Integrating InterSystems IRIS with Source Control Systems

Version 2024.2
2024-09-05

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

Integrating InterSystems IRIS with Source Control Systems
PDF generated on 2024-09-05
InterSystems IRIS® Version 2024.2

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

Integrating InterSystems IRIS with Source Control Systems

This page explains briefly how to place InterSystems IRIS® data platform code under source control by connecting to a third-party source control system. It does not discuss how to choose an appropriate strategy or tools for your use case.

1 Basics

The process of placing InterSystems IRIS code under source control consists of the following steps:

- If necessary, creating a custom source control class.
- Configuring the namespace or namespaces to use the source control class you have chosen.
- Also configuring the same namespaces to specify the location of the files to be loaded into the code databases. The way to perform this configuration generally depends on the source control class being used, but a common approach is to contain the relevant information in a global.

1.1 InterSystems IRIS Documents

On this page, the term *document* refers to any discrete item that you place within a source control system; this includes class definitions, routines, and data transformations. InterSystems IRIS records information such as whether a document has changed since the last compilation. Your source control system treats each document as a separate unit.

Each document has two names:

- An internal name.
- An external name, which is the fully qualified path to the file in the file system.

2 Creating a Source Control Class

To create a source control class, do the following:

1. Create a subclass of %Studio.Extension.Base or %Studio.SourceControl.Base.
2. If you started with %Studio.Extension.Base, create an XDATA block named `Menu` in your subclass. (Copy and paste from %Studio.SourceControl.Base to start this.)
3. Implement the methods of this class as needed: **AddToSourceControl()**, **CheckIn()**, **CheckOut()**, and so on. These methods would typically do the following, at a minimum:
    - If appropriate, import or export the InterSystems IRIS document to a file.
    - Call the appropriate function or method of your source control software, to act on the file.
    - Update internal information in InterSystems IRIS about the status of the given file.

- Control whether the InterSystems IRIS document is editable.

The details depend upon the source control system.

4. Implement the **GetStatus()** method of your source control class. This is required. You might also need to implement the **IsInSourceControl()** method, if the default implementation is not suitable.

2.1 Deciding How to Map Internal and External Names

Creating a bidirectional mapping between internal and external names may be one of the most challenging parts of implementing a source control system. Your code will need to define this mapping, which means that it must do the following:

- Given the name of an internal item, determine the corresponding filename.
- Given a changed file, determine the corresponding internal item and update it by importing the changed file.

For example, you could use a directory structure like this:

- Class files are in the cls subdirectory, which contains subdirectories corresponding to the package hierarchy of the classes.
- .INT routines are in the int subdirectory.
- .MAC routines are in the mac subdirectory.

So that multiple developers in different environments can use your custom source control class, your code must also provide a way to configure the basic location of the files (rather than simply hardcoding the mapping completely).

2.2 Tools for Managing Documents and Files

InterSystems IRIS provides the following tools for managing documents and external files:

- To export InterSystems IRIS documents, use one of the following, depending on the file format you want to use:
  - For UDL files, use **$system.OBJ.ExportUDL()**.
  - For XML files, use **$system.OBJ.Export()**.

  UDL files more closely resemble the code as seen in an IDE.
- To import files (of either format), use **$system.OBJ.Load()**.
- To obtain the timestamp and compile time for an InterSystems IRIS document, use the **%RoutineMgr.TS** class method.

2.3 Accessing Your Source Control System

The API for your source control system provides methods or functions to perform source control activities such as checking files out. Your source control class will need to make the appropriate calls to this API, and the InterSystems IRIS server will need to be able to locate the shared library or other file that defines the API itself.

Also, it is important to remember that InterSystems IRIS will execute the source control commands on the InterSystems IRIS server. This means that your XML files will be on the InterSystems IRIS server, and your file mapping must work on the operating system used on that server.

2.3.1 Example 1

For the following fragment, we have created wrapper methods for the API for VSS. Then we can include code like the following within the source control methods:

```
 do ..VSSFile.CheckIn(..VSSFile.LocalSpec,Description)
```

The details depend on the source control software, its API, and your needs.

2.3.2 Example 2

The following fragment uses a Windows command-line interface to check out a file. In this example, the source control system is Perforce:

```
/// Check this routine/class out of source control.
Method CheckOut(IntName As %String, Description As %String) As %Status
{
  Set file=..ExternalName(IntName)
  If file="" Quit $$$OK
  //...
 Set cmd="p4 edit """_file_""""


  #; execute the actual command
  Set sc=..RunCmd(cmd)
  If $$$ISERR(sc) Quit sc

  #; If the file still does not exist or
  #; if it is not writable then checkout failed
  If '##class(%File).Exists(file)||(##class(%File).ReadOnly(file)) {
    Quit $$$ERROR($$$GeneralError,
                  "Failure: '"_IntName_"' not writeable in file sys")
  }

  #; make sure we have latest version
  Set sc=..OnBeforeLoad(IntName)
  If $$$ISERR(sc) Quit sc

  //...
  Quit sc
}
```

In this example, **RunCmd()** is another method, which executes the given command and does some generic error checking. (**RunCmd()** issues the OS command via the callout interface.)

Also, this **CheckOut()** method calls the **OnBeforeLoad()** method, which ensures that the InterSystems IRIS document and the external XML file are synchronized.

3 Activating a Source Control Class

To activate a source control class for a given namespace, do the following in the Management Portal:

1. Select **System Administration** > **Configuration** > **Additional Settings** > **Source Control**.
2. On the left, select the namespace to which this setting should apply.
3. Select the name of the extension class to use and select **OK**.

   This list includes all compiled subclasses of %Studio.Extension.Base in the selected namespace.

In your IDE, you may need to reopen your workspace in order to see the new menu options.

The Management Portal Production Configuration Page also supports source control, see Configuring Source Control Settings.

4 See Also

- Configuring Source Control Settings (options related to interoperability productions)
- Introduction to Visual Studio Code
- Video: Visual Studio Code for ObjectScript: Choosing an IDE/Source Code Combination on the Developer Community
- Using Studio