
Article

[Nicholai Mitchko](#) · Apr 12, 2022 7m read

Some Programmatic Interoperability Examples

Programmatic Production Access

To Programmatically Edit Productions (interfaces) you can use a combination of the interoperability apis and SQL queries.

Current Namespace

At a high level, it is important to know the namespace and production you are working in at the moment.

```
// Object script
// The active namespace is stored in this variable
$$$NAMESPACE
// Print namespace
Write $$$NAMESPACE

# Python
import iris
# The active namespace is returned from this method
iris.utils._OriginalNamespace()
# Print namespace
print(iris.utils._OriginalNamespace())
>>> DEMONSTRATION
```

Current Production (active or last run production)

Also, it is important to know the name of your production, you can get the active production in a namespace using the following APIs

```
// ObjectScript
USER>ZN "DEMONSTRATION"
// Get current or last run production
DEMONSTRATION>W ##class(Ens.Director).GetActiveProductionName()
>>> Hospital.HospitalProduction

# Python
import os
os.environ['IRISNAMESPACE'] = 'DEMONSTRATION'
import iris
active_production = iris.cls('Ens.Director').GetActiveProductionName()
print(active_production)
>>> Hospital.HospitalProduction
```

Finding Items in a production

You can use objectscript or python to find active items in the production

1. SQL Query for Items in production

```
SELECT Name FROM Ens_Config.Item Where Production = 'Hospital.HospitalProduction'
--
['From_Athena_Multi']
['From_Athena_Multi_Router']
['From_Cerner_ADT']
['From_Cerner_ADT_Router']
['From_Cerner_Orders']
['From_Cerner_Orders_Router']
['From_Dictaphone_Results']
['From_Dictaphone_Results_Router']
['From_Lab_Results']
['From_Lab_Results_Router']
['From_Radiology_Results']
['From_Radiology_Results_Router']
['HS.IHE.XDSb.DocumentSource.Operations']
['HS.IHE.XDSb.Repository.Operations']
['To_Cerner_Results']
['To_Dictaphone']
['To_Intellilab']
['To_Lab']
['To_Radiology']
--
```

2. SQL Query for Active Items in Production

```
SELECT Name, ClassName
FROM Ens_Config.Item
WHERE Production = 'Hospital.HospitalProduction'
      AND Enabled = 1
```

```
--
Name                               ClassName
To_Radiology                       EnsLib.HL7.Operation.FileOperation
To_Lab                             EnsLib.HL7.Operation.FileOperation
To_Dictaphone                      EnsLib.HL7.Operation.FileOperation
From_Cerner_ADT                    EnsLib.HL7.Service.FileService
From_Cerner_ADT_Router              EnsLib.HL7.MsgRouter.RoutingEngine
From_Radiology_Results_Router       EnsLib.HL7.MsgRouter.RoutingEngine
From_Lab_Results_Router             EnsLib.HL7.MsgRouter.RoutingEngine
From_Dictaphone_Results_Router      EnsLib.HL7.MsgRouter.RoutingEngine
To_Intellilab                      EnsLib.HL7.Operation.FileOperation
To_Cerner_Results                   EnsLib.HL7.Operation.FileOperation
From_Cerner_Orders_Router           EnsLib.HL7.MsgRouter.RoutingEngine
From_Athena_Multi_Router            EnsLib.HL7.MsgRouter.RoutingEngine
HS.IHE.XDSb.DocumentSource.Operations HS.IHE.XDSb.DocumentSource.Operations
--
```

3. Object Access for Production Items

Some Programmatic Interoperability Examples

Published on InterSystems Developer Community (<https://community.intersystems.com>)

```
// ObjectScript
// Access to get all items in the active production
// Returns list of items
ClassMethod ListItemsInProduction()
{
    Set productionName = ##class(Ens.Director).GetActiveProductionName()
    Set items = []
    &sql(Declare curr cursor FOR Select Name into :newId from Ens_Config.Item Where P
roduction = :productionName)
    &sql(OPEN curr)
    For {
        &sql(FETCH curr)
        Quit:SQLCODE
        Do items.%Push(newId)
    }
    &sql(CLOSE curr)
    quit items
}
```

```
>>> zw ##class(ISC.SE.ProductionTools).ListItemsInProduction()
```

```
["From_Athena_Multi","From_Athena_Multi_Router","From_Cerner_ADT","From_Cerner_ADT_Ro
uter","From_Cerner_Orders","From_Cerner_Orders_Router","From_Dictaphone_Results","Fro
m_Dictaphone_Results_Router"
,"From_Lab_Results","From_Lab_Results_Router","From_Radiology_Results","From_Radiolog
y_Results_Router","HS.IHE.XDSb.DocumentSource.Operations","HS.IHE.XDSb.Repository.Ope
rations","To_Cerner_Results"
,"To_Dictaphone","To_Intellilab","To_Lab","To_Radiology"] ; <DYNAMIC ARRAY>
```

```
# Python
```

```
# Get Dataframe of active production items
```

```
import os
```

```
# Set environment variables
```

```
os.environ['IRISNAMESPACE'] = 'DEMONSTRATION'
```

```
import iris
```

```
def getActiveProductionItems():
```

```
    productionName = iris.cls('Ens.Director').GetActiveProductionName()
```

```
    df = iris.sql.exec("SELECT Name FROM Ens_Config.Item Where Production = '{ }'.for
mat(productionName))
```

```
    return df
```

```
production_items_df = getActiveProductionItems().dataframe()
```

```
#                                     name
# 0                                From_Athena_Multi
# 1                        From_Athena_Multi_Router
# 2                                From_Cerner_ADT
# 3                        From_Cerner_ADT_Router
# 4                                From_Cerner_Orders
# 5                        From_Cerner_Orders_Router
# 6                                From_Dictaphone_Results
# 7                        From_Dictaphone_Results_Router
# 8                                From_Lab_Results
# 9                        From_Lab_Results_Router
# 10                               From_Radiology_Results
# 11                       From_Radiology_Results_Router
```

```
# 12 HS.IHE.XDSb.DocumentSource.Operations
# 13     HS.IHE.XDSb.Repository.Operations
# 14     To_Cerner_Results
# 15     To_Dictaphone
# 16     To_Intellilab
# 17     To_Lab
# 18     To_Radiology
```

Manipulating Production via API

1. Adding Component

```
// ObjectScript
set productionName = ##class(Ens.Director).GetActiveProductionName()
//create a new xml file service
set classname="EnsLib.XML.FileService" //class of this item
set name="NewService" //config name
set item=##class(Ens.Config.Item).%New(classname)

set item.Name=name
set item.Comment = "Test Service"
set item.PoolSize = "1"
set item.Enabled = 1
do item.%Save()
//
// open the production class
// set prod="Test.configtest" //production name set manually
// OR
set prod = productionName
set prodObj=##class(Ens.Config.Production).%OpenId(prod)
//save the new item
set tSC=prodObj.Items.Insert(item)
set tSC=prodObj.SaveToClass(item)
set tSC=prodObj.%Save()

// DELETE item from above
set tSC = prodObj.RemoveItem(item)
set tSC = prodObj.SaveToClass()
set tSC=prodObj.%Save()

# Python
import os
os.environ['IRISNAMESPACE'] = 'DEMONSTRATION'
import iris
active_production = iris.cls('Ens.Director').GetActiveProductionName()
print("Current Production {}".format(active_production))

# Metadata about component
classname="EnsLib.XML.FileService" # class of this item
name="NewService" # config name
item=iris.cls('Ens.Config.Item')._New(classname) # Make new component
item.Name=name
item.Comment = "Test Service"
item.PoolSize = "1"
item.Enabled = 1
item._Save()
```

```
# open the production class
# prod="Test.configtest"      # production name manually set
# OR use the active production from above
prod = active_production

prodObj=iris.cls('Ens.Config.Production')._OpenId(prod)
# save the production after we insert that item to it
tSC=prodObj.Items.Insert(item)
tSC=prodObj.SaveToClass(item)
tSC=prodObj._Save()

# DELETE item from above
tSC = prodObj.RemoveItem(item)
tSC = prodObj.SaveToClass()
tSC=prodObj._Save()
```

2. Disabling / Enabling Component

```
// ObjectScript
set productionName = ##class(Ens.Director).GetActiveProductionName()
set itemName = "My.Inbound.HL7"
// Required for Enable Item
Set componentName = productionName _ "||" _ itemName _ "|"
// Disable or enable
Set enable = 1 // or 0
Do ##class(Ens.Director).EnableConfigItem(componentName, enable, 1)

/// Enable or disable a ConfigItem in a Production. The Production may be running or
not.
/// The pConfigItemName argument gives the name of the config item to be enabled or d
isabled
/// In the case of multiple matching items with the same config name, if any is alrea
dy enabled then
/// the pEnable=1 option will do nothing and the pEnable=0 option will disable the r
unning matching
/// production item, or if not running then the first matching enabled item that it
finds.
///
/// See method Ens.Director.ParseConfigName() for full syntax of the ConfigItem name
specification string.
ClassMethod EnableConfigItem(pConfigItemName As %String, pEnable As %Boolean = 1, pDo
Update As %Boolean = 1)

# Python
import os
os.environ['IRISNAMESPACE'] = 'DEMONSTRATION'
import iris

active_production = iris.cls('Ens.Director').GetActiveProductionName()
item_name = "My.Inbound.HL7"
componentName = active_production + "||" + item_name + "|"

enable = 1 # or 0
iris.cls('Ens.Director').EnableConfigItem(componentName, enable, 1)
```

Production Status via API

```
// ObjectScript
/// This method returns the production status via the output parameters.
/// pProductionName: Returns the production name when the status is running, suspended or troubled.
/// pState: Outputs production status. The valid values are:
///         $$$eProductionStateRunning == 1
///         $$$eProductionStateStopped == 2
///         $$$eProductionStateSuspended == 3
///         $$$eProductionStateTroubled == 4
Set sc = ##class(Ens.Director).GetProductionStatus(.productionName, .productionState)
```

```
Write productionName, " -- ", productionState
```

```
import os
# Set namespace the hard way
os.environ['IRISNAMESPACE'] = 'DEMONSTRATION'

import iris

# TEST 2 with output variables
productionName, productionState = iris.ref('productionName'), iris.ref('productionState')
status = iris.cls('Ens.Director').GetProductionStatus(productionName, productionState)

print("Status: {}".format(status))
# see .value
print("Production: {}".format(productionName.value))
# see .value
print("Production State: {}".format(productionState.value))
```

[#Embedded Python](#) [#Interoperability](#) [#ObjectScript](#) [#InterSystems IRIS](#) [#InterSystems IRIS for Health](#)

Source URL: <https://community.intersystems.com/post/some-programmatic-interoperability-examples>