Article
Eduard Lebedyuk · Dec 17, 2019 3m read

Open Exchange

Python Gateway IV: Interoperability Adapter

This series of articles would cover Python Gateway for InterSystems Data Platforms. Execute Python code and more from InterSystems IRIS. This project brings you the power of Python right into your InterSystems IRIS environment:

- Execute arbitrary Python code
- Seamlessly transfer data from InterSystems IRIS into Python
- Build intelligent Interoperability business processes with Python Interoperability Adapter
- Save, examine, modify and restore Python context from InterSystems IRIS

Other articles

The plan for the series so far (subject to change).

- Part I: Overview, Landscape, and Introduction
- Part II: Installation and Troubleshooting
- Part III: Basic functionality
- Part IV: Interoperability Adapter <-- you're here
- Part V: Execute function
- Part VI: Dynamic Gateway
- Part VII: Proxy Gateway
- Part VIII: Use cases and ML Toolkit

Intro

You now have tried Python Gateway from a terminal, time to start using it via Interoperability Productons.
In this article, I would cover the main Interoperability interface to Python - isc.py.ens.Operation. It allows us to:

- Execute Python code and return requested variables (string/stream)
- Save/Restore context
- Load data into Python

Generally speaking, it's an Interoperability wrapper over isc.py.Main. Interoperability adapter isc.py.ens.Operation offers the ability to interact with the Python process from Interoperability productions. Currently, five requests are supported:

- Execute Python code via isc.py.msg.ExecutionRequest. Returns isc.py.msg.ExecutionResponse with requested variable values
- Execute Python code via isc.py.msg.StreamExecutionRequest. Returns isc.py.msg.StreamExecutionResponse with requested variable values. Same as above, but accepts and returns streams instead of strings.
- Set dataset from SQL Query with isc.py.msg.QueryRequest. Returns Ens.Response.
- Set dataset faster from Global/Class/Table with isc.py.msg.GlobalRequest/isc.py.msg.ClassRequest/isc.py.msg.TableRequest. Returns Ens.Response.

- Save the Python context via isc.py.msg.SaveRequest. Returns Ens.StringResponse with context id.
- Restore Python context via isc.py.msg.RestoreRequest.

  isc.py.ens.Operation has but two settings:

  - Initializer - select a class implementing isc.py.init.Abstract. It can be used to load functions, modules, classes and so on. It would be executed at process start.
  - PythonLib - (Linux only) if you see loading errors set it to libpython3.6m.so or even to a full path to the shared library.

Writing Business Process

There are two utility classes to ease BP development:
- isc.py.ens.ProcessUtils allows annotation fetching with variable substitution
- isc.py.util.BPEmulator allows easy testing of Python Interoperability business processes. It can execute a business process (python parts) in a current job.

Variable substitution

All business processes inheriting from isc.py.ens.ProcessUtils can use GetAnnotation(name) method to get value of activity annotation by activity name. Activity annotation can contain variables which would be calculated on ObjectScript side before being passed to Python. This is the syntax for variable substitution:

- ${class:method:arg1:...:argN} - execute method
- #{expr} - execute ObjectScript code

Check test isc.py.test.Process business process for example in Correlation Matrix: Graph activity:
f.savefig(r'#{process.WorkDirectory}SHOWCASE${%PopulateUtils:Integer:1:100}.png')

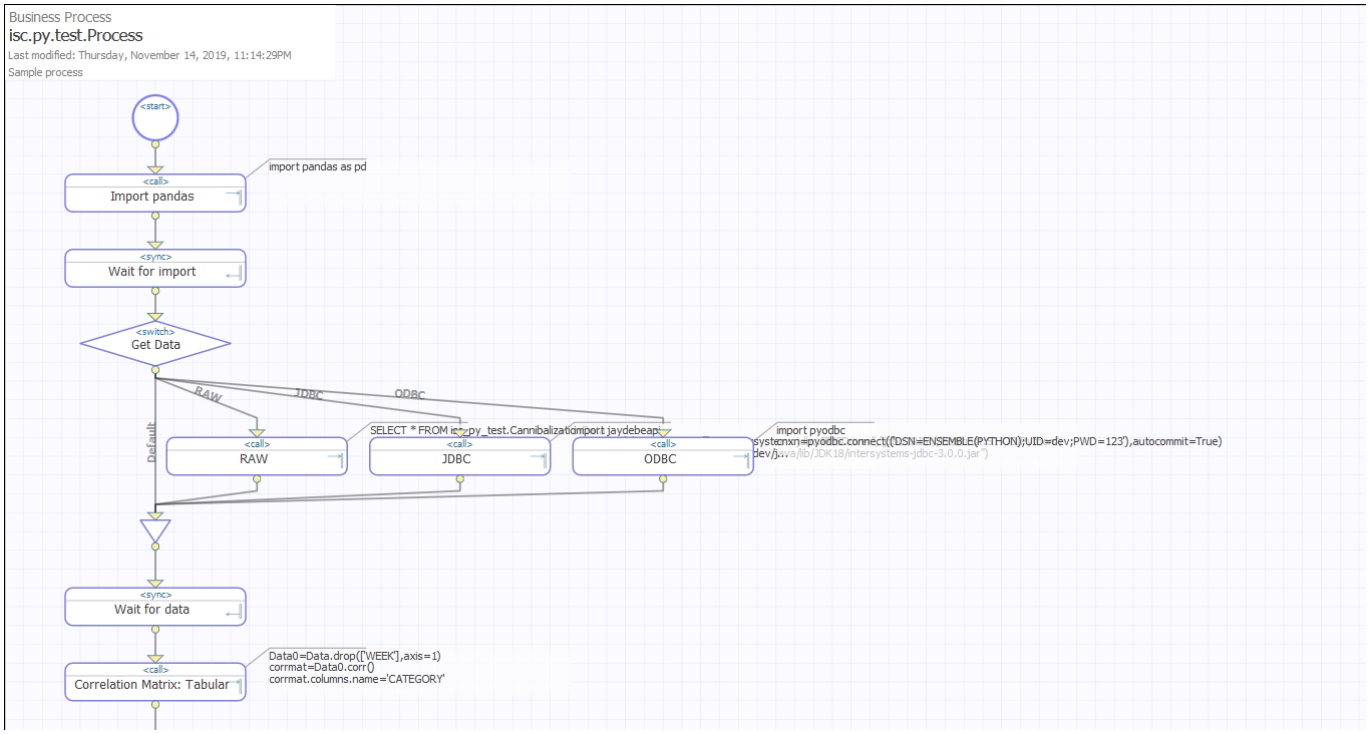
In this example:
- #{process.WorkDirectory} returns WorkDirectory property of process object which is an instance of isc.py.test.Process class and current business process.
- ${%PopulateUtils:Integer:1:100} calls Integer method of %PopulateUtils class passing arguments 1 and 100, returning random integer in range 1...100.

Test Business Process

Test Interoperability Production and test Business Process are available by default as a part of the Python Gateway. To use them:

1. In OS bash execute pip install pandas matplotlib seaborn.
2. Execute: do ##class(isc.py.test.CannibalizationData).Import() to populate test data.
3. Start isc.py.test.Production production.
4. Send empty Ens.Request message to the isc.py.test.Process.

Let's see how it all works together. Open isc.py.test.Process in BPL editor (or Studio):



Code execution call

Here's the most important call - to execute Python code:

Request is isc.py.msg.ExecutionRequest and the properties are:
- Code - Python code to execute.
- SeparateLines - Separate incoming message into lines for execution. $c(10) (\n) is used for line separation. Note that it's NOT recommended to process the whole message at once, this feature is only for def and similar multi-line expressions processing. Defaults to 0.
- Variables - Comma-separated list of variables to get in response message.
- Serialization - How to serialize variables we want to return with Str, Repr, JSON, Pickle and Dill options, defaulting to Str.

In our case, we're only setting Code property so all other properties are defaults. We set it by calling process.GetAnnotation("Import pandas") which at runtime returns annotation after performing variable substitution. Eventually, import pandas as pd string would be passed to Python. GetAnnotation can be useful to set up multiline python scripts, but there's no restriction on it. You can set Code property any way you like.

Variable retrieval call

Another interesting call using isc.py.msg.ExecutionRequest is Correlation Matrix: Tabular:

It calculates Correlation Matrix on a Python side and retrieves corrmat baack into InterSystems IRIS in a JSON format, by setting request properties:
- Variables: "corrmat"
- Serialization: "JSON"

We can see results in a visual trace:

And if we need it down the line in BP it can be saved with: callresponse.Variables.GetAt("corrmat")

Data transfer call

Next, let's talk about InterSystems IRIS -> Python data transfer, all data transfer requests extend isc.py.msg.DataRequest which supplies the following common properties:
- Variable - Python variable to set.
- Type - Variable type (Currently supported: dataframe (pandas dataframe) and list.
- Namespace - Namespace in which we get the data. 'isc.py' package must be available in this namespace.

Building on that are 4 concrete classes:
- isc.py.msg.QueryRequest - set Query property to transfer SQL resultset.
- isc.py.msg.ClassRequest - set Class property to transfer class data.
- isc.py.msg.TableRequest - set Table property to transfer a whole table.
- isc.py.msg.GlobalRequest - set Global property to transfer a global.

In the test process check RAW call to see isc.py.msg.QueryRequest in action.

Save/Restore Python context calls

Finally, we can persist Python context to InterSystems IRIS, to do that send isc.py.msg.SaveRequest with:
- Mask - Only variables that satisfy Mask are saved. Wildcards * and ? are accepted. Example: "Data*,Figure?". Defaults to *.
- MaxLength - Maximum length of the saved variable. If variable serialization is longer than that, it would be ignored. Set to 0 to get them all. Defaults to $$$MaxStringLength.
- Name - Context name (optional).
- Description - Extended context info (optional).

Check Save Context call in the test process for example. Returns Ens.StringResponse with Id of a saved context.

Corresponding isc.py.msg.RestoreRequest loads context from InterSystems IRIS to Python:
- ContextId - Context identifier to restore.
- Clear - Clear context before the restore.

Summary

Python Gateway allows seamless integration between InterSystems IRIS and Python. Use it to add Python functionality to your Interoperability productions.

Links

- Python Gateway
- Python Gateway Samples
- Install Python 3.6.7 64 bit
- Python documentation/tutorial

Illustrated guide

There's also an illustrated guide in ML Toolkit user group. ML Toolkit user group is a private GitHub repository set up as part of the InterSystems corporate GitHub organization. It is addressed to the external users that are installing, learning or are already using ML Toolkit components including Python Gateway. To join ML Toolkit user group, please send a short e-mail at the following address: MLToolkit@intersystems.com and indicate in your e-mail the following details (needed for the group members to get to know and identify you during discussions):

- GitHub username
- Full Name (your first name followed by your last name in Latin script)
- Organization (you are working for, or you study at, or your home office)
- Position (your actual position in your organization, or “ Student ” , or “ Independent ”)
- Country (you are based in)

#Business Operation #Business Process (BPL) #Interoperability #InterSystems IRIS
Check the related application on InterSystems Open Exchange