Article
Mihoko Iijima · Mar 5, 2021 4m read

[InterSystems IRIS for the first time] Interoperability: Message

This article is a follow-up to this post.

In the previous article, [What is a Production?], we checked the production contents. We ran the sample code and checked the flowing messages' contents into the production on the Visual Trace page.

This article will review the concept and the definition of the **messages** used to send and receive data between components from the required development content for system integration.

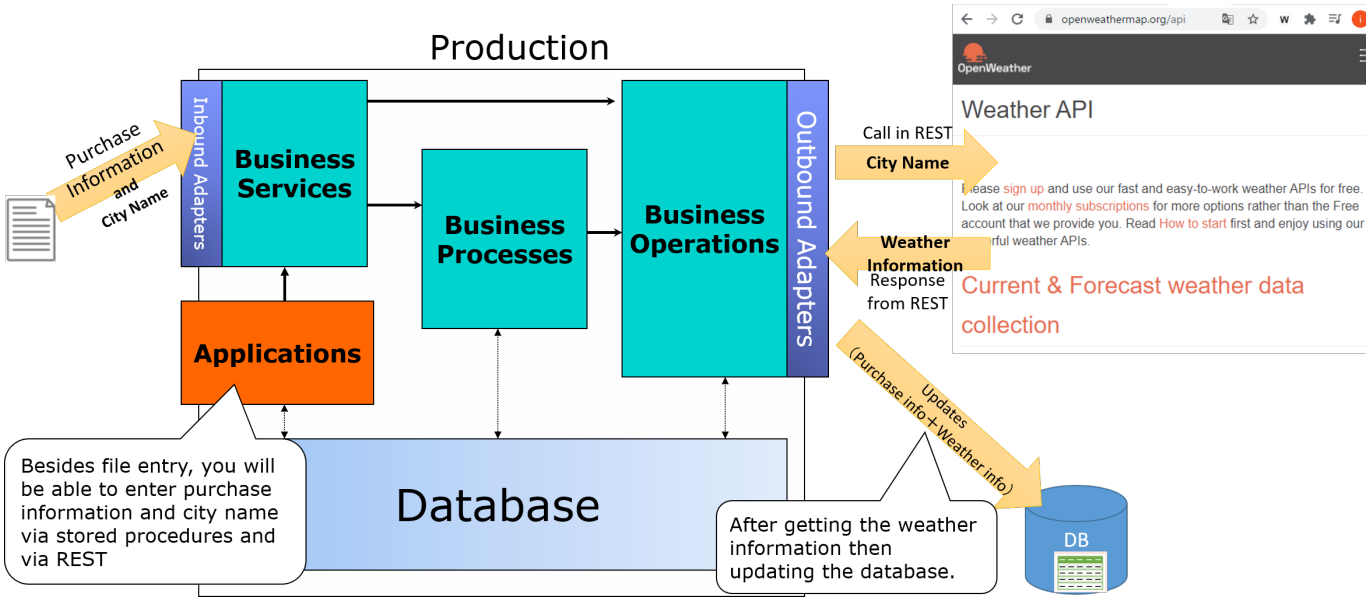
- Production[previous post]
- **Message**
- Components
  - Business Services
  - Business Processes
  - Business Operations

Before creating a **message**, let's review the case study.

> A company operates a shopping site and is changing the order of displaying product information to match the seasons.
> However, some items sell well regardless of the season, while others sell at unexpected times, which does not match the current display rule of changing the order.
> Therefore, we looked into the possibility of changing the order of display to match the day's temperature instead of the season. It became necessary to survey the temperature of the purchased products at that time.
> Since an external Web API is available for checking weather information, we plan to collect the weather information at the time of purchase and register it in the later review database.

From this case, you can see the following:

Information to be received from the outside is "purchased product name and name of the city."
The information to be sent from IRIS to an external system to request processing is the "name of the city." The result of this process will be "weather information" for the city used as an input.



From this case study, we will implement the components needed for system integration. But before that, to run the components, you need to send and receive **messages**, which are relay data, and to use the **messages**, you need to define the message class.

A message class is designed to consider **what information (i.e., messages) should be sent and received to make the component to run.**

In this process, we need the following two types of information:

A) The **name of the city** to send to an external Web API to obtain **weather information.**

B) **weather information** and **purchased product name** for recording in the DB.

The **name of the city** in A) and the **purchased product name** in B) can be included in the input information to IRIS.

The **weather information** in B) can be retrieved from the response information of an **external Web API.**

Here is a diagram that considers what information would be needed to send and receive each component from the available data.

The **first line** of the **yellow balloons** describes the **message class name**, and the second line onwards states what to set for the property.

In the sample code, we have the following three types of **messages**:

Start.Request **(Request message)**
It is used to send the name of the ***purchased product*** and the **city** to acquire the weather information.

Start.Response **(Response message)**
They are used to return the results of operations (**weather information**) to obtain weather information.

Start.InsertRequest **(Request message)**
It is used to send the **weather information** and the **name of the purchased products** for DB registration.

The **messages** are specified in a superclass, Request message, and Response message are derived from **Ens.Request** and **Ens.Response**, respectively.

The following is an example of the definition of the Request message Start.Request.

Below is an example of the definition of the Response message.

The Request message, Start.InsertRequest to be sent with the DB registration request is as follows:

(We plan to set the WeatherInfo property to the information in Start.Response, which will be returned after the weather information is obtained.)

If you want to create it in Studio, you can also use the Message Creation Wizard.

Reference) Steps to create a response class in Studio

The key point so far

A message class is designed with the idea of "what information (i.e., **messages**) should be sent and received" to make the component RUN.

Once the message class (which is the information to drive each component) is implemented, the next step is to create a class for the component.

#Beginner #Interoperability #InterSystems IRIS #InterSystems IRIS for Health