

Article

[Mihoko Iijima](#) · Mar 5, 2021 10m read

[InterSystems IRIS for the First Time] Interoperability: Creating Components (Business Operation)


This article is a continuation of [this post](#).

In the [previous article](#), we reviewed how to create and define **messages** used to send and receive data between components.

In this article, I will explain how to create a business operation from the component creation methods.

- [Production](#)
- [Message \(previous article\)](#)
- **Components**
 - Business Services
 - Business Processes
 - **Business Operations**

We will quickly check the code by referring to the sample..



Management Portal

Home About Help Log

Server 76328623e918 Namespace USER [Switch](#) User _SYSTEM Licensed To InterSystems IRIS Community Instance IRIS

Interoperability > Production Configuration - (Start.Production)

Production Configuration

Start Stop

Production Running

Category: All Legend Production Settings

Services	Processes	Operations
<ul style="list-style-type: none">EnsLib.JavaGateway.ServiceStart.FileBSStart.NonAdapterBSStart.WS.WebServiceBS	<ul style="list-style-type: none">Start.WeatherCheckProcess	<ul style="list-style-type: none">Start.GetKionOperationStart.InsertOperationStart.SQLInsertOperation

Verifies the specified directory at regular intervals and passes the information to the BS if a file is found.
File format : PurchasedProductName, CityName<line break>

BS
FileInboundAdapter

Ex) Call from REST
<http://localhost:52773/start/weather/Takoyaki/Osaka>

BS without adapters
(calls from stored procedures/REST)

Ex) call from stored procedures
Start.Util.CallProduction("Tempura", "Tokyo")

BS
Web Service

Run from the Web Service Test page.
<http://localhost:52773/csp/ensemble/%25SOAP.WebServiceInvoke.cls?CLS=Start.WS.WebServiceBS&OP=CallEnsemble>

BP

Start

気象情報取得

気象情報DB登録

End

BO
HttpOutboundAdapter(REST)

BO
Non Adapter
(DB Registration)

BO
SQLOutboundAdapter
(DB Registration via xDBC)

DB

via xDBC

Start.Request

Start.Request

Start.Request

Start.Request

Start.Response

Start.InsertRequest

Start.InsertRequest

Start.InsertRequest

Update directly in BO

https://api.openweathermap.org/data/2.5/weather?appid=<APIkey>&units=metric&q=Tokyo&lang=en)

Component Name

Role

Start.FileBS	A business service that uses file inbound adapter to read files placed in a specified directory at regular intervals.
Start.NonAdapterBS	Business services that allow applications and users to input information directly without using an adapter.
Start.WS.WebServiceBS	Business services that allow people to enter information using web services.
Start.WeatherCheckProcess	A business process that controls the procedure for acquiring weather information and then registering it in a database.
Start.GetKionOperation	Business operation to pass the city name to the web service that provides weather information and sent back.
Start.SQLInsertOperation	Business operations using SQL outbound adapters to request registration of weather and purchase information into the database.
Start.InsertOperation	Business operations that perform updates to tables in InterSystems IRIS without the use of adapters.

Note: BS stands for Business Services, BP for Business Processes, and BO for Business Operations.

You need to write ObjectScript in Business Services and Business Operations and can be created in VSCode or Studio. Business Processes can also be made in the Management Portal (see [this article](#) for more information on using VSCode).

There is no particular order of creation, but the external site to be connected to is a public site and can be used immediately in this sample. In this case, it is convenient to start with the business operation to make testing easier.

After creating the components, there are test page in the production for business processes and business operations.

However, testing is disabled by default in the production definition to avoid random testing in the production environment.

For details on how to allow "Testing Enables" in Production, use the following settings (the sample Production has been set to "Testing Enabled" in advance):

1) Business Operations

In the sample, two types of business operations are provided.

One operation is to pass the city ' s name to an external Web API via REST and request the acquisition of weather information. The other operation is to give the weather information and the name of the purchased product to the InterSystems IRIS database and ask for the update process.

1)-1 REST Business Operations

Let ' s start by creating an operation that calls an external Web API via REST.

This operation starts the GetKion() method when a [Start.Request](#) message is entered, queries an external site, and returns the weather information in a [Start.Response](#) message.

See [here](#) for code details.

To create a business operation for REST, inherit from EnsLib.REST.Operation .

```
Class Start.GetKionOperation Extends EnsLib.REST.Operation
```

Inheritance of this class provides the following methods in IRIS that match the HTTP methods. Please refer to the [documentation](#) for details.

- GetURL()— used for HTTP GET operations.
- PostURL()— used in HTTP POST operations.
- PutURL()— used in a HTTP PUT operations.
- DeleteURL()— used in a HTTP DELETE operations.

For REST, use the adapter EnsLib.HTTP.OutboundAdapter. Set the adapter name to the **ADAPTER** parameter and the **Adapter** property, as shown in the example.

The **INVOCATION** parameter configures the Queue.

```
Parameter ADAPTER = "EnsLib.HTTP.OutboundAdapter";  
  
Property Adapter As EnsLib.HTTP.OutboundAdapter;  
  
Parameter INVOCATION = "Queue";
```

It is necessary to specify the OpenWeather API key to be obtained at runtime. There is a way to display settings that vary depending on the environment in the production settings.

The procedure is as follows:

1. Define the properties
2. Specify the name of the property you created in the SETTINGS parameter (if there are multiple properties, separate them with commas). Optionally, you can also specify a category (use “ property name: category name ”).

An example code is shown below.

```
/// API?????????  
Property appid As %String;  
/// specify lang option for OpenWeather API (default = ja = japanese)  
Property lang As %String [ InitialExpression = "ja" ];  
  
Parameter SETTINGS = "lang:OpenWeatherMap,appid:OpenWeatherMap";
```

The Production Settings page displays the following. The description in the line immediately before the property definition is also displayed in the production settings page, as shown in the figure.

Then, we will review the message map, which is an essential setting for business operations.

The above definition is defined so that the **GetKion() method** will work when the **Start.Request message** is sent.

In the GetKion() method, the city name can be obtained from the request message ' s Area property passed as input information.

By setting the city name as a parameter of the URL published by the external Web API and calling it, you can obtain **weather information**.

The HTTP server and URL settings are configured in the Production page of the Management Portal. To obtain the settings, use the Adapter property provided by the HTTP outbound adapter.

Example) to specify a URL, use ..Adapter.URL

Use the `GetURL()` method provided by Business Operations for REST to call an external site. The first parameter is the URL to be executed (i.e., the URL specified in the required parameters such as city name). The second parameter is the HTTP response with parameters passed by reference.

Since the weather information is stored in JSON format in the HTTP response, the operation is complete when the data is registered in the response message (=pResponse).

The response message class 's name is specified in the second parameter of the created method by passing reference.

```
Method GetKion(pRequest As Start.Request, Output pResponse As Start.Response) As %Sta
tus
```

To return a response message to the caller, create an instance of the response message, store it in the second parameter variable (pResponse), and set the necessary information in the properties.

```
set pResponse.AreaDescription=weatherinfo.weather.%Get(0).description
set pResponse.KionMax=weatherinfo.main."temp_max"
set pResponse.KionMin=weatherinfo.main."temp_min"
set pResponse.Area=weatherinfo.name
// this code is fit to Japan time because weatherinfo.dt is UTC
set unixEpochFormat=weatherinfo.dt+32400
set dt=$system.SQL.Functions.DATEADD("s",unixEpochFormat,"1970-01-01 00:00:00")
set pResponse.AreaPublicTime=dt
```

Since HTTP responses from external sites are returned in JSON format, the stream that could be obtained from the HTTP response is used to convert it into a dynamic object that is convenient for JSON operations within IRIS.

```
set weatherinfo={}.%FromJSON(tHttpResponse.Data)
```

An example of a returned JSON string is shown below:

```
{
  "coord": {
    "lon": 135.5022,
    "lat": 34.6937
  },
  "weather": [
    {
      "id": 803,
      "main": "Clouds",
      "description": "broken clouds",
      "icon": "04d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 11.38,
    "feels_like": 8.33,
    "temp_min": 11,
    "temp_max": 12.22,
```

```
        "pressure": 1007,  
        "humidity": 62  
    },  
    "visibility": 10000,  
    "wind": {  
        "speed": 2.57,  
        "deg": 220  
    },  
    "clouds": {  
        "all": 75  
    },  
    "dt": 1611820991,  
    "sys": {  
        "type": 1,  
        "id": 8032,  
        "country": "JP",  
        "sunrise": 1611784750,  
        "sunset": 1611822143  
    },  
    "timezone": 32400,  
    "id": 1853909,  
    "name": "Osaka",  
    "cod": 200  
}
```

The maximum temperature, minimum temperature, and weather can be obtained as follows:

```
set pResponse.AreaDescription=weatherinfo.weather.%Get(0).description  
set pResponse.KionMax=weatherinfo.main."temp_max"  
set pResponse.KionMin=weatherinfo.main."temp_min"
```

If you would like to learn more about JSON manipulation in IRIS, please refer to [this article](#) and [documentation](#).

Now, let 's use the production testing tool to see if we can get the weather information properly.

Open the Production page (Management Portal> Interoperability> Configuration> Production), click Start.GetKionOperation, and then click the "Test" button on the "Action" tab.

Specify a city name (Naha, Sapporo, Nagano, Shinjuku, etc.) for Area, and click the "Run Test Service" button.

You can see the test results below, with the maximum and minimum temperatures and the weather listed.

Continue to learn how to use the Trace page.

Selecting a horizontal rectangle such as in the left screen causes the information, in the right screen, to change.

Messages sent and received during the system integration process are automatically saved in the database. Using the message Visual Trace page, you can see in detail what messages were passed to which components in chronological order and whether there was a response or not.

Besides, if an error occurs,

“ An error occurred while sending/receiving/receiving message to the component from to . ”

a red mark will appear where the error occurred so that you can see it. Of course, in addition to tracing, we also have an event log page.

(Management Portal > [Interoperability] > [View] > [Event Log])

Moving on, let ' s check out the operation to request an update to the database.

1)-2 Business operations that request updates to the database

The sample provides two types of operations: [Start.SQLInsertOperation](#) and [Start.InsertOperation](#).

Each of them is an operation to request a database update, but [Start.SQLInsertOperation](#) uses the SQL outbound adapter, while [Start.InsertOperation](#) has no adapter.

The difference between the two is,

operation using the SQL outbound adapter is assumed to be accessed via ODBC/JDBC connections so that the database connection destination can be switched in the production settings.

For operations that do not use adapters, it is assumed that the DB update target is a database within the range of visibility from the production configuration and that no connection destination switching occurs.

The IRIS database can be used to store arbitrary data during system integration. However, suppose the system configuration changes for some reason a few years later, and the need to connect to a database on a different server arises. In that case, the operation without the adapter cannot be continued.

On the other hand, operations using the SQL outbound adapter can be operated if there are no changes processing the content of the destination specification (if there is no problem with the SQL statement to be executed, it can be connected to databases of different products).

During system integration, there may be cases where connection information changes due to external system reasons. Therefore it is vital to have a design that can flexibly respond to changes. For this reason, it is recommended to create components that support external connections in a [loosely coupled](#) manner.

However, suppose there is no change in the configuration in the future. In that case, you can access the database in IRIS without using the ODBC/JDBC connection, so you can choose to use the adapter or not, depending on your usage.

Let ' s take a look at the [Start.SQLInsertOperation](#) code that uses the adapter.

The adapter used in the sample is an SQL outbound adapter, which allows you to request the database to execute SQL statements.

Different adapters provide different methods. Please refer to the [documentation](#) for details on the methods provided by the adapters.

Then review the code for [Start.InsertOperation](#), without using the adapter.

Whether you use an adapter or not, the message map and method definitions for the operation are required. If you do not use an adapter, you do not need to define the " Paramter " and " Property " for the adapter.

Business operations without adapters: In [Start.InsertOperation](#), SQL is executed using ObjectScript (the comment statement is the update process in object operations).

The implementation is satisfactory if the database to be updated is not detached from IRIS.

We found out that operations using adapters provide a reliable method to request processing from the destination. We also confirmed that it is possible to create operations without using adapters and freely write code for them.

Next, I would like to explain how to create a business process that calls the operations for getting weather information and updating the database in the correct order.

[#Beginner](#) [#Business Operation](#) [#Interoperability](#) [#InterSystems IRIS](#) [#InterSystems IRIS for Health](#)

Source

URL:<https://community.intersystems.com/post/intersystems-iris-first-time-interoperability-creating-components-business-operation>