Red Hat Enterprise Linux 9

使用 IdM API

通过 Python 脚本使用 IdM API

Last Updated: 2024-06-26

Red Hat Enterprise Linux 9 使用 IdM API

通过 Python 脚本使用 IdM API

法律通告

摘要

IdM API 包含使用各种类型请求的示例。管理员和开发人员可以使用 IdM API 用 Python 编写自定义脚本，来将 IdM 与第三方应用程序集成。

目录

对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

通过 Jira 提交反馈（需要帐户）

1. 登录到 Jira 网站。
2. 在顶部导航栏中点 **Create**
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您对改进的建议。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

第 1 章 IDM API 简介

您可以使用命令行和基于 Web 的界面访问红帽身份管理的服务。使用身份管理 API，您可以通过第三方应用程序和用 Python 编写的脚本来与身份管理服务进行交互。

身份管理 API 具有 JavaScript Object Notation Remote Procedure Call (JSON-RPC)接口。要对各种重要部分使用自动化，请通过 Python 访问身份管理 API。例如，您可以使用所有可用的命令从服务器检索元数据。

第 2 章 IDM API 的基础知识

您可以使用 IdM API，使用自定义脚本自动访问 IdM 环境。

2.1. 初始化 IDM API

要使用 IdM API，首先在您的环境中初始化它。

先决条件

- IdM 服务器或 IdM 客户端软件包已安装。
- 一个有效的 Kerberos 票据已发布。

流程

1. 要初始化 IdM API，请在脚本的开头包含以下代码：

```
from ipalib import api

api.bootstrap(context="server")
api.finalize()
```

2. 要与 LDAP 服务器建立连接，请在 API 初始化后在脚本中添加以下逻辑：

```
if api.env.in_server:
    api.Backend.ldap2.connect()
else:
    api.Backend.rpcclient.connect()
```

   - 如果您在 IdM 服务器上运行脚本，则此逻辑允许您的脚本直接连接到 LDAP 服务器。
   - 如果您在 IdM 客户端上运行脚本，则脚本将使用远程过程调用(RPC)客户端。

其他资源

- IdM API 上下文

2.2. 运行 IDM API 命令

您可以在脚本中运行 IdM API 命令。要运行 IdM API 命令，请在脚本中使用 **api.Command** 结构。

先决条件

- IdM API 已初始化。如需更多信息，请参阅 初始化 IdM API。

流程

- 例如，要列出有关用户的信息，请在脚本中包含以下代码：

```
api.Command.user_show("user_name", no_members=True, all=True)
```

  在本例中，您还将参数和选项传给命令 **user_show**。

其他资源

- 有关 **api.Command** 命令的完整列表，请参阅 IPA API 命令 web 源。

2.3. IDM API 命令输出结构

每个 IdM API 命令的输出都有四个部分。这些部分包含有关命令执行的各种信息。

IdM API 输出结构

result

这部分提供命令的结果。它包含有关命令操作的各种详细信息，如传递给命令的选项和参数。

values

这部分表示命令的参数。

messages

这部分显示在执行命令后 **ipa** 工具提供的各种信息。

summary

这部分显示操作的总结。

在本例中，您的脚本执行 **add_user** 命令：

```
api.Command.user_add("test", givenname="a", sn="b")
```

该命令的输出结构如下：

```
{
    "result": {
        "displayname": ["a b"],
        "objectclass": [
            "top",
            "person",
            "organizationalperson",
            "inetorgperson",
            "inetuser",
            "posixaccount",
            "krbprincipalaux",
            "krbticketpolicyaux",
            "ipaobject",
            "ipasshuser",
            "ipaSshGroupOfPubKeys",
            "mepOriginEntry",
            "ipantuserattrs",
        ],
        "cn": ["a b"],
        "gidnumber": ["1445000004"],
        "mail": ["test@ipa.test"],
        "krbprincipalname": [ipapython.kerberos.Principal("test@IPA.TEST")],
        "loginshell": ["/bin/sh"],
        "initials": ["ab"],
        "uid": ["test"],
        "uidnumber": ["1445000004"],
        "sn": ["b"],
```

```
        "krbcanonicalname": [ipapython.kerberos.Principal("test@IPA.TEST")],
        "homedirectory": ["/home/test"],
        "givenname": ["a"],
        "gecos": ["a b"],
        "ipauniqueid": ["9f9c1df8-5073-11ed-9a56-fa163ea98bb3"],
        "mepmanagedentry": [
            ipapython.dn.DN("cn=test,cn=groups,cn=accounts,dc=ipa,dc=test")
        ],
        "has_password": False,
        "has_keytab": False,
        "memberof_group": ["ipausers"],
        "dn": ipapython.dn.DN("uid=test,cn=users,cn=accounts,dc=ipa,dc=test"),
    },
    "value": "test",
    "messages": [
        {
            "type": "warning",
            "name": "VersionMissing",
            "message": "API Version number was not sent, forward compatibility not guaranteed.
Assuming server's API version, 2.248",
            "code": 13001,
            "data": {"server_version": "2.248"},
        }
    ],
    "summary": 'Added user "test"',
}
```

2.4. 列出 IDM API 命令和参数

您可以使用 **command_show** 和 **param_show** 来列出 IdM API 命令及其参数的信息。

先决条件

- IdM API 已初始化。如需更多信息，请参阅 初始化 IdM API。

流程

1. 要显示 **user_add** 命令的信息，请执行以下代码：

```
api.Command.command_show("user_add")
```

这个命令的结果如下：

```
{
    "result": {
        "name": "user_add",
        "version": "1",
        "full_name": "user_add/1",
        "doc": "Add a new user.",
        "topic_topic": "user/1",
        "obj_class": "user/1",
        "attr_name": "add",
    },
    "value": "user_add",
```

```
    "messages": [
        {
            "type": "warning",
            "name": "VersionMissing",
            "message": "API Version number was not sent, forward compatibility not guaranteed.
Assuming server's API version, 2.251",
            "code": 13001,
            "data": {"server_version": "2.251"},
        }
    ],
    "summary": None,
}
```

2. 要显示 **user_add** 命令的 **givenname** 参数的信息，请执行以下代码：

```
api.Command.param_show("user_add", name="givenname")
```

这个命令的结果如下：

```
{
    "result": {
        "name": "givenname",
        "type": "str",
        "positional": False,
        "cli_name": "first",
        "label": "First name",
    },
    "value": "givenname",
    "messages": [
        {
            "type": "warning",
            "name": "VersionMissing",
            "message": "API Version number was not sent, forward compatibility not guaranteed.
Assuming server's API version, 2.251",
            "code": 13001,
            "data": {"server_version": "2.251"},
        }
    ],
    "summary": None,
}
```

2.5. 使用批处理执行 IDM API 命令

您可以使用 **batch** 命令,通过一个调用执行多个 IdM API 命令。以下示例演示了如何创建多个 IdM 用户。

先决条件

- IdM API 已初始化。如需更多信息，请参阅 初始化 IdM API。

流程

- 要在一个批处理中创建 100 个 IdM 用户，请在脚本中包含以下代码：

```
batch_args = []
```

```
for i in range(100):
    user_id = "user%i" % i
    args = [user_id]
    kw = {
        'givenname' : user_id,
        'sn' : user_id
    }
    batch_args.append({
        'method' : 'user_add',
        'params' : [args, kw]
    })
ret = api.Command.batch(*batch_args)
```

2.6. IDM API 上下文

IdM API 上下文决定了 API 使用哪个插件。在 API 初始化期间指定上下文。有关如何使用 IdM API 上下文的示例，请参阅 初始化 IdM API。

IdM API 上下文

server

插件集合，其验证传递给 IdM API 命令执行的参数和选项。

client

插件集合，其验证转发给 IdM 服务器执行的参数和选项。

installer

特定于安装过程的插件集合。

updates

特定于更新过程的插件集合。

第 3 章 IDM API 和 IDM CLI 命令比较

您可以在 Python 交互控制台中使用 IdM API 命令。IdM API 命令与 **ipa** 工具命令不同。

IdM CLI 和 IdM API 命令的区别

命令命名结构

ipa CLI 命令使用连字符，如 **user-add**，但 IdM API 命令改为使用下划线，如 **user_add**。

参数命名

IdM CLI 命令和 IdM API 命令的参数不同。例如，IdM CLI **user-add** 命令首先有一个参数 **first**，但 IdM API **user_add** 命令有一个参数 **givenname**。

日期格式

IdM CLI 提供了以下日期格式：

- **%Y%m%d%H%M%SZ**
- **%Y-%m-%dT%H:%M:%SZ**
- **%Y-%m-%dT%H:%MZ**
- **%Y-%m-%dZ**
- **%Y-%m-%d %H:%M:%SZ**
- **%Y-%m-%d %H:%MZ**
  另外，IdM API 可以使用 Python 内置类 **datetime**。

有用的 CLI 工具

- **console** 启动交互式 Python 控制台，您可以使用它来运行 IdM API 命令。
- **help** 命令显示主题和命令的描述，并包含各种示例。
- **show-mapping** 命令显示 CLI 参数名称和 LDAP 属性之间的映射。

第 4 章 IDM API 示例场景

以下示例为您提供了使用 IdM API 命令的常见场景。

4.1. 使用 IDM API 命令管理用户

以下示例显示了如何使用 IdM API 命令管理 IdM 用户的常见场景。

使用 IdM API 命令管理 IdM 用户的示例

创建 IdM 用户

在本例中，您可以使用用户名 **exampleuser** 和支持的用户 **一次性密码(OTP)** 身份验证创建一个 IdM 用户。

```
api.Command.user_add("exampleuser", givenname="Example", sn="User",
ipauserauthtype="otp")
```

显示 IdM 用户信息

在本例中，您显示有关 IdM 用户 **exampleuser** 的所有可用信息。

```
api.Command.user_show("exampleuser", all=True)
```

修改 IdM 用户

在本例中，您将更改 IdM 用户 **exampleuser** 的电子邮件地址。

```
api.Command.user_mod("exampleuser", mail="exampleuser@example.org")
```

搜索 IdM 用户

在本例中，您将搜索与 IdM 组 **admins** 中 **exampleuser** 匹配的所有 IdM 用户。

```
api.Command.user_find(criteria="exampleuser", in_group="admins")
```

删除 IdM 用户

在本例中，您将删除 IdM 用户 **exampleuser**。

```
api.Command.user_del("exampleuser")
```

要在以后恢复用户，请使用 **preserve** 选项。如果使用这个选项，您可以使用 **user_undel** 命令恢复用户。

为 IdM 用户添加和删除证书

您可以使用 **user_add_cert** 和 **user_remove_cert** 命令为用户添加或删除 **Base64 编码的** 证书。在本例中，您为用户 **exampleuser** 添加了一个证书。

```
args = ["exampleuser"]
kw = {
    "usercertificate": """

MIICYzCCAcygAwIBAgIBADANBgkqhkiG9w0BAQUFADAuMQswCQYDVQQGEwJVUzEMMAoGA
1UEC
```

```
hMDSUJNMREwDwYDVQQLEwhMb2NhbCBDQTAeFw05OTEyMjIwNTAwMDBaFw0wMDEyMjMwNDU5NT

laMC4xCzAJBgNVBAYTAlVTMQwwCgYDVQQKEwNJQk0xETAPBgNVBAsTCExvY2FsIENBMIGfMA0

GCSqGSIb3DQEBATOPA4GNADCBiQKBgQD2bZEo7xGaX2/0GHkrNFZvlxBou9v1Jmt/PDiTMPve

    8r9FeJAQ0QdvFST/0JPQYD20rH0bimdDLgNdNynmyRoS2S/IInfpmf69iyc2G0TPyRvmHIiOZ

bdCd+YBHQi1adkj17NDcWj6S14tVurFX73zx0sNoMS79q3tuXKrDsxeuwIDAQABo4GQMIGNME

sGCVUdDwGG+EIBDQQ+EzxHZW5lcmF0ZWQgYnkgdGhlIFNlY3VyZVdheSBTZWN1cml0eSBTZXJ

2ZXIgZm9yIE9TLzM5MCAoUkFDRikwDgYDVR0PAQH/BAQDAgAGMA8GA1UdEwEB/wQFMAMBAf8w

HQYDVR0OBBYEFJ3+ocRyCTJw067dLSwr/nalx6YMMA0GCSqGSIb3DQEBBQUAA4GBAMaQzt+za
    j1GU77yzlr8iiMBXgdQrwsZZWJo5exnAucJAEYQZmOfyLiMD6oYq+ZnfvM0n8G/Y79q8nhwvu
    xpYOnRSAXFp6xSkrIOeZtJMY1h00LKp/JX3Ng1svZ2agE126JHsQ0bhzN5TKsYfbwfTwfjdWA
    Gy6Vf1nYi/rO+ryMO
  """
}

api.Command.user_add_cert(*args, **kw)
```

启用和禁用 IdM 用户

您可以使用 **user_enable** 和 **user_disable** 命令启用或禁用 IdM 用户。在本例中，您禁用了 IdM 用户 **exampleuser**。

```
api.Command.user_disable("exampleuser")
```

4.2. 使用 IDM API 命令管理组

以下示例显示了如何使用 IdM API 命令管理 IdM 组的常见场景。

使用 IdM API 命令管理 IdM 用户的示例

创建 IdM 组

在本例中，您可以创建一个具有指定组 ID 号 的 IdM 组 **developers**。

```
api.Command.group_add("developers", gidnumber=500, description="Developers")
```

将用户作为成员添加到 IdM 组

在本例中，您将 **admin** 用户添加到 **developers** 组。

```
api.Command.group_add_member("developers", user="admin")
```

将服务作为成员添加到 IdM 组

在本例中，您要将 **HTTP/server.ipa.test** 服务添加到 **developers** 组。

```
api.Command.group_add_member("developers", service="HTTP/server.ipa.test")
```

将组作为子组添加到 IdM 组

在本例中，您将另一个组 **admins** 添加到 **developers** 组。

```
api.Command.group_add_member("developers", group="admins")
```

添加 IdM 组管理者

在这个示例中，您将 **bob** 用户添加为 **developers** 组的组管理者。

```
api.Command.group_add_member_manager("developers", user="bob")
```

查找 IdM 组

您可以使用各种参数搜索 IdM 组。在这个示例中，您查找到用户 **bob** 管理的所有组。

```
api.Command.group_find(membermanager_user="bob")
```

显示 IdM 组信息

在本例中，您显示有关 **developers** 组的组信息，而不显示成员列表。

```
api.Command.group_show("developers", no_members=True)
```

修改 IdM 组

在本例中，您将非 POSIX 组 **testgroup** 转换为 POSIX 组。

```
api.Command.group_mod("testgroup", posix=True)
```

从 IdM 组中删除成员

在本例中，您从 **developers** 组中删除 **admin** 用户。

```
api.Command.group_remove_member("developers", user="admin")
```

删除 IdM 组管理者

在这个示例中，您将用户 **bob** 作为管理器者从 **developers** 组中删除。

```
api.Command.group_remove_member_manager("developers", user="bob")
```

删除 IdM 组

在本例中，您删除 **developers** 组。

```
api.Command.group_del("developers")
```

4.3. 使用 IDM API 命令管理访问控制

以下示例显示了如何使用 IdM API 命令管理访问控制的常见场景。

使用 IdM API 命令管理访问控制的示例

添加创建用户的权限

在本例中，添加创建用户的权限。

```
api.Command.permission_add("Create users", ipapermright='add', type='user')
```

添加管理组成员资格的权限

在本例中，您添加将用户添加到组的权限。

```
api.Command.permission_add("Manage group membership", ipapermright='write', type='group',
attrs="member")
```

为用户创建进程添加特权

在本例中，您添加了创建用户的特权，将它们添加到组，以并管理用户证书。

```
api.Command.permission_add("Create users", ipapermright='add', type='user')
api.Command.permission_add("Manage group membership", ipapermright='write', type='group',
attrs="member")
api.Command.permission_add("Manage User certificates", ipapermright='write', type='user',
attrs='usercertificate')

api.Command.privilege_add("User creation")
api.Command.privilege_add_permission("User creation", permission="Create users")
api.Command.privilege_add_permission("User creation", permission="Manage group
membership")
api.Command.privilege_add_permission("User creation", permission="Manage User certificates")
```

使用特权添加角色

在本例中，您要使用上例中创建的特权添加角色。

```
api.Command.role_add("usermanager", description="Users manager")
api.Command.role_add_privilege("usermanager", privilege="User creation")
```

向用户分配角色

在本例中，您将 **usermanager** 角色分配给用户 **bob**。

```
api.Command.role_add_member("usermanager", user="bob")
```

向组分配角色

在本例中，您将 **usermanager** 角色分配给 **managers** 组。

```
api.Command.role_add_member("usermanager", group="managers")
```

4.4. 使用 IDM API 命令管理 SUDO 规则

以下示例显示了如何使用 IdM API 命令管理 sudo 规则的常见场景。

使用 IdM API 命令管理 sudo 规则的示例

创建 sudo 规则

在本例中，您可以创建一个保持时间更改命令的 sudo 规则。

```
api.Command.sudorule_add("timechange")
```

创建 sudo 命令

在本例中，您创建 **date** sudo 命令。

```
api.Command.sudocmd_add("/usr/bin/date")
```

将 sudo 命令附加到 sudo 规则

在本例中，您要将 **date** sudo 命令附加到 **timechange** sudo 规则。

```
api.Command.sudorule_add_allow_command("timechange", sudocmd="/usr/bin/date")
```

创建并附加 sudo 命令组

在本例中，您创建多个 sudo 命令，将它们添加到新创建的 **timecmds** sudo 命令组中，并将组附加到 **timechange** sudo 规则。

```
api.Command.sudocmd_add("/usr/bin/date")
api.Command.sudocmd_add("/usr/bin/timedatectl")
api.Command.sudocmd_add("/usr/sbin/hwclock")
api.Command.sudocmdgroup_add("timecmds")
api.Command.sudocmdgroup_add_member("timecmds", sudocmd="/usr/bin/date")
api.Command.sudocmdgroup_add_member("timecmds", sudocmd="/usr/bin/timedatectl")
api.Command.sudocmdgroup_add_member("timecmds", sudocmd="/usr/sbin/hwclock")
api.Command.sudorule_add_allow_command("timechange", sudocmdgroup="timecmds")
```

拒绝 sudo 命令

在本例中，您拒绝要作为 sudo 运行的 **rm** 命令。

```
api.Command.sudocmd_add("/usr/bin/rm")
api.Command.sudorule_add_deny_command("timechange", sudocmd="/usr/bin/rm")
```

将用户添加到 sudo 规则

在这个示例中，您将用户 **bob** 添加到 **timechange** sudo 规则中。

```
api.Command.sudorule_add_user("timechange", user="bob")
```

使 sudo 规则仅对指定的主机可用

在本例中，您将 **timechange** 规则限制为仅对 **client.ipa.test** 主机可用。

```
api.Command.sudorule_add_host("timechange", host="client.ipa.test")
```

将 sudo 规则设置为以不同用户身份运行

默认情况下，sudo 规则以 **root** 用户身份运行。在本例中，您将 **timechange** sudo 规则设置为以 **alice** 用户身份运行。

```
api.Command.sudorule_add_runasuser("timechange", user="alice")
```

将 sudo 规则设置为以组身份运行

在本例中，您将 **timechange** sudo 规则设置为以 **sysadmins** 组身份运行。

```
api.Command.sudorule_add_runasgroup("timechange", group="sysadmins")
```

为 sudo 规则设置 sudo 选项

在本例中，您为 **timechange** sudo 规则设置 sudo 选项。

```
api.Command.sudorule_add_option("timechange", ipasudoopt="logfile='/var/log/timechange_log'")
```

启用 sudo 规则

在本例中，您启用 **timechange** sudo 规则。

```
api.Command.sudorule_enable("timechange")
```

禁用 sudo 规则

在本例中，您禁用 **timechange** sudo 规则。

```
api.Command.sudorule_disable("timechange")
```

4.5. 使用 IDM API 命令管理基于主机的访问控制

以下示例显示了如何使用 IdM API 命令管理基于主机的访问控制(HBAC)的常见场景。

使用 IdM API 命令管理 HBAC 的示例

创建 HBAC 规则

在本例中，您创建一个将处理 SSH 服务访问的基本规则。

```
api.Command.hbacrule_add("sshd_rule")
```

将用户添加到 HBAC 规则中

在本例中，您将用户 **john** 添加到 **sshd_rule** HBAC 规则中。

```
api.Command.hbacrule_add_user("sshd_rule", user="john")
```

将组添加到 HBAC 规则中

在本例中，您将组 **developers** 添加到 **sshd_rule** HBAC 规则中。

```
api.Command.hbacrule_add_user("sshd_rule", group="developers")
```

从 HBAC 规则中删除用户

在本例中，您从 **sshd_rule** HBAC 规则中删除用户 **john**。

```
api.Command.hbacrule_remove_user("sshd_rule", user="john")
```

注册新的目标 HBAC 服务

在将其附加到 HBAC 规则前，您必须注册一个目标服务。在本例中，您注册 **chronyd** 服务。

```
api.Command.hbacsvc_add("chronyd")
```

将注册的服务附加到 HBAC 规则中

在本例中，您将 **sshd** 服务附加到 **sshd_rule** HBAC 规则中。默认情况下，该服务在 IPA 中注册，因此无需事先使用 **hbacsvc_add** 进行注册。

```
api.Command.hbacrule_add_service("sshd_rule", hbacsvc="sshd")
```

将主机添加到 HBAC 规则中

在本例中，您要 **workstations** 主机组添加到 **sshd_rule** HBAC 规则中。

```
api.Command.hbacrule_add_host("sshd_rule", hostgroup="workstations")
```

测试 HBAC 规则

在本例中，您对 **workstation.ipa.test** 主机使用 **sshd_rule** HBAC 规则。它以来自用户 **john** 的服务 **sshd** 为目标。

```
api.Command.hbactest(user="john", targethost="workstation.ipa.test", service="sshd",
rules="sshd_rule")
```

启用 HBAC 规则

在本例中，您启用 **sshd_rule** HBAC 规则。

```
api.Command.hbacrule_enable("sshd_rule")
```

禁用 HBAC 规则

在本例中，您禁用 **sshd_rule** HBAC 规则。

```
api.Command.hbacrule_disable("sshd_rule")
```