



プロダクションでの REST サービスと REST オペレーションの 使用法

Version 2024.1

2024-06-06

プロダクションでの REST サービスと REST オペレーションの使用法
InterSystems IRIS Data Platform Version 2024.1 2024-06-06
Copyright © 2024 InterSystems Corporation
All rights reserved.

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble®, InterSystems HealthShare®, InterSystems IRIS®, および TrakCare は、InterSystems Corporation の登録商標です。HealthShare® CMS Solution Pack™ HealthShare® Health Connect Cloud™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, および InterSystems TotalView™ For Asset Management は、InterSystems Corporation の商標です。TrakCare は、オーストラリアおよび EU における登録商標です。

ここで使われている他の全てのブランドまたは製品名は、各社および各組織の商標または登録商標です。

このドキュメントは、インターシステムズ社(住所: One Memorial Drive, Cambridge, MA 02142)あるいはその子会社が所有する企業秘密および秘密情報を含んでおり、インターシステムズ社の製品を稼働および維持するためにのみ提供される。この発行物のいかなる部分も他の目的のために使用してはならない。また、インターシステムズ社の書面による事前の同意がない限り、本発行物を、いかなる形式、いかなる手段で、その全てまたは一部を、再発行、複製、開示、送付、検索可能なシステムへの保存、あるいは人またはコンピュータ言語への翻訳はしてはならない。

かかるプログラムと関連ドキュメントについて書かれているインターシステムズ社の標準ライセンス契約に記載されている範囲を除き、ここに記載された本ドキュメントとソフトウェアプログラムの複製、使用、廃棄は禁じられている。インターシステムズ社は、ソフトウェアライセンス契約に記載されている事項以外にかかるソフトウェアプログラムに関する説明と保証をするものではない。さらに、かかるソフトウェアに関する、あるいはかかるソフトウェアの使用から起こるいかなる損失、損害に対するインターシステムズ社の責任は、ソフトウェアライセンス契約にある事項に制限される。

前述は、そのコンピュータソフトウェアの使用およびそれによって起こるインターシステムズ社の責任の範囲、制限に関する一般的な概略である。完全な参照情報は、インターシステムズ社の標準ライセンス契約に記載され、そのコピーは要望によって入手することができる。

インターシステムズ社は、本ドキュメントにある誤りに対する責任を放棄する。また、インターシステムズ社は、独自の裁量にて事前通知なしに、本ドキュメントに記載された製品および実行に対する代替と変更を行う権利を有する。

インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

目次

1 プロダクションでの REST サービスの作成	1
2 プロダクションでの REST オペレーションの作成	3
2.1 基本	3
2.2 例	4
2.3 バリエーション：JSON データのポスト操作	5

1

プロダクションでの REST サービスの作成

このページでは、REST サービスでもあるビジネス サービスの作成方法を簡単に説明します。REST サービスは、REST 要求を受け取り、プロダクションの他の場所にあるビジネス・プロセスまたはビジネス・オペレーションに渡すことができます。

この方法は実際のニーズに応じて異なります。以下の操作を希望する場合は、次のようにします。

- ・ プロダクション内の要求を解析および処理する：**%CSP.REST** のサブクラスを使用し、`Ens.Director.CreateBusinessService()` メソッドを呼び出して、このクラスをビジネス・サービスとしてインスタンス化します。このサービスでは Web ポートが使用されます。**%CSP.REST** のサブクラスの実装に関する詳細は、“REST サービスの作成” を参照してください。
- ・ 最小限の変更で REST URL を外部サーバに渡す：パススルー REST サービス **EnsLib.REST.GenericService** を使用します。パススルー REST サービスの使用に関する詳細は、“ESB としてのプロダクションの使用” の“[ESB のサービスおよびオペレーションの構成](#)” および“[パススルー・サービスとパススルー・オペレーションのウォークスルー](#)” で、パススルー・ビジネス・サービスに関する節を参照してください。

%CSP.REST のサブクラスの実装に関する詳細は、“REST サービスの作成” を参照してください。

注釈 InterSystems IRIS には、**%CSP.REST** のサブクラスを作成する代わりに使用できる組み込みのビジネス・サービス **EnsLib.REST.Service** が用意されています。このビジネス・サービスは HTTP/REST 受信アダプタと連携するので、ディスパッチ・メソッドは、HTTP アダプタが使用する入出力ストリームが含まれる追加引数を受け取ります。転送された <Map> 要求を受信するクラスは、**EnsLib.REST.Service** ではなく、**%CSP.REST** を拡張する必要があります。

2

プロダクションでの REST オペレーションの作成

このページでは、REST オペレーションの作成方法を簡単に説明します。これは、外部 REST サービスを呼び出すビジネス・オペレーションです。

2.1 基本

1. REST オペレーション・クラスを定義します。InterSystems IRIS® の送信 HTTP アダプタを使用する、**EnsLib.REST.Operation** のサブクラスを作成します。詳細は、“[HTTP 送信アダプタの使用法](#)”を参照してください。
2. このビジネス・オペレーションの動作を目的のクラスに定義します。一般的な詳細は、“[ビジネス・オペレーションの定義](#)”を参照してください。このビジネス・オペレーションで外部 REST サービスを呼び出す必要があるので、使用する HTTP オペレーションに応じて、以下の HTTP アダプタのメソッドを 1 つまたは複数呼び出します。
 - ・ `GetURL()` – HTTP GET オペレーションを使用します。
 - ・ `PostURL()` – HTTP POST オペレーションを使用します。
 - ・ `PutURL()` – HTTP PUT オペレーションを使用します。
 - ・ `DeleteURL()` – HTTP DELETE オペレーションを使用します。
 - ・ `SendFormDataArray()` – HTTP オペレーションをパラメータとして指定できます。

これらのオペレーションはすべて、プロダクション構成で指定したベース URL を基準として動作します。

3. このビジネス・オペレーションをプロダクションに追加し、通常の手順に従って構成します。これで、外部 REST サービスの場所を指定できます。例えば、天気サービスを呼び出す REST オペレーションを定義するには、以下のようオペレーションを構成できます。

実行中のビジネス・プロセスがない場合は、[Interoperability]→[構成する]→[プロダクション] ページで、このオペレーションおよび他のオペレーションを実行およびテストできます。そのためには、オペレーションを選択した後、[アクション] タブで [テスト] を選択します。

2.2 例

例えば、以下の `EnsLib.REST.Operation` の拡張は、天気 REST サービスを呼び出して、都市名をパラメータとして提供します。

Class Definition

```
Class Test.REST.WeatherOperation Extends EnsLib.REST.Operation
{
    Parameter INVOCATION = "Queue";

    Method getWeather(
        pRequest As Test.REST.WeatherRequest,
        Output pResponse As Test.REST.WeatherResponse) As %Status
    {
        try {
            // Prepare and log the call
            // Append the city to the URL configured for adapter
            Set tURL=..Adapter.URL_"?q=" _pRequest.City_"&units=imperial"

            // Execute the call
            Set tSC=..Adapter.GetURL(tURL,.tHttpResponse)

            // Return the response
            If $$$ISERR(tSC)&&$$isObject(tHttpResponse)&&$$isObject(tHttpResponse.Data)&&tHttpResponse.Data.Size
            {
                Set tSC=$$$ERROR($$$EnsErrGeneral,$$$StatusDisplayString(tSC)_"_"_tHttpResponse.Data.Read())
            }
            Quit:$$$ISERR(tSC)
            If $isObject(tHttpResponse) {
                // Instantiate the response object
                set pResponse = ##class(Test.REST.WeatherResponse).%New()
                // Convert JSON into a Proxy Object
                set tSC = ..JSONStreamToObject(tHttpResponse.Data, .tProxy)
                if (tSC){
                    // Set response properties from the Proxy Object
                    set pResponse.Temperature = tProxy.main.temp_"F"
                    set pResponse.Humidity = tProxy.main.humidity_"%"
                    set pResponse.MaxTemp = tProxy.main."temp_max_"_"F"
                    set pResponse.MinTemp = tProxy.main."temp_min_"_"F"
```



```

        set pResponse.Pressure = tProxy.main.pressure_" mbar"
        set pResponse.WindSpeed = tProxy.wind.speed_" MPH"
        set pResponse.WindDirection = tProxy.wind.deg_" degrees"
        // Convert from POSIX time
        set pResponse.Sunrise = $ZT($PIECE($ZDTH(tProxy.sys.sunrise, -2),",",2),3)
        set pResponse.Sunset = $ZT($PIECE($ZDTH(tProxy.sys.sunset, -2),",",2),3)
    }
}
}catch{
    Set tSC=$$$SystemError
}
Quit tSC
}

XData MessageMap
{
    <MapItems>
        <MapItem MessageType="Test.REST.WeatherRequest">
            <Method>getWeather</Method>
        </MapItem>
    </MapItems>
}
}

```

オペレーションに送信されたメッセージは、都市を指定します。

Class Definition

```

Class Test.REST.WeatherRequest Extends (%Persistent, Ens.Util.MessageBodyMethods)
{
    Property City As %String;
}

```

以下の操作は、JSONStreamToObject() メソッドを呼び出し、JSON の要素をアクセス可能にする InterSystems IRIS オブジェクトを返します。このサンプルによって返されたメッセージは、JSON ストリームから取得された以下のプロパティを返します。

Class Definition

```

Class Test.REST.WeatherResponse Extends (%Persistent, Ens.Util.MessageBodyMethods)
{
    Property Temperature As %String;
    Property MinTemp As %String;
    Property MaxTemp As %String;
    Property Pressure As %String;
    Property Humidity As %String;
    Property WindSpeed As %String;
    Property WindDirection As %String;
    Property Sunrise As %String;
    Property Sunset As %String;
}

```

2.3 バリエーション：JSON データのポスト操作

JSON データをポストする REST オペレーションが必要な場合は何らかの適応処理が必要です。

1. HTTP アダプタで HTTP コンテンツタイプ ヘッダを適切に指定する必要があります。つまり、以下のように専用の HTTP アダプタ・クラスを作成する必要があります。

Class Definition

```
Class My.REST.Client.HTTPOutboundAdapter Extends EnsLib.HTTP.OutboundAdapter
{
  /// Send a POST to the configured Server, Port and URL, sending form data to the named form variables.
  Method Post(Output pHttpResponse As %Net.HttpResponse, pFormVarNames As %String, pData...) As
  %Status {
    quit ..SendFormDataArray(.pHttpResponse, "POST", ..GetRequest(), .pFormVarNames, .pData)
  }

  ClassMethod GetRequest() As %Net.HttpRequest
  {
    set request = ##class(%Net.HttpRequest).%New()
    set request.ContentType = "application/json"
    quit request
  }
}
```

2. 作成した新しいアダプタ・クラスを使用するカスタム・ビジネス・オペレーション・クラスを作成します。
3. 以下の手順で JSON 形式の文字列を作成します。
 - a. %DynamicObject のインスタンスを作成します。
 - b. このオブジェクトのプロパティを設定します。
 - c. %ToJSON() メソッドを使用して、このオブジェクトをシリアル化します。

以下の例をご覧ください。

ObjectScript

```
//Use a %Library.DynamicObject to prepare the REST POST request
Set tRequest = ##class(%DynamicObject).%New()
Set tRequest.transactionid=pRequest.transactionid
Set tRequest.participantid=pRequest.participantid
Set tRequest.authstatus=pRequest.authstatus
Set tRequest.reason=pRequest.reason
set tPayload = tRequest.%ToJSON()
```

4. アダプタの Post メソッドを呼び出して、この文字列を目的の REST サービスにポストします。以下に例を示します。

ObjectScript

```
Set tSC=..Adapter.Post(.tHttpResponse, , tPayload)
```

この例では、2 番目のパラメータが空です。