Red Hat Satellite 6.15

API 指南

使用 Satellite REST API 开发自定义应用程序或集成

Last Updated: 2024-06-21

Red Hat Satellite 6.15 API 指南

使用 Satellite REST API 开发自定义应用程序或集成

Red Hat Satellite Documentation Team
satellite-doc-list@redhat.com

法律通告

摘要

Red Hat Satellite Representational State Transfer (REST) API 指南解释了 REST API 背后的概念，并为各种类型请求提供示例用法。这为管理员和开发人员提供了编写自定义脚本并将 Red Hat Satellite 与第三方应用程序集成的基础。

目录

对红帽文档提供反馈

我们感谢您对我们的文档提供的信息。请让我们了解如何改进文档。

您可以通过在 Bugzilla 中记录一个 ticket 来提交反馈：

1. 导航到 Bugzilla 网站。
2. 在 **Component** 字段中，使用 **Documentation**。
3. 在 **Description** 字段中，输入您要改进的建议。包括文档相关部分的链接。
4. 点 **Submit Bug**。

第 1 章 简介

Red Hat Satellite 提供了一个 Representational State Transfer (REST) API。API 为软件开发人员和系统管理员提供在标准 Web 界面之外控制其 Red Hat Satellite 环境的权限。REST API 适用于旨在将 Red Hat Satellite 的功能与通过 HTTP 访问 API 的自定义脚本或外部应用程序集成的开发人员和管理员。

1.1. RED HAT SATELLITE API 概述

使用 REST API 的优点包括：

- 广泛的客户端支持 - 任何支持 HTTP 协议的编程语言、框架或系统都可以使用 API。
- 自我指定 - 客户端应用程序需要最少了解 Red Hat Satellite 基础架构，因为用户在运行时发现很多详情。
- 基于资源的模式 - 基于资源的 REST 模型提供了一种管理虚拟化平台的自然方法。

您可以使用 REST API 执行以下任务：

- 与企业 IT 系统集成。
- 与第三方应用程序集成。
- 执行自动维护或错误检查任务。
- 使用脚本自动执行重复性任务。

在准备升级 Satellite 服务器时，请确保使用包含 Satellite API 命令的任何脚本都为最新版本。API 命令因 Satellite 版本而异。

1.2. 与 HAMMER CLI 工具相比，SATELLITE API

对于许多任务，您可以使用 Hammer 和 Satellite API。您可以使用 Hammer 作为 Satellite API 的人类可读接口。例如，要在在脚本中应用 API 调用前测试对 API 调用的响应，请使用 **--debug** 选项检查 Hammer 问题：**hammer --debug 组织列表**.

在后台，每个 Hammer 命令首先建立与 API 的绑定，然后发送请求。在按顺序执行大量 Hammer 命令时，这可能会具有性能影响。相反，使用 API 命令的脚本直接与 Satellite API 通信。

请注意，您必须手动更新使用 API 命令的脚本，而 Hammer 会自动反映 API 中的更改。如需更多信息，请参阅 Hammer CLI 指南。

第 2 章 API 参考

您的 Satellite 服务器上位于 **https://*satellite.example.com*/apidoc/v2.html** 的完整 API 引用。请注意，虽然 Satellite 6 API 版本 1 和 2 可用，但红帽只支持版本 2。

2.1. 了解 API 语法

内置的 API 引用显示 API 路由或路径，前面带有 HTTP 动词：

```
HTTP_VERB API_ROUTE
```

要使用 API，请使用 **curl** 命令语法和参考文档中的 API 路由来构建命令：

```
$ curl --request HTTP_VERB \                    (1)
--insecure \                                    (2)
--user sat_username:sat_password \              (3)
--data @file.json \                             (4)
--header "Accept:application/json" \            (5)
--header "Content-Type:application/json" \      (6)
--output file                                   (7)
API_ROUTE \                                     (8)
| python -m json.tool                           (9)
```

1. 要将 **curl** 用于 API 调用，请使用 **--request** 选项指定 HTTP 动词。例如，`--request POST`。
2. 添加 **--insecure** 选项以跳过 SSL peer 证书验证检查。
3. 使用 **--user** 选项为用户提供凭据。
4. 对于 **POST** 和 **PUT** 请求，请使用 **--data** 选项传递 JSON 格式的数据。更多信息请参阅 第 4.1.1 节 “将 JSON 数据传递给 API 请求”。
5. 6. 使用 **--data** 选项传递 JSON 数据时，您必须使用 **--header** 选项指定以下标头。更多信息请参阅 第 4.1.1 节 “将 JSON 数据传递给 API 请求”。
7. 从 Satellite 服务器下载内容时，请使用 **--output** 选项指定输出文件。
8. 使用以下格式的 API 路由：**https://*satellite.example.com*/katello/api/activation_keys**。在 Satellite 6 中，API 的版本 2 是默认的。因此，不需要在 URL 中使用 **v2** 进行 API 调用。
9. 将输出重定向到 Python **json.tool** 模块，以便更轻松地读取输出。

2.1.1. 使用 GET HTTP 动词

使用 GET HTTP 动词从 API 获取有关现有条目或资源的数据。

示例

本例请求 Satellite 主机数量：

请求示例：

```
$ curl --request GET --insecure --user sat_username:sat_password \
https://satellite.example.com/api/hosts | python -m json.tool
```

响应示例：

```
{
  "total": 2,
  "subtotal": 2,
  "page": 1,
  "per_page": 20,
  "search": null,
  "sort": {
    "by": null,
    "order": null
  },
  "results":
output truncated
```

API 的响应表示结果总计有两个，这是结果的第一个页面，每个页面的最大结果设置为 20。如需更多信息，请参阅 第 2.2 节 “了解 JSON 响应格式”。

2.1.2. 使用 POST HTTP 动词

使用 POST HTTP 动词将数据提交到 API，以创建条目或资源。您必须使用 JSON 格式提交数据。更多信息请参阅 第 4.1.1 节 “将 JSON 数据传递给 API 请求”。

示例

这个示例创建了一个激活码。

1. 创建包含以下内容的测试文件，如 **activation-key.json** ：

   ```
   {"organization_id":1, "name":"TestKey", "description":"Just for testing"}
   ```

2. 通过在 **activation-key.json** 文件中应用数据来创建激活码：
   请求示例：

   ```
   $ curl --header "Accept:application/json" \
   --header "Content-Type:application/json" --request POST \
   --user sat_username:sat_password --insecure \
   --data @activation-key.json \
   https://satellite.example.com/katello/api/activation_keys \
   | python -m json.tool
   ```

   响应示例：

   ```
   {
     "id": 2,
     "name": "TestKey",
     "description": "Just for testing",
     "unlimited_hosts": true,
     "auto_attach": true,
     "content_view_id": null,
     "environment_id": null,
   ```

```
    "usage_count": 0,
    "user_id": 3,
    "max_hosts": null,
    "release_version": null,
    "service_level": null,
    "content_overrides": [

    ],
    "organization": {
        "name": "Default Organization",
        "label": "Default_Organization",
        "id": 1
    },
    "created_at": "2017-02-16 12:37:47 UTC",
    "updated_at": "2017-02-16 12:37:48 UTC",
    "content_view": null,
    "environment": null,
    "products": null,
    "host_collections": [

    ],
    "permissions": {
        "view_activation_keys": true,
        "edit_activation_keys": true,
        "destroy_activation_keys": true
    }
}
```


3. 验证是否存在新的激活码。在 Satellite Web UI 中，进入到 **Content** > **Activation keys** 以查看您的激活码。

2.1.3. 使用 PUT HTTP 动词

使用 PUT HTTP 动词更改现有值或附加到现有资源。您必须使用 JSON 格式提交数据。更多信息请参阅 第 4.1.1 节 “将 JSON 数据传递给 API 请求”。

示例

本例更新上例中创建的 **TestKey** 激活码。

1. 编辑之前创建的 **activation-key.json** 文件，如下所示：

```
{"organization_id":1, "name":"TestKey", "description":"Just for testing","max_hosts":"10" }
```

2. 应用 JSON 文件中的更改：
请求示例：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" --request PUT \
--user sat_username:sat_password --insecure \
--data @activation-key.json \
https://satellite.example.com/katello/api/activation_keys/2 \
| python -m json.tool
```

响应示例：

```
{
    "id": 2,
    "name": "TestKey",
    "description": "Just for testing",
    "unlimited_hosts": false,
    "auto_attach": true,
    "content_view_id": null,
    "environment_id": null,
    "usage_count": 0,
    "user_id": 3,
    "max_hosts": 10,
    "release_version": null,
    "service_level": null,
    "content_overrides": [

    ],
    "organization": {
        "name": "Default Organization",
        "label": "Default_Organization",
        "id": 1
    },
    "created_at": "2017-02-16 12:37:47 UTC",
    "updated_at": "2017-02-16 12:46:17 UTC",
    "content_view": null,
    "environment": null,
    "products": null,
    "host_collections": [

    ],
    "permissions": {
        "view_activation_keys": true,
        "edit_activation_keys": true,
        "destroy_activation_keys": true
    }
}
```


3. 在 Satellite Web UI 中，导航到 **Content** > **Activation keys** 来验证更改。

2.1.4. 使用 DELETE HTTP 动词

要删除资源，请使用 DELETE 动词以及包含您要删除的资源 ID 的 API 路由。

示例

这个示例删除 ID 为 2 的 **TestKey** 激活码：

请求示例：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" --request DELETE \
--user sat_username:sat_password --insecure \
https://satellite.example.com/katello/api/activation_keys/2 \
| python -m json.tool
```

响应示例：

```
output omitted
    "started_at": "2017-02-16 12:58:17 UTC",
    "ended_at": "2017-02-16 12:58:18 UTC",
    "state": "stopped",
    "result": "success",
    "progress": 1.0,
    "input": {
        "activation_key": {
            "id": 2,
            "name": "TestKey"
output truncated
```

2.1.5. 将 API 错误消息与 API 引用相关联

API 使用 RAILs 格式来指示错误：

```
Nested_Resource.Attribute_Name
```

这会转换为 API 引用中使用的以下格式：

```
Resource[Nested_Resource_attributes][Attribute_Name_id]
```

2.2. 了解 JSON 响应格式

对 API 的调用会以 JSON 格式返回结果。API 调用返回单个选项响应或响应集合的结果。

单个对象的 JSON 响应格式

您可以使用单对象 JSON 响应来处理单个对象。对单个对象的 API 请求需要对象的唯一标识符 **:id**。

这是 ID 为 23 的单对象请求的单对象请求的格式示例：

请求示例：

```
$ curl --request GET --insecure --user sat_username:sat_password \
https://satellite.example.com/api/domains/23 | python -m json.tool
```

响应示例：

```
{
    "id": 23,
    "name": "qa.lab.example.com",
    "fullname": "QA",
    "dns_id": 10,
    "created_at": "2013-08-13T09:02:31Z",
    "updated_at": "2013-08-13T09:02:31Z"
}
```

集合的 JSON 响应格式

集合是主机和域等对象列表。集合 JSON 响应的格式由 metadata 字段和 results 部分组成。

这是 Satellite 域列表集合请求的格式示例：

请求示例：

```
$ curl --request GET --insecure --user sat_username:sat_password \
https://satellite.example.com/api/domains | python -m json.tool
```

响应示例：

```
{
    "total": 3,
    "subtotal": 3,
    "page": 1,
    "per_page": 20,
    "search": null,
    "sort": {
        "by": null,
        "order": null
    },
    "results": [
        {
            "id": 23,
            "name": "qa.lab.example.com",
            "fullname": "QA",
            "dns_id": 10,
            "created_at": "2013-08-13T09:02:31Z",
            "updated_at": "2013-08-13T09:02:31Z"
        },
        {
            "id": 25,
            "name": "sat.lab.example.com",
            "fullname": "SATLAB",
            "dns_id": 8,
            "created_at": "2013-08-13T08:32:48Z",
            "updated_at": "2013-08-14T07:04:03Z"
        },
        {
            "id": 32,
            "name": "hr.lab.example.com",
            "fullname": "HR",
            "dns_id": 8,
            "created_at": "2013-08-16T08:32:48Z",
            "updated_at": "2013-08-16T07:04:03Z"
        }
    ]
}
```

响应元数据字段

API 响应使用以下 metadata 字段：

- **total** - 没有搜索参数的对象总数。
- **Subtotal** - 使用给定搜索参数返回的对象数量。如果没有搜索，则 subtotal 等于 total。
- **页面** - 页面编号。
- **per_page** - 每个页面返回的最大对象数。

- **limit** - 在集合响应中返回的指定数量的对象。
- **offset** - 返回集合前跳过的对象数量。
- **search** - 基于 **scoped_scoped** 语法的搜索字符串。
- **sort**
  - **by** - 根据 API 对集合进行排序的字段指定。
  - **order** - 排序顺序，ASC 代表升序或 DESC 用于降序。
- **results** - 对象的集合。

第 3 章 验证 API 调用

与 Satellite API 交互需要与 Satellite 服务器 CA 证书进行 SSL 身份验证，并使用有效的 Satellite 用户凭据进行身份验证。本章概述了您可以使用的身份验证方法。

3.1. SSL 验证概述

Red Hat Satellite 使用 HTTPS，在与 Red Hat Satellite Server 通信时提供一定程度的加密和身份验证。Satellite {ProductVersion} 不支持非 SSL 通信。

每个 Red Hat Satellite 服务器都使用自签名证书。此证书同时充当服务器证书，以验证加密密钥和证书颁发机构(CA)来信任 Satellite 服务器的身份。

3.1.1. 配置 SSL 身份验证

使用以下步骤为 API 请求配置到 Satellite 服务器的 SSL 身份验证。

流程

1. 使用以下选项之一从 Satellite 服务器获取证书：
   - 如果您从远程服务器执行命令，请使用 SSH 获取证书：

     ```
     $ scp root@satellite.example.com:/var/www/html/pub/katello-server-ca.crt /etc/pki/ca-trust/source/anchors/satellite.example.com-katello-server-ca.crt
     ```

   - 如果您直接在 Satellite 服务器上执行命令，请将证书复制到 **/etc/pki/ca-trust/source/anchors** 目录中：

     ```
     $ cp /var/www/html/pub/katello-server-ca.crt /etc/pki/ca-trust/source/anchors/satellite.example.com-katello-server-ca.crt
     ```

2. 将证书添加到可信 CA 列表中：

   ```
   update-ca-trust extract
   ```

验证

- 在没有 **--cacert** 选项的情况下，输入 API 请求来验证 NSS 数据库中是否存在证书：

  ```
  $ curl --request GET \
  --user sat_username:sat_password \
  https://satellite.example.com/api/v2/hosts
  ```

3.2. HTTP 验证概述

对 Satellite API 的所有请求都需要有效的 Satellite 用户名和密码。API 使用 HTTP 基本身份验证来对这些凭证进行编码，并添加到 **Authorization** 标头中。有关基本身份验证的更多信息，请参阅 RFC 2617 HTTP 身份验证： Basic 和 Digest 访问身份验证。如果请求没有包括适当的 **Authorization** 标头，API 会返回 **401 Authorization Required** 错误

重要

基本身份验证涉及潜在的敏感信息，例如，它会以纯文本形式发送密码。REST API 需要 HTTPS 用于纯文本请求的传输级别加密。

有些 base64 库将编码的凭证分为多行，并使用换行符终止每行。这会使标头无效，并导致错误请求。Authorization 标头要求编码的凭据位于标头中的一行中。

3.3. 个人访问令牌身份验证概述

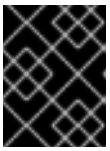
Red Hat Satellite 支持可用于验证 API 请求的个人访问令牌，而不使用您的密码。您可以为个人访问令牌设置过期日期。如果您决定它应在过期日期前过期，可以撤销它。

3.3.1. 创建一个个人访问令牌

使用这个流程创建个人访问令牌。

流程

1. 在 Satellite Web UI 中，进入到 **Administer** > **Users**。
2. 选择您要为其创建个人访问令牌的用户。
3. 在 **Personal Access Tokens** 选项卡中，点 **Add Personal Access Token**。
4. 输入个人访问令牌的**名称**。
5. 可选：选择 **Expires** 日期来设置过期日期。如果您没有设置过期日期，您的个人访问令牌将永远不会过期，除非被撤销。
6. 点 Submit。到此，在 **Personal Access Tokens** 选项卡会包括您可用的个人访问令牌。



重要

确保存储您的个人访问令牌，因为您将无法在离开页面或创建新的个人访问令牌后再次访问它。您可以点 **Copy to clipboard** 复制个人访问令牌。

验证

1. 向 Satellite 服务器发出 API 请求，并使用您的个人访问令牌进行身份验证：

```
# curl https://satellite.example.com/api/status --user
My_Username:My_Personal_Access_Token
```

2. 您应该收到状态为 **200** 的响应，例如：

```
{"satellite_version":"6.15.0","result":"ok","status":200,"version":"3.5.1.10","api_version":2}
```

如果您再次返回 **Personal Access Tokens** 选项卡，您可以在个人访问令牌旁看到**最后使用**时间。

3.3.2. 撤销个人访问令牌

使用这个流程在过期日期前撤销个人访问令牌。

流程

1. 在 Satellite Web UI 中，进入到 **Administer** > **Users**。
2. 选择您要撤销个人访问令牌的用户。
3. 在 **Personal Access Tokens** 选项卡中，找到您要撤销的个人访问令牌。
4. 在您要撤销的个人访问令牌旁边的 **Actions** 列中点 **Revoke**。

验证

1. 向 Satellite 服务器发出 API 请求，并尝试使用撤销的个人访问令牌进行身份验证：

```
# curl https://satellite.example.com/api/status --user
My_Username:My_Personal_Access_Token
```

2. 您会收到以下出错信息：

```
{
  "error": {"message":"Unable to authenticate user My_Username"}
}
```

3.4. OAUTH 身份验证概述

作为基本身份验证的替代选择，您可以使用有限的 OAuth 1.0 身份验证。这有时在协议的版本 1.0a 中被称为 1 委派的 OAuth。

要在 Satellite Web UI 中查看 OAuth 设置，请导航到 **Administer** > **Settings** > **Authentication**。**OAuth 使用者密钥是** 所有 OAuth 客户端要使用的令牌。

Satellite 将 OAuth 设置存储在 **/etc/foreman/settings.yaml** 文件中。使用 **satellite-installer** 脚本配置这些设置，因为 Satellite 在升级时会覆盖对此文件的任何手动更改。

3.4.1. 配置 OAuth

要更改 OAuth 设置，请使用所需选项输入 **satellite-installer**。输入以下命令列出所有 OAuth 相关安装程序选项：

```
# satellite-installer --full-help | grep oauth
```

启用 OAuth 映射

默认情况下，Satellite 会授权所有 OAuth API 请求作为内置的匿名 API 管理员帐户。因此，API 响应包含所有 Satellite 数据。但是，您也可以指定 Foreman 用户，以发出请求并限制对该用户访问的数据。

要启用 OAuth 用户映射，请输入以下命令：

```
# satellite-installer --foreman-oauth-map-users true
```

重要

Satellite 不签署 OAuth 请求中的标头。具有有效使用者密钥的任何人都可以模拟任何 Foreman 用户。

3.4.2. OAuth 请求格式

每个 OAuth API 请求都需要具有现有 Foreman 用户和 **Authorization** 标头的 **FOREMAN-USER** 标头，格式为：

```
--header 'FOREMAN-USER: sat_username' \
--header 'Authorization: OAuth
oauth_version="1.0",oauth_consumer_key="secretkey",oauth_signature_method="hmac-
sha1",oauth_timestamp=timestamp,oauth_signature=signature'
```

重要

使用 OAuth 客户端库来构建所有 OAuth 参数。有关使用 **requests_oauthlib** Python 模块的示例，请参阅红帽知识库中的 如何通过 Red Hat Satellite 6 中的 python 脚本使用 OAuth 身份验证方法执行 API 调用。

Example

本例列出了使用 OAuth 进行身份验证的架构。该请求在 **FOREMAN-USER** 标头中使用 *sat_username* 用户名。将 **--foreman-oauth-map-users** 设置为 **true** 时，响应仅包含用户有权访问查看的架构。签名反映了每个参数、HTTP 方法和 URI 更改。

请求示例：

```
$ curl 'https://satellite.example.com/api/architectures' \
--header 'Content-Type: application/json' \
--header 'Accept:application/json' \
--header 'FOREMAN-USER: sat_username' \
--header 'Authorization: OAuth
oauth_version="1.0",oauth_consumer_key="secretkey",oauth_signature_method="hmac-
sha1",oauth_timestamp=1321473112,oauth_signature=Il8hR8/ogj/XVuOqMPB9qNjSy6E='
```

其他资源

- OAuth Core 1.0 修订 A 的文档

第 4 章 使用不同语言的 API 请求

本章概述了使用 curl、Ruby 和 Python 向 Red Hat Satellite 发送 API 请求，并提供了示例。

4.1. 使用 CURL 的 API 请求

本节概述了如何将 **curl** 与 Satellite API 搭配使用来执行各种任务。

Red Hat Satellite 需要使用 HTTPS，默认是主机识别的证书。如果您还没有添加 Satellite 服务器证书，如 第 3.1 节 “SSL 验证概述” 所述，您可以使用 **--insecure** 选项绕过证书检查。

对于用户身份验证，您可以使用 **--user** 选项以 **--user *username:password*** 格式提供 Satellite 用户凭据，或者如果您不包含密码，则命令会提示您输入它。要降低安全风险，请不要将密码包含在命令中，因为它会成为 shell 历史记录的一部分。本节中的示例仅包含简单性的密码。

请注意，如果您使用 **--silent** 选项，**curl** 不会显示进度量表或任何错误消息。

本章中的示例使用 Python **json.tool** 模块来格式化输出。

4.1.1. 将 JSON 数据传递给 API 请求

您可以使用 API 请求将数据传递给 Satellite 服务器。数据必须采用 JSON 格式。使用 **--data** 选项指定 JSON 数据时，您必须使用 **--header** 选项设置以下 HTTP 标头：

```
--header "Accept:application/json" \
--header "Content-Type:application/json"
```

使用以下选项之一包含带有 **--data** 选项的数据：

1. 以大括号 **{}** 括起的带引号的 JSON 格式数据。为 JSON 类型参数传递值时，您必须转义引号 **"**，带有反斜杠 \。例如，在大括号中，您必须将 **"示例 JSON 变量"** 格式化为 **\"Example JSON Variable\"** ：

   ```
   --data {"id":44, "smart_class_parameter":{"override":"true", "parameter_type":"json",
   "default_value":"{\"GRUB_CMDLINE_LINUX\": {\"audit\":\"1\",\"crashkernel\":\"true\"}}"}}
   ```

2. 不加引号的 JSON 格式数据包含在文件中，并由 @ 符号和文件名指定。例如：

   ```
   --data @file.json
   ```

   将外部文件用于 JSON 格式的数据有以下优点：

   - 您可以使用您首选的文本编辑器。
   - 您可以使用语法检查程序来查找和避免错误。
   - 您可以使用工具来检查 JSON 数据的有效性或重新格式化。

验证 JSON 文件

使用 **json_verify** 工具检查 JSON 文件的有效性：

```
$ json_verify < test_file.json
```

4.1.2. 检索资源列表

本节概述了如何将 **curl** 与 Satellite 6 API 搭配使用，以从 Satellite 部署请求信息。这些示例包括请求和响应。预计每个部署的结果都不同。

列出用户

本例是返回 Satellite 资源列表的基本请求，本例中为 Satellite 用户。此类请求返回元数据中嵌套的数据列表，其他请求类型仅返回实际对象。

请求示例：

```
$ curl --request GET --insecure --user sat_username:sat_password \
https://satellite.example.com/api/users | python -m json.tool
```

响应示例：

```
{
    "page": 1,
    "per_page": 20,
    "results": [
        {
            "admin": false,
            "auth_source_id": 1,
            "auth_source_name": "Internal",
            "created_at": "2018-09-21 08:59:22 UTC",
            "default_location": null,
            "default_organization": null,
            "description": "",
            "effective_admin": false,
            "firstname": "",
            "id": 5,
            "last_login_on": "2018-09-21 09:03:25 UTC",
            "lastname": "",
            "locale": null,
            "locations": [],
            "login": "test",
            "mail": "example@domain.com",
            "organizations": [
                {
                    "id": 1,
                    "name": "Default Organization"
                }
            ],
            "ssh_keys": [],
            "timezone": null,
            "updated_at": "2018-09-21 09:04:45 UTC"
        },
        {
            "admin": true,
            "auth_source_id": 1,
            "auth_source_name": "Internal",
            "created_at": "2018-09-20 07:09:41 UTC",
            "default_location": null,
            "default_organization": {
                "description": null,
```

```
            "id": 1,
            "name": "Default Organization",
            "title": "Default Organization"
        },
        "description": "",
        "effective_admin": true,
        "firstname": "Admin",
        "id": 4,
        "last_login_on": "2018-12-07 07:31:09 UTC",
        "lastname": "User",
        "locale": null,
        "locations": [
            {
                "id": 2,
                "name": "Default Location"
            }
        ],
        "login": "admin",
        "mail": "root@example.com",
        "organizations": [
            {
                "id": 1,
                "name": "Default Organization"
            }
        ],
        "ssh_keys": [],
        "timezone": null,
        "updated_at": "2018-11-14 08:19:46 UTC"
    }
  ],
  "search": null,
  "sort": {
    "by": null,
    "order": null
  },
  "subtotal": 2,
  "total": 2
}
```


4.1.3. 创建和修改资源

本节概述了如何将 **curl** 与 Satellite 6 API 搭配使用，以操作 Satellite 服务器上的资源。这些 API 调用需要您使用 **json** 格式通过 API 调用来传递数据。如需更多信息，请参阅 第 4.1.1 节 "将 JSON 数据传递给 API 请求"。

创建用户

这个示例使用 **--data** 选项创建一个用户来提供所需信息。

请求示例：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" --request POST \
--user sat_username:sat_password --insecure \
--data "{\"firstname\":\"Test
```

```
Name\",\"mail\":\"test@example.com\",\"login\":\"test_user\",\"password\":\"password123\",\"auth_sour
ce_id\":1}" \
https://satellite.example.com/api/users | python -m json.tool
```

修改用户

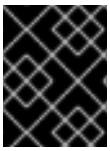
这个示例修改 创建用户 中创建的 **test_user** 的名字和登录。

请求示例：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" --request PUT \
--user sat_username:sat_password --insecure \
--data "{\"firstname\":\"New Test
Name\",\"mail\":\"test@example.com\",\"login\":\"new_test_user\",\"password\":\"password123\",\"auth
_source_id\":1}" \
https://satellite.example.com/api/users/8 | python -m json.tool
```

4.2. 使用 RUBY 的 API 请求

本节概述了如何将 Ruby 与 Satellite API 搭配使用来执行各种任务。



重要

以下是脚本和命令示例。在使用前，请确保仔细检查这些脚本，并替换任何变量、用户名、密码和其他信息以适应您自己的部署。

4.2.1. 使用 Ruby 创建对象

此脚本连接到 Red Hat Satellite 6 API 并创建一个机构，然后在组织中创建三个环境。如果机构已存在，脚本将使用该组织。如果机构中已存在任何环境，该脚本会引发错误并退出。

```
#!/usr/bin/ruby

require 'rest-client'
require 'json'

url = 'https://satellite.example.com/api/v2/'
katello_url = "#{url}/katello/api/v2/"

$username = 'admin'
$password = 'changeme'

org_name = "MyOrg"
environments = [ "Development", "Testing", "Production" ]

# Performs a GET using the passed URL location
def get_json(location)
  response = RestClient::Request.new(
    :method => :get,
    :url => location,
    :user => $username,
    :password => $password,
    :headers => { :accept => :json,
```

```
    :content_type => :json }
  ).execute
  JSON.parse(response.to_str)
end

# Performs a POST and passes the data to the URL location
def post_json(location, json_data)
  response = RestClient::Request.new(
    :method => :post,
    :url => location,
    :user => $username,
    :password => $password,
    :headers => { :accept => :json,
    :content_type => :json},
    :payload => json_data
  ).execute
  JSON.parse(response.to_str)
end

# Creates a hash with ids mapping to names for an array of records
def id_name_map(records)
  records.inject({}) do |map, record|
    map.update(record['id'] => record['name'])
  end
end

# Get list of existing organizations
orgs = get_json("#{katello_url}/organizations")
org_list = id_name_map(orgs['results'])

if !org_list.has_value?(org_name)
  # If our organization is not found, create it
  puts "Creating organization: \t#{org_name}"
  org_id = post_json("#{katello_url}/organizations", JSON.generate({"name"=> org_name}))["id"]
else
  # Our organization exists, so let's grab it
  org_id = org_list.key(org_name)
  puts "Organization \"#{org_name}\" exists"
end

# Get list of organization's lifecycle environments
envs = get_json("#{katello_url}/organizations/#{org_id}/environments")
env_list = id_name_map(envs['results'])
prior_env_id = env_list.key("Library")

# Exit the script if at least one life cycle environment already exists
environments.each do |e|
  if env_list.has_value?(e)
    puts "ERROR: One of the Environments is not unique to organization"
    exit
  end
end

 # Create life cycle environments
environments.each do |environment|
  puts "Creating environment: \t#{environment}"
```

```
  prior_env_id = post_json("#{katello_url}/organizations/#{org_id}/environments",
JSON.generate({"name" => environment, "organization_id" => org_id, "prior_id" => prior_env_id}))
["id"]
end
```

4.2.2. 在 Ruby 中使用 apipie 绑定

apipie 绑定是 apipie 记录的 API 调用的 Ruby 绑定。它们从 Satellite 获取并缓存 API 定义，然后按需生成 API 调用。这个示例创建了一个机构，然后在机构中创建三个环境。如果机构已存在，脚本将使用该组织。如果机构中已存在任何环境，该脚本会引发错误并退出。

```
#!/usr/bin/tfm-ruby

require 'apipie-bindings'

org_name = "MyOrg"
environments = [ "Development", "Testing", "Production" ]

# Create an instance of apipie bindings
@api = ApipieBindings::API.new({
  :uri => 'https://satellite.example.com/',
  :username => 'admin',
  :password => 'changeme',
  :api_version => 2
})

# Performs an API call with default options
def call_api(resource_name, action_name, params = {})
  http_headers = {}
  apipie_options = { :skip_validation => true }
  @api.resource(resource_name).call(action_name, params, http_headers, apipie_options)
end

# Creates a hash with IDs mapping to names for an array of records
def id_name_map(records)
  records.inject({}) do |map, record|
    map.update(record['id'] => record['name'])
  end
end

# Get list of existing organizations
orgs = call_api(:organizations, :index)
org_list = id_name_map(orgs['results'])

if !org_list.has_value?(org_name)
  # If our organization is not found, create it
  puts "Creating organization: \t#{org_name}"
  org_id = call_api(:organizations, :create, {'organization' => { :name => org_name }})['id']
else
  # Our organization exists, so let's grab it
  org_id = org_list.key(org_name)
  puts "Organization \"#{org_name}\" exists"
end

# Get list of organization's life cycle environments
```

```
envs = call_api(:lifecycle_environments, :index, {'organization_id' => org_id})
env_list = id_name_map(envs['results'])
prior_env_id = env_list.key("Library")

# Exit the script if at least one life cycle environment already exists
environments.each do |e|
  if env_list.has_value?(e)
    puts "ERROR: One of the Environments is not unique to organization"
    exit
  end
end

 # Create life cycle environments
environments.each do |environment|
  puts "Creating environment: \t#{environment}"
  prior_env_id = call_api(:lifecycle_environments, :create, {"name" => environment, "organization_id"
=> org_id, "prior_id" => prior_env_id })['id']
end
```

4.3. 使用 PYTHON 的 API 请求

本节概述了如何将 Python 与 Satellite API 搭配使用来执行各种任务。

重要

以下是脚本和命令示例。在使用前，请确保仔细检查这些脚本，并替换任何变量、用户名、密码和其他信息以适应您自己的部署。

本节中的脚本示例不使用 SSL 验证与 REST API 交互。

4.3.1. 使用 Python 创建对象

此脚本连接到 Red Hat Satellite 6 API 并创建一个机构，然后在组织中创建三个环境。如果机构已存在，脚本将使用该组织。如果机构中已存在任何环境，该脚本会引发错误并退出。

Python 2 示例

```
#!/usr/bin/python

import json
import sys

try:
    import requests
except ImportError:
    print "Please install the python-requests module."
    sys.exit(-1)

# URL to your Satellite 6 server
URL = "https://satellite.example.com"
# URL for the API to your deployed Satellite 6 server
SAT_API = "%s/katello/api/v2/" % URL
# Katello-specific API
KATELLO_API = "%s/katello/api/" % URL
```

```
POST_HEADERS = {'content-type': 'application/json'}
# Default credentials to login to Satellite 6
USERNAME = "admin"
PASSWORD = "changeme"
# Ignore SSL for now
SSL_VERIFY = False

# Name of the organization to be either created or used
ORG_NAME = "MyOrg"
# Name for life cycle environments to be either created or used
ENVIRONMENTS = ["Development", "Testing", "Production"]


def get_json(location):
    """
    Performs a GET using the passed URL location
    """

    r = requests.get(location, auth=(USERNAME, PASSWORD), verify=SSL_VERIFY)

    return r.json()


def post_json(location, json_data):
    """
    Performs a POST and passes the data to the URL location
    """

    result = requests.post(
        location,
        data=json_data,
        auth=(USERNAME, PASSWORD),
        verify=SSL_VERIFY,
        headers=POST_HEADERS)

    return result.json()


def main():
    """
    Main routine that creates or re-uses an organization and
    life cycle environments. If life cycle environments already
    exist, exit out.
    """

    # Check if our organization already exists
    org = get_json(SAT_API + "organizations/" + ORG_NAME)

    # If our organization is not found, create it
    if org.get('error', None):
        org_id = post_json(
            SAT_API + "organizations/",
            json.dumps({"name": ORG_NAME}))["id"]
        print "Creating organization: \t" + ORG_NAME
    else:
        # Our organization exists, so let's grab it
```

```
        org_id = org['id']
        print "Organization '%s' exists." % ORG_NAME

    # Now, let's fetch all available life cycle environments for this org...
    envs = get_json(
        SAT_API + "organizations/" + str(org_id) + "/environments/")

    # ... and add them to a dictionary, with respective 'Prior' environment
    prior_env_id = 0
    env_list = {}
    for env in envs['results']:
        env_list[env['id']] = env['name']
        prior_env_id = env['id'] if env['name'] == "Library" else prior_env_id

    # Exit the script if at least one life cycle environment already exists
    if all(environment in env_list.values() for environment in ENVIRONMENTS):
        print "ERROR: One of the Environments is not unique to organization"
        sys.exit(-1)

    # Create life cycle environments
    for environment in ENVIRONMENTS:
        new_env_id = post_json(
            SAT_API + "organizations/" + str(org_id) + "/environments/",
            json.dumps(
                {
                    "name": environment,
                    "organization_id": org_id,
                    "prior": prior_env_id}
            ))["id"]

        print "Creating environment: \t" + environment
        prior_env_id = new_env_id


if __name__ == "__main__":
    main()
```

4.3.2. 使用 Python 从 API 请求信息

这是一个示例脚本，它使用 Python 进行各种 API 请求。

Python 2 示例

```
#!/usr/bin/python
import json
import sys
try:
    import requests
except ImportError:
    print "Please install the python-requests module."
    sys.exit(-1)

SAT_API = 'https://satellite.example.com/api/v2/'
USERNAME = "admin"
PASSWORD = "password"
```

```
SSL_VERIFY = False   # Ignore SSL for now

def get_json(url):
    # Performs a GET using the passed URL location
    r = requests.get(url, auth=(USERNAME, PASSWORD), verify=SSL_VERIFY)
    return r.json()

def get_results(url):
    jsn = get_json(url)
    if jsn.get('error'):
        print "Error: " + jsn['error']['message']
    else:
        if jsn.get('results'):
            return jsn['results']
        elif 'results' not in jsn:
            return jsn
        else:
            print "No results found"
    return None

def display_all_results(url):
    results = get_results(url)
    if results:
        print json.dumps(results, indent=4, sort_keys=True)

def display_info_for_hosts(url):
    hosts = get_results(url)
    if hosts:
        for host in hosts:
            print "ID: %-10d Name: %-30s IP: %-20s OS: %-30s" % (host['id'], host['name'], host['ip'],
host['operatingsystem_name'])

def main():
    host = 'satellite.example.com'
    print "Displaying all info for host %s ..." % host
    display_all_results(SAT_API + 'hosts/' + host)

    print "Displaying all facts for host %s ..." % host
    display_all_results(SAT_API + 'hosts/%s/facts' % host)

    host_pattern = 'example'
    print "Displaying basic info for hosts matching pattern '%s'..." % host_pattern
    display_info_for_hosts(SAT_API + 'hosts?search=' + host_pattern)

    environment = 'production'
    print "Displaying basic info for hosts in environment %s..." % environment
    display_info_for_hosts(SAT_API + 'hosts?search=environment=' + environment)

    model = 'RHEV Hypervisor'
    print "Displaying basic info for hosts with model name %s..." % model
    display_info_for_hosts(SAT_API + 'hosts?search=model="' + model + '"')

if __name__ == "__main__":
    main()
```

Python 3 示例

```
#!/usr/bin/env python3

import json
import sys

try:
    import requests
except ImportError:
    print("Please install the python-requests module.")
    sys.exit(-1)

SAT = "satellite.example.com"
# URL for the API to your deployed Satellite 6 server
SAT_API = f"https://{SAT}/api/"
KATELLO_API = f"https://{SAT}/katello/api/v2/"

POST_HEADERS = {'content-type': 'application/json'}
# Default credentials to login to Satellite 6
USERNAME = "admin"
PASSWORD = "password"
# Ignore SSL for now
SSL_VERIFY = False
#SSL_VERIFY = "./path/to/CA-certificate.crt" # Put the path to your CA certificate here to allow
SSL_VERIFY


def get_json(url):
    # Performs a GET using the passed URL location
    r = requests.get(url, auth=(USERNAME, PASSWORD), verify=SSL_VERIFY)
    return r.json()

def get_results(url):
    jsn = get_json(url)
    if jsn.get('error'):
        print("Error: " + jsn['error']['message'])
    else:
        if jsn.get('results'):
            return jsn['results']
        elif 'results' not in jsn:
            return jsn
        else:
            print("No results found")
    return None

def display_all_results(url):
    results = get_results(url)
    if results:
        print(json.dumps(results, indent=4, sort_keys=True))

def display_info_for_hosts(url):
    hosts = get_results(url)
    if hosts:
        print(f"{'ID':10}{'Name':40}{'IP':30}{'Operating System':30}")
        for host in hosts:
            print(f"{str(host['id']):10}{host['name']:40}{str(host['ip']):30}
{str(host['operatingsystem_name']):30}")
```

```
def display_info_for_subs(url):
    subs = get_results(url)
    if subs:
        print(f"{'ID':10}{'Name':90}{'Start Date':30}")
        for sub in subs:
            print(f"{str(sub['id']):10}{sub['name']:90}{str(sub['start_date']):30}")

def main():
    host = SAT
    print(f"Displaying all info for host {host} ...")
    display_all_results(SAT_API + 'hosts/' + host)

    print(f"Displaying all facts for host {host} ...")
    display_all_results(SAT_API + f'hosts/{host}/facts')

    host_pattern = 'example'
    print(f"Displaying basic info for hosts matching pattern '{host_pattern}'...")
    display_info_for_hosts(SAT_API + 'hosts?per_page=1&search=name~' + host_pattern)

    print(f"Displaying basic info for subscriptions")
    display_info_for_subs(KATELLO_API + 'subscriptions')

    environment = 'production'
    print(f"Displaying basic info for hosts in environment {environment}...")
    display_info_for_hosts(SAT_API + 'hosts?search=environment=' + environment)


if __name__ == "__main__":
    main()
```

第 5 章 使用 RED HAT SATELLITE API

本章介绍了如何使用 Red Hat Satellite API 执行不同任务的示例。您可以通过 HTTPS 在端口 443 上通过 HTTPS 使用 Satellite 服务器上的 API，或通过 HTTPS 在端口 8443 上通过 HTTPS 使用。

您可以在脚本本身中解决这些不同的端口要求。例如，在 Ruby 中，您可以指定 Satellite 和 Capsule URL，如下所示：

```
url = 'https://satellite.example.com/api/v2/'
capsule_url = 'https://capsule.example.com:8443/api/v2/'
katello_url = 'https://satellite.example.com/katello/api/v2/'
```

对于订阅了 Satellite 服务器或 Capsule 服务器的主机，您可以在 **[server]** 部分的端口条目中确定从 **/etc/rhsm/rhsm.conf** 文件访问 API 所需的正确端口。您可以使用这些值来完全自动化脚本，无需无需验证要使用的端口。

本章使用 **curl** 发送 API 请求。更多信息请参阅 第 4.1 节 “使用 curl 的 API 请求”。

本章中的示例使用 Python **json.tool** 模块来格式化输出。

5.1. 使用主机

列出主机

本例返回 Satellite 主机列表。

请求示例：

```
$ curl --request GET --insecure --user sat_username:sat_password \
https://satellite.example.com/api/v2/hosts | python3 -m json.tool
```

响应示例：

```
{
      ...
      "total" => 2,
    "subtotal" => 2,
       "page" => 1,
    "per_page" => 1000,
     "search" => nil,
       "sort" => {
         "by" => nil,
      "order" => nil
    },
     "results" => [
      ...
}
```

请求主机信息

此请求返回主机 **satellite.example.com** 的信息。

请求示例：

```
$  curl --request GET --insecure --user sat_username:sat_password \
https://satellite.example.com/api/v2/hosts/satellite.example.com \
| python -m json.tool
```

响应示例：

```
{
    "all_puppetclasses": [],
    "architecture_id": 1,
    "architecture_name": "x86_64",
    "build": false,
    "capabilities": [
        "build"
    ],
    "certname": "satellite.example.com",
    "comment": null,
    "compute_profile_id": null,
    ...
}
```

列出主机事实

此请求返回主机 **satellite.example.com** 的所有事实。

请求示例：

```
$ curl --request GET --insecure --user sat_username:sat_password \
https://satellite.example.com/api/v2/hosts/satellite.example.com/facts \
| python -m json.tool
```

响应示例：

```
{
    ...
    "results": {
        "satellite.example.com": {
            "augeasversion": "1.0.0",
            "bios_release_date": "01/01/2007",
            "bios_version": "0.5.1",
            "blockdevice_sr0_size": "1073741312",
            "facterversion": "1.7.6",
            ...
}
```

搜索具有匹配模式的主机

此查询返回与"example"模式匹配的所有主机。

请求示例：

```
$ curl --request GET --insecure --user sat_username:sat_password \
https://satellite.example.com/api/v2/hosts?search=example \
| python -m json.tool
```

响应示例：

```
{
    ...
    "results": [
        {
            "name": "satellite.example.com",
            ...
        }
    ],
    "search": "example",
    ...
}
```

搜索环境中的主机

此查询会返回**生产环境**中的所有主机。

请求示例：

```
$ curl --request GET --insecure --user sat_username:sat_password \
https://satellite.example.com/api/v2/hosts?search=environment=production \
| python -m json.tool
```

响应示例：

```
{
    ...
    "results": [
        {
            "environment_name": "production",
            "name": "satellite.example.com",
            ...
        }
    ],
    "search": "environment=production",
    ...
}
```

搜索具有特定事实值的主机

此查询返回所有主机，其模型名称为 **RHEV Hypervisor**。

请求示例：

```
$ curl --request GET --insecure --user sat_username:sat_password \
https://satellite.example.com/api/v2/hosts?search=model=\"RHEV+Hypervisor\" \
| python -m json.tool
```

响应示例：

```
{
    ...
    "results": [
        {
            "model_id": 1,
            "model_name": "RHEV Hypervisor",
```

```
            "name": "satellite.example.com",
            ...
        }
    ],
    "search": "model=\"RHEV Hypervisor\"",
    ...
}
```

删除主机

此请求将删除名为 *host1.example.com* 的主机。

请求示例：

```
$ curl --request DELETE --insecure --user sat_username:sat_password \
https://satellite.example.com/api/v2/hosts/host1.example.com \
| python -m json.tool
```

下载完整引导磁盘镜像

此请求通过其 ID 为主机下载完整引导磁盘镜像。

请求示例：

```
$ curl --request GET --insecure --user sat_username:sat_password \
https://satellite.example.com/api/bootdisk/hosts/host_ID?full=true \
--output image.iso
```

5.2. 使用生命周期环境

Satellite 将应用生命周期划分为生命周期环境，后者代表应用程序生命周期的每个阶段。生命周期环境从环境路径链接到。要使用 API 创建链接的生命周期环境，请使用 **prior_id** 参数。

您可以在 **https://*satellite.example.com*/apidoc/v2/lifecycle_environments.html** 中找到生命周期环境的内置 API 参考。API 路由包括 **/katello/api/environments** 和 **/katello/api/organizations/:organization_id/environments**。

列出生命周期环境

使用此 API 调用，为 ID 为 **1** 的默认组织列出您的 Satellite 上所有当前生命周期环境。

请求示例：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" \
--request GET --user sat_username:sat_password --insecure \
https://satellite.example.com/katello/api/organizations/1/environments \
| python -m json.tool`
```

响应示例：

```
    output omitted
  "description": null,
  "id": 1,
  "label": "Library",
```

```
"library": true,
"name": "Library",
"organization": {
     "id": 1,
     "label": "Default_Organization",
     "name": "Default Organization"
},
"permissions": {
    "destroy_lifecycle_environments": false,
    "edit_lifecycle_environments": true,
    "promote_or_remove_content_views_to_environments": true,
    "view_lifecycle_environments": true
},
"prior": null,
"successor": null,
output truncated
```

创建链接的生命周期环境

使用本示例创建生命周期环境的路径。

此流程使用 ID 为 **1** 的默认 Library 环境，作为创建生命周期环境的起点。

1. 选择您要用作起点的现有生命周期环境。使用其 ID 列出环境，本例中为 ID 为 **1** 的环境：
   请求示例：

   ```
   $ curl --request GET --user sat_username:sat_password --insecure \
   https://satellite.example.com/katello/api/environments/1 \
   | python -m json.tool
   ```

   响应示例：

   ```
   output omitted
     "id": 1,
     "label": "Library",
   output omitted
      "prior": null,
      "successor": null,
    output truncated
   ```

2. 创建包含以下内容的 JSON 文件，如 **life-cycle.json** ：

   ```
   {"organization_id":1,"label":"api-dev","name":"API Development","prior":1}
   ```

3. 将 **prior** 选项设置为 **1** 来创建生命周期环境。
   请求示例：

   ```
   $ curl --header "Accept:application/json" \
   --header "Content-Type:application/json" \
   --request POST --user sat_username:sat_password --insecure \
   --data @life-cycle.json \
   https://satellite.example.com/katello/api/environments \
   | python -m json.tool
   ```

响应示例：

```
  output omitted
 "description": null,
 "id": 2,
 "label": "api-dev",
 "library": false,
 "name": "API Development",
 "organization": {
     "id": 1,
     "label": "Default_Organization",
     "name": "Default Organization"
 },
 "permissions": {
     "destroy_lifecycle_environments": true,
     "edit_lifecycle_environments": true,
     "promote_or_remove_content_views_to_environments": true,
     "view_lifecycle_environments": true
 },
"prior": {
     "id": 1,
     "name": "Library"
 },
 output truncated
```

在命令输出中，您可以看到此生命周期环境的 ID 为 **2**，在此生命周期环境是 **1**。使用 ID 为 **2** 的生命周期环境创建此环境的后续版本。

4. 编辑之前创建的 **life-cycle.json** 文件，更新 标签、**name** 和 **prior** 值。

```
{"organization_id":1,"label":"api-qa","name":"API QA","prior":2}
```

5. 将 **prior** 选项设置为 **2** 来创建生命周期环境。
请求示例：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" \
--request POST --user sat_username:sat_password --insecure \
--data @life-cycle.json \
https://satellite.example.com/katello/api/environments \
| python -m json.tool
```

响应示例：

```
  output omitted
 "description": null,
"id": 3,
 "label": "api-qa",
 "library": false,
 "name": "API QA",
 "organization": {
     "id": 1,
     "label": "Default_Organization",
     "name": "Default Organization"
 },
```

```
    "permissions": {
        "destroy_lifecycle_environments": true,
        "edit_lifecycle_environments": true,
        "promote_or_remove_content_views_to_environments": true,
        "view_lifecycle_environments": true
    },
    "prior": {
        "id": 2,
        "name": "API Development"
    },
    "successor": null,
    output truncated
```

在命令输出中，您可以看到此生命周期环境的 ID 为 **3**，在此生命周期环境是 **2**。

更新生命周期环境

您可以使用 PUT 命令更新生命周期环境。

这个示例请求更新 ID 为 **3** 的生命周期环境的描述。

请求示例：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" \
--request POST --user sat_username:sat_password --insecure \
--data '{"description":"Quality Acceptance Testing"}' \
https://satellite.example.com/katello/api/environments/3 \
| python -m json.tool
```

响应示例：

```
    output omitted
    "description": "Quality Acceptance Testing",
    "id": 3,
    "label": "api-qa",
    "library": false,
    "name": "API QA",
    "organization": {
        "id": 1,
        "label": "Default_Organization",
        "name": "Default Organization"
    },
    "permissions": {
        "destroy_lifecycle_environments": true,
        "edit_lifecycle_environments": true,
        "promote_or_remove_content_views_to_environments": true,
        "view_lifecycle_environments": true
    },
    "prior": {
        "id": 2,
        "name": "API Development"
    },
    output truncated
```

删除生命周期环境

您可以删除一个没有后续者的生命周期环境。因此，使用以下格式的命令以反向顺序删除它们：

请求示例：

```
$ curl --request DELETE --user sat_username:sat_password --insecure \
https://satellite.example.com/katello/api/environments/:id
```

5.3. 将内容上传到 SATELLITE 服务器

本节概述了如何使用 Satellite 6 API 上传和导入大型文件到 Satellite 服务器。这个过程涉及四个步骤：

1. 创建上传请求。
2. 上传内容。
3. 导入内容。
4. 删除上传请求。

您可以上传的最大文件大小为 2MB。有关上传较大的内容的详情，请参考 上传大于 2 MB 的内容。

流程

1. 为变量 **name** 分配软件包名称：
   请求示例：

   ```
   $ export name=jq-1.6-2.el7.x86_64.rpm
   ```

2. 将文件的校验和分配给变量 **checksum** ：
   请求示例：

   ```
   $ export checksum=$(sha256sum $name|cut -c 1-65)
   ```

3. 将文件大小分配给变量 **大小** ：
   请求示例：

   ```
   $ export size=$(du -bs $name|cut -f 1)
   ```

4. 以下命令创建一个新的上传请求，并使用 **大小和 校验和** 返回请求的上传 ID。
   请求示例：

   ```
   $ curl -H 'Content-Type: application/json' -X POST -k \
   -u sat_username:sat_password -d "{\"size\": \"$size\", \
   \"checksum\":\"$checksum\"}" \
   https://$(hostname -f)/katello/api/v2/repositories/76/content_uploads
   ```

   其中 76，本例中为 Repository ID 示例。

   请求示例：

   ```
   {"upload_id":"37eb5900-597e-4ac3-9bc5-2250c302fdc4"}
   ```

5. 将上传 ID 分配给变量 **upload_id** ：
请求示例：

```
$ export upload_id=37eb5900-597e-4ac3-9bc5-2250c302fdc4
```

6. 分配您要上传到变量 **路径 的软件包路径** ：

```
$ export path=/root/jq/jq-1.6-2.el7.x86_64.rpm
```

7. 上传您的内容。在上传数据时，确保使用正确的 MIME 类型。API 将 application/json MIME 类型用于对 Satellite 6 的大多数请求。组合 upload_id、MIME 类型和其他参数以上传内容。
请求示例：

```
$ curl -u sat_username:sat_password -H Accept:application/json -H \
Content-Type:multipart/form-data -X PUT --data-urlencode size=$size --data-urlencode
offset=0 \
--data-urlencode content@${path} \
https://$(hostname -f)/katello/api/v2/repositories/76/content_uploads/$upload_id
```

8. 将内容上传到 Satellite 服务器后，您需要将其导入到相应的存储库。在完成此步骤前，Satellite 服务器不会检测到新内容。
请求示例：

```
$ curl -H "Content-Type:application/json" -X PUT -u \
sat_username:sat_password -k -d \
"{\"uploads\":[{\"id\": \"$upload_id\", \"name\": \"$name\", \
\"checksum\": \"$checksum\" }]}" \
https://$(hostname -f)/katello/api/v2/repositories/76/import_uploads
```

9. 成功上传并导入内容后，您可以删除上传请求。这会在上传过程中释放数据使用的任何临时磁盘空间。
请求示例：

```
$ curl -H 'Content-Type: application/json' -X DELETE -k \
-u sat_username:sat_password -d "{}" \
https://$(hostname -f)/katello/api/v2/repositories/76/content_uploads/$upload_id
```

上传大于 2 MB 的内容

以下示例演示了如何将大型文件分成块，创建一个上传请求，上传单个文件，将它们导入到 Satellite，然后删除上传请求。请注意，本示例使用示例内容、主机名、用户名、存储库 ID 和文件名。

1. 为变量 **name** 分配软件包名称：

```
$ export name=bpftool-3.10.0-1160.2.1.el7.centos.plus.x86_64.rpm
```

2. 将文件的校验和分配给变量 **checksum** ：

```
$ export  checksum=$(sha256sum $name|cut -c 1-65)
```

3. 将文件大小分配给变量 **大小** ：

```
$ export  size=$(du -bs $name|cut -f 1)
```

4. 以下命令创建一个新的上传请求，并使用 **大小和 校验和** 返回请求的上传 ID。
请求示例：

```
$ curl -H 'Content-Type: application/json' -X POST -k \
-u sat_username:sat_password -d "{\"size\": \"$size\", \
\"checksum\":\"$checksum\"}" \
https://$(hostname -f)/katello/api/v2/repositories/76/content_uploads
```

其中 76，本例中为 Repository ID 示例。

输出示例

```
{"upload_id":"37eb5900-597e-4ac3-9bc5-2250c302fdc4"}
```

5. 将上传 ID 分配给变量 **upload_id** ：

```
$ export upload_id=37eb5900-597e-4ac3-9bc5-2250c302fdc4
```

6. 以 2MB 块分割文件：

```
$ split --bytes 2MB --numeric-suffixes \
--suffix-length=1 bpftool-3.10.0-1160.2.1.el7.centos.plus.x86_64.rpm bpftool
```

输出示例

```
$ ls bpftool[0-9] -l
-rw-r--r--. 1 root root 2000000 Mar 31 14:15 bpftool0
-rw-r--r--. 1 root root 2000000 Mar 31 14:15 bpftool1
-rw-r--r--. 1 root root 2000000 Mar 31 14:15 bpftool2
-rw-r--r--. 1 root root 2000000 Mar 31 14:15 bpftool3
-rw-r--r--. 1 root root  868648 Mar 31 14:15 bpftool4
```

7. 将分割文件的前缀分配给变量路径。

```
$ export path=/root/tmp/bpftool
```

8. 上传文件块。对于第一个块，偏移从 0 开始，并为每个文件增加 2000000。请注意，使用 offset 参数及其与文件大小的关系。另请注意，索引会在路径变量后使用，如 ${path}0, ${path}1。
请求示例：

```
$ curl -u sat_username:sat_password -H Accept:application/json -H \
Content-Type:multipart/form-data  \
-X PUT --data-urlencode size=$size --data-urlencode offset=0 \
--data-urlencode content@${path}0 https://$(hostname -
f)/katello/api/v2/repositories/76/content_uploads/$upload_id

$ curl -u sat_username:sat_password -H Accept:application/json -H \
Content-Type:multipart/form-data \
-X PUT --data-urlencode size=$size --data-urlencode offset=2000000 \
--data-urlencode content@${path}1 https://$(hostname -
f)/katello/api/v2/repositories/76/content_uploads/$upload_id
```

```
$ curl -u sat_username:sat_password -H Accept:application/json -H \
Content-Type:multipart/form-data \
-X PUT --data-urlencode size=$size --data-urlencode offset=4000000 \
--data-urlencode content@${path}2 https://$(hostname -
f)/katello/api/v2/repositories/76/content_uploads/$upload_id

$curl -u sat_username:sat_password -H Accept:application/json -H \
Content-Type:multipart/form-data \
-X PUT --data-urlencode size=$size --data-urlencode offset=6000000
--data-urlencode content@${path}3 https://$(hostname -
f)/katello/api/v2/repositories/76/content_uploads/$upload_id

$ curl -u sat_username:sat_password -H Accept:application/json -H \
Content-Type:multipart/form-data \
-X PUT --data-urlencode size=$size --data-urlencode offset=8000000 \
--data-urlencode content@${path}4 https://$(hostname -
f)/katello/api/v2/repositories/76/content_uploads/$upload_id
```

9. 将完整上传导入到存储库：

```
$ curl -H "Content-Type:application/json" -X PUT -u \
sat_username:sat_password -k -d \
"{\"uploads\":[{\"id\": \"$upload_id\", \
\"name\": \"$name\", \"checksum\": \"$checksum\" }]}" \
https://$(hostname -f)/katello/api/v2/repositories/76/import_uploads
```

10. 删除上传请求：

```
$ curl -H 'Content-Type: application/json' -X DELETE -k \
-u sat_username:sat_password -d "{}" \
https://$(hostname -f)/katello/api/v2/repositories/76/content_uploads/$upload_id
```

上传重复内容

请注意，如果您尝试使用以下方法上传重复内容：

请求示例：

```
$ curl -H 'Content-Type: application/json' -X POST -k \
-u sat_username:sat_password -d "{\"size\": \"$size\", \"checksum\":\"$checksum\"}" \
https://$(hostname -f)/katello/api/v2/repositories/76/content_uploads
```

调用将返回内容单元 ID 而不是上传 ID，如下所示：

```
{"content_unit_href":"/pulp/api/v3/content/file/files/c1bcdfb8-d840-4604-845e-86e82454c747/"}
```

您可以复制此输出并直接调用导入上传，将内容添加到存储库中：

请求示例：

```
$ curl -H "Content-Type:application/json" -X PUT -u \
sat_username:sat_password -k \-d \
"{\"uploads\":[{\"content_unit_id\": \"/pulp/api/v3/content/file/files/c1bcdfb8-d840-4604-845e-
```

```
86e82454c747/\", \
\"name\": \"$name\", \ \"checksum\": \"$checksum\" }]}" https://$(hostname -
f)/katello/api/v2/repositories/76/import_uploads
```

请注意，调用将从 使用 **upload_id** 更改为使用 **content_unit_id**。

5.4. 将勘误应用到主机或主机集合

您可以使用 API 将勘误表应用到主机、主机组或主机集合。以下是 PUT 请求的基本语法：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" --request PUT \
--user sat_username:sat_password --insecure \
--data json-formatted-data https://satellite7.example.com
```

您可以浏览 API 中内置的文档，以查找用于应用勘误表的 URL。您可以使用 Satellite Web UI 来帮助发现搜索查询的格式。导航到 **Hosts** > **Host Collections** 并选择主机集合。进入 **Collection Actions** > **Errata Installation** 并注意搜索查询框内容。例如，对于名为 *my-collection* 的 Host Collection，搜索框包含 **host_collection="my-collection"。**

将勘误应用到主机

本例使用 API URL 进行批量操作 **/katello/api/hosts/bulk/install_content** 来显示简单搜索所需的格式。

请求示例：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" --request PUT \
--user sat_username:sat_password --insecure \
--data "{\"organization_id\":1,\"included\":{\"search\":\"my-host\"},\"content_type\":\"errata\",\"content\":
[\"RHBA-2016:1981\"]}" \
https://satellite.example.com/api/v2/hosts/bulk/install_content
```

将勘误应用到主机集合

在本例中，请注意所需的转义级别，以传递搜索字符串 **host_collection="my-collection"**，如 Satellite Web UI 中所示。

请求示例：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" --request PUT \
--user sat_username:sat_password --insecure \
--data "{\"organization_id\":1,\"included\":{\"search\":\"host_collection=\\\"my-
collection\\\"\"},\"content_type\":\"errata\",\"content\":[\"RHBA-2016:1981\"]}" \
https://satellite.example.com/api/v2/hosts/bulk/install_content
```

5.5. 使用扩展搜索

您可以在 Web UI 中找到可用于构建搜索查询的搜索参数。如需更多信息，请参阅管理 *Red Hat Satellite 中的* 构建搜索查询。

例如，要搜索主机，请完成以下步骤：

1. 在 Satellite Web UI 中，进入到 **Hosts** > **All Hosts**，点 **Search** 字段显示搜索参数列表。
2. 找到您要使用的搜索参数。在本例中，找到 **os_title** 和 **model**。
3. 组合 API 查询中的搜索参数，如下所示：
   请求示例：

```
$ curl --insecure --user sat_username:sat_password \
https://satellite.example.com/api/v2/hosts?
search=os_title=\"RedHat+7.7\",model=\"PowerEdge+R330\" \
| python -m json.tool
```

响应示例：

```
{
  ...
  "results": [
    {
      "model_id": 1,
      "model_name": "PowerEdge R330",
      "name": "satellite.example.com",
      "operatingsystem_id": 1,
      "operatingsystem_name": "RedHat 7.7",
      ...
    }
  ],
  "search": "os_title=\"RedHat 7.7\",model=\"PowerEdge R330\"",
  "subtotal": 1,
  "total": 11
}
```

5.6. 使用带有分页控制的搜索

您可以使用 **per_page** 和 **page** 分页参数来限制 API 搜索查询返回的搜索结果。**per_ page** 参数指定每个页面的结果数，page 参数指定按 **per_page** 参数计算的结果数，以返回。

当您没有指定任何分页参数时，要返回的默认项目数设置为 1000，但 **per_page** 值在指定 **page** 参数时应用默认值为 20。

列出内容视图

本例在页面中返回内容视图的列表。列表中每页包含 10 个密钥，并返回第三页。

请求示例：

```
$ curl --request GET --user sat_username:sat_password \
https://satellite.example.com/katello/api/content_views?per_page=10&amp;page=3
```

列出激活码

这个示例返回有 ID 为 **1** 的机构的激活码列表。列表中包含每个页面 30 个密钥，并返回第二个页面。

请求示例：

```
$ curl --request GET --user sat_username:sat_password \
https://satellite.example.com/katello/api/activation_keys?
organization_id=1&amp;amp;per_page=30&amp;amp;page=2
```

返回多个页面

您可以使用 **for** 循环结构来获取多个结果页面。

这个示例将第 1 页返回到内容视图的第 3 页，每个页面有 5 个结果：

```
$ for i in seq 1 3; do \
curl --request GET --user sat_username:sat_password \
https://satellite.example.com/katello/api/content_views?per_page=5&amp;amp;page=$i; \
done
```

5.7. 覆盖智能类参数

您可以使用 API 搜索智能参数，并提供一个值来覆盖类中的智能参数。您可以在 **https://*satellite.example.com*/apidoc/v2/smart_class_parameters/update.html** 中找到的内置 API 参考中可以修改的属性的完整列表。

1. 找到您要更改的 Smart Class 参数的 ID：
   - 列出所有智能类参数.
     请求示例：

     ```
     $ curl --request GET --insecure --user sat_username:sat_password \
     https://satellite.example.com/api/smart_class_parameters
     ```

   - 如果您知道 Puppet 类 ID，如 5，您可以限制范围：
     请求示例：

     ```
     $ curl --request GET --insecure --user sat_username:sat_password \
     https://satellite.example.com/api/puppetclasses/5/smart_class_parameters
     ```

     这两个调用都接受搜索参数。您可以在 Satellite Web UI 中查看可搜索字段的完整列表。导航到 **Configure** > **Smart variables** 并点搜索查询框，以显示字段列表。

     两个特别有用的搜索参数是 **puppetclass_name** 和 **key**，可用于搜索特定参数。例如，使用 **--data** 选项传递 URL 编码数据。

     请求示例：

     ```
     $ curl --request GET --insecure --user sat_username:sat_password \
     --data 'search=puppetclass_name = access_insights_client and key = authmethod' \
     https://satellite.example.com/api/smart_class_parameters
     ```

     Satellite 支持标准范围搜索语法。

2. 找到 参数的 ID 时，列出包括当前覆盖值在内的完整详情。
   请求示例：

```
$ curl --request GET --insecure --user sat_username:sat_password \
https://satellite.example.com/api/smart_class_parameters/63
```

3. 启用覆盖参数值。
请求示例：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" \
--request PUT --insecure --user sat_username:sat_password \
--data '{"smart_class_parameter":{"override":true}}' \
https://satellite.example.com/api/smart_class_parameters/63
```

请注意，您无法手动创建或删除参数。您只能修改其属性。Satellite 仅在从代理导入类时创建和删除参数。

4. 添加自定义覆盖匹配器。
请求示例：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" \
--request PUT --insecure --user sat_username:sat_password \
--data '{"smart_class_parameter":{"override_value":
{"match":"hostgroup=Test","value":"2.4.6"}}}' \
https://satellite.example.com/api/smart_class_parameters/63
```

有关覆盖值的更多信息，请参阅 **https://*satellite.example.com*/apidoc/v2/override_values.html**。

5. 您可以删除覆盖值。
请求示例：

```
$ curl --request DELETE --user sat_username:sat_password \
https://satellite.example.com/api/smart_class_parameters/63/override_values/3
```

5.8. 使用外部文件修改智能类参数

使用外部文件简化了使用 JSON 数据的过程。使用带有语法高亮显示的编辑器可帮助您避免和查找错误。

使用外部文件修改智能类参数

本例使用 MOTD Puppet 清单。

1. 根据名称搜索 Puppet 类，本例中为 **motd**。
请求示例：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" \
--request GET --user sat_user:sat_password --insecure \
https://satellite.example.com/api/smart_class_parameters?search=puppetclass_name=motd
\
| python -m json.tool
```

2. 检查以下输出：每个智能类参数都有一个 ID，ID 为同一 Satellite 实例全局使用。**motd** 类的 **content** 参数在此卫星服务器中有 **id=3**。不要将此 ID 与 Puppet 类名称之前显示的 Puppet 类 ID 混淆。
响应示例：

```
{
 "avoid_duplicates": false,
  "created_at": "2017-02-06 12:37:48 UTC", # Remove this line.
   "default_value": "", # Add a new value here.
   "description": "",
  "hidden_value": "",
  "hidden_value?": false,
  "id": 3,
  "merge_default": false,
  "merge_overrides": false,
  "override": false, # Set the override value to true.
   "override_value_order": "fqdn\nhostgroup\nos\ndomain",
  "override_values": [], # Remove this line.
   "override_values_count": 0,
  "parameter": "content",
  "parameter_type": "string",
  "puppetclass_id": 3,
  "puppetclass_name": "motd",
  "required": false,
  "updated_at": "2017-02-07 11:56:55 UTC", # Remove this line.
   "use_puppet_default": false,
  "validator_rule": null,
  "validator_type": ""
}
```

3. 使用参数 ID **3** 获取特定于 **motd** 参数的信息，并将输出重定向到文件，如 **output_file.json**。
请求示例：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" --request GET \
--user sat_user:sat_password --insecure \`
https://satellite.example.com/api/smart_class_parameters/3 \
| python -m json.tool > output_file.json
```

4. 将上一步中创建的文件复制到新文件以进行编辑，例如 **changed_file.json** ：

```
$ cp output_file.json changed_file.json
```

5. 修改文件中所需的值。在本例中，更改 **motd** 模块的 content 参数，这需要将 **override** 选项从 **false** 改为 **true** ：

```
{
 "avoid_duplicates": false,
  "created_at": "2017-02-06 12:37:48 UTC", # Remove this line.
   "default_value": "", # Add a new value here.
   "description": "",
  "hidden_value": "",
  "hidden_value?": false,
  "id": 3,
```

```
  "merge_default": false,
  "merge_overrides": false,
  "override": false, # Set the override value to true.
   "override_value_order": "fqdn\nhostgroup\nos\ndomain",
  "override_values": [], # Remove this line.
   "override_values_count": 0,
  "parameter": "content",
  "parameter_type": "string",
  "puppetclass_id": 3,
  "puppetclass_name": "motd",
  "required": false,
  "updated_at": "2017-02-07 11:56:55 UTC", # Remove this line.
   "use_puppet_default": false,
  "validator_rule": null,
  "validator_type": ""
}
```

6. 编辑文件后，验证它是否如下所示，然后保存更改：

```
{
 "avoid_duplicates": false,
  "default_value": "No Unauthorized Access Allowed",
   "description": "",
  "hidden_value": "",
  "hidden_value?": false,
  "id": 3,
  "merge_default": false,
  "merge_overrides": false,
  "override": true,
   "override_value_order": "fqdn\nhostgroup\nos\ndomain",
  "override_values_count": 0,
  "parameter": "content",
  "parameter_type": "string",
  "puppetclass_id": 3,
  "puppetclass_name": "motd",
  "required": false,
  "use_puppet_default": false,
  "validator_rule": null,
  "validator_type": ""
}
```

7. 将更改应用到 Satellite 服务器：

```
$ curl --header "Accept:application/json" \
--header "Content-Type:application/json" \
--request PUT --user sat_username:sat_password --insecure \
--data @changed_file.json \
https://satellite.example.com/api/smart_class_parameters/3
```

5.9. 删除 OPENSCAP 报告

在 Satellite 服务器中，您可以删除一个或多个 OpenSCAP 报告。但是，当您删除报告时，您必须一次删除一个页面。如果要删除所有 Openscap 报告，请使用如下 bash 脚本。

删除 OpenSCAP 报告

要删除 OpenSCAP 报告，请完成以下步骤：

1. 列出所有 OpenSCAP 报告。请注意您要删除的报告的 ID。
   请求示例：

```
curl --insecure --user username:_password_ \
https://satellite.example.com/api/v2/compliance/arf_reports/ | python -m json.tool
```

   响应示例：

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  3252    0  3252    0     0   4319      0 --:--:-- --:--:-- --:--:--  4318
{
    "page": 1,
    "per_page": 20,
    "results": [
        {
            "created_at": "2017-05-16 13:27:09 UTC",
            "failed": 0,
            "host": "host1.example.com",
            "id": 404,
            "othered": 0,
            "passed": 0,
            "updated_at": "2017-05-16 13:27:09 UTC"
        },
        {
            "created_at": "2017-05-16 13:26:07 UTC",
            "failed": 0,
            "host": "host2.example.com,
            "id": 405,
            "othered": 0,
            "passed": 0,
            "updated_at": "2017-05-16 13:26:07 UTC"
        },
        {
            "created_at": "2017-05-16 13:25:07 UTC",
            "failed": 0,
            "host": "host3.example.com",
            "id": 406,
            "othered": 0,
            "passed": 0,
            "updated_at": "2017-05-16 13:25:07 UTC"
        },
        {
            "created_at": "2017-05-16 13:24:07 UTC",
            "failed": 0,
            "host": "host4.example.com",
            "id": 407,
            "othered": 0,
            "passed": 0,
            "updated_at": "2017-05-16 13:24:07 UTC"
        },
    ],
```

```
    "search": null,
    "sort": {
        "by": null,
        "order": null
    },
    "subtotal": 29,
    "total": 29
```

2. 使用上一步中的 ID，删除 OpenSCAP 报告。对您要删除的每个 ID 重复此操作。
请求示例：

```
# curl --insecure --user username:_password_ \
--header "Content-Type: application/json" \
--request DELETE https://satellite.example.com/api/v2/compliance/arf_reports/405
```

响应示例：

```
HTTP/1.1 200 OK
Date: Thu, 18 May 2017 07:14:36 GMT
Server: Apache/2.4.6 (Red Hat Enterprise Linux)
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Foreman_version: 1.11.0.76
Foreman_api_version: 2
Apipie-Checksum: 2d39dc59aed19120d2359f7515e10d76
Cache-Control: max-age=0, private, must-revalidate
X-Request-Id: f47eb877-35c7-41fe-b866-34274b56c506
X-Runtime: 0.661831
X-Powered-By: Phusion Passenger 4.0.18
Set-Cookie: request_method=DELETE; path=/
Set-Cookie: _session_id=d58fe2649e6788b87f46eabf8a461edd; path=/; secure; HttpOnly
ETag: "2574955fc0afc47cb5394ce95553f428"
Status: 200 OK
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8
```

删除所有 OpenSCAP 报告的 BASH 脚本示例

使用以下 bash 脚本删除所有 OpenSCAP 报告：

```
#!/bin/bash

#this script removes all the arf reports from the satellite server

#settings
USER=username
PASS=password
URI=https://satellite.example.com

#check amount of reports
 while [ $(curl --insecure --user $USER:$PASS $URI/api/v2/compliance/arf_reports/ | python -m
json.tool | grep \"\total\": | cut --fields=2 --delimiter":" | cut --fields=1 --delimiter"," | sed "s/ //g") -gt 0 ];
do
```

```
#fetch reports
 for i in $(curl --insecure --user $USER:$PASS $URI/api/v2/compliance/arf_reports/ | python -m
json.tool | grep \"\id\": | cut --fields=2 --delimiter":" | cut --fields=1 --delimiter"," | sed "s/ //g")

#delete reports
  do
  curl --insecure --user $USER:$PASS --header "Content-Type: application/json" --request DELETE
$URI/api/v2/compliance/arf_reports/$i
  done
done
```

5.10. 使用 SATELLITE API 使用 PULP

将 API 请求发送到与 Satellite 集成的 Pulp 请求时，使用基于证书的身份验证。

以下 Pulp API 请求示例包括如何使用 Pulp CLI 作为替代方案。当您以 root 用户身份运行 **pulp** 命令时，Pulp CLI 会使用 **/root/.config/pulp/cli.toml** 中配置的系统证书。

列出软件仓库

要列出所有存储库的端点为 **/pulp/api/v3/repositories/**。以下查询从 *satellite.example.com* 获取存储库列表，同时提供从 Satellite 服务器发出请求所需的证书。

请求示例：

```
curl --cacert /etc/pki/katello/certs/katello-server-ca.crt \
--cert /etc/foreman/client_cert.pem --key /etc/foreman/client_key.pem \
https://<satellite.example.com>/pulp/api/v3/repositories/ \
| python3 -m json.tool
```

响应示例：

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "pulp_href": "/pulp/api/v3/repositories/rpm/rpm/018cd05a-4b83-73db-b71c-587c6181d89b/",
            "pulp_created": "2024-01-03T17:23:47.715882Z",
            "versions_href": "/pulp/api/v3/repositories/rpm/rpm/018cd05a-4b83-73db-b71c-
587c6181d89b/versions/",
            "pulp_labels": {},
            "latest_version_href": "/pulp/api/v3/repositories/rpm/rpm/018cd05a-4b83-73db-b71c-
587c6181d89b/versions/1/",
            "name": "Red_Hat_Enterprise_Linux_8_for_x86_64_-_BaseOS_Kickstart_8_9-49838",
            "description": null,
            "retain_repo_versions": null,
            "remote": null
        }
    ]
}
```

或者，使用 Pulp CLI 列出存储库：

```
# pulp repository list
[
  {
    "pulp_href": "/pulp/api/v3/repositories/rpm/rpm/018cd025-c6ef-7237-a99e-70bab3d30941/",
    "pulp_created": "2024-01-03T16:26:25.904682Z",
    "versions_href": "/pulp/api/v3/repositories/rpm/rpm/018cd025-c6ef-7237-a99e-
70bab3d30941/versions/",
    "pulp_labels": {},
    "latest_version_href": "/pulp/api/v3/repositories/rpm/rpm/018cd025-c6ef-7237-a99e-
70bab3d30941/versions/1/",
    "name": "Red_Hat_Enterprise_Linux_8_for_x86_64_-_AppStream_RPMs_8-2875",
    "description": null,
    "retain_repo_versions": null,
    "remote": null
  }
]
```

检查 Pulp 状态

返回有关 Pulp 的状态信息的端点是 **/pulp/api/v3/status/**。对 Pulp 状态的请求不需要身份验证。

请求示例：

```
curl https://<satellite.example.com>/pulp/api/v3/status/ \
| python3 -m json.tool
```

响应示例：

```
{
    "versions": [
        {
            "component": "core",
            "version": "3.39.4",
            "package": "pulpcore",
            "domain_compatible": true
        },
        {
            "component": "rpm",
            "version": "3.23.0",
            "package": "pulp-rpm",
            "domain_compatible": true
        },
...
```

或者，使用 Pulp CLI 检索 Pulp 状态：

```
# pulp status
{
  "versions": [
    {
      "component": "core",
      "version": "3.39.4",
      "package": "pulpcore",
      "domain_compatible": true
    },
```

```
    {
      "component": "rpm",
      "version": "3.23.0",
      "package": "pulp-rpm",
      "domain_compatible": true
    },
...
```

其他资源

- 使用 Pulp CLI 的详细信息，运行 **pulp --help**。
- 在 **https://** ***<satellite.example.com>*** **/pulp/api/v3/docs/** 的 Satellite 服务器上提供了 Pulp 的完整 API 引用。

附录 A. API 响应代码

Red Hat Satellite 6 API 为 API 调用提供 HTTP 响应状态代码。以下代码对于 Satellite API 中的所有资源都是通用的。

表 A.1. API 响应代码

响应	解释
200 OK	对于成功的请求操作： show、index、update 或 delete (GET、PUT、DELETE 请求)。
201 created	用于成功创建操作(POST 请求)。
301 永久移动	当 Satellite 限制为使用 HTTPS 但尝试使用 HTTP 时，会进行重定向。
400 错误请求	缺少需要的参数，或者搜索查询具有无效的语法。
401 未授权	授权用户失败（例如，凭证不正确）。
403 Forbidden	用户没有足够的权限来执行操作或读取资源，或者一般不支持该操作。
404 not Found	带有给定 ID 的记录不存在。当请求的记录不存在时，它可能会出现在 show 和 delete 操作中；或者在创建、更新和删除操作时显示、更新和删除操作。
409 冲突	无法因为存在的依赖项而删除记录（例如，带有主机的主机组）。
415 不支持的 Media 类型	HTTP 请求的内容类型是 JSON。
422 Unprocessable Entity	由于一些验证错误，创建实体失败。仅适用于创建或更新操作。
500 内部服务器错误	意外的内部服务器错误。
503 服务不可用	服务器没有运行。

附录 B. API 权限列表

Red Hat Satellite 6 API 支持许多操作，其中许多需要特定权限。下表列出了 API 权限名称、与这些权限关联的操作以及关联的资源类型。

表 B.1. API 权限列表

权限名称	Actions	资源类型
view_activation_keys	• katello/activation_keys/all • katello/activation_keys/index • katello/activation_keys/auto_complete_search • katello/api/v2/activation_keys/index • katello/api/v2/activation_keys/show • katello/api/v2/activation_keys/available_host_collections • katello/api/v2/activation_keys/available_releases • katello/api/v2/activation_keys/product_content	Katello::ActivationKey
create_activation_keys	• katello/api/v2/activation_keys/create • katello/api/v2/activation_keys/copy	Katello::ActivationKey
edit_activation_keys	• katello/api/v2/activation_keys/update • katello/api/v2/activation_keys/content_override • katello/api/v2/activation_keys/add_subscriptions • katello/api/v2/activation_keys/remove_subscriptions	Katello::ActivationKey

权限名称	Actions	资源类型
destroy_activation_keys	• katello/api/v2/activation_keys/destroy	Katello::ActivationKey
logout	• users/logout	
view_architectures	• architectures/index • architectures/show • architectures/auto_complete_search • api/v2/architectures/index • api/v2/architectures/show	
create_architectures	• architectures/new • architectures/create • api/v2/architectures/create	
edit_architectures	• architectures/edit • architectures/update • api/v2/architectures/update	
destroy_architectures	• architectures/destroy • api/v2/architectures/destroy	

权限名称	Actions	资源类型
view_audit_logs	• audits/index • audits/show • audits/auto_complete_search • api/v2/audits/index • api/v2/audits/show	
view_authenticators	• auth_source_ldaps/index • auth_source_ldaps/show • api/v2/auth_source_ldaps/index • api/v2/auth_source_ldaps/show	
create_authenticators	• auth_source_ldaps/new • auth_source_ldaps/create • api/v2/auth_source_ldaps/create	
edit_authenticators	• auth_source_ldaps/edit • auth_source_ldaps/update • api/v2/auth_source_ldaps/update	
destroy_authenticators	• auth_source_ldaps/destroy • api/v2/auth_source_ldaps/destroy	

权限名称	Actions	资源类型
view_bookmarks	• bookmarks/index • bookmarks/show • api/v2/bookmarks/index • api/v2/bookmarks/show	
create_bookmarks	• bookmarks/new • bookmarks/create • api/v2/bookmarks/new • api/v2/bookmarks/create	
edit_bookmarks	• bookmarks/edit • bookmarks/update • api/v2/bookmarks/edit • api/v2/bookmarks/update	
destroy_bookmarks	• bookmarks/destroy • api/v2/bookmarks/destroy	

权限名称	Actions	资源类型
download_bootdisk	• foreman_bootdisk/disks/generic • foreman_bootdisk/disks/host • foreman_bootdisk/disks/full_host • foreman_bootdisk/disks/subnet • foreman_bootdisk/disks/help • foreman_bootdisk/api/v2/disks/generic • foreman_bootdisk/api/v2/disks/host	
manage_capsule_content	• katello/api/v2/capsule_content/lifecycle_environments • katello/api/v2/capsule_content/available_lifecycle_environments • katello/api/v2/capsule_content/add_lifecycle_environment • katello/api/v2/capsule_content/remove_lifecycle_environment • katello/api/v2/capsule_content/sync • katello/api/v2/capsule_content/sync_status • katello/api/v2/capsule_content/cancel_sync	SmartProxy

权限名称	Actions	资源类型
view_capsule_content	• smart_proxies/pulp_storage • smart_proxies/pulp_status • smart_proxies/show_with_content	SmartProxy
view_compute_profiles	• compute_profiles/index • compute_profiles/show • compute_profiles/auto_complete_search • api/v2/compute_profiles/index • api/v2/compute_profiles/show	
create_compute_profiles	• compute_profiles/new • compute_profiles/create • api/v2/compute_profiles/create	
edit_compute_profiles	• compute_profiles/edit • compute_profiles/update • api/v2/compute_profiles/update	
destroy_compute_profiles	• compute_profiles/destroy • api/v2/compute_profiles/destroy	

权限名称	Actions	资源类型
view_compute_resources	• compute_resources/index • compute_resources/show • compute_resources/auto_complete_search • compute_resources/ping • compute_resources/available_images • api/v2/compute_resources/index • api/v2/compute_resources/show • api/v2/compute_resources/available_images • api/v2/compute_resources/available_clusters • api/v2/compute_resources/available_folders • api/v2/compute_resources/available_flavors • api/v2/compute_resources/available_networks • api/v2/compute_resources/available_resource_pools • api/v2/compute_resources/available_security_groups • api/v2/compute_resources/available_storage_domains • api/v2/compute_resources/available_zones • api/v2/compute_resources/available_storage_pods	

权限名称	Actions	资源类型
create_compute_resources	• compute_resources/new • compute_resources/create • compute_resources/test_connection • api/v2/compute_resources/create	
edit_compute_resources	• compute_resources/edit • compute_resources/update • compute_resources/test_connection • compute_attributes/new • compute_attributes/create • compute_attributes/edit • compute_attributes/update • api/v2/compute_resources/update • api/v2/compute_attributes/create • api/v2/compute_attributes/update	
destroy_compute_resources	• compute_resources/destroy • api/v2/compute_resources/destroy	
view_compute_resources_vms	• compute_resources_vms/index • compute_resources_vms/show	

权限名称	Actions	资源类型
create_compute_resources_vms	• compute_resources_vms/new • compute_resources_vms/create	
edit_compute_resources_vms	• compute_resources_vms/edit • compute_resources_vms/update	
destroy_compute_resources_vms	• compute_resources_vms/destroy	
power_compute_resources_vms	• compute_resources_vms/power • compute_resources_vms/pause	
console_compute_resources_vms	• compute_resources_vms/console	
view_config_groups	• config_groups/index • config_groups/auto_complete_search • api/v2/config_groups/index • api/v2/config_groups/show	
create_config_groups	• config_groups/new • config_groups/create • api/v2/config_groups/create	

权限名称	Actions	资源类型
edit_config_groups	• config_groups/edit • config_groups/update • api/v2/config_groups/update	
destroy_config_groups	• config_groups/destroy • api/v2/config_groups/destroy	
view_config_reports	• config_reports/index • config_reports/show • config_reports/auto_complete_search • api/v2/config_reports/index • api/v2/config_reports/show • api/v2/config_reports/last	
destroy_config_reports	• config_reports/destroy • api/v2/config_reports/destroy	
upload_config_reports	• api/v2/config_reports/create	
view_containers	• containers/index • containers/show • api/v2/containers/index • api/v2/containers/show • api/v2/containers/logs	Container

权限名称	Actions	资源类型
commit_containers	• containers/commit	Container
create_containers	• containers/steps/show • containers/steps/update • containers/new • api/v2/containers/create • api/v2/containers/power	Container
destroy_containers	• containers/destroy • api/v2/containers/destroy	Container
power_compute_resources_vms	• containers/power • api/v2/containers/create • api/v2/containers/power	ComputeResource
view_content_views	• katello/api/v2/content_views/index • katello/api/v2/content_views/show • katello/api/v2/content_views/available_puppet_modules • katello/api/v2/content_views/available_puppet_module_names • katello/api/v2/content_view_filters/index • katello/api/v2/content_view_filters/show • katello/api/v2/content_view_filter_rules/index • katello/api/v2/content_view_filter_rules/show	Katello::ContentView

权限名称	Actions	资源类型
	• katello/api/v2/content_view_histories/index • katello/api/v2/content_view_puppet_modules/index • katello/api/v2/content_view_puppet_modules/show • katello/api/v2/content_view_versions/index • katello/api/v2/content_view_versions/show • katello/api/v2/package_groups/index • katello/api/v2/package_groups/show • katello/api/v2/errata/index • katello/api/v2/errata/show • katello/api/v2/puppet_modules/index • katello/api/v2/puppet_modules/show • katello/content_views/auto_complete • katello/content_views/auto_complete_search • katello/errata/short_details • katello/errata/auto_complete • katello/packages/details • katello/packages/auto_complete • katello/products/auto_complete • katello/repositories/auto_complete_library • katello/content_search/index • katello/content_search/products • katello/content_search/repos	

权限名称	Actions	资源类型
	• katello/content_search/packages • katello/content_search/errata • katello/content_search/puppet_modules • katello/content_search/packages_items • katello/content_search/errata_items • katello/content_search/puppet_modules_items • katello/content_search/view_packages • katello/content_search/view_puppet_modules • katello/content_search/repo_packages • katello/content_search/repo_errata • katello/content_search/repo_puppet_modules • katello/content_search/repo_compare_errata • katello/content_search/repo_compare_packages • katello/content_search/repo_compare_puppet_modules • katello/content_search/view_compare_errata • katello/content_search/view_compare_packages • katello/content_search/view_compare_puppet_modules • katello/content_search/views	

权限名称	Actions	资源类型
create_content_views	• katello/api/v2/content_views/create • katello/api/v2/content_views/copy	Katello::ContentView
edit_content_views	• katello/api/v2/content_views/update • katello/api/v2/content_view_filters/create • katello/api/v2/content_view_filters/update • katello/api/v2/content_view_filters/destroy • katello/api/v2/content_view_filter_rules/create • katello/api/v2/content_view_filter_rules/update • katello/api/v2/content_view_filter_rules/destroy • katello/api/v2/content_view_puppet_modules/create • katello/api/v2/content_view_puppet_modules/update • katello/api/v2/content_view_puppet_modules/destroy	Katello::ContentView
destroy_content_views	• katello/api/v2/content_views/destroy • katello/api/v2/content_views/remove • katello/api/v2/content_view_versions/destroy	Katello::ContentView

权限名称	Actions	资源类型
publish_content_views	• katello/api/v2/content_views/publish • katello/api/v2/content_view_versions/incremental_update	Katello::ContentView
promote_or_remove_content_views	• katello/api/v2/content_view_versions/promote • katello/api/v2/content_views/remove_from_environment • katello/api/v2/content_views/remove	Katello::ContentView
export_content_views	• katello/api/v2/content_view_versions/export	Katello::ContentView
access_dashboard	• dashboard/index • dashboard/save_positions • dashboard/reset_default • dashboard/create • dashboard/destroy • api/v2/dashboard/index	
view_discovered_hosts	• discovered_hosts/index • discovered_hosts/show • discovered_hosts/auto_complete_search • api/v2/discovered_hosts/show	主机

权限名称	Actions	资源类型
submit_discovered_hosts	• api/v2/discovered_hosts/facts • api/v2/discovered_hosts/create	主机
auto_provision_discovered_hosts	• discovered_hosts/auto_provision • discovered_hosts/auto_provision_all • api/v2/discovered_hosts/auto_provision • api/v2/discovered_hosts/auto_provision_all	主机
provision_discovered_hosts	• discovered_hosts/edit • discovered_hosts/update • api/v2/discovered_hosts/update	主机

权限名称	Actions	资源类型
edit_discovered_hosts	• discovered_hosts/update_multiple_location • discovered_hosts/select_multiple_organization • discovered_hosts/update_multiple_organization • discovered_hosts/select_multiple_location • discovered_hosts/refresh_facts • discovered_hosts/reboot • discovered_hosts/reboot_all • api/v2/discovered_hosts/refresh_facts • api/v2/discovered_hosts/reboot • api/v2/discovered_hosts/reboot_all	主机
destroy_discovered_hosts	• discovered_hosts/destroy • discovered_hosts/submit_multiple_destroy • discovered_hosts/multiple_destroy • api/v2/discovered_hosts/destroy	主机
view_discovery_rules	• discovery_rules/index • discovery_rules/show • discovery_rules/auto_complete_search • api/v2/discovery_rules/index • api/v2/discovery_rules/show	DiscoveryRule

权限名称	Actions	资源类型
create_discovery_rules	• discovery_rules/new • discovery_rules/create • api/v2/discovery_rules/create	DiscoveryRule
edit_discovery_rules	• discovery_rules/edit • discovery_rules/update • discovery_rules/enable • discovery_rules/disable • api/v2/discovery_rules/create • api/v2/discovery_rules/update	DiscoveryRule
execute_discovery_rules	• discovery_rules/auto_provision • discovery_rules/auto_provision_all • api/v2/discovery_rules/auto_provision • api/v2/discovery_rules/auto_provision_all	DiscoveryRule
destroy_discovery_rules	• discovery_rules/destroy • api/v2/discovery_rules/destroy	DiscoveryRule

权限名称	Actions	资源类型
view_domains	• domains/index • domains/show • domains/auto_complete_search • api/v2/domains/index • api/v2/domains/show • api/v2/parameters/index • api/v2/parameters/show	
create_domains	• domains/new • domains/create • api/v2/domains/create	
edit_domains	• domains/edit • domains/update • api/v2/domains/update • api/v2/parameters/create • api/v2/parameters/update • api/v2/parameters/destroy • api/v2/parameters/reset	
destroy_domains	• domains/destroy • api/v2/domains/destroy	

权限名称	Actions	资源类型
view_environments	• environments/index • environments/show • environments/auto_complete_search • api/v2/environments/index • api/v2/environments/show	
create_environments	• environments/new • environments/create • api/v2/environments/create	
edit_environments	• environments/edit • environments/update • api/v2/environments/update	
destroy_environments	• environments/destroy • api/v2/environments/destroy	
import_environments	• environments/import_environments • environments/obsolete_and_new • api/v2/environments/import_puppetclasses • api/v2/smart_proxies/import_puppetclasses	

权限名称	Actions	资源类型
view_external_usergroups	• external_usergroups/index • external_usergroups/show • api/v2/external_usergroups/index • api/v2/external_usergroups/show	
create_external_usergroups	• external_usergroups/new • external_usergroups/create • api/v2/external_usergroups/new • api/v2/external_usergroups/create	
edit_external_usergroups	• external_usergroups/edit • external_usergroups/update • external_usergroups/refresh • api/v2/external_usergroups/update • api/v2/external_usergroups/refresh	
destroy_external_usergroups	• external_usergroups/destroy • api/v2/external_usergroups/destroy	

权限名称	Actions	资源类型
view_external_variables	• lookup_keys/index • lookup_keys/show • lookup_keys/auto_complete_search • puppetclass_lookup_keys/index • puppetclass_lookup_keys/show • puppetclass_lookup_keys/auto_complete_search • variable_lookup_keys/index • variable_lookup_keys/show • variable_lookup_keys/auto_complete_search • lookup_values/index • api/v2/smart_variables/index • api/v2/smart_variables/show • api/v2/smart_class_parameters/index • api/v2/smart_class_parameters/show • api/v2/override_values/index • api/v2/override_values/show	

权限名称	Actions	资源类型
create_external_variables	• lookup_keys/new • lookup_keys/create • puppetclass_lookup_keys/new • puppetclass_lookup_keys/create • variable_lookup_keys/new • variable_lookup_keys/create • lookup_values/create • api/v2/smart_variables/create • api/v2/smart_class_parameters/create • api/v2/override_values/create	

权限名称	Actions	资源类型
edit_external_variables	• lookup_keys/edit • lookup_keys/update • puppetclass_lookup_keys/edit • puppetclass_lookup_keys/update • variable_lookup_keys/edit • variable_lookup_keys/update • lookup_values/create • lookup_values/update • lookup_values/destroy • api/v2/smart_variables/update • api/v2/smart_class_parameters/update • api/v2/override_values/create • api/v2/override_values/update • api/v2/override_values/destroy	

权限名称	Actions	资源类型
destroy_external_variables	• lookup_keys/destroy • puppetclass_lookup_keys/destroy • variable_lookup_keys/destroy • lookup_values/destroy • api/v2/smart_variables/destroy • api/v2/smart_class_parameters/destroy • api/v2/override_values/create • api/v2/override_values/update • api/v2/override_values/destroy	
view_facts	• facts/index • facts/show • fact_values/index • fact_values/show • fact_values/auto_complete_search • api/v2/fact_values/index • api/v2/fact_values/show	
upload_facts	• api/v2/hosts/facts	
view_filters	• filters/index • filters/auto_complete_search • api/v2/filters/index • api/v2/filters/show	

权限名称	Actions	资源类型
create_filters	• filters/new • filters/create • api/v2/filters/create	
edit_filters	• filters/edit • filters/update • permissions/index • api/v2/filters/update • api/v2/permissions/index • api/v2/permissions/show	
destroy_filters	• filters/destroy • api/v2/filters/destroy	
view_arf_reports	• arf_reports/index • arf_reports/show • arf_reports/parse_html • arf_reports/show_html • arf_reports/parse_bzip • arf_reports/auto_complete_search • api/v2/compliance/arf_reports/index • api/v2/compliance/arf_reports/show • compliance_hosts/show	

权限名称	Actions	资源类型
destroy_arf_reports	• arf_reports/destroy • arf_reports/delete_multiple • arf_reports/submit_delete_multiple • api/v2/compliance/arf_reports/destroy	
create_arf_reports	• api/v2/compliance/arf_reports/create	
view_policies	• policies/index • policies/show • policies/parse • policies/auto_complete_search • policy_dashboard/index • compliance_dashboard/index • api/v2/compliance/policies/index • api/v2/compliance/policies/show • api/v2/compliance/policies/content	ForemanOpenscap::Policy
edit_policies	• policies/edit • policies/update • policies/scap_content_selected • api/v2/compliance/policies/update	ForemanOpenscap::Policy

权限名称	Actions	资源类型
create_policies	• policies/new • policies/create • api/v2/compliance/policies/create	ForemanOpenscap::Policy
destroy_policies	• policies/destroy • api/v2/compliance/policies/destroy	ForemanOpenscap::Policy
assign_policies	• policies/select_multiple_hosts • policies/update_multiple_hosts • policies/disassociate_multiple_hosts • policies/remove_policy_from_multiple_hosts	ForemanOpenscap::Policy
view_scap_contents	• scap_contents/index • scap_contents/show • scap_contents/auto_complete_search • api/v2/compliance/scap_contents/index • api/v2/compliance/scap_contents/show	ForemanOpenscap::ScapContent
view_scap_contents	• scap_contents/index • scap_contents/show • scap_contents/auto_complete_search • api/v2/compliance/scap_contents/index • api/v2/compliance/scap_contents/show	ForemanOpenscap::ScapContent

权限名称	Actions	资源类型
edit_scap_contents	• scap_contents/edit • scap_contents/update • api/v2/compliance/scap_contents/update	ForemanOpenscap::ScapContent
create_scap_contents	• scap_contents/new • scap_contents/create • api/v2/compliance/scap_contents/create	ForemanOpenscap::ScapContent
destroy_scap_contents	• scap_contents/destroy • api/v2/compliance/scap_contents/destroy	ForemanOpenscap::ScapContent

权限名称	Actions	资源类型
view_job_templates	• job_templates/index • job_templates/show • job_templates/revision • job_templates/auto_complete_search • job_templates/auto_complete_job_category • job_templates/preview • job_templates/export • api/v2/job_templates/index • api/v2/job_templates/show • api/v2/job_templates/revision • api/v2/job_templates/export • api/v2/template_inputs/index • api/v2/template_inputs/show • api/v2/foreign_input_sets/index • api/v2/foreign_input_sets/show	JobTemplate
create_job_templates	• job_templates/new • job_templates/create • job_templates/clone_template • job_templates/import • api/v2/job_templates/create • api/v2/job_templates/clone • api/v2/job_templates/import	JobTemplate

权限名称	Actions	资源类型
edit_job_templates	• job_templates/edit • job_templates/update • api/v2/job_templates/update • api/v2/template_inputs/create • api/v2/template_inputs/update • api/v2/template_inputs/destroy • api/v2/foreign_input_sets/create • api/v2/foreign_input_sets/update • api/v2/foreign_input_sets/destroy	
edit_job_templates	• job_templates/edit • job_templates/update • api/v2/job_templates/update • api/v2/template_inputs/create • api/v2/template_inputs/update • api/v2/template_inputs/destroy • api/v2/foreign_input_sets/create • api/v2/foreign_input_sets/update • api/v2/foreign_input_sets/destroy	

权限名称	Actions	资源类型
edit_remote_execution_features	• remote_execution_features/index • remote_execution_features/show • remote_execution_features/update • api/v2/remote_execution_features/index • api/v2/remote_execution_features/show • api/v2/remote_execution_features/update	RemoteExecutionFeature
destroy_job_templates	• job_templates/destroy • api/v2/job_templates/destroy	JobTemplate
lock_job_templates	• job_templates/lock • job_templates/unlock	JobTemplate
create_job_invocations	• job_invocations/new • job_invocations/create • job_invocations/refresh • job_invocations/rerun • job_invocations/preview_hosts • api/v2/job_invocations/create	JobInvocation

权限名称	Actions	资源类型
view_job_invocations	• job_invocations/index • job_invocations/show • template_invocations/show • api/v2/job_invocations/index • api/v2/job_invocations/show • api/v2/job_invocations/output	JobInvocation
execute_template_invocation		TemplateInvocation
filter_autocompletion_for_template_invocation	• template_invocations/auto_complete_search • job_invocations/show • template_invocations/index	TemplateInvocation
view_foreman_tasks	• foreman_tasks/tasks/auto_complete_search • foreman_tasks/tasks/sub_tasks • foreman_tasks/tasks/index • foreman_tasks/tasks/show • foreman_tasks/api/tasks/bulk_search • foreman_tasks/api/tasks/show • foreman_tasks/api/tasks/index • foreman_tasks/api/tasks/summary	ForemanTasks::Task

权限名称	Actions	资源类型
edit_foreman_tasks	• foreman_tasks/tasks/resume • foreman_tasks/tasks/unlock • foreman_tasks/tasks/force_unlock • foreman_tasks/tasks/cancel_step • foreman_tasks/api/tasks/bulk_resume	ForemanTasks::Task
create_recurring_logics		ForemanTasks::RecurringLogic
view_recurring_logics	• foreman_tasks/recurring_logics/index • foreman_tasks/recurring_logics/show • foreman_tasks/api/recurring_logics/index • foreman_tasks/api/recurring_logics/show	ForemanTasks::RecurringLogic
edit_recurring_logics	• foreman_tasks/recurring_logics/cancel • foreman_tasks/api/recurring_logics/cancel	ForemanTasks::RecurringLogic
view_globals	• common_parameters/index • common_parameters/show • common_parameters/auto_complete_search • api/v2/common_parameters/index • api/v2/common_parameters/show	

权限名称	Actions	资源类型
create_globals	• common_parameters/new • common_parameters/create • api/v2/common_parameters/create	
edit_globals	• common_parameters/edit • common_parameters/update • api/v2/common_parameters/update	
destroy_globals	• common_parameters/destroy • api/v2/common_parameters/destroy	
view_gpg_keys	• katello/gpg_keys/all • katello/gpg_keys/index • katello/gpg_keys/auto_complete_search • katello/api/v2/gpg_keys/index • katello/api/v2/gpg_keys/show	Katello::GpgKey
create_gpg_keys	• katello/api/v2/gpg_keys/create	Katello::GpgKey
edit_gpg_keys	• katello/api/v2/gpg_keys/update • katello/api/v2/gpg_keys/content	Katello::GpgKey

权限名称	Actions	资源类型
destroy_gpg_keys	• katello/api/v2/gpg_keys/destroy	Katello::GpgKey
view_host_collections	• katello/api/v2/host_collections/index • katello/api/v2/host_collections/show • katello/host_collections/auto_complete_search	Katello::HostCollection
create_host_collections	• katello/api/v2/host_collections/create • katello/api/v2/host_collections/copy	Katello::HostCollection
edit_host_collections	• katello/api/v2/host_collections/update • katello/api/v2/host_collections/add_systems • katello/api/v2/host_collections/remove_systems	Katello::HostCollection
destroy_host_collections	• katello/api/v2/host_collections/destroy	Katello::HostCollection
edit_classes	• host_editing/edit_classes • api/v2/host_classes/index • api/v2/host_classes/create • api/v2/host_classes/destroy	

权限名称	Actions	资源类型
create_params	• host_editing/create_params • api/v2/parameters/create	
edit_params	• host_editing/edit_params • api/v2/parameters/update	
destroy_params	• host_editing/destroy_params • api/v2/parameters/destroy • api/v2/parameters/reset	
view_hostgroups	• hostgroups/index • hostgroups/show • hostgroups/auto_complete_search • api/v2/hostgroups/index • api/v2/hostgroups/show	

权限名称	Actions	资源类型
create_hostgroups	• hostgroups/new • hostgroups/create • hostgroups/clone • hostgroups/nest • hostgroups/process_hostgroup • hostgroups/architecture_selected • hostgroups/domain_selected • hostgroups/environment_selected • hostgroups/medium_selected • hostgroups/os_selected • hostgroups/use_image_selected • hostgroups/process_hostgroup • hostgroups/puppetclass_parameters • host/process_hostgroup • puppetclasses/parameters • api/v2/hostgroups/create • api/v2/hostgroups/clone	
edit_hostgroups	• hostgroups/edit • hostgroups/update • hostgroups/architecture_selected • hostgroups/process_hostgroup • hostgroups/architecture_selected	

权限名称	Actions	资源类型
	• hostgroups/domain_selected • hostgroups/environment_selected • hostgroups/medium_selected • hostgroups/os_selected • hostgroups/use_image_selected • hostgroups/process_hostgroup • hostgroups/puppetclass_parameters • hostgroups/openscap_proxy_changed • host/process_hostgroup • puppetclasses/parameters • api/v2/hostgroups/add_ansible_role • api/v2/hostgroups/remove_ansible_role • api/v2/hostgroups/update • api/v2/parameters/create • api/v2/parameters/update • api/v2/parameters/destroy • api/v2/parameters/reset • api/v2/hostgroup_classes/index • api/v2/hostgroup_classes/create • api/v2/hostgroup_classes/destroy	
destroy_hostgroups	• hostgroups/destroy • api/v2/hostgroups/destroy	
view_hosts	• hosts/index	

权限名称	Actions	资源类型
	• hosts/show • hosts/errors • hosts/active • hosts/out_of_sync • hosts/disabled • hosts/pending • hosts/vm • hosts/externalNodes • hosts/pxe_config • hosts/storeconfig_klasses • hosts/auto_complete_search • hosts/bmc • hosts/runtime • hosts/resources • hosts/templates • hosts/overview • hosts/nics • dashboard/OutOfSync • dashboard/errors • dashboard/active • unattended/host_template • unattended/hostgroup_template • api/v2/hosts/index • api/v2/hosts/show • api/v2/hosts/status/configuration • api/v2/hosts/get_status • api/v2/hosts/vm_compute_attributes • api/v2/hosts/template • api/v2/interfaces/index • api/v2/interfaces/show	

权限名称	Actions	资源类型
	• locations/mismatches • organizations/mismatches • hosts/puppet_environment_for_content_view • katello/api/v2/host_autocomplete/auto_complete_search • katello/api/v2/host_errata/index • katello/api/v2/host_errata/show • katello/api/v2/host_errata/auto_complete_search • katello/api/v2/host_subscriptions/index • katello/api/v2/host_subscriptions/events • katello/api/v2/host_subscriptions/product_content • katello/api/v2/hosts/applicable_errata • katello/api/v2/hosts/installable_errata • katello/api/v2/hosts/bulk/available_incremental_updates • katello/api/v2/host_packages/index	
create_hosts	• hosts/new • hosts/create • hosts/clone • hosts/architecture_selected • hosts/compute_resource_selected • hosts/domain_selected • hosts/environment_selected	

权限名称	Actions	资源类型
	• hosts/hostgroup_or_environment_selected • hosts/medium_selected • hosts/os_selected • hosts/use_image_selected • hosts/process_hostgroup • hosts/process_taxonomy • hosts/current_parameters • hosts/puppetclass_parameters • hosts/template_used • hosts/interfaces • compute_resources/cluster_selected • compute_resources/template_selected • compute_resources/provider_selected • compute_resources/resource_pools • puppetclasses/parameters • subnets/freeip • interfaces/new • api/v2/hosts/create • api/v2/interfaces/create • api/v2/tasks/index	
edit_hosts	• hosts/openscap_proxy_changed • hosts/edit • hosts/update • hosts/multiple_actions • hosts/reset_multiple	

权限名称	Actions	资源类型
	• hosts/submit_multiple_enable • hosts/select_multiple_hostgroup • hosts/select_multiple_environment • hosts/submit_multiple_disable • hosts/multiple_parameters • hosts/multiple_disable • hosts/multiple_enable • hosts/update_multiple_environment • hosts/update_multiple_hostgroup • hosts/update_multiple_parameters • hosts/toggle_manage • hosts/select_multiple_organization • hosts/update_multiple_organization • hosts/disassociate • hosts/multiple_disassociate • hosts/update_multiple_disassociate • hosts/select_multiple_owner • hosts/update_multiple_owner • hosts/select_multiple_power_state • hosts/update_multiple_power_state • hosts/select_multiple_puppet_proxy • hosts/update_multiple_puppet_proxy • hosts/select_multiple_puppet_ca_proxy	

权限名称	Actions	资源类型
	• hosts/update_multiple_puppet_ca_proxy • hosts/select_multiple_location • hosts/update_multiple_location • hosts/architecture_selected • hosts/compute_resource_selected • hosts/domain_selected • hosts/environment_selected • hosts/hostgroup_or_environment_selected • hosts/medium_selected • hosts/os_selected • hosts/use_image_selected • hosts/process_hostgroup • hosts/process_taxonomy • hosts/current_parameters • hosts/puppetclass_parameters • hosts/template_used • hosts/interfaces • compute_resources/associate • compute_resources/[:cluster_selected, :template_selected, :provider_selected, :resource_pools] • compute_resources_vms/associate • puppetclasses/parameters • subnets/freeip • interfaces/new	

权限名称	Actions	资源类型
	• api/v2/hosts/add_ansible_role • api/v2/hosts/remove_ansible_role • api/v2/hosts/update • api/v2/hosts/disassociate • api/v2/interfaces/create • api/v2/interfaces/update • api/v2/interfaces/destroy • api/v2/compute_resources/associate • api/v2/hosts/host_collections • katello/api/v2/host_errata/apply • katello/api/v2/host_packages/install • katello/api/v2/host_packages/upgrade • katello/api/v2/host_packages/upgrade_all • katello/api/v2/host_packages/remove • katello/api/v2/host_subscriptions/auto_attach • katello/api/v2/host_subscriptions/add_subscriptions • katello/api/v2/host_subscriptions/remove_subscriptions • katello/api/v2/host_subscriptions/content_override • katello/api/v2/hosts/bulk/add_host_collections • katello/api/v2/hosts/bulk/remove_host_collections • katello/api/v2/hosts/bulk/install_content	

权限名称	Actions	资源类型
	• katello/api/v2/hosts/bulk/update_content • katello/api/v2/hosts/bulk/remove_content	
destroy_hosts	• katello/api/v2/hosts/bulk/environment_content_view • hosts/destroy • hosts/multiple_actions • hosts/reset_multiple • hosts/multiple_destroy • hosts/submit_multiple_destroy • api/v2/hosts/destroy • api/v2/interfaces/destroy • katello/api/v2/hosts/bulk/destroy	
build_hosts	• hosts/setBuild • hosts/cancelBuild • hosts/multiple_build • hosts/submit_multiple_build • hosts/review_before_build • hosts/rebuild_config • hosts/submit_rebuild_config • tasks/show • api/v2/tasks/index • api/v2/hosts/rebuild_config	
power_hosts	• hosts/power • api/v2/hosts/power	
console_hosts	• hosts/console	

权限名称	Actions	资源类型
ipmi_boot	• hosts/ipmi_boot • api/v2/hosts/boot	
puppetrun_hosts	• hosts/puppetrun • hosts/multiple_puppetrun • hosts/update_multiple_puppetrun • api/v2/hosts/puppetrun	
search_repository_image_search	• image_search/auto_complete_repository_name • image_search/auto_complete_image_tag • image_search/search_repository	Docker/ImageSearch
view_images	• images/index • images/show • images/auto_complete_search • api/v2/images/index • api/v2/images/show	
create_images	• images/new • images/create • api/v2/images/create	
edit_images	• images/edit • images/update • api/v2/images/update	

权限名称	Actions	资源类型
destroy_images	• images/destroy • api/v2/images/destroy	
view_lifecycle_environments	• katello/api/v2/environments/index • katello/api/v2/environments/show • katello/api/v2/environments/paths • katello/api/v2/environments/repositories • katello/api/rhsm/candlepin_proxies/rhsm_index • katello/environments/auto_complete_search	Katello::KTEnvironment
create_lifecycle_environments	• katello/api/v2/environments/create	Katello::KTEnvironment
edit_lifecycle_environments	• katello/api/v2/environments/update	Katello::KTEnvironment
destroy_lifecycle_environments	• katello/api/v2/environments/destroy	Katello::KTEnvironment
promote_or_remove_content_views_to_environments		Katello::KTEnvironment
view_locations	• locations/index • locations/show • locations/auto_complete_search • api/v2/locations/index • api/v2/locations/show	

权限名称	Actions	资源类型
create_locations	• locations/new • locations/create • locations/clone_taxonomy • locations/step2 • locations/nest • api/v2/locations/create	
edit_locations	• locations/edit • locations/update • locations/import_mismatches • locations/parent_taxonomy_selected • api/v2/locations/update	
destroy_locations	• locations/destroy • api/v2/locations/destroy	
assign_locations	• locations/assign_all_hosts • locations/assign_hosts • locations/assign_selected_hosts	
view_mail_notifications	• mail_notifications/index • mail_notifications/auto_complete_search • mail_notifications/show • api/v2/mail_notifications/index • api/v2/mail_notifications/show	

权限名称	Actions	资源类型
view_media	• media/index • media/show • media/auto_complete_search • api/v2/media/index • api/v2/media/show	
create_media	• media/new • media/create • api/v2/media/create	
edit_media	• media/edit • media/update • api/v2/media/update	
destroy_media	• media/destroy • api/v2/media/destroy	
view_models	• models/index • models/show • models/auto_complete_search • api/v2/models/index • api/v2/models/show	
create_models	• models/new • models/create • api/v2/models/create	

权限名称	Actions	资源类型
edit_models	• models/edit • models/update • api/v2/models/update	
destroy_models	• models/destroy • api/v2/models/destroy	
view_operatingsystems	• operatingsystems/index • operatingsystems/show • operatingsystems/bootfiles • operatingsystems/auto_complete_search • api/v2/operatingsystems/index • api/v2/operatingsystems/show • api/v2/operatingsystems/bootfiles • api/v2/os_default_templates/index • api/v2/os_default_templates/show	
create_operatingsystems	• operatingsystems/new • operatingsystems/create • api/v2/operatingsystems/create • api/v2/os_default_templates/create	

权限名称	Actions	资源类型
edit_operatingsystems	• operatingsystems/edit • operatingsystems/update • api/v2/operatingsystems/update • api/v2/parameters/create • api/v2/parameters/update • api/v2/parameters/destroy • api/v2/parameters/reset • api/v2/os_default_templates/create • api/v2/os_default_templates/update • api/v2/os_default_templates/destroy	
destroy_operatingsystems	• operatingsystems/destroy • api/v2/operatingsystems/destroy • api/v2/os_default_templates/create	

权限名称	Actions	资源类型
view_organizations	• organizations/index • organizations/show • organizations/auto_complete_search • api/v2/organizations/index • api/v2/organizations/show • katello/api/v2/organizations/index • katello/api/v2/organizations/show • katello/api/v2/organizations/redhat_provider • katello/api/v2/organizations/download_debug_certificate • katello/api/v2/tasks/index	
create_organizations	• organizations/new • organizations/create • organizations/clone_taxonomy • organizations/step2 • organizations/nest • api/v2/organizations/create • katello/api/v2/organizations/create	

权限名称	Actions	资源类型
edit_organizations	• organizations/edit • organizations/update • organizations/import_mismatches • organizations/parent_taxonomy_selected • api/v2/organizations/update • katello/api/v2/organizations/update • katello/api/v2/organizations/autoattach_subscriptions	
destroy_organizations	• organizations/destroy • api/v2/organizations/destroy • katello/api/v2/organizations/destroy	
assign_organizations	• organizations/assign_all_hosts • organizations/assign_hosts • organizations/assign_selected_hosts	
view_ptables	• ptables/index • ptables/show • ptables/auto_complete_search • ptables/revision • ptables/preview • api/v2/ptables/show • api/v2/ptables/revision	

权限名称	Actions	资源类型
create_ptables	• ptables/new • ptables/create • ptables/clone_template • api/v2/ptables/create • api/v2/ptables/clone	
edit_ptables	• ptables/edit • ptables/update • api/v2/ptables/update	
destroy_ptables	• ptables/destroy • api/v2/ptables/destroy	
lock_ptables	• ptables/lock • ptables/unlock • api/v2/ptables/lock • api/v2/ptables/unlock	
view_plugins	• plugins/index • api/v2/plugins/index	
view_products	• katello/products/auto_complete • katello/products/auto_complete_search • katello/api/v2/products/index • katello/api/v2/products/show • katello/api/v2/repositories/index • katello/api/v2/repositories/show	Katello::Product

权限名称	Actions	资源类型
	• katello/api/v2/packages/index • katello/api/v2/packages/show • katello/api/v2/distributions/index • katello/api/v2/distributions/show • katello/api/v2/package_groups/index • katello/api/v2/package_groups/show • katello/api/v2/errata/index • katello/api/v2/errata/show • katello/api/v2/puppet_modules/index • katello/api/v2/puppet_modules/show • katello/errata/short_details • katello/errata/auto_complete • katello/packages/details • katello/packages/auto_complete • katello/puppet_modules/show • katello/repositories/auto_complete_library • katello/repositories/repository_types • katello/content_search/index • katello/content_search/products • katello/content_search/repos • katello/content_search/packages • katello/content_search/errata	

权限名称	Actions	资源类型
	• katello/content_search/puppet_modules • katello/content_search/packages_items • katello/content_search/errata_items • katello/content_search/puppet_modules_items • katello/content_search/repo_packages • katello/content_search/repo_errata • katello/content_search/repo_puppet_modules • katello/content_search/repo_compare_errata • katello/content_search/repo_compare_packages • katello/content_search/repo_compare_puppet_modules	
create_products	• katello/api/v2/products/create • katello/api/v2/repositories/create	Katello::Product

权限名称	Actions	资源类型
edit_products	• katello/api/v2/products/update • katello/api/v2/repositories/update • katello/api/v2/repositories/remove_content • katello/api/v2/repositories/import_uploads • katello/api/v2/repositories/upload_content • katello/api/v2/products_bulk_actions/update_sync_plans • katello/api/v2/content_uploads/create • katello/api/v2/content_uploads/update • katello/api/v2/content_uploads/destroy • katello/api/v2/organizations/repo_discover • katello/api/v2/organizations/cancel_repo_discover	Katello::Product
destroy_products	• katello/api/v2/products/destroy • katello/api/v2/repositories/destroy • katello/api/v2/products_bulk_actions/destroy_products • katello/api/v2/repositories_bulk_actions/destroy_repositories	Katello::Product

权限名称	Actions	资源类型
sync_products	• katello/api/v2/products/sync • katello/api/v2/repositories/sync • katello/api/v2/products_bulk_actions/sync_products • katello/api/v2/repositories_bulk_actions/sync_repositories • katello/api/v2/sync/index • katello/api/v2/sync_plans/sync • katello/sync_management/index • katello/sync_management/sync_status • katello/sync_management/product_status • katello/sync_management/sync • katello/sync_management/destroy	Katello::Product
export_products	• katello/api/v2/repositories/export	Katello::Product

权限名称	Actions	资源类型
view_provisioning_templates	• provisioning_templates/index • provisioning_templates/show • provisioning_templates/revision • provisioning_templates/auto_complete_search • provisioning_templates/preview • api/v2/provisioning_templates/index • api/v2/provisioning_templates/show • api/v2/provisioning_templates/revision • api/v2/template_combinations/index • api/v2/template_combinations/show • api/v2/template_kinds/index	
create_provisioning_templates	• provisioning_templates/new • provisioning_templates/create • provisioning_templates/clone_template • api/v2/provisioning_templates/create • api/v2/provisioning_templates/clone • api/v2/template_combinations/create	

权限名称	Actions	资源类型
edit_provisioning_templates	• provisioning_templates/edit • provisioning_templates/update • api/v2/provisioning_templates/update • api/v2/template_combinations/update	
destroy_provisioning_templates	• provisioning_templates/destroy • api/v2/provisioning_templates/destory • api/v2/template_combinations/destory	
deploy_provisioning_templates	• provisioning_templates/build_pxe_default • api/v2/provisioning_templates/build_pxe_default	
lock_provisioning_templates	• provisioning_templates/lock • provisioning_templates/unlock • api/v2/provisioning_templates/lock • api/v2/provisioning_templates/unlock	
user_logout	• users/logout	
my_account	• users/edit • katello/api/v2/tasks/show	

权限名称	Actions	资源类型
api_status	• api/v2/home/status/	
view_puppetclasses	• puppetclasses/index • puppetclasses/show • puppetclasses/auto_complete_search • api/v2/puppetclasses/index • api/v2/puppetclasses/show • api/v2/smart_variables/index • api/v2/smart_variables/show • api/v2/smart_class_parameters/index • api/v2/smart_class_parameters/show	
create_puppetclasses	• puppetclasses/new • puppetclasses/create • api/v2/puppetclasses/create	

权限名称	Actions	资源类型
edit_puppetclasses	• puppetclasses/edit • puppetclasses/update • puppetclasses/override • api/v2/puppetclasses/update • api/v2/smart_variables/create • api/v2/smart_variables/update • api/v2/smart_variables/destroy • api/v2/smart_class_parameters/create • api/v2/smart_class_parameters/update • api/v2/smart_class_parameters/destroy	
destroy_puppetclasses	• puppetclasses/destroy • api/v2/puppetclasses/destroy	
import_puppetclasses	• puppetclasses/import_environments • puppetclasses/obsolete_and_new • api/v2/environments/import_puppetclasses • api/v2/smart_proxies/import_puppetclasses	

权限名称	Actions	资源类型
view_realms	• realms/index • realms/show • realms/auto_complete_search • api/v2/realms/index • api/v2/realms/show	
create_realms	• realms/new • realms/create • api/v2/realms/create	
edit_realms	• realms/edit • realms/update • api/v2/realms/update	
destroy_realms	• realms/destroy • api/v2/realms/destroy	
view_search	• redhat_access/search/index	
view_cases	• redhat_access/cases/index • redhat_access/cases/create	
attachments	• redhat_access/attachments/index • redhat_access/attachments/create	

权限名称	Actions	资源类型
配置	• redhat_access/configuration/index	
app_root	• redhat_access/redhat_access/index	
view_log_viewer	• redhat_access/logviewer/index	
logs	• redhat_access/logs/index	
rh_telemetry_api	• redhat_access/api/telemetry_api/proxy • redhat_access/api/telemetry_api/connection_status	
rh_telemetry_view	• redhat_access/analytics_dashboard/index	
rh_telemetry_configurations	• redhat_access/telemetry_configurations/show • redhat_access/telemetry_configurations/update	
view_roles	• roles/index • roles/auto_complete_search • api/v2/roles/index • api/v2/roles/show	

权限名称	Actions	资源类型
create_roles	• roles/new • roles/create • roles/clone • api/v2/roles/create	
edit_roles	• roles/edit • roles/update • api/v2/roles/update	
destroy_roles	• roles/destroy • api/v2/roles/destroy	
access_settings	• home/settings	

权限名称	Actions	资源类型
view_smart_proxies	• smart_proxies/index • smart_proxies/ping • smart_proxies/auto_complete_search • smart_proxies/version • smart_proxies/show • smart_proxies/plugin_version • smart_proxies/tftp_server • smart_proxies/puppet_environments • smart_proxies/puppet_dashboard • smart_proxies/log_pane • smart_proxies/failed_modules • smart_proxies/errors_card • smart_proxies/modules_card • api/v2/smart_proxies/index • api/v2/smart_proxies/show • api/v2/smart_proxies/version • api/v2/smart_proxies/log	
create_smart_proxies	• smart_proxies/new • smart_proxies/create • api/v2/smart_proxies/create	

权限名称	Actions	资源类型
edit_smart_proxies	• smart_proxies/edit • smart_proxies/update • smart_proxies/refresh • smart_proxies/expire_logs • api/v2/smart_proxies/update • api/v2/smart_proxies/refresh	
destroy_smart_proxies	• smart_proxies/destroy • api/v2/smart_proxies/destroy	
view_smart_proxies_autosign	• autosign/index • autosign/show • autosign/counts • api/v2/autosign/index	
create_smart_proxies_autosign	• autosign/new • autosign/create	
destroy_smart_proxies_autosign	• autosign/destroy	
view_smart_proxies_puppetca	• puppetca/index • puppetca/counts • puppetca/expiry	
edit_smart_proxies_puppetca	• puppetca/update	
destroy_smart_proxies_puppetca	• puppetca/destroy	

权限名称	Actions	资源类型
view_subnets	• subnets/index • subnets/show • subnets/auto_complete_search • api/v2/subnets/index • api/v2/subnets/show	
create_subnets	• subnets/new • subnets/create • api/v2/subnets/create	
edit_subnets	• subnets/edit • subnets/update • api/v2/subnets/update	
destroy_subnets	• subnets/destroy • api/v2/subnets/destroy	
import_subnets	• subnets/import • subnets/create_multiple	

权限名称	Actions	资源类型
view_subscriptions	• katello/api/v2/subscriptions/index • katello/api/v2/subscriptions/show • katello/api/v2/subscriptions/available • katello/api/v2/subscriptions/manifest_history • katello/api/v2/subscriptions/auto_complete_search • katello/api/v2/repository_sets/index • katello/api/v2/repository_sets/show • katello/api/v2/repository_sets/available_repositories	机构（Organization）
attach_subscriptions	• katello/api/v2/subscriptions/create	机构（Organization）
unattach_subscriptions	• katello/api/v2/subscriptions/destroy	机构（Organization）

权限名称	Actions	资源类型
import_manifest	• katello/products/available_repositories • katello/products/toggle_repository • katello/providers/redhat_provider • katello/providers/redhat_provider_tab • katello/api/v2/subscriptions/upload • katello/api/v2/subscriptions/refresh_manifest • katello/api/v2/repository_sets/enable • katello/api/v2/repository_sets/disable	机构（Organization）
delete_manifest	• katello/api/v2/subscriptions/delete_manifest	机构（Organization）
view_sync_plans	• katello/sync_plans/all • katello/sync_plans/index • katello/sync_plans/auto_complete_search • katello/api/v2/sync_plans/index • katello/api/v2/sync_plans/show • katello/api/v2/sync_plans/add_products • katello/api/v2/sync_plans/remove_products • katello/api/v2/sync_plans/available_products • katello/api/v2/products/index	Katello::SyncPlan

权限名称	Actions	资源类型
create_sync_plans	• katello/api/v2/sync_plans/create	Katello::SyncPlan
edit_sync_plans	• katello/api/v2/sync_plans/update	Katello::SyncPlan
destroy_sync_plans	• katello/api/v2/sync_plans/destroy	Katello::SyncPlan
	my_organizations	• katello/api/rhsm/candlepin_proxies/list_owners
	view_usergroups	• usergroups/index • usergroups/show • usergroups/auto_complete_search • api/v2/usergroups/index • api/v2/usergroups/show
	create_usergroups	• usergroups/new • usergroups/create • api/v2/usergroups/create
	edit_usergroups	• usergroups/edit • usergroups/update • api/v2/usergroups/update
	destroy_usergroups	• usergroups/destroy • api/v2/usergroups/destroy

权限名称	Actions	资源类型
	view_users	• users/index • users/show • users/auto_complete_search • api/v2/users/index • api/v2/users/show
	create_users	• users/new • users/create • users/auth_source_selected • api/v2/users/create
	edit_users	• users/edit • users/update • users/auth_source_selected • users/test_mail • api/v2/users/update
	destroy_users	• users/destroy • api/v2/users/destroy