文件编号：DSP2050

日期：2017 年 6 月 30 日

版本：1.0.0

Redfish 可组合性白皮书

文件分级：供参考文件

文件状态：已出版

文件语言：中文（中国）

DMTF 是一家非营利性组织，其行业成员的主要目的在于促进企业和系统的管理与互通性。成员和非成员都可以复制 DMTF 的规范和文件，但在引用相关文件时必须明确说明其具体出处。DMTF 规范文件可能会随时进行修改，因此用户必须随时注意相关文件的具体版本号和发行日期。

实施本标准或建议标准中的部分要素可能会涉及到第三方的专利权，包括临时性专利权（简称为“专利权”）。DMTF 不负责向标准用户告知或说明上述专利权，也不负责识别、披露。或者确定任何或所有上述第三方专利权、所有者或索赔人；也不对在确定或者披露上述权利、所有者或索赔人过程中出现的不完整或不准确信息负责。在任何情况下，DMTF 都不对未能识别、披露或者确定上述第三方专利权的行为、也不对相关方依赖标准文件，或者将其纳入到其产品、协议或测试流程中的行为承担任何形式的责任。DMTF 不对实施本标准文件的任何一方承担责任，不论相关实施行为是否可以被预见；也不对任何专利所有者或索赔人承担责任；也不对标准在出版之后被撤销或者进行修改导致的任何成本或损失承担责任。对于实施本标准的任何方而言，如果其在实施标准的过程中侵犯了任何专利权，则其必须针对专利所有者提出的索赔要求向 DMTF 进行赔偿，并保证 DMTF 免受任何不利影响。

更多关于向 DMTF 发出通知，认为其持有的专利权可能与 DMTF 标准相关、或者会影响标准实施的持有专利权的第三方的信息，请参阅下列网站中的相关内容：http://www.dmtf.org/about/policies/disclosures.php。

本标准文件使用的语言为英语，但允许将其翻译为其他语言。

目录

序言

“Redfish 可组合性白皮书”是由 DMTF 的可扩展平台管理论坛编制的。

DMTF 是一家非营利性组织，其行业成员的主要目的在于促进企业和系统的管理与互通性。更多关于 DMTF 的信息，请参阅：http://www.dmtf.org。

致谢

DMTF 在此感谢下列人员在本文件编制过程中做出的贡献：

- 拉菲克.艾哈迈德.K：惠普公司；
- 杰夫.奥特尔：惠普公司；
- 迈克尔.杜：联想公司；
- 杰夫.西兰德：惠普公司；
- 约翰.梁：英特尔公司；
- 史蒂夫.莱尔：惠普公司；
- 迈克尔. 拉伊内里：戴尔公司；
- 保罗.冯.贝赫伦：英特尔公司。

概述

由于世界不断变迁为软件定义的范式，这就需要硬件管理能力能够随之进化，以便应对数据中心的上述变迁。对于诸如托盘、模块、硅器件等分解式硬件而言，需要将其组合起来以便创建组合的逻辑系统。这些逻辑系统的功能与传统行业中使用的标准机架式系统相同。通过这种方式可以让用户能够动态配置其硬件，以便满足其工作负载的要求。除此之外，用户还必须能够对系统的全寿命周期进行管理，例如在不移动任何实体设备的情况下，在逻辑系统中加入更多的计算资源。

Redfish 是一项不断进化的硬件管理标准，其设计方式具有灵活性、可扩展性和可互通性。Redfish 中包括一个用于描述可组合硬件的数据模型，以及供客户端管理组合资源的界面。此规范文件将帮助实施者和客户端理解 Redfish 可组合性数据模型，以及组合请求应该采取何种具体形式。

可组合性模型

如果 Redfish 服务支持可组合性，则在服务根目录（Service Root）资源中将会含有一个 CompositionService（组合服务）属性。在 CompositionService 属性中，客户端可以找到能够组合为新资源的所有部件的列表目录（资源块），含有不同部件组合限制条件的描述符（资源区），以及向客户端说明如何提出组合请求的注释项（集合能力）。在下文各条中，将详细介绍 Redfish 服务如何处理这些事情。

1. 组合服务

组合（Composition）服务是所有与可组合性相关的事宜的顶层资源。其中含有状态和控制指标属性，例如 Status（状态）和 ServiceEnabled（服务启用）属性。在各种 Redfish 服务实例中都可以找到这些通用属性。其中还含有分别通过 ResourceBlocks（资源块）和 ResourceZones（资源区）属性链接到资源块和资源区集合的链接。其中资源块将在“资源块”一条中进行介绍，资源区将在“资源区”一条中进行介绍。

组合服务资源示例：


```
{
    "@odata.context": "/redfish/v1/
$metadata#CompositionService.CompositionService",
    "@odata.type": "#CompositionService.v1_0_0.CompositionService",
    "@odata.id": "/redfish/v1/CompositionService",
    "Id": "CompositionService",
    "Name": "Composition Service",
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    },
    "ServiceEnabled": true,
    "ResourceBlocks": {
        "@odata.id": "/redfish/v1/CompositionService/ResourceBlocks"
    },
    "ResourceZones": {
        "@odata.id": "/redfish/v1/CompositionService/ResourceZones"
    }
}
```


2. 资源块

资源块是组合请求中最低级别的构建模块。资源块中含有关于资源块实例的状态和控制信息。其中还有在该资源块实例中可以找到的部件列表。例如，如果资源块中含有 1 个处理器和四根内存条，则所有这些部件都可以成为组合请求中的一部分，即使只需要其中的一个部件。在完全分解化的系统中，客户端很可能在每个资源模块中只能找到一个部件实例。资源模块及其部件将无法被系统软件使用，直到其共同构成一个组合资源为止。例如，如果资源块中包括一个驱动器实例，则在提出使用其资源块的组合请求之前，该驱动器不属于任何特定的计算机系统。

在 ResourceBlockType（资源块类型）属性中，含有资源块中部件类型的分级信息，此属性可以帮助客户端快速识别资源块。每个 ResourceBlockType 属性都与包括在该资源块内的具体架构要素相关。例如，如果此属性的取值为“Storage（存储）”，则客户端可以在不需要查看具体部件资源的情况下，判断此资源块中含有存储相关设备，例如存储控制器或者驱动器。此属性的“Compute（计算）”取值具有特殊含义，此取值用于描述包括有绑定的处理器和内存部件、可以作为计算子系统共同运行的资源块。

CompositionStatus（组合状态）属性是一个包含下列两个属性的对象：CompositionState（组合状态）和 Reserved（已保留），其中 CompositionState 用于向客户端说明此资源模块在部件中的应用状态；Reserved 是一个可写入的标记项，客户端可以利用其来说明此资源块已经被客户端识别、且将被用于组合中。如果

第二台客户端试图识别此资源块中的资源用于组合，则在其看到 Reserved 的取值为“true”时，则第二台客户端就会知道此资源已经配置，无法使用。之后，第二台客户端将会转移到下一个资源块进行进一步处理。Redfish 服务不会为 Reserved 标记项提供任何类型的保护。任何客户端都可以修改其状态，所有客户端都具有公平的权限。

在资源块中具有多个各种部件类型的链接数组，例如 Porcessor（处理器）、Memory（内存）、以及 Storage（存储器）数组。所有这些链接都最终指向资源块内的具体部件。在提出组合请求之后，这些部件可以用于组合为新的组合资源。在资源块中含有一套或多套完整的计算机系统时，资源块将会使用 ComputerSystem（计算机系统）数组。通过此数组，客户端可以利用小规模的计算机系统，创建一个单一的组合式计算机系统。

Links（链接）属性中含有对相关资源的引用。Chassis（机架）数组中含有囊括资源模块内资源的机架实例。ComputerSystems（计算机系统）数组中含有使用资源模块作为其组合的一部分的计算机系统实例。Zones（区域）数组中含有囊括资源模块的资源区链接。

资源块资源的示例如下所述：


```
{}{
    "@odata.context": "/redfish/v1/$metadata#ResourceBlock.ResourceBlock",
    "@odata.type": "#ResourceBlock.v1_0_0.ResourceBlock",
    "@odata.id": "/redfish/v1/CompositionService/ResourceBlocks/DriveBlock3",
    "Id": "DriveBlock3",
    "Name": "Drive Block 3",
    "ResourceBlockType": [ "Storage" ],
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    },
    "CompositionStatus": {
        "Reserved": false,
        "CompositionState": "Composed"
    },
    "Processors": [],
    "Memory": [],
    "Storage": [
        {}{
            "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/DriveBlock3/Storage/Block3NVMe"
```

```
            }
        ],
        "Links": {
            "ComputerSystems": [
                {{
                    "@odata.id": "/redfish/v1/Systems/ComposedSystem"
                }
            ],
            "Chassis": [
                {{
                    "@odata.id": "/redfish/v1/Chassis/ComposableModule3"
                }
            ],
            "Zones": [
                {{
                    "@odata.id": "/redfish/v1/CompositionService/ResourceZones/1"
                },
                {{
                    "@odata.id": "/redfish/v1/CompositionService/ResourceZones/2"
                }
            ]
        }
    }
```


在上面的示例中，资源块的类型为 Storage（存储），其中含有一台存储器实体。从 CompositionStatus 属性，可以看出此资源块在组合中使用。从 Links 属性，可以看出该资源块正在被名为 ComposedSystem 的计算机系统使用。

3. 资源区

资源区的主要功能是向客户端说明服务识别到的资源块的各种组合限制条件。同属于一个资源区的资源块可以组合在一起。通过此功能，可以让客户端在不需要执行尝试-失败逻辑的情况下，就能找出特定实施中存在的各种组合限制条件。除此之外，每个资源区中都带有集合能力（Collection Capabilities）注释项，用于描述每个资源区能够进行的具体组合。更多这方面的信息将在本文件“集合能力”一条中进行更多详细介绍。

资源区资源示例如下所述：


```
{{
    "@odata.context": "/redfish/v1/$metadata#Zone.Zone",
    "@odata.type": "#Zone.v1_1_0.Zone",
    "@odata.id": "/redfish/v1/CompositionService/ResourceZones/1",
    "Id": "1",
    "Name": "Resource Zone 1",
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    },
    "Links": {
```

```
        "ResourceBlocks": [
            {}{
                "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/ComputeBlock1"
            },
            {}{
                "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/DriveBlock3"
            },
            {}{
                "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/DriveBlock4"
            },
            {}{
                "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/DriveBlock5"
            },
            {}{
                "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/DriveBlock6"
            },
            {}{
                "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/DriveBlock7"
            }
        ]
    },
    "@Redfish.CollectionCapabilities": {
        "@odata.type": "#CollectionCapabilities.v1_0_0.CollectionCapabilities",
        "Capabilities": [
            {}{
                "CapabilitiesObject": {
                    "@odata.id": "/redfish/v1/Systems/Capabilities"
                },
                "UseCase": "ComputerSystemComposition",
                "Links": {
                    "TargetCollection": {
                        "@odata.id": "/redfish/v1/Systems"
                    }
                }
            }
        ]
    }
}
```


在上面的示例中，资源块 ComputeBlock1、DriveBlock3、DriveBlock4、DriveBlock5、DriveBlock6、以及 DriveBlock7 同属于一个资源区。除此之外，此资源区的集合能力注释项中说明，此资源区可以用于组合形成/redfish/v1/Systems 集合中的计算机系统。

4. 集合能力

集合能力可以在资源区以及集合能力本身中找到。这是因为集合能力还适用于资源区之外的其他语境。集合能力可以通过响应消息体中的@Redfish.CollectionCapabilities 注释项进行识别。此注释项的主要功能是向客户端说明在特定集合中、根据具体的用途示例创建（POST）操作的请求消息体应该采取何种具体形式，如果此操作可能会在特定集合中加入新的成员。

4.1 集合能力注释项

在集合能力（Collection Capabilities）注释项中，有一个名为 Capabilities（能力）的属性。此属性是一个数组，其中包括特定资源区或资源集合内的所有能力。在每个 Capabilities 数组实例内部，都有一个用于描述具体能力的对象。

CapabilitiesObject（能力对象）属性中含有指向潜在对象实例的 URI，用于描述负载的格式。更多相关信

息将在下一节中进行介绍。

UseCase（用途实例）属性用于向客户端说明具体创建（POST）操作的环境。下表中列出了可组合性服务中使用的 UseCase 属性的不同取值。每个取值都与一种具体类型的可组合资源对应、也与请求中规定的组合类型对应。

UseCase 取值	组合资源	组合类型
ComputerSystemComposition	ComputerSystem	自定义组合

在 Links 对象中的 TargetCollection（目标集合）属性中，包含有可以接受特定能力的资源集合的 URI。客户端可以根据 CapabilitiesObject（能力对象）中的描述内容，对此 URI 执行创建（POST）操作。

集合能力注释项示例如下所述：

```
{
    "@Redfish.CollectionCapabilities": {
        "@odata.type": "#CollectionCapabilities.v1_0_0.CollectionCapabilities",
        "Capabilities": [
            {
                "CapabilitiesObject": {
                    "@odata.id": "/redfish/v1/Systems/Capabilities"
                },
                "UseCase": "ComputerSystemComposition",
                "Links": {
                    "TargetCollection": {
                        "@odata.id": "/redfish/v1/Systems"
                    }
                }
            }
        ]
    },
    ...
}
```

上述注释项中包含了一个单一的能力。从 UseCase 开始，此能力描述了如何通过具体的资源模块创建新的计算机系统的创建（POST）请求。除此之外，TargetCollection（对象集合）属性表明，客户端可以向资源集合/redfish/v1/Systems 提出请求，客户端创建的新的资源实例将被包含在此集合中

4.2 集合能力对象

集合能力对象将遵守客户端能够创建的新资源的架构。例如，如果对象描述的是创建新的计算机系统实例的请求应该采用何种形式，则对象的类型将为 ComputerSystem.vX_Y_Z. ComputerSystem，其中 vX_Y_Z 是指服务器支持的 ComputerSystem（计算机系统）的版本号。

对象本身中也含有带注释的属性，客户端可以在创建（POST）请求的消息体中予以使用。在对象中还会列出可选属性，以及在新资源创建之后可能会有的限制性属性。下表中列出了在集合能力对象属性中使用的各种不同注释项。

属性注释项	描述

Redfish.RequiredOnCreate	客户端必须在创建（POST）请求消息体中提供特定的属性。
Redfish.OptionalOnCreate	客户端可以在创建（POST）请求消息体中提供特定的属性。
Redfish.SetOnlyOnCreate	如果客户端具有属性需要的取值，则其必须在创建（POST）请求消息体中提供；此属性有点类似于资源创建之后的“只读”属性。
Redfish.UpdatableAfterCreate	在资源创建之后，允许客户端更新资源。
Redfish.AllowableValues	在特定属性的创建（POST）请求消息体中，允许客户端使用任何规定的取值。

集合能力对象示例如下所述：


```
{
    "@odata.context": "/redfish/v1/$metadata#ComputerSystem.ComputerSystem",
    "@odata.type": "#ComputerSystem.v1_4_0.ComputerSystem",
    "@odata.id": "/redfish/v1/Systems/Capabilities",
    "Id": "Capabilities",
    "Name": "Capabilities for the Zone",
    "Name@Redfish.RequiredOnCreate": true,
    "Name@Redfish.SetOnlyOnCreate": true,
    "Description@Redfish.OptionalOnCreate": true,
    "Description@Redfish.SetOnlyOnCreate": true,

    "HostName@Redfish.OptionalOnCreate": true,
    "HostName@Redfish.UpdatableAfterCreate": true,
    "Boot@Redfish.OptionalOnCreate": true,
    "Boot": {
        "BootSourceOverrideEnabled@Redfish.OptionalOnCreate": true,
        "BootSourceOverrideEnabled@Redfish.UpdatableAfterCreate": true,
        "BootSourceOverrideTarget@Redfish.OptionalOnCreate": true,
        "BootSourceOverrideTarget@Redfish.UpdatableAfterCreate": true,
        "BootSourceOverrideTarget@Redfish.AllowableValues": [
            "None",
            "Pxe",
            "Usb",
            "Hdd"
        ]
    },
    "Links@Redfish.RequiredOnCreate": true,
    "Links": {
        "ResourceBlocks@Redfish.RequiredOnCreate": true,
        "ResourceBlocks@Redfish.UpdatableAfterCreate": true
    }
}
```

在上面的示例中，有三个属性使用了 Redfish.RequiredOnCreate 注释项进行了注释，分别是 Links 对象内部的 Name（名称）、Links（链接）和 ResourceBlocks（资源块）属性。所有其他属性都使用 Redfish.OptionalOnCreate 注释项进行注释。尽管如此，Name（名称）和 Description（描述）属性使用了 Redfish.SetOnlyOnCreate 注释项进行注释，说明在新资源创建之后将无法修改。

组合类型

Redfish 可组合性数据模型具有一定的灵活性，确保服务实施者能够根据具体需求提出不同的组合类型。服务器会根据集合能力注释项中的 UseCase 属性的取值，向客户端说明允许进行了组合的类型。在现有的 Redfish 可组合性模型中，只定义了一种称为“自定义组合”的组合类型。

5. 自定义组合

自定义组合允许客户端通过预设的资源块和资源区，创建组合资源，并在其寿命周期内对其进行管理。由于资源块是资源区内的自包含实体，因此客户端在提出组合请求时，能够选取和选择具体的资源块。

根据绑定规则选取资源块、以及创建（POST）请求中具体资源块的详细信息，请参阅“创建组合资源”一节相关内容。

另一种能够与自定义组合实例相匹配的行业标准服务器设计将在“刀片分区模型”一节中进行定义。在此示例中，可以将多个带有刀片式外壳的分解式硬件机架组合在一起，构成被称为“分区服务器”的系统。这些分区可以通过自定义组合功能进行组合。在 Redfish 服务实施过程中，壳体内的每个刀片都可以定义为一个资源块，其 ResourceBlockType 属性的取值应该设定为“Compute”或者“Storage”，这样就能够允许客户端将多个资源块组合起来，创建组合计算机系统，也就是分区服务器。

自定义组合创建（POST）请求消息体示例如下所述：

```
{
    "Name": "Sample Composed System",
    "Links": {
        "ResourceBlocks": [
            { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/BladeComputeBlock1" },
            { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/BladeComputeBlock5" },
            { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/BladeStorageBlock8" }
        ]
    }
}
```

附录

6. 客户端基本工作流程

在使用 Redfish 可组合性模型创建和管理组合系统的过程中，客户端预计会使用到数种操作模式。在下面的示例中，客户端需要发起有效的会话、或者提供基本认证数据头。

6.1 识别 Redfish 服务是否支持组合

应用代码应该随时以下列根目录开始：/redfish/v1/。

1. 读取服务器根目录资源：
    1. 查找 CompositionService（组合服务）属性；
    2. 对该属性提供的 URI 执行 GET 操作；
    3. 查看 ServiceEnabled（服务启用）属性的取值是否为“true”。

```
Client|                                                | Redfish
Service
   |---- GET /redfish/v1/CompositionService ----->|
   |<--- { ..., "ServiceEnabled": true, ... } <---|
```

6.2 读取可用于组合的资源列表

客户端需要能够通过读取资源块和资源区集合，理解组合服务报告的组合模型。此关系可以用于执行 Redfish 服务支持的已报告 UseCase，更多信息将在后面的“创建组合资源”一节中进行介绍。

1. 读取资源块：
    1. 对组合服务 URI 执行 GET 操作；
    2. 查看 ResourceBlocks（资源块）属性；
    3. 对该 URI 执行 GET 操作，获得所有资源块列表；
    4. 为了获得特定资源块的访问详细信息，对在 Member（成员）数组中列出的特定条目的 URI 执行 GET 操作；
    5. 每个资源块中的 CompositionStatus（组合状态）属性可以用于识别组合请求中规定的资源块的可用性：
        - 在确定在组合请求中规定哪些资源块时，客户端应该考虑上述信息；
        - 根据 CompositionStatus 的具体取值，特定资源块当前可能不能用于组合。

```
{
 "@odata.type": "#ResourceBlockCollection.ResourceBlockCollection",
 "@odata.id": "/redfish/v1/CompositionService/ResourceBlocks",
 "Name": "Resource Block Collection",
 "Members@odata.count": 9,
 "Members": [
     { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/ComputeBlock1" },
     { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/ComputeBlock2" },
     { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/DriveBlock3" },
     { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/DriveBlock4" },
     { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/DriveBlock5" },
     { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/DriveBlock6" },
     { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/DriveBlock7" },
     { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/NetworkBlock8" },
     { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/OffloadBlock9" }
 ]
}
```


2. 读取资源区：

    1. 对组合服务 URI 执行 GET 操作；
    2. 查看 ResourceZones（资源区）属性；
    3. 对该 URI 执行 GET 操作，获得所有资源区列表；
    4. 为了获得特定资源区的访问详细信息，对在 Member（成员）数组中列出的特定条目的 URI 执行 GET 操作；


```
{
 "@odata.type": "#ZoneCollection.ZoneCollection",
 "@odata.id": "/redfish/v1/CompositionService/ResourceZones",
 "Name": "Resource Zone Collection",
 "Members@odata.count": 2,
 "Members": [
     { "@odata.id":
"/redfish/v1/CompositionService/ResourceZones/1" },
     { "@odata.id":
"/redfish/v1/CompositionService/ResourceZones/2" }
 ]
}
```


3. 读取每个资源区的集合能力：

    1. 使用在 Member（成员）数组中每个条目中查找到的 URI，对每个资源区执行 GET 操作；

2. 查看每个资源区的@Redfish.CollectionCapabilities 注释项：

   ■ 在客户端确定了需要创建哪种类型的组合之后，需要使用 UseCase（用途实例）属 性；

   ■ 在提出了组合请求之后，需要使用 TargetCollection（目标集合）属性。


```
{
 "@odata.context": "/redfish/v1/$metadata#Zone.Zone",
 "@odata.type": "#Zone.v1_1_0.Zone",
 "@odata.id": "/redfish/v1/CompositionService/ResourceZones/1",
 "Id": "1",
 "Name": "Resource Zone 1",
 "Status": {},
 "Links": {},
 "@Redfish.CollectionCapabilities": {
     "@odata.type":
"#CollectionCapabilities.v1_0_0.CollectionCapabilities",
     "Capabilities": [
         {
             "CapabilitiesObject": { "@odata.id":
"/redfish/v1/Systems/Capabilities" },
             "UseCase":"ComputerSystemComposition",
             "Links": {
                 "TargetCollection": { "@odata.id":
"/redfish/v1/Systems" },
                 "RelatedItem": [
                     { "@odata.id":
"/redfish/v1/CompositionService/ResourceZones/1" }
                 ]
             }
         }
     ]
 }
}
```


4. 读取每个集合能力对象：

JRI 执行 GET 操作。


```
{
 "@odata.context": "/redfish/v1/
$metadata#ComputerSystem.ComputerSystem",
 "@odata.type": "#ComputerSystem.v1_4_0.ComputerSystem",
```

```
"@odata.id": "/redfish/v1/Systems/Capabilities",
"Id": "Capabilities",
"Name": "Capabilities for the Zone",
"Name@Redfish.RequiredOnCreate": true,
"Name@Redfish.SetOnlyOnCreate": true,
"Description@Redfish.OptionalOnCreate": true,
"Description@Redfish.SetOnlyOnCreate": true,
"HostName@Redfish.OptionalOnCreate": true,
"HostName@Redfish.UpdatableAfterCreate": true,
"Boot@Redfish.OptionalOnCreate": true,
"Boot": {
    "BootSourceOverrideEnabled@Redfish.OptionalOnCreate": true,
    "BootSourceOverrideEnabled@Redfish.UpdatableAfterCreate": true,
    "BootSourceOverrideTarget@Redfish.OptionalOnCreate": true,
    "BootSourceOverrideTarget@Redfish.UpdatableAfterCreate": true,
    "BootSourceOverrideTarget@Redfish.AllowableValues": [
        "None",
        "Pxe",
        "Usb",
        "Hdd"
    ]
},
"Links@Redfish.RequiredOnCreate": true,
"Links": {
    "ResourceBlocks@Redfish.RequiredOnCreate": true,
    "ResourceBlocks@Redfish.UpdatableAfterCreate": true
}
}
```


6.3 创建组合资源

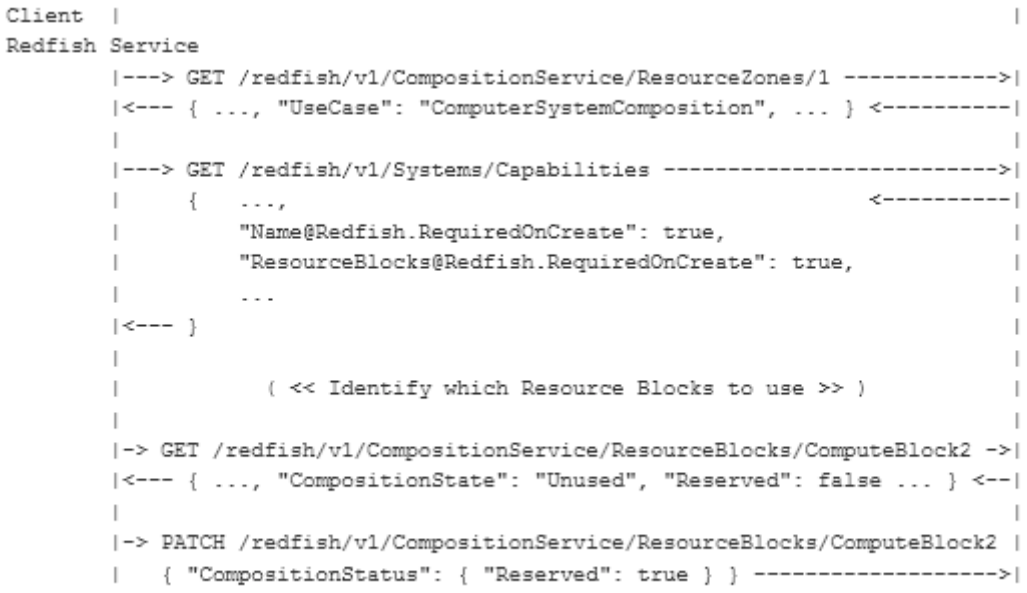
客户端可以通过下列步骤来提出组合请求：

1. 根据上文示例中所述的操作方法，对其集合 URI 执行 GET 操作，列出特定资源区内包含的所有资源块：
    - 在读取资源块时，需要注意其 CompositionStatus（组合状态）属性；
    - 根据 CompositionStatus 的具体取值，特定资源块当前可能不能用于组合。
2. （可选步骤）保留组合请求中说明的、已经识别到的每个资源块：
    - 对资源块执行 PATCH 操作，将其 Reserved（已保留）属性的取值设定为“true”；
    - 在有多台客户端提出组合请求时，可以实施此步骤操作。
3. 识别确定具体组合的 UseCase（用途实例）需求：
    1. 对相关的资源区执行 GET 操作；
    2. 在@Redfish.CollectionCapabilities 注释项中查看相匹配的 UseCase 取值：
        - 例如，如果需要通过特定的资源块列表创建新的计算机系统，就需要查看 ComputerSystemComposition（计算机系统组合）的取值；

3. 对在 CapabilitiesObject（能力对象）中查找到的 URI 执行 GET 操作；
4. 记下所有带有 RequiredOnCreate 注释项的属性：
    - 这些属性是需要纳入到组合请求中的属性。
5. 记下 TargetCollection（目标集合）的 URI：
    - 这是提出创建（POST）新组合的请求的目标 URI。

4. 使用所有带有 RequiredOnCreate 注释项的属性，编制创建（POST）请求消息体，并将其发送到 TargetCollection（目标集合）的 URI 处：
    - 在上文中所述示例的步骤 4 中，只需要查看 Links（链接）属性内的 Name （名称）和 ResourceBlocks（资源块）属性；
    - Redfish 服务可以接受在请求中提出其他属性，如果这些属性在后续不需要进行更新。
5. 在服务器响应中的 Location HTTP 数据头中，应该包括组合资源的 URI。

基本流程图：



客户端请求示例：

```
POST /redfish/v1/Systems HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: <computed-length>
OData-Version: 4.0
{
    "Name": "Sample Composed System",
    "Links": {
        "ResourceBlocks": [
            { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/ComputeBlock0" },
            { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/DriveBlock2" }
        ]
    }
}
```

服务器响应示例：

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8
Content-Length: <computed-length>
Location: /redfish/v1/Systems/NewSystem
```

在上述客户端请求示例中，列出了客户端针对/redfish/v1/Systems 处的计算机系统集合提出的组合请求。在此请求中，客户端使用 ComputeBlock0 和 DriveBlock2 资源块创建了新的计算机系统。在上述服务器响应示例中，服务器使用 201（创建成功）状态代码进行响应，说明新的计算机系统已经创建到了/redfish/v1/Systems/NewSystem 处。

6.4 更新组合资源

如果 Redfish 服务支持对现有的组合资源进行更新，则客户端可以通过 PUT/PATCH 指令更新已经创建的组合资源。具体做法为对组合资源内的 ResourceBlocks（资源块）数组进行更新。在使用 PATCH 指令时，所使用的数组语义与 Redfish 规范中规定的语义相同。

客户端请求示例：

```
PATCH /redfish/v1/Systems/NewSystem HTTP/1.1
Content-Type: application/json; charset=utf-8
Content-Length: <computed-length>
OData-Version: 4.0
{
    "Links": {
        "ResourceBlocks": [
            {},
            {},
            { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/NetworkBlock8" }
        ]
    }
}
```

在上述示例中，更新操作将保留组合资源中要素 0 和 1 中的现有资源块，同时在数组元素 2 处新增一个 NetWorkBlock8 资源块。

6.5 删除组合资源

客户端可以使用 DELETE 指令删除已经创建的组合资源，或者将其分解。

客户端请求示例：

```
DELETE /redfish/v1/Systems/NewSystem HTTP/1.1
```

在上述示例中，客户端要求将名为 NewSystem 的组合系统删除。在删除之后，此组合系统中使用的资源块将被释放，可以在未来的组合中使用。尽管如此，每个资源块中的 CompositionStatus（组合状态）中的 Reserved（已保留）标记项的取值将保持不变。如果客户端已经使用完相关的资源块，应该将 Reserved（已保留）标记项的取值修改为“false”。

7. 参考文献

- “可组合系统”和“刀片分区”模型：http://redfish.dmtf.org/redfish/v1；
- 组合服务架构：http://redfish.dmtf.org/schemas/v1/CompositionService_v1.xml；
- 资源块架构：http://redfish.dmtf.org/schemas/v1/ResourceBlock_v1.xml；
- 资源区架构：http://redfish.dmtf.org/schemas/v1/Zone_v1.xml；
- 集合能力架构：http://redfish.dmtf.org/schemas/v1/CollectionCapabilities_v1.xml。