Integrating InterSystems IRIS with Source Control Systems

Version 2024.1
2024-05-02

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

Integrating InterSystems IRIS with Source Control Systems
InterSystems IRIS Data Platform Version 2024.1 2024-05-02

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 how to place InterSystems IRIS® data platform code under source control by connecting to a third-party source control system.

1 Overview

Placing an InterSystems IRIS development project under source control involves the following steps:

- Export code units to a file system as XML files. In the context of source control, each code unit is a *document*.
- Place the XML files under source control.
- Keep XML files synchronized with InterSystems and in terms of both content and read-write state.
- Ensure that you can perform source control activities from within InterSystems IRIS.
- Keep InterSystems IRIS and the source control system synchronized as to whether the document has been checked out, and by whom.

2 InterSystems IRIS Documents

Documents include 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.

In InterSystems IRIS and your source control system, you work within one namespace at a time.

2.1 Tools for Managing Documents and Files

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

- The %Studio.Extension.Base class provides methods for basic document management. %Studio.SourceControl.Base is a subclass of %Studio.Extension.Base that also provides a source control menu. See Creating and Activating a Source Control Class.
- The **$system.OBJ.Export** function exports an InterSystems IRIS document to an XML file in the external document system. The XML file contains sufficient information to reconstruct the document.
- The **$system.OBJ.Load** function loads an external XML file and overwrites the corresponding InterSystems IRIS document, if one exists.
- The **%RoutineMgr.TS** class method returns the timestamp and compile time for an InterSystems IRIS document.

2.2 Deciding How to Map Internal and External Names

Each document has two names:

- An internal name. For example, the name used in the **Open** dialog box in Studio.
- An external name. The fully-qualified path to the file in the file system. Because of differences in how file paths are constructed between supported InterSystems IRIS platforms, it is not possible to provide a meaningful default.

Creating a bidirectional mapping between internal and external names may be one of the most challenging parts of implementing a source control system. This mapping is customer-specific and should be considered carefully. You want to group similar items. For example, the sample uses the following directory structure:

- 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.

For example:

- Internal name — MyApp.Addresses.HomeAddress
- External name — C:\sources\cls\MyApp\Addresses\HomeAddress.xml

If you have large numbers of routines, you might prefer to group routines into subdirectories, perhaps by function.

3 Creating and Activating a Source Control Class

This section describes the basic requirements for creating and activating a source control class.

3.1 Extending Studio

You can use either %Studio.Extension.Base or %Studio.SourceControl.Base to add a source code control menu to Studio. You can add up two menus with 19 menu items each.

Note: Users must have the %Developer role in order to use the Studio source control menus.

The %Studio.Extension.Base class provides the following methods, which all use the internal name of the InterSystems IRIS document:

- Empty **Login** and **Logout** methods that you can implement as needed. The variable *$username* records the current user. (In the **Login** method, the *Username* argument is provided for backward compatibility; it is recommended that you use the variable *$username* instead.)
- Basic methods to indicate the status of a given InterSystems IRIS document: **GetStatus**, and **IsInSourceControl**. Implement these methods as needed.
- Callback methods that are executed when a user performs some action on an InterSystems IRIS document. These methods include **OnBeforeLoad**, **OnAfterLoad**, **OnBeforeCompile**, **OnAfterCompile**, **ItemIconState**.

Note: Studio class compilation can use multiple processes. Therefore, do not use properties of %Studio.Extension.Base to pass information from MenuItem to **OnBeforeCompile**. Instead, use a temporary global.

%Studio.SourceControl.Base is a subclass of %Studio.Extension.Base and provides the following additional elements:

- An XDATA block named `Menu` that defines an additional menu for InterSystems IRIS: **Source Control**. By default, this menu contains the menu items **Check In**, **Check Out**, **Undo Check Out**, **Get Latest**, and **Add To Source Control**. This XDATA block also defines additional menu items for the context menu in Studio.

  All these menu items call methods also defined in this class.

- Methods named **CheckIn()**, **CheckOut()**, **UndoCheckOut()**, **GetLatest()**, and **AddToSourceControl()**, which do nothing by default.

To extend InterSystems IRIS, you define a new class that extends one of these classes. As you see in Activating a Source Control Class, the Management Portal provides a way to indicate which extension class is currently active in a given namespace. If an extension class is active in a given namespace, and if that class defines an XDATA menu block, those menu items are added to InterSystems IRIS.

3.2 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 XML.
   - Call the appropriate function or method of your source control software, to act on the XML 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.

3.3 Activating a Source Control Class

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

1. Use the Management Portal to specify which extension class, if any, InterSystems IRIS should use for a given namespace. To specify the class to use:
   a. Select **System Administration** > **Configuration** > **Additional Settings** > **Source Control**.
   b. On the left, select the namespace to which this setting should apply.
   c. 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.
2. If Studio is currently open, close it and reopen it, or switch to another namespace and then switch back.

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

4 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.

4.1 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.

4.2 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 **$ZF(-1)** interface.)

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

5 See Also

- Introduction to Visual Studio Code
- Using Studio