Red Hat Enterprise Linux 9

IdM API の使用

Python スクリプトによる IdM API の使用

Last Updated: 2024-05-02

Red Hat Enterprise Linux 9 IdM API の使用

Python スクリプトによる IdM API の使用

法律上の通知

概要

IdM API には、さまざまなタイプのリクエストを使用するための例が含まれています。管理者と開発者は、IdM API を使用して Python でカスタムスクリプトを作成し、IdM をサードパーティーアプリケーションと統合できます。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ を参照してください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに関するご意見やご感想をお寄せください。また、改善点があればお知らせください。

Jira からのフィードバック送信 (アカウントが必要)

1. Jira の Web サイトにログインします。
2. 上部のナビゲーションバーで **Create** をクリックします。
3. **Summary** フィールドにわかりやすいタイトルを入力します。
4. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも追加してください。
5. ダイアログの下部にある **Create** をクリックします。

第1章 IDM API の概要

Red Hat Identity Management のサービスには、コマンドラインおよび Web ベースのインターフェイスを使用してアクセスできます。Identity Management API を使用すると、Python で記述されたサードパーティーのアプリケーションやスクリプトを通じて Identity Management サービスと対話できます。

Identity Management API は、JavaScript Object Notation Remote Procedure Call (JSON-RPC) インターフェイスを備えています。さまざまな重要な部分で自動化を使用するには、Python を介して Identity Management API にアクセスします。たとえば、利用可能なすべてのコマンドを使用してサーバーからメタデータを取得できます。

第2章 IDM API の基本

IdM API を使用すると、カスタムスクリプトにより IdM 環境へのアクセスを自動化できます。

2.1. IDM API の初期化

IdM API を使用するには、まず環境内で IdM API を初期化します。

前提条件

- IdM サーバーまたは IdM クライアントパッケージがインストールされている。
- 有効な Kerberos チケットが発行されている。

手順

1. IdM API を初期化するには、スクリプトの先頭に次のコードを含めます。

```
from ipalib import api

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

2. LDAP サーバーとの接続を確立するには、API の初期化後に次のロジックをスクリプトに追加します。

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

   - IdM サーバーでスクリプトを実行する場合、このロジックにより、スクリプトが LDAP サーバーに直接接続できるようになります。
   - IdM クライアントでスクリプトを実行する場合、スクリプトはリモートプロシージャコール (RPC) クライアントを使用します。

関連情報

- IdM API コンテキスト

2.2. IDM API コマンドの実行

スクリプト内で IdM API コマンドを実行できます。IdM API コマンドを実行するには、スクリプトで **api.Command** 構造を使用します。

前提条件

- IdM API が初期化されている。詳細は、IdM API の初期化 を参照してください。

手順

- たとえば、ユーザーに関する情報をリスト表示するには、スクリプトに次のコードを含めます。

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

この例では、引数とオプションもコマンド **user_show** に渡しています。

関連情報

- **api.Command** コマンドの完全なリストについては、IPA API Commands Web ソースを参照してください。

2.3. IDM API コマンドの出力構造

各 IdM API コマンドの出力には、4 つのセクションがあります。これらのセクションには、コマンドの実行に関するさまざまな情報が含まれています。

IdM API の出力構造

result
このセクションには、コマンドの結果が示されます。これには、コマンドに渡されたオプションや引数など、コマンド操作に関するさまざまな詳細が含まれます。

values
このセクションには、コマンドの引数が示されます。

messages
このセクションには、コマンド実行後に **ipa** ツールが提供するさまざまな情報が示されます。

summary
このセクションには、操作の概要が示されます。

この例では、スクリプトは **add_user** コマンドを実行します。

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

このコマンドの出力構造は次のとおりです。

```
{
    "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. IDM API コマンドとパラメーターのリスト

コマンド **command_show** および **param_show** を使用すると、IdM API コマンドとそのパラメーターに関する情報をリスト表示できます。

前提条件

- IdM API が初期化されている。詳細は、IdM API の初期化 を参照してください。

手順

1. **user_add** コマンドに関する情報を表示するには、次のコードを実行します。

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

このコマンドの結果は次のようになります。

```
{
    "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. **user_add** コマンドの **givenname** パラメーターに関する情報を表示するには、次のコードを実行します。

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

このコマンドの結果は次のようになります。

```
{
    "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. バッチを使用した IDM API コマンドの実行

batch コマンドを使用すると、複数の IdM API コマンドを 1 回の呼び出しで実行できます。次の例は、複数の IdM ユーザーを作成する方法を示しています。

前提条件

- IdM API が初期化されている。詳細は、IdM API の初期化 を参照してください。

手順

- 1 つのバッチで 100 人の IdM ユーザーを作成するには、次のコードをスクリプトに含めます。

```
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 コンテキスト

IdM API コンテキストは、API が使用するプラグインを決定するものです。コンテキストは API の初期化中に指定します。IdM API コンテキストの使用方法の例は、IdM API の初期化 を参照してください。

IdM API コンテキスト

server
実行のために IdM API コマンドに渡される引数とオプションを検証するプラグインのセット。

client
実行のために IdM サーバーに転送される引数とオプションを検証するプラグインのセット。

installer
インストールプロセスに固有のプラグインのセット。

updates
更新プロセスに固有のプラグインのセット。

第3章 IDM API と IDM CLI コマンドの比較

IdM API コマンドは、Python 対話型コンソールで使用できます。IdM API コマンドは、**ipa** ツールのコマンドとは異なります。

IdM CLI と IdM API コマンドの違い

コマンドの命名構造

ipa CLI コマンドでは、**user-add** のようにハイフンが使用されますが、IdM API コマンドでは、**user_add** のように、代わりにアンダースコアが使用されます。

パラメーターの命名

パラメーターは、IdM CLI コマンドと IdM API コマンドで異なります。たとえば、IdM CLI **user-add** コマンドには **first** パラメーターがありますが、IdM API **user_add** コマンドには **givenname** パラメーターがあります。

日付書式

IdM CLI では次の日付形式を使用できます。

- **%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**
  さらに、IdM API では Python のビルトインクラス **datetime** を使用できます。

便利な CLI ツール

- **console** は対話型の Python コンソールを起動します。これを使用すると、IdM API コマンドを実行できます。
- **help** コマンドは、さまざまな例を含むトピックとコマンドの説明を表示します。
- **show-mapping** コマンドは、CLI パラメーター名と LDAP 属性の間のマッピングを表示します。

第4章 IDM API のサンプルシナリオ

次の例では、IdM API コマンドを使用する一般的なシナリオを示します。

4.1. IDM API コマンドを使用したユーザーの管理

以下の例は、IdM API コマンドを使用して IdM ユーザーを管理する方法の一般的なシナリオを示しています。

IdM API コマンドを使用した IdM ユーザーの管理の例

IdM ユーザーの作成

この例では、ユーザー名 **exampleuser** とサポートされているユーザーの **one-time password (OTP)** 認証を使用して IdM ユーザーを作成します。

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

IdM ユーザー情報の表示

この例では、IdM ユーザー **exampleuser** に関する入手可能な情報をすべて表示します。

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

IdM ユーザーの変更

この例では、IdM ユーザー **exampleuser** のメールアドレスを変更します。

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

IdM ユーザーの検索

この例では、IdM グループ **admins** 内の **exampleuser** に一致するすべての IdM ユーザーを検索します。

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

IdM ユーザーの削除

この例では、IdM ユーザー **exampleuser** を削除します。

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

後でユーザーを復元するには、**preserve** オプションを使用します。このオプションを使用すると、**user_undel** コマンドでユーザーを復元できます。

IdM ユーザーの証明書の追加と削除

user_add_cert および **user_remove_cert** コマンドを使用すると、ユーザーの **Base64 encoded** 証明書を追加または削除できます。この例では、ユーザー **exampleuser** の証明書を追加します。

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

```
MIICYzCCAcygAwIBAgIBADANBgkqhkiG9w0BAQUFADAuMQswCQYDVQQGEwJVUzEMMAoGA
1UEC

hMDSUJNMREwDwYDVQQLEwhMb2NhbCBDQTAeFw05OTEyMjIwNTAwMDBaFw0wMDEyMjM
wNDU5NT

laMC4xCzAJBgNVBAYTAlVTMQwwCgYDVQQKEwNJQk0xETAPBgNVBAsTCExvY2FsIENBMIGf
MA0

GCSqGSIb3DQEBATOPA4GNADCBiQKBgQD2bZEo7xGaX2/0GHkrNFZvlxBou9v1Jmt/PDiTMPve

    8r9FeJAQ0QdvFST/0JPQYD20rH0bimdDLgNdNynmyRoS2S/IInfpmf69iyc2G0TPyRvmHIiOZ

bdCd+YBHQi1adkj17NDcWj6S14tVurFX73zx0sNoMS79q3tuXKrDsxeuwIDAQABo4GQMIGNME

sGCVUdDwGG+EIBDQQ+EzxHZW5lcmF0ZWQgYnkgdGhlIFNlY3VyZVdheSBTZWN1cml0eSBTZ
XJ

2ZXIgZm9yIE9TLzM5MCAoUkFDRikwDgYDVR0PAQH/BAQDAgAGMA8GA1UdEwEB/wQFMAMB
Af8w

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

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

IdM ユーザーの有効化と無効化

user_enable および **user_disable** コマンドを使用すると、IdM ユーザーを有効または無効にできます。この例では、IdM ユーザー **exampleuser** を無効にします。

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

4.2. IDM API コマンドを使用したグループの管理

以下の例は、IdM API コマンドを使用して IdM グループを管理する方法の一般的なシナリオを示しています。

IdM API コマンドを使用した IdM ユーザーの管理の例

IdM グループの作成

この例では、指定のグループ ID 番号を使用して IdM グループ **developers** を作成します。

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

ユーザーをメンバーとして IdM グループに追加する

この例では、**admin** ユーザーを **developers** グループに追加します。

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

サービスをメンバーとして IdM グループに追加する

この例では、**HTTP/server.ipa.test** サービスを **developers** グループに追加します。

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

グループをサブグループとして IdM グループに追加する

この例では、別のグループ **admins** を **developers** グループに追加します。

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

IdM グループマネージャーの追加

この例では、**bob** ユーザーを **developers** グループのグループマネージャーとして追加します。

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

IdM グループの検索

さまざまなパラメーターを使用して IdM グループを検索できます。この例では、ユーザー **bob** が管理しているすべてのグループが検索されます。

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

IdM グループ情報の表示

この例では、メンバーリストを表示せずに、**developers** グループに関するグループ情報を表示します。

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

IdM グループの変更

この例では、非 POSIX グループの **testgroup** をPOSIX グループに変換します。

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

IdM グループからのメンバーの削除

この例では、**admin** ユーザーを **developers** グループから削除します。

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

IdM グループマネージャーの削除

この例では、マネージャーであるユーザー **bob** を **developers** グループから削除します。

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

IdM グループの削除

この例では、**developers** グループを削除します。

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

4.3. IDM API コマンドを使用したアクセス制御の管理

以下の例は、IdM API コマンドを使用して、アクセス制御を管理する方法の一般的なシナリオを示しています。

IdM API コマンドを使用したアクセス制御の管理の例

ユーザー作成用の権限の追加

この例では、ユーザーを作成するための権限を追加します。

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

グループメンバーシップを管理する権限の追加

この例では、ユーザーをグループに追加するための権限を追加します。

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

ユーザー作成プロセスでの権限の追加

この例では、ユーザーの作成、グループへの追加、およびユーザー証明書の管理のための権限を追加します。

```
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")
```

権限を使用したロールの追加

この例では、前の例で作成した権限を使用してロールを追加します。

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

ユーザーへのロールの割り当て

この例では、**usermanager** ロールをユーザー **bob** に割り当てます。

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

グループへのロールの割り当て

この例では、**usermanager** ロールを **managers** グループに割り当てます。

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

4.4. IDM API コマンドを使用した SUDO ルールの管理

以下の例は、IdM API コマンドを使用して sudo ルールを管理する方法の一般的なシナリオを示しています。

IdM API コマンドを使用した sudo ルールの管理の例

sudo ルールを作成する

この例では、時間変更コマンドを保持する sudo ルールを作成します。

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

sudo コマンドを作成する

この例では、**date** sudo コマンドを作成します。

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

sudo コマンドを sudo ルールに割り当てる

この例では、**date** sudo コマンドを **timechange** sudo ルールに割り当てます。

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

sudo コマンドのグループを作成して割り当てる

この例では、複数の sudo コマンドを作成して、新しく作成した **timecmds** sudo コマンドグループに追加し、そのグループを **timechange** sudo ルールに割り当てます。

```
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")
```

sudo コマンドを拒否する

この例では、**rm** コマンドが sudo として実行されることを拒否します。

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

sudo ルールにユーザーを追加する

この例では、ユーザー **bob** を **timechange** sudo ルールに追加します。

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

指定したホストでのみ sudo ルールを使用できるようにする

この例では、**timechange** ルールを **client.ipa.test** ホストでしか使用できないように制限します。

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

sudo ルールを別のユーザーとして実行するように設定する

デフォルトでは、sudo ルールは **root** として実行されます。この例では、代わりに **alice** ユーザーとして実行されるように **timechange** sudo ルールを設定します。

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

sudo ルールをグループとして実行するように設定する

この例では、**sysadmins** グループとして実行されるように **timechange** sudo ルールを設定します。

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

sudo ルールの sudo オプションを設定する

この例では、**timechange** sudo ルールの sudo オプションを設定します。

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

sudo ルールを有効にする

この例では、**timechange** sudo ルールを有効にします。

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

sudo ルールを無効にする

この例では、**timechange** sudo ルールを無効にします。

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

4.5. IDM API コマンドを使用したホストベースのアクセス制御の管理

以下の例は、IdM API コマンドを使用してホストベースのアクセス制御 (HBAC) を管理する方法の一般的なシナリオを示しています。

IdM API コマンドを使用した HBAC の管理の例

HBAC ルールの作成

この例では、SSH サービスアクセスを処理する基本ルールを作成します。

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

HBAC ルールへのユーザーの追加

この例では、ユーザー **john** を **sshd_rule** HBAC ルールに追加します。

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

HBAC ルールへのグループの追加

この例では、グループ **developers** を **sshd_rule** HBAC ルールに追加します。

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

HBAC ルールからのユーザーの削除

この例では、ユーザー **john** を **sshd_rule** HBAC ルールから削除します。

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

新しいターゲット HBAC サービスの登録

ターゲットサービスを HBAC ルールに割り当てる前に、ターゲットサービスを登録する必要があります。この例では、**chronyd** サービスを登録します。

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

登録済みサービスの HBAC ルールへの割り当て

この例では、**sshd** サービスを **sshd_rule** HBAC ルールに割り当てます。このサービスはデフォルトで IPA に登録されているため、事前に **hbacsvc_add** で登録する必要はありません。

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

HBAC ルールへのホストの追加

この例では、**workstations** ホストグループを **sshd_rule** HBAC ルールに追加します。

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

HBAC ルールのテスト

この例では、**workstation.ipa.test** ホストに対して **sshd_rule** HBAC ルールを使用します。ユーザー **john** からのサービス **sshd** をターゲットとしています。

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

HBAC ルールの有効化

この例では、**sshd_rule** HBAC ルールを有効にします。

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

HBAC ルールの無効化

この例では、**sshd_rule** HBAC ルールを無効にします。

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