First Look: InterSystems IRIS Native API for Python

Version 2019.2
2019-09-12

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

First Look: InterSystems IRIS Native API for Python
InterSystems IRIS Data Platform Version 2019.2 2019-09-12
Copyright © 2019 InterSystems Corporation
All rights reserved.

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

InterSystems IRIS Data Platform, InterSystems IRIS, InterSystems iKnow, Zen, and Caché Server Pages are 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

Table of Contents

First Look: InterSystems IRIS Native API for Python

This First Look guide explains how to access InterSystems IRIS globals from a Python application using the InterSystems IRIS™ Native functionality. IRIS Native also allows you to run ObjectScript methods, functions, and routines. In this exploration, you will first connect to InterSystems IRIS. You will then set and retrieve the value of a global node in InterSystems IRIS and iterate over the nodes of another global. You will also call an InterSystems IRIS class method. All of these activities will be performed from a Python 3 application.

To give you a taste of IRIS Native without bogging you down in details, this exploration is intentionally simple. These activities are designed to only use the default settings and features, so that you can acquaint yourself with the fundamentals of the feature without having to deal with details that are off-topic or overly complicated. When you bring IRIS Native to your production systems, there may be things you will need to do differently. Be sure not to confuse this exploration of IRIS Native with the real thing! The sources provided at the end of this document will give you a good idea of what is involved in using IRIS Native in production.

To browse all of the First Looks, including those that can be performed on a free evaluation instance of InterSystems IRIS, see InterSystems First Looks.

1 Introduction to InterSystems IRIS Storage Structures

InterSystems IRIS provides an easy-to-use way to store data in persistent, multidimensional arrays. A *global* is a named multidimensional array that is stored within a physical InterSystems IRIS database. Within an application, the mappings of globals to physical databases is based on the current namespace — a namespace provides a logical, unified view of one or more physical databases. As an example, to associate the value "Red" with the key "Color" using a global named ^*Settings*, open the InterSystems IRIS Terminal using the procedure described for your instance in *InterSystems IRIS Basics: Connecting an IDE*, and enter the following code:

```
set ^Settings("Color")="Red"
```

You can take advantage of the multidimensional nature of globals to define a more complex structure:

```
set ^Settings("Auto1","Properties","Color") = "Red"
set ^Settings("Auto1","Properties","Model") = "SUV"
set ^Settings("Auto2","Owner") = "Mo"
set ^Settings("Auto2","Properties","Color") = "Green"
```

For more information on globals, see *Using Globals*.

2 Exploring IRIS Native for Python

At this point, you are ready to experiment with IRIS Native. The following brief demo shows you how to work with IRIS Native in a simple Python application.

Want to try an online video-based demo of the InterSystems Native API for Python? Check out the Python QuickStart!

2.1 Before You Begin

To use the procedure, you need a running instance of InterSystems IRIS and a system to work on, with Python 3 with the PIP utility, and your favorite Python IDE. (On Windows systems, make sure the Python interpreter is included in your PATH environment variable.) Your choices for InterSystems IRIS include several types of licensed and free evaluation instances; the instance need not be hosted by the system you are working on (although they must have network access to each other). For information on how to deploy each type of instance if you do not already have one to work with, see Deploying InterSystems IRIS in *InterSystems IRIS Basics: Connecting an IDE*. Connect Visual Studio to your InterSystems IRIS instance using the information in InterSystems IRIS Connection Information and Python IDEs in the same document.

2.2 Install the Native API Package

Before running the sample Python program, you need to use the PIP package manager to install the `irisnative` package, which you can download from https://github.com/intersystems/quickstarts-python/tree/master/Solutions/nativeAPI_wheel. If InterSystems IRIS is installed on your local system, the wheel file is already installed in the subdirectory *install-dir*\dev\python, where *install-dir* is the installation directory for your instance of InterSystems IRIS and the name of the Python 3 package (.whl file) includes `cp34`.

To install the wheel file, from the command line, enter the command **pip install** *irisnative-1.0.0 whl file*, where *irisnative-1.0.0 whl file* is the wheel file that corresponds to your Python version and operating system. For example, if you are using Python 3 on Linux, use `irisnative-1.0.0-cp34-abi3-linux_x86_64.whl`.

2.3 The IRIS Native Application

Now that you've created your project, you will create a small application that demonstrates a few of the features of the Native API.

1. In your IDE, create a new source file in a IRISNative directory, saving the file as IRISNative.py.
2. Paste the following code into IRISNative.py, substituting the connection information for your InterSystems IRIS instance for the values in `kwargs`. You can specify the **USER** namespace as shown, or you can choose another that you have created on your instance:

```
import irisnative

# Modify connection info based on your environment
ip = "127.0.0.1"
port = 51773
namespace = "USER"
username = "_SYSTEM"
password = "SYS"

# create database connection and IRIS instance
connection = irisnative.createConnection(ip,port,namespace,username,password)
dbnative = irisnative.createIris(connection)

print("[1. Setting and getting a global]")
# setting and getting a global
# ObjectScript equivalent: set ^testglobal("1") = 8888
dbnative.set(8888, "testglobal", "1")

# ObjectScript equivalent: set globalValue = $get(^testglobal("1"))
globalValue = dbnative.get("testglobal","1")

print("The value of testglobal is ", globalValue)
print()

print("[2 Iterating over a global]")
# modify global to iterate over
# ObjectScript equivalent: set ^testglobal("1") = 8888
# ObjectScript equivalent: set ^testglobal("2") = 9999
dbnative.set(8888, "testglobal", "1")
dbnative.set(9999, "testglobal", "2")

Iter = dbnative.iterator("testglobal")
print("walk forwards")
```

```
    for subscript, value in Iter.items():
        print("subscript= {}, value={}".format(subscript, value))
    print()

    print("[3. Calling a class method]")
    # calling a class  method
    # ObjectScript equivalent: set returnValue = ##class(%Library.Utility).Date(5)
    returnValue = dbnative.classMethodValue("%Library.Utility", "Date", 5)
    print(returnValue)

    # close connection
    connection.close()
```

The example code is split into three sections:

1. The first section shows how you set the value of a global and later retrieve it. The commands executed in this section are equivalent to the ObjectScript commands **SET** and **GET**.
2. The second section shows how to iterate through the subnodes of a global, similar to the **$ORDER** ObjectScript function.
3. The third section shows how you call an ObjectScript class method from your Python application using IRIS Native.

Note: Globals in ObjectScript begin with the caret character (^). This is not a requirement in your Python applications that use the InterSystems IRIS Native API.

2.4 Running the Exercise

Now you are ready to run the demo application. If the example executes successfully, you should see printed output with the results of the sample code:

```
[1. Setting and getting a global]
The value of testglobal(1) is 8888

[2. Iterating over a global]
walk forwards
subscript=1, value=8888
subscript=2, value=9999

[3. Calling a class method]
Jan 30, 2018
```

2.5 Confirming the Changes in the Management Portal

Next, confirm your results in the Management Portal, using the following procedure:

1. Open the Management Portal for your instance in your browser, using the URL described for your instance in *InterSystems IRIS Basics: Connecting an IDE*.
2. If you are not in the namespace you specified in the code, switch to it.
3. Navigate to the **Globals** page (**System Explorer** > **Globals**). You should see the *testglobal* global created in the example code. Click **View** to see its contents. You should see the two nodes of the global: `^testglobal(1) = 8888` and `^testglobal(2) = 9999`.

3 Learn More About IRIS Native

For more information on IRIS Native, globals, and InterSystems IRIS, see:

Using the Native API for Python

Using Globals