NPM レジストリのテストを HTTP リクエストを使用して行う

NPM レジストリを `curl` や Insomnia などのツールを使用して生の HTTP リクエストでテストすることで、その機能を確認できます。これにより、接続性、認証、パッケージデータの提供能力を確認できます。NPM クライアントを使用せずに行うことができます。レジストリがユーザー名とパスワードを要求するため、リクエストで認証を手動で処理する必要があります。以下は、生の HTTP リクエストを使用してレジストリをテストするための完全なガイドです。NPM 互換のレジストリ（例：Verdaccio、Nexus、またはカスタム実装）を前提としています。トークンベースまたは基本認証をサポートしています。

生の HTTP リクエストを使用してレジストリをテストする手順

1. 認証とトークンの取得（必要な場合）

ほとんどの NPM レジストリはトークンベースの認証を使用し、トークンを取得するためにログインする必要があります。一部のレジストリは基本認証を直接サポートする場合もあります。以下に、`curl` を使用して認証する方法を示します。

`curl` を使用してログイン　認証エンドポイントに PUT リクエストを送信してトークンを取得します：

```
curl -X PUT \
  -H "Content-Type: application/json" \
  -d '{"name": "<username>", "password": "<password>"}' \
  <registry-url>/-/user/org.couchdb.user:<username>
```

- **置換**:
  - `<username>`: レジストリのユーザー名。
  - `<password>`: レジストリのパスワード。
  - `<registry-url>`: レジストリの完全な URL（例：`https://my-registry.example.com`）。
- **期待される応答**: 成功すると、トークンを含む JSON 応答が返されます：

  ```
  {
    "token": "your-auth-token"
  }
  ```

- **トークンの保存**: `your-auth-token` の値を後続のリクエストで使用するためにコピーします。

注意: レジストリが異なる認証エンドポイントや方法（例：基本認証またはカスタム API）を使用している場合は、そのドキュメントを参照してください。基本認証を直接サポートしている場合は、このステップをスキップし、後続のリクエストに `-u "<username>:<password>"` を含めることができます。

---

2. レジストリの Ping

レジストリへの基本的な接続をテストするために、ルート URL または Ping エンドポイントに GET リクエストを送信します。

`curl` を使用して Ping

```
curl -H "Authorization: Bearer your-auth-token" <registry-url>
```

- **置換**:
  - `your-auth-token`: ステップ 1 のトークン。
  - `<registry-url>`: レジストリの URL。
- **期待される応答**: 成功した応答（HTTP 200）は、レジストリのホームページまたは簡単なステータスメッセージ（例：CouchDB ベースのレジストリの場合は `{"db_name":"registry"}`）を返すことがあります。
- **代替**: 一部のレジストリは`/-/ping` エンドポイントを提供しています：

  ```
  curl -H "Authorization: Bearer your-auth-token" <registry-url>/-/ping
  ```

基本認証を使用する場合: トークンを使用せずに基本認証をサポートするレジストリの場合：

```
curl -u "<username>:<password>" <registry-url>
```

---

3. パッケージメタデータの取得

レジストリがパッケージメタデータを提供できることを確認するために、特定のパッケージの詳細をリクエストします。

`curl` を使用してメタデータを取得

```
curl -H "Authorization: Bearer your-auth-token" <registry-url>/<package-name>
```

- **置換**:
  - `<package-name>`: レジストリに存在するパッケージ（例：公共レジストリをプロキシする場合は `lodash`、またはプライベートパッケージ `my-org-utils`）。

- **期待される応答**: パッケージのメタデータを含む JSON オブジェクト、バージョン、依存関係、tarball URL を含む。例：

```
{
  "name": "lodash",
  "versions": {
    "4.17.21": {
      "dist": {
        "tarball": "<registry-url>/lodash/-/lodash-4.17.21.tgz"
      }
    }
  }
}
```

基本認証を使用する場合:

```
curl -u "<username>:<password>" <registry-url>/<package-name>
```

- **成功**: メタデータを含む 200 OK 応答は、レジストリがパッケージデータを正しく提供していることを確認します。

---

4. パッケージターバルのダウンロード（オプション）

レジストリが実際のパッケージファイルを提供できることを完全にテストするために、パッケージターバルをダウンロードします。

`curl` を使用してターバルをダウンロード

1. ステップ3のメタデータから、特定のバージョンの `tarball` URL を探します（例：`<registry-url>/lodash/-/lodash-`
2. ダウンロードします：

```
curl -H "Authorization: Bearer your-auth-token" -O <tarball-url>
```

- **置換**: メタデータからの URL。
- **`-O` フラグ**: ファイルを元の名前で保存します（例：`lodash-4.17.21.tgz`）。
- **基本認証を使用する場合**:

  ```
  curl -u "<username>:<password>" -O <tarball-url>
  ```

- **成功**: ファイルが正常にダウンロードされ、内容を確認するために抽出できます（例：`tar -xzf <filename>`）。

Insomnia を使用したテスト

GUI ツールである Insomnia を使用する場合は、以下の手順に従ってください：

1. 認証の設定

- Insomnia で新しいリクエストを作成します。
- **Auth** タブに移動します：
  - **Bearer Token**: ステップ 1 でトークンを取得した場合は、「Bearer Token」を選択し、`your-auth-token` を貼り付けます。
  - **Basic Auth**: レジストリが基本認証を使用する場合は、「Basic Auth」を選択し、`<username>` と `<password>` を入力します。

2. レジストリの Ping

- **Method**: GET
- **URL**: `<registry-url>` または `<registry-url>/-/ping`
- **Send** をクリックします。
- **期待される応答**: 200 OK ステータスと簡単な応答ボディ。

3. パッケージメタデータの取得

- **Method**: GET
- **URL**: `<registry-url>/<package-name>`
- **Auth** タブで認証が設定されていることを確認します。
- **Send** をクリックします。
- **期待される応答**: 200 OK ステータスと JSON 形式のパッケージメタデータ。

4. ターバルのダウンロード

- **Method**: GET
- **URL**: メタデータからのターバル URL(例：`<registry-url>/<package-name>/-/<package-name>-<version>.tgz`)。
- **Send and Download** をクリックしてファイルをローカルに保存します。
- **成功**: ファイルがダウンロードされ、レジストリがパッケージを提供していることを確認します。

トラブルシューティング

- **401 Unauthorized**:
  - トークンまたは資格情報を確認します。
  - `Authorization` ヘッダーまたは基本認証が正しく形式化されていることを確認します。
- **404 Not Found**:
  - パッケージがレジストリに存在することを確認します。
  - 公共パッケージ（例：`lodash`）をテストしている場合は、レジストリが公共 NPM レジストリをプロキシしていることを確認します。
- **接続エラー**:
  - `<registry-url>` を確認します（例：必要に応じて `https://`を含めます）。
  - HTTPS レジストリの場合、`curl` で SSL エラーが発生する場合は、テスト用に`-k` を追加して証明書のチェックをバイパスします。
- **カスタムレジストリの動作**:
  - これらの手順が失敗する場合は、特定のエンドポイントや認証方法についてレジストリのドキュメントを参照してください。

結論

`curl` または Insomnia を使用してこれらの手順に従うことで、生の HTTP リクエストを使用して NPM レジストリの接続性、認証、パッケージ提供機能をテストできます。まず認証（必要な場合）、次にレジストリの Ping、知っているパッケージのメタデータを取得し、オプションでターバルをダウンロードします。このアプローチにより、レジストリが HTTP レベルで完全に動作していることを確認できます。