記事
Toshihiko Minamoto · 2023年1月31日 10m read

セキュリティパッケージのREST API

開発者の皆さん、こんにちは！

この記事では、IRIS Security Package 用 REST API をセットアップする方法を学習します。 簡単な HTTP リクエストによって、ユーザー、役割、アプリケーションの追加などを作成し、ObjectScript でクライアントアプリケーションを生成できるようになります。

必要条件

必要なのは:

1. IRISインスタンス(インストールキットまたはDocker)
2. ObjectScript package manager (ZPM)
3. ObjectScriptクライアントを生成するための2つ目のIRISインスタンス(オプション)

OpenExchangeで既存のアプリケーションとライブラリのセットを使用する予定です。 package manager (ZPM)を使用すると、それらのインストールが非常に簡単になります。 インスタンスに ZPM がない場合は、IRIS ターミナルにこのラインをコピーすることで簡単にインストールすることができます。

```
set $namespace="%SYS" do ##class(Security.SSLConfigs).Create("ssl") set r=##class(%Net.HttpRequest).%New(),r.Server="pm.community.intersystems.com",r.SSLConfiguration="ssl" do r.Get("/packages/zpm/latest/installer"),$system.OBJ.LoadStream(r.HttpResponse.Data,"c")
```

Webアプリケーションの作成

パッケージ関連の REST サービスは、Config-API アプリケーションのバージョン 1.4.0 以降で利用できます。
このアプリケーションをZPMコマンドで簡単にインストールすることにします。

```
zpm "install config-api"
```

デフォルトでは、Config-API をインストールしても REST サービスは公開されないので、Web アプリケーション「/config-api」を作成する必要があります。
管理ポータルでの手動操作を避けるために、即時使用可能なスクリプトが用意されています。

```
Do ##class(Api.Config.Developers.Install).installMainRESTApp()
```

セキュリティパッケージのRESTサービスが、「/config-api/security/」というパスで利用できるようになりました。

APIセキュリティ

これで、「/config-
api」Webアプリケーションが作成されると、管理ポータルで詳細を確認できるようになります。

デフォルトでは、ユーザが%AdminSecureリソースを持っている場合、アプリケーションはログインとパスワード（基本認証）によってのみアクセスできます。

もちろん、これだけでは十分ではありません。 リクエストにはHTTPSプロトコルを使用し、WebGatewayとIRISインスタンス間の通信を暗号化する必要があります。API Manager(IAM)を利用するのも有効かもしれません。そうすることで、例えば特定のIPアドレスからのHTTPリクエストのみを受け付けるなど、よりスムーズなアクセス制御が可能になります。 設定方法については、この記事の範囲外なので、詳しくは説明しません。しかし、このテーマについては、コミュニティや公式ドキュメントでより多くの記事を見つけることができます。しかし、もしあなたが私にコメントを残して、私はあなたにWebGateway
HTTPSとSSLTLSを持つDockerベースのリポジトリを提供することを望みます。

OnPreDispatchのカスタマイズ (オプション)

RESTディスパッチクラス "Api.Config.REST.Main" は、 "OnPreDispatch" メソッドを実装しています。このメソッドは、リクエスト処理の前に呼び出されます。各リクエストに対して実行したい共通のコードをここに置くだけです。pContinueに0を設定すると、リクエストは処理されないことを覚えておいてください。

```
Class Api.Config.REST.Main Extends %CSP.REST
{

....

ClassMethod OnPreDispatch(pUrl As %String, pMethod As %String, ByRef pContinue As %Bo
olean) As %Status
{
    Set sc = $$$OK, class = ##class(Api.Config.REST.OnPreDispatchAbstract).GetSubClas
s()

    Set pContinue = $$$YES

    Return:class="" sc

    Return $CLASSMETHOD(class, "OnPreDispatch", pUrl, pMethod, .pContinue)
}
}
```

デフォルトの実装では、"Api.Config.REST.OnPreDispatchAbstract
"のサブクラスが存在するかどうかをチェックします。
もし存在すれば、それが実行されます。カスタムコードを実行するフックがあるので、API
Managerがない場合、追加のアクセスチェックやロギングを行う代替手段にもなります。

以下は、受信したリクエストのみをログに記録する実装例です。

```
Class dc.sample.RestSecurity Extends Api.Config.REST.OnPreDispatchAbstract
{

ClassMethod OnPreDispatch(pUrl As %String, pMethod As %String, ByRef pContinue As %Bo
olean) As %Status
{
    Set sc = $$$OK
```

```
    /// ??????????????????????

    Set key = $Increment(^RestSecurity.log)
    Set ^RestSecurity.log(key) = $ZDateTime($Horolog,3,1) _ " " _ pMethod _ " " _pUrl
 _ "( IP : " _ $Get(%request.CgiEnvs("REMOTE_ADDR")) _ ")"

    merge ^RestSecurity.log(key, "CgiEnvs") = %request.CgiEnvs
    merge ^RestSecurity.log(key, "Data") = %request.Data

    // ????????:
    // Set %response.Status = "401 Unauthorized"
    // Set pContinue = $$$NO

    Quit sc
}
}
```

REST APIのテスト

このAPIは http://localhost:52773/config-api/security/ で公開されている 仕様（swagger 2.0）で提供されています（必要に応じてポート番号を修正してください）。そのため、例えばOpenExchangeで公開されているswagger-uiというアプリケーションを使用すれば、簡単にクライアントアプリケーションを生成することができます。

swagger-uiをインストールします。

```
zpm "install swagger-ui"
```

次に、URLhttp://localhost:52773/csp/swagger-ui/index.htmlでブラウザを開きます。

ページを開くと、ほとんどの場合、エラーが表示されます。

このエラーは、デフォルトでswagger-uiが存在しないURLから仕様を取得しようとするために発生します。このエラーを回避するには、RESTサービスのURLを調査することで、swagger-uiを強制的に開かせることができます：

```
Do ##class(Api.Config.Developers.Install).SetSwaggerUIDefaultPath("/config-
api/security/")
```

ここでブラウザを更新してください。 %Adminsecure リソースを持つユーザーでログインしてください。

この段階では、ユーザー、役割、リソース、SSL構成、およびWebアプリケーションに対して、**C**reate **R**ead **U**pdate **D**elete の操作が可能であることが、このインターフェイスで確認することができます。

サービスやシステム設定の場合は少し違っていて、読み込み（GET）と変更（PUT）しかできません。

このインターフェースは、かなり詳細なswagger仕様のおかげで自己文書化されています。

POST /user をクリックすれば、詳細な説明、リクエストのサンプルボディ、モデルの完全なドキュメントを得ることができるので、ここで各リクエストをテストする方法を説明することは有益ではありません。
この記事では、SQL 特権という特殊なケースについてのみ説明します。

SQL特権の追加

POST /sqlprivileges

SQL 特権を設定します。

以下は、"select " 特権を追加するためのボディの例です。

```
{
  "Grantable": "1",
  "Grantee": "MyRoleName",
  "Grantor": "_system",
  "Namespace": "USER",
  "Privilege": "s",
  "SQLObject": "1,schema_name.table_name"
}
```

特権は次の値を取ることができます：s (select)、 i (insert)、 u (update)、 d (delete)、 r (reference)、 および e (execute).

SQLObjectは、テーブルを表す "1"、ビューを表す "3"、ストアドプロシージャを表す "9 "から始まります。

多数のテーブルに特権を割り当てる必要がある場合、あまり便利ではありません.....。
この場合は「(PUT)/sqlhelper」を使用するのがよいでしょう。

特権を削除する

DELETE /sqlprivileges/{id}

削除は、名前空間、SQLObject、特権、付与者、付与者から構成されるIDによって行われます。前回の特権の作成に対応するIDは「USER||1,schemaname.tablename||s||MyRoleName|system」であります。特殊文字のエスケープに注意してください。

スキーマ内のすべてのテーブルに特権を追加する

PUT /sqlhelper

このサービスでは、1つまたは複数のスキーマの全テーブルに一連の権限を割り当てることができます。以下はボディの例です。

```
{
  "Grantable": "1",
  "Grantee": "MyNewRole",
  "Grantor": "_system",
  "Namespace": "USER",
  "Table": {
    "Schemas": [
      "schema_name",
      "schema_name_2"
    ],
    "Privileges": [
      "S",
      "I",
      "U",
      "D"
    ]
  }
```

```
}
```

HTTP ObjectScript クライアントを生成する

Swagger editorでは、swaggerの仕様から様々な言語でクライアントを生成することができますが、残念ながらObjectScriptはそのリストに含まれていません。 ただし、OpenExchangeにあるアプリケーション「OpenApi-client-gen」を使えば生成することが可能です。現在のところ、swagger 2.0仕様にのみ対応しています。

Install openapi-client-gen:

```
zpm "install openapi-client-gen"
```

クライアントアプリケーションを生成します：

```
Set features("simpleHttpClientOnly") = 1
Set sc = ##class(dc.openapi.client.Spec).generateApp("IrisSecurity", "https://raw.gitHubusercontent.com/lscalese/iris-config-api/master/swagger-security.json", .features)
Write !,"Status : ",$SYSTEM.Status.GetOneErrorText(sc)
```

以下に、HTTPリクエストでWebアプリケーションの一覧を取得し、その結果をターミナルに表示するコードの例を示します。

```
Class iris.dc.sample.ObjectScriptRestClient
{

ClassMethod GetRequestObj() As %Net.HttpRequest
{
    Set httpRequest = ##class(%Net.HttpRequest).%New()
    Set httpRequest.Username = "_system"
    Set httpRequest.Password = "SYS"
    Set httpRequest.Server = "iris-security-rest-server"
    Set httpRequest.Port = 52773
    Set httpRequest.Https = 0
    Quit httpRequest
}

ClassMethod ExampleGetWebAppList() As %Status
{
    Set sc = $$$OK
    Set httpClient = ##class(IrisSecurity.HttpClient).%New()
    Set httpRequest = ..GetRequestObj()
    Do httpRequest.SetHeader("accept", "application/json")
    Set msg = ##class(IrisSecurity.msg.GetListOfWebAppsRequest).%New()
    Set sc = httpClient.GETGetListOfWebApps(msg,.response,.httpRequest,.httpresponse)
    Write !,"Status           : ",$SYSTEM.Status.GetOneErrorText(sc)
    Quit:$$$ISERR(sc) sc
    Write !,"Http Status Code : ",response.httpStatusCode
    Write !
    zw response

    If response.httpStatusCode = 200 {
        Set formatter=##class(%JSON.Formatter).%New()
        Do formatter.Format({}.%FromJSON(response.body))
```

```
    }

    Quit sc
}

}
```

GitHub

もしあなたがDockerユーザーなら、この記事で見てきたことはすべて次のGitHubリポジトリで公開されています：https://github.com/lscalese/iris-sample-security-rest-api.

このように実行します。

```
git clone https://github.com/lscalese/iris-sample-security-rest-api.git
cd iris-sample-security-rest-api
docker-compose up -d
```

これで、URL http://localhost:32773/swagger-ui/index.html でブラウザを開き、REST API を直接テストすることができます。

また、iris-cliサービスにおいてターミナルを開き、生成されたObjectScriptクライアントアプリケーションをテストすることもできます。

```
docker exec -it iris-security-rest-client iris session iris
Do ##class(iris.dc.sample.ObjectScriptRestClient).ExampleGetWebAppList()
```

JSON形式で表示されたWebアプリケーションのリストが表示されるはずです。

#DevOps #REST API #セキュリティ #デプロイ #InterSystems IRIS

ソースURL:
https://jp.community.intersystems.com/post/%E3%82%BB%E3%82%AD%E3%83%A5%E3%83%AA%E3%83%86%E3%82%A3%E3%83%91%E3%83%83%E3%82%B1%E3%83%BC%E3%82%B8%E3%81%AErest-api