



# Using REST Services and Operations in Productions

Version 2019.4  
2020-01-28

*Using REST Services and Operations in Productions*

InterSystems IRIS Data Platform Version 2019.4 2020-01-28

Copyright © 2020 InterSystems Corporation

All rights reserved.

InterSystems, InterSystems IRIS, InterSystems Caché, InterSystems Ensemble, and InterSystems HealthShare are registered trademarks of InterSystems Corporation.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

**InterSystems Worldwide Response Center (WRC)**

Tel: +1-617-621-0700

Tel: +44 (0) 844 854 2917

Email: [support@InterSystems.com](mailto:support@InterSystems.com)

# Table of Contents

<b>About This Book .....</b>	<b>1</b>
<b>1 Overview of Using REST in Productions .....</b>	<b>3</b>
<b>2 Creating REST Services in Productions .....</b>	<b>5</b>
<b>3 Creating REST Operations in Productions .....</b>	<b>7</b>



# About This Book

This book describes, to programmers, how to create REST business services and REST business operations for productions. Before reading this book, you should read *First Look: Developing REST Interfaces in InterSystems IRIS*, which provides an introduction to creating REST services and *Creating REST Services*. You may also want to take the online class [Setting Up RESTful Services](#). You need to be logged into [learning.intersystems.com](http://learning.intersystems.com) to take this course. If you do not have an account, you can create one.

The *First Look* and online class present an introduction to using REST with InterSystems IRIS. This document provides a brief overview and then describes more advanced topics. It includes the following sections:

- [Overview of Using REST in Productions](#)
- [Creating REST Services in Productions](#)
- [Creating REST Operations in Productions](#)

For a detailed outline, see the [table of contents](#).



# 1

## Overview of Using REST in Productions

InterSystems IRIS® provides the capabilities to implement a REST service that can be invoked from a REST call.

REST, which is named from “Representational State Transfer,” has the following attributes:

- REST is an architectural style rather than a clearly defined format. Although REST is frequently implemented using HTTP for transporting messages and JSON for passing data, you can also pass data as XML or plain text. REST makes use of existing web standards such as HTTP, URL, XML, and JSON.
- REST is resource oriented. Typically a resource is identified by a URL and uses operations based explicitly on HTTP methods, such as GET, POST, PUT, and DELETE.
- REST typically has a small overhead. Although it can use XML to describe data, it typically uses JSON which is a lightweight data wrapper. JSON identifies data with tags but the tags are not specified in a formal schema definition and do not have explicit data types.





# 2

## Creating REST Services in Productions

This section describes how to provide a REST service from an Business Service and to pass the request to a Business Process or Business Operation. It lists the best practice for different requirements.

If you want to:

- Parse and process the request in the production—use a subclass of `%CSP.REST` and call the **Ens.Director.CreateBusinessService()** method to instantiate the class as a business service. This service uses the Web port.
- Pass through a REST URL to an external server with minimal changes—use the pass-through REST service, `EnsLib.REST.GenericService`.

For details on implementing a subclass of `%CSP.REST`, see *Creating REST Services*.

For an example of using the **Ens.Director.CreateBusinessService()** method, use Studio to examine the class `Demo.ZenService.Zen.WeatherReportForm.cls`. Although this is a Zen service rather than a REST service, you would use the **Ens.Director.CreateBusinessService()** method in the same way.

For details on using the pass-through REST service, see the sections on pass-through business services in “[Configuring ESB Services and Operations](#)” and “[Pass-through Service and Operation Walkthrough](#)” in *Using a Production as an ESB*.

**Note:** Although InterSystems IRIS defines a class `EnsLib.REST.Service`, that is a subclass of `%CSP.REST`, we recommend that you not use this class because it provides an incomplete implementation of `%CSP.REST`. The only feature that `EnsLib.REST.Service` provides that is not available from `%CSP.REST` is the ability to use a special port, but we recommend against using a special port because it does not provide the robustness and security you get by using a commercial web server and the Web port.

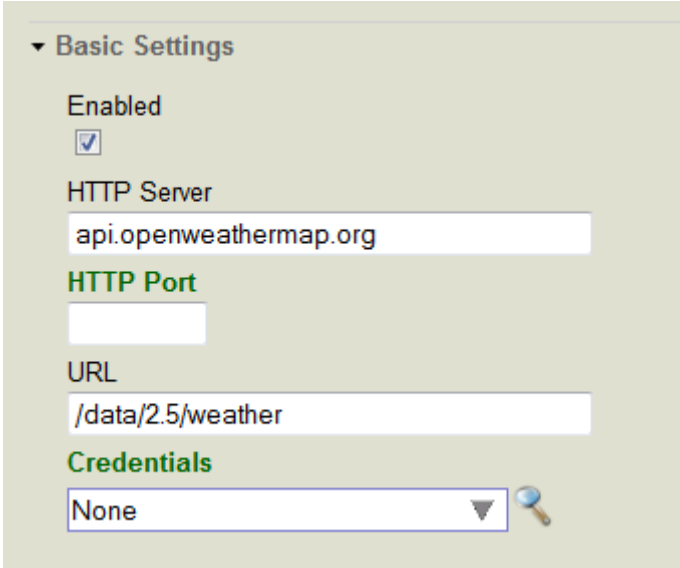


# 3

## Creating REST Operations in Productions

To develop a REST operation, you extend the class `EnsLib.REST.Operation`. The REST operation uses the InterSystems IRIS® outbound HTTP adapter, which is described in [Using the HTTP Outbound Adapter](#).

Using the HTTP Outbound Adapter, you specify the server, port, and address of the external web service in the HTTP adapter configuration. For example, to define a REST operation that calls a weather service, you could configure the operation as follows:



The screenshot shows the configuration interface for the HTTP Outbound Adapter. It is titled "Basic Settings" and includes the following fields:

- Enabled:** A checkbox that is checked.
- HTTP Server:** A text input field containing the value "api.openweathermap.org".
- HTTP Port:** An empty text input field.
- URL:** A text input field containing the value "/data/2.5/weather".
- Credentials:** A dropdown menu set to "None", with a search icon to its right.

In your extension of `EnsLib.REST.Operation`, you specify the remaining parts of the URL by appending the value to the adapter URL property. Then you call one of the following adapter methods depending on what HTTP operation you want to use:

- **GetURL()**—uses the HTTP GET operation.
- **PostURL()**—uses the HTTP POST operation.
- **PutURL()**—uses the HTTP PUT operation.
- **DeleteURL()**—uses the HTTP DELETE operation.
- **SendFormDataArray()**—allows you to specify the HTTP operation as a parameter.

For example, the following extension of `EnsLib.REST.Operation` calls the weather REST service and provides a city name as a parameter:

```

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>
}
}

```

The message sent to the operation specifies the city:

```

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

```

This operation calls the **JSONStreamToObject()** method and returns an InterSystems IRIS object that makes the elements of the JSON accessible. The message returned by this sample returns the following properties taken from the JSON stream:

```
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;
}
```

If you do not have a business process running, you can run and test this and other operations in **Interoperability > Configure > Production** page by selecting your operation and then selecting **Test** on the **Actions** tab.

