso incluye un vínculo a una colección de suscripciones (denominada `EventSubscriptions` y que se encuentra en `/rest/v1/EventService/EventSubscriptions`).

Ejemplos de suscripción a eventos

Para poder recibir eventos, debe proporcionar un servidor HTTPS accesible a la red de iLO con un URI designado como destino para las operaciones POST de HTTPS iniciadas por iLO.

Construya un objeto JSON de tipo `ListenerDestination` (vea el ejemplo) y ejecute una operación POST a la colección indicada por el vínculo `EventSubscriptions` situado en `/rest/v1/EventService/EventSubscriptions`. Si recibe una respuesta HTTP 201 Created, significa que se ha agregado una suscripción nueva. Tenga en cuenta que iLO no comprueba el URI de destino durante esta fase, por lo que si el URI especificado no es válido, esta circunstancia no se indicará hasta que se generen los eventos y falle la conexión con el destino.

Ejemplo 1 Ejemplo de carga útil POST para la colección EventSubscriptions (`ListenerDestination`)

```
{
  "Destination": "https://myeventreciever/eventreceiver",
  "EventTypes": [
    "ResourceAdded",
    "ResourceRemoved",
    "ResourceUpdated",
    "StatusChange",
    "Alert"
  ],
  "HttpHeaders": {
    "Header”:”HeaderValue”
  },
  "TTLCount": 1440,
  "TTLUnits": "minutes",
  "Context": "context string",
  "Oem": {
    "Hp": {
      "DeliveryRetryIntervalInSeconds": 30,
      "RequestedMaxEventsToQueue": 20,
      "DeliveryRetryAttempts": 5,
      "RetireOldEventInMinutes": 10
    }
  }
}
```

Gran parte del contenido anterior depende por completo de su configuración y sus necesidades:

- **`Destination`** debe ser un URI HTTPS accesible para la red de iLO.
- **`EventTypes`** en el ejemplo incluye todos los tipos, pero puede quitar los que desee de la matriz.
- **`HttpHeaders`** le permite especificar cualquier encabezado HTTP arbitrario que necesite para la operación POST del evento. Tenga en cuenta que un usuario autorizado de iLO puede leer la suscripción a través de GET.

- **`Context`** puede ser cualquier cadena.

Consulte el esquema **`ListenerDestination`** para obtener más detalles sobre cada propiedad. La suscripción caducará automáticamente después de la información de TTL especificada y debe renovarse.

9 Solución de problemas

Restablecimiento de la API de RESTful utilizando un cliente web de REST de otros fabricantes

Síntoma

Es posible que los servidores ProLiant Gen9 experimenten un error de la API de RESTful durante el arranque del sistema que imposibilitará cambiar la configuración del BIOS mediante la RESTful API. Además, el siguiente mensaje de error persistente puede aparecer durante el arranque del sistema (POST) y se registra en el registro de gestión integrado (IML):

`335 RESTful API Error- RESTful API PUT request failed (HTTP: Status Code = 404) (335 Error de la API de RESTful- Fallo en la solicitud PUT de la API de RESTful (HTTP: Código de estado = 404))`

Con la versión v2.20 o posteriores del firmware de iLO, es posible restablecer la API de REST. Puede hacerlo a través de la RESTful API usando cualquier cliente web de REST de otros fabricantes, RESTful Interface Tool o desde el comando `restclient` de HPE Embedded UEFI Shell.

Acción

1. Ejecute una operación POST en el recurso en el URI `<ip-de-ilo>/rest/v1/managers/1` con el JSON siguiente en el cuerpo de la solicitud.

```
--------------- Inicio de la copia ----------------
{
    "Action":"ClearRestApiState",
    "Target":"/Oem/Hp"
}
---------------- Fin de la copia -----------------
```

2. Reinicie el servidor.

Restablecimiento de la API de RESTful para iLO utilizando RESTful Interface Tool

Síntoma

Es posible que los servidores ProLiant Gen9 experimenten un error de la API de RESTful durante el arranque del sistema que imposibilitará cambiar la configuración del BIOS mediante la RESTful API. Además, el siguiente mensaje de error persistente puede aparecer durante el arranque del sistema (POST) y se registra en el registro de gestión integrado (IML):

`335 RESTful API Error- RESTful API PUT request failed (HTTP: Status Code = 404) (335 Error de la API de RESTful- Fallo en la solicitud PUT de la API de RESTful (HTTP: Código de estado = 404))`

Con la versión v2.20 o posteriores del firmware de iLO, es posible restablecer la API de REST. Puede hacerlo a través de la RESTful API usando cualquier cliente web de REST de otros fabricantes, RESTful Interface Tool o desde el comando `restclient` de HPE Embedded UEFI Shell.

Causa

Acción

1. Descargue e instale RESTful Interface Tool. Para obtener más información sobre el uso de esta herramienta, consulte **http://www.hpe.com/info/resttool**.

2. Copie y pegue el JSON siguiente en un archivo de texto y guárdelo como `hprest_tool_clear_api.json`.

```
--------------- Inicio de la copia ----------------
{
    "path": "/rest/v1/managers/1",
    "body": {
        "Action":"ClearRestApiState",
        "Target":"/Oem/Hp"
    }
}
---------------- Fin de la copia -----------------
```

3. Inicie la herramienta hprest.

```
hprest
```

4. Inicie sesión en iLO.

```
hprest> login <ip-de-ilo>
```

5. Ejecute el comando siguiente con el archivo `hprest_tool_clear_api.json`.

```
hprest> rawpost hprest_tool_clear_api.json
```

6. Reinicie el servidor.

Restablecimiento de la API de RESTful para iLO utilizando el comando `restclient` de Embedded UEFI Shell

Síntoma

Es posible que los servidores ProLiant Gen9 experimenten un error de la API de RESTful durante el arranque del sistema que imposibilitará cambiar la configuración del BIOS mediante la RESTful API. Además, el siguiente mensaje de error persistente puede aparecer durante el arranque del sistema (POST) y se registra en el registro de gestión integrado (IML):

`335 RESTful API Error- RESTful API PUT request failed (HTTP: Status Code = 404) (335 Error de la API de RESTful- Fallo en la solicitud PUT de la API de RESTful (HTTP: Código de estado = 404))`

Con la versión v2.20 o posteriores del firmware de iLO, es posible restablecer la API de REST. Puede hacerlo a través de la RESTful API usando cualquier cliente web de REST de otros fabricantes, RESTful Interface Tool o desde el comando `restclient` de HPE Embedded UEFI Shell.

Causa

Acción

1. Entre en Embedded UEFI Shell. Para obtener más información, consulte la *Guía de usuario de UEFI Shell* en **http://www.hpe.com/servers/proliant/uefi**.
2. Copie y pegue el JSON siguiente en un archivo de texto ASCII y guárdelo como `clear_api.json` en un soporte USB con formato FAT.

```
--------------- Inicio de la copia ----------------
{
  "Action":"ClearRestApiState",
  "Target":"/Oem/Hp"
}
---------------- Fin de la copia -----------------
```

3. Conecte el soporte USB al servidor.
4. Encienda el servidor y arranque con Embedded UEFI Shell.

5. En la línea de comandos de UEFI Shell, utilice el comando `partitions` para localizar el sistema de archivos que corresponda al soporte USB. Por ejemplo, `FS0`, `FS1`, etc.
6. Para cambiar al sistema de archivos, escriba el nombre de este (por ejemplo, `shell>FS0:`).
7. Ejecute el comando siguiente:

   ```
   Fs0:> restclient –m POST –uri "/rest/v1/managers/1" –i clear_api.json
   ```
8. Reinicie el servidor.

Restablecimiento de la API de RESTful utilizando la CLI SSH de iLO

Síntoma

Es posible que los servidores ProLiant Gen9 experimenten un error de la API de RESTful durante el arranque del sistema que imposibilitará cambiar la configuración del BIOS mediante la RESTful API. Además, el siguiente mensaje de error persistente puede aparecer durante el arranque del sistema (POST) y se registra en el registro de gestión integrado (IML):

```
335 RESTful API Error- RESTful API PUT request failed (HTTP: Status
Code = 404) (335 Error de la API de RESTful- Fallo en la solicitud PUT
de la API de RESTful (HTTP: Código de estado = 404))
```

Con la versión v2.20 o posteriores del firmware de iLO, es posible restablecer la API de REST. Puede hacerlo a través de la RESTful API usando cualquier cliente web de REST de otros fabricantes, RESTful Interface Tool o desde el comando `restclient` de HPE Embedded UEFI Shell.

Causa

Acción

1. Abra una conexión SSH con iLO e inicie sesión utilizando una cuenta con privilegios de administrador. Para obtener más información, consulte la *Guía de secuencias y línea de comandos de HPE iLO 4* en **http://www.hpe.com/info/iLO**.
2. En la línea de comandos de la CLI, ejecute el comando `oemhp_clearRESTAPIstate`. Tenga en cuenta que este comando puede tardar unos cuantos segundos en completarse.
3. Reinicie el servidor.

Caracteres extraños en JSON al suscribirse a un EventType Alert

Síntoma

Al suscribirse a un EventType Alert, se incluyen caracteres extraños o adicionales al final del archivo JSON.

Causa

La escucha de eventos está realizando el análisis más allá del valor de `Content-Length` del encabezado.

Acción

10 Funcionalidad de la API de RESTful para iLO para los servidores Gen9

La tabla siguiente enumera las funciones adicionales de la API de REST disponibles para los servidores Gen9 que ejecutan iLO 4 2.30 en comparación con los servidores Gen8 que ejecutan iLO 4 2.30.

# Detalles en memoria mejorada de UEFI	
/rest/v1/Systems/{item}/Memory/{item}	HpMemory.1.0.0 #/Manufacturer
# Detalles de dispositivo PCI mejorado UEFI	
/rest/v1/Systems/{item}/PCIDevices/{item}	HpServerPciDevice.1.0.0 #/ClassCode
/rest/v1/Systems/{item}/PCIDevices/{item}	HpServerPciDevice.1.0.0 #/DeviceID
/rest/v1/Systems/{item}/PCIDevices/{item}	HpServerPciDevice.1.0.0 #/DeviceInstance
/rest/v1/Systems/{item}/PCIDevices/{item}	HpServerPciDevice.1.0.0 #/DeviceSubInstance
/rest/v1/Systems/{item}/PCIDevices/{item}	HpServerPciDevice.1.0.0 #/SegmentNumber
/rest/v1/Systems/{item}/PCIDevices/{item}	HpServerPciDevice.1.0.0 #/StructuredName
/rest/v1/Systems/{item}/PCIDevices/{item}	HpServerPciDevice.1.0.0 #/SubclassCode
/rest/v1/Systems/{item}/PCIDevices/{item}	HpServerPciDevice.1.0.0 #/SubsystemDeviceID
/rest/v1/Systems/{item}/PCIDevices/{item}	HpServerPciDevice.1.0.0 #/SubsystemVendorID
/rest/v1/Systems/{item}/PCIDevices/{item}	HpServerPciDevice.1.0.0 #/UEFIDevicePath
/rest/v1/Systems/{item}/PCIDevices/{item}	HpServerPciDevice.1.0.0 #/VendorID
# Detalles de la ranura PCI UEFI	
/rest/v1/Systems/{item}/PCISlots/{item}	HpServerPCISlot.1.0.0 #/UEFIDevicePath
# Recursos relacionados con el BIOS	
/rest/v1/Systems/{item}/SecureBoot	HpSecureBoot.1.0.0
/rest/v1/Systems/{item}/bios	HpBios.1.2.0
/rest/v1/Systems/{item}/bios/Settings	HpBios.1.2.0
/rest/v1/Systems/{item}/bios/iScsi/Settings	HpiSCSISoftwareInitiator.1.1.0
/rest/v1/Systems/{item}/bios/Mappings	HpBiosMapping.1.2.0
/rest/v1/Systems/{item}/bios/iScsi/BaseConfigs	HpBaseConfigs.0.10.0
/rest/v1/Systems/{item}/bios/iScsi	HpiSCSISoftwareInitiator.1.1.0
/rest/v1/Systems/{item}/bios/Boot/Settings	HpServerBootSettings.1.2.0
/rest/v1/Systems/{item}/bios/BaseConfigs	HpBaseConfigs.0.10.0
# Versiones de firmware del dispositivo PCI	
/rest/v1/Systems/{item}/FirmwareInventory	FwSwVersionInventory.1.0.0 #/Current/{PCIDeviceID}[]/ImageSizeBytes
/rest/v1/Systems/{item}/FirmwareInventory	FwSwVersionInventory.1.0.0 #/Current/{PCIDeviceID}[]/Key
/rest/v1/Systems/{item}/FirmwareInventory	FwSwVersionInventory.1.0.0 #/Current/{PCIDeviceID}[]/Location
/rest/v1/Systems/{item}/FirmwareInventory	FwSwVersionInventory.1.0.0 #/Current/{PCIDeviceID}[]/Updateable

/rest/v1/Systems/{item}/FirmwareInventory	FwSwVersionInventory.1.0.0 #/Current/{PCIDeviceID}[]/VersionString
# Nuevos componentes de firmware de Gen9	
/rest/v1/Systems/{item}/FirmwareInventory	FwSwVersionInventory.1.0.0 #/Current/SASProgrammableLogicDevice[]/Location
/rest/v1/Systems/{item}/FirmwareInventory	FwSwVersionInventory.1.0.0 #/Current/SASProgrammableLogicDevice[]/VersionString
/rest/v1/Systems/{item}/FirmwareInventory	FwSwVersionInventory.1.0.0 #/Current/StorageBattery[]/Location
/rest/v1/Systems/{item}/FirmwareInventory	FwSwVersionInventory.1.0.0 #/Current/StorageBattery[]/VersionString
/rest/v1/Chassis/{item}	HpServerChassis.1.0.0 #/Oem/Hp/Firmware/SASProgrammableLogicDevice/Current/VersionString

11 Asistencia y otros recursos

Acceso al soprte de Hewlett Packard Enterprise

- Para obtener asistencia en tiempo real, vaya a la página web Contact Hewlett Packard Enterprise Worldwide (Póngase en contacto con Hewlett Packard Enterprise en todo el mundo):

  www.hpe.com/assistance

- Para acceder a la documentación y los servicios de soporte técnico, vaya a la página web del centro de soporte de Hewlett Packard Enterprise:

  www.hpe.com/support/hpesc

Información que debe recopilar

- Número de registro de asistencia técnica (si corresponde)
- Nombre del producto, modelo o versión y número de serie
- Nombre y versión del sistema operativo
- Versión de firmware
- Mensajes de error
- Informes y registros específicos del producto
- Productos o componentes adicionales
- Productos o componentes de otros fabricantes

Acceso a las actualizaciones

- Algunos productos de software proporcionan un mecanismo para acceder a las actualizaciones de software a través de la interfaz del producto. Revise la documentación del producto para identificar el método recomendado de actualización del software.
- Para descargar actualizaciones del producto, vaya a cualquiera de las páginas web siguientes:
  - Página **Get connected with updates from HPE** (Conéctese con las actualizaciones de HPE) del centro de soporte de Hewlett Packard Enterprise:

    www.hpe.com/support/e-updates

  - Página web de Software Depot:

    www.hpe.com/support/softwaredepot

- Para ver y actualizar sus concesiones, así como para vincular sus contratos y garantías con su perfil, vaya a la página **More Information on Access to HP Support Materials** (Más información sobre cómo acceder a los materiales de soporte de HP) del centro de soporte de Hewlett Packard Enterprise:

  www.hpe.com/support/AccessToSupportMaterials

IMPORTANTE: El acceso a algunas actualizaciones podría requerir la concesión de producto cuando se accede a través del centro de soporte de Hewlett Packard Enterprise. Debe disponer de una cuenta de HP Passport configurada con las concesiones correspondientes.

Páginas web y documentos

Tabla 3 Páginas web

Página web	Enlace
Biblioteca de información de Hewlett Packard Enterprise	**http://www.hpe.com/info/enterprise/docs**
UEFI	**http://www.hpe.com/info/ProLiantUEFI/docs**
HPE Service Pack para ProLiant	**http://www.hpe.com/servers/spp/documentation**
HPE iLO 4	**http://www.hpe.com/info/ilo/docs**
HPE iLO University videos	**http://www.hpe.com/info/ilo/videos**
HPE Systems Insight Manager	**http://www.hpe.com/info/hpsim**
HPE Onboard Administrator	**http://www.hpe.com/info/oa**
HPE VMware Vibs Depot	**http://vibsdepot.hpe.com/**
Biblioteca de información de Hewlett Packard Enterprise	**www.hpe.com/info/enterprise/docs**
Centro de soporte de Hewlett Packard Enterprise	**www.hpe.com/support/hpesc**
Contacto con Hewlett Packard Enterprise en todo el mundo	**www.hpe.com/assistance**
Servicio de suscripción/alertas de soporte	**www.hpe.com/support/e-updates**
Software Depot	**www.hpe.com/support/softwaredepot**
Reparaciones del propio cliente	**www.hpe.com/support/selfrepair**
Insight Remote Support	**www.hpe.com/info/insightremotesupport/docs**
Soluciones Serviceguard para HP-UX	**www.hpe.com/info/hpux-serviceguard-docs**
Matriz de compatibilidad de dispositivos de almacenamiento de Single Point of Connectivity Knowledge (SPOCK)	**www.hpe.com/storage/spock**
Documentos técnicos e informes analíticos de almacenamiento	**www.hpe.com/storage/whitepapers**

Tabla 4 Documentos

Documento
Referencia del modelo de datos de la API de RESTful para iLO
Guía de secuencias y línea de comandos de HPE iLO 4
Notas de la versión de HPE iLO 4
Guía de usuario de las utilidades del sistema UEFI
Guía de solución de problemas de los servidores HPE ProLiant Gen9, Volumen I: Solución de problemas
Guía de usuario de HPE iLO Federation
Guía de inicio rápido de HPE Service Pack para ProLiant

Reparaciones del propio cliente

El programa de reparaciones del propio cliente (CSR) de Hewlett Packard Enterprise le permite reparar su producto. Si es necesario reemplazar una pieza incluida en el programa CSR, se le enviará directamente para que pueda instalarla cuando le resulte más cómodo. Algunas piezas

no entran en el programa CSR. El servicio técnico autorizado de Hewlett Packard Enterprise determinará si una reparación entra en el programa CSR.

Para obtener más información sobre el programa CSR, póngase en contacto con el proveedor de servicios local o vaya a la página web del CSR:

www.hpe.com/support/selfrepair

Remote Support

El soporte remoto está disponible con los dispositivos compatibles como parte de su garantía o de un contrato de soporte. Proporciona diagnóstico inteligente de eventos y envío automático y seguro de notificaciones de eventos de hardware a Hewlett Packard Enterprise, que iniciará un proceso de solución rápido y preciso basándose en el nivel de servicio de su producto. Hewlett Packard Enterprise le recomienda que registre su dispositivo en Remote Support.

Para obtener más información y conocer los detalles de los dispositivos compatibles, vaya a la página web siguiente:

www.hpe.com/info/insightremotesupport/docs

Comentarios sobre la documentación

Hewlett Packard Enterprise se compromete a proporcionar documentación que se adapte a sus necesidades. Para ayudarnos a mejorar la documentación, envíe cualquier error, sugerencia o comentario a Comentarios sobre la documentación (**docsfeedback@hpe.com**). Cuando envíe sus comentarios, incluya el título del documento, el número de referencia, la edición y la fecha de publicación, que se encuentran en la portada del documento. Para el contenido de ayuda en línea, incluya el nombre y la versión del producto, la edición y la fecha de publicación de la ayuda, que se encuentran en la página de avisos legales.

A Información normativa y sobre la garantía

Para obtener información importante de seguridad, medioambiental y normativa, consulte *Información de seguridad y certificación para productos de servidor, almacenamiento, alimentación, red y en bastidor*, disponible en **www.hpe.com/support/Safety-Compliance-EnterpriseProducts**.

Información de garantía

Servidores y opciones HPE ProLiant y x86

www.hpe.com/support/ProLiantServers-Warranties

HPE Enterprise Servers

www.hpe.com/support/EnterpriseServers-Warranties

Productos de almacenamiento de HPE Storage Products

www.hpe.com/support/Storage-Warranties

Productos de conexión de redes HPE

www.hpe.com/support/Networking-Warranties

Información normativa

Marca para Belarús, Kazajistán y Rusia

EAC

Información del fabricante y del representante local

Información del fabricante:

- Hewlett Packard Enterprise Company, 3000 Hanover Street, Palo Alto, California 94304, EE. UU.

Información del representante local ruso:

- **Rusia**:

  ООО «Хьюлетт Паккард Энтерпрайз», Российская Федерация, 125171, г. Москва, Ленинградское шоссе, 16А, стр.3, Телефон/факс: +7 495 797 35 00

- **Bielorrusia**:

  ИООО «Хьюлетт-Паккард Бел», Республика Беларусь, 220030, г. Минск, ул. Интернациональная, 36-1, Телефон/факс: +375 17 392 28 18

- **Kazajistán**:

  ТОО «Хьюлетт-Паккард (К)», Республика Казахстан, 050040, г. Алматы, Бостандыкский район, проспект Аль-Фараби, 77/7, Телефон/факс: + 7 727 355 35 50

Información del representante local kazajo:

- **Rusia**:

  ЖШС "Хьюлетт Паккард Энтерпрайз" Ресей Федерациясы, 125171, Мәскеу, Ленинград тас жолы, 16А блок 3, Телефон/факс: +7 495 797 35 00

- **Bielorrusia**:

  «HEWLETT-PACKARD Bel» ЖШС, Беларусь Республикасы, 220030, Минск қ., Интернациональная көшесі, 36/1, Телефон/факс: +375 17 392 28 18

- **Kazajistán**:

  ЖШС «Хьюлетт-Паккард (К)», Қазақстан Республикасы, 050040, Алматы к., Бостандык ауданы, Әл-Фараби даңғ ылы, 77/7, Телефон/факс: +7 727 355 35 50

Fecha de fabricación:

La fecha de fabricación está definida en el número de serie.

CCSYWWZZZZ (formato de número de serie de este producto)

Los formatos de fecha válida incluyen:

- YWW, donde Y indica el año a partir de cada nueva década, con 2000 como punto de partida; por ejemplo, 238:2 para 2002 y 38 para la semana del 9 de septiembre. Además, 2010 se indica con 0, 2011 con 1, 2012 con 2, 2013 con 3 y así sucesivamente.
- YYWW, donde YY indica el año a partir del año 2000; por ejemplo, 0238:02 para 2002 y 38 para la semana del 9 de septiembre.

Declaración de contenido de materiales RoHS para Turquía

Türkiye Cumhuriyeti: EEE Yönetmeliğine Uygundur

Declaración de contenido de materiales RoHS para Ucrania

Обладнання відповідає вимогам Технічного регламенту щодо обмеження використання деяких небезпечних речовин в електричному та електронному обладнанні, затвердженого постановою Кабінету Міністрів України від 3 грудня 2008 № 1057