



Frequently Asked Questions About Caché Studio

Version 5.1
15 June 2006

Frequently Asked Questions About Caché Studio

Caché Version 5.1 15 June 2006

Copyright © 2006 InterSystems Corporation.

All rights reserved.

This book was assembled and formatted in Adobe Page Description Format (PDF) using tools and information from the following sources: Sun Microsystems, RenderX, Inc., Adobe Systems, and the World Wide Web Consortium at www.w3c.org. The primary document development tools were special-purpose XML-processing applications built by InterSystems using Caché and Java.



The Caché product and its logos are trademarks of InterSystems Corporation.



The Ensemble product and its logos are trademarks of InterSystems Corporation.



The InterSystems name and logo are trademarks of InterSystems Corporation.

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.

Caché, InterSystems Caché, Caché SQL, Caché ObjectScript, Caché Object, Ensemble, InterSystems Ensemble, Ensemble Object, and Ensemble Production are trademarks of InterSystems Corporation. All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Customer Support

Tel: +1 617 621-0700

Fax: +1 617 374-9391

Email: support@InterSystems.com

Table of Contents

Frequently Asked Questions About Caché Studio	1
General	1
Projects	1
Opening Files	2
Debugging	4
Editing	6
Importing Files	6
Printing	7
Templates	7
Multiuser Support	8
Classes	8
Routines	9
SQL	9
Source Control	10
Compatibility	11
Studio Implementation	11

Frequently Asked Questions About Caché Studio

General

How can I learn more about Caché Studio?

The best place to start is to read the [Using Caché Studio](#) guide included with the Caché online documentation.

Projects

What is a project?

A project is simply a collection of class definitions, routines, and/or CSP files that you can group together for the sake of convenience.

Using projects gives you an easy way to return to your work when you start a Studio session. For example, you can place all the classes related to an application, or part of an application, in a project. When you start Studio, open this project and the Project tab of the Workspace window will display all the classes in a convenient list.

You can also export and import entire projects to and from a single external file making it easy to save or pass around application code.

How do I add an item to a project?

There are several ways to add items to a the current project.

- When you open an item or items (using the **Open** command in the **File** menu), select the **Add to Project** check box in the Open Dialog before opening the file.
- Use the **Add Item** command from the **Project** menu to add the item in the current editor window to the current project.

Can I add something from another namespace to my project?

No. A project can only contain items that are visible to the current Caché namespace.

Can an item belong to multiple projects?

Yes. A project is simply a list of items (class definitions, routines, and CSP files). The items themselves have no link back to the projects they may belong to. There is no limit to how many projects an item can belong to.

What if I don't want to use projects?

You are not required to use projects with Studio; you can completely ignore them if you like. Simply do not add any items to the default project and ignore the prompt asking you if you want to save your project when you exit Studio.

Can I export a project?

Yes. Use the **Export** command in the **Tools** menu. In the Export dialog select **Export Project**, enter a file name, and press **OK**. This will export the entire contents of the current project (including the project definition) to a single XML file.

How do I delete a project?

Use the File Open Dialog and list all your projects. Right-click on a project and select **Delete** from the pop-up menu.

Note that you can use the File Open Dialog to delete any type of item on the server in this way.

Opening Files

How do I open a class definition?

To open an existing class definition (i.e., one saved in the Caché server) do the following:

1. Make sure you are connected to the Caché namespace and server containing the class definition.

2. Using the **Open** command in the **File** menu, invoke the Open Dialog.
3. Make sure that class definitions are listed by selecting “Class Definitions” (.CLS) or “All” in the File Types combo box.
4. Package names are listed in the file list as folders. Click on the desired package name to list all the classes (or sub-packages) within the package. When you see the class you wish to open you can double-click on it to open it.
5. Alternatively, you can enter the name of the class you want directly into the filename edit box with a .cls extension (such as Sample.Person.cls) and press **Open**.

How do I open a routine?

To open an existing routine (i.e., one saved in the Caché server) do the following:

1. Make sure you are connected to the Caché namespace and server containing the routine.
2. Using the **Open** command in the **File** menu, invoke the Open Dialog.
3. Make sure that routines are listed by selecting either “MAC routines” (.MAC), “INT routines” (.INT), or “All” in the File Types combo box.
4. Double-click on the desired routine name to open it.
5. Alternatively, you can enter the name of the routine you want directly into the filename edit box with correct extension (such as MyRoutine.MAC) and press **Open**.

How do I open a CSP file?

You can open a CSP file in the same way that you open a class definition or a routine. The main difference is that the Open Dialog lists CSP Applications (e.g., /csp/samples) as folders; click on application to see the CSP pages within it.

What does the Show System check box in the Open Dialog do?

If the **Show System** check box is checked, then the Open Dialog will list system items (items whose names start with the “%” character and are stored in the %CACHELIB database) along with items in the current namespace.

Can I use pattern matching in the Open Dialog?

Yes. You can use the “*” character as you would in a standard File Open dialog. You can use file extensions to filter certain items; for example, “*.cls” will list all Class Definitions in the selected package.

How do I open a routine from a different namespace?

The Studio Open Dialog only lists items from the current namespace and server. To open a routine from a different namespace or server you can either:

1. Connect to the new namespace and/or server using the **Connect** command in the **File** menu and then open the desired routine.
2. Open a routine using the **Open Remote** command in the **File** menu. This allows you to open the routine without first connecting to the new namespace or server.

Can I open a % class?

Yes. You can list “%” classes (classes whose package name starts with a “%” character and are stored within the %CACHELIB database) in the Open Dialog by selecting the **Show System** check box at the bottom of the dialog.

Studio will open “% ” classes as read-only if you open them while connected to a namespace other than %CACHELIB.

What does the Connect command in the File menu do?

Studio maintains a connection to a specific Caché namespace and server. It uses this connection to provide a list of classes (such as for specifying property types, super classes, etc.). It also uses this connection for debugging. The **Connect** command in the **File** menu lets you connect to a different namespace or server.

Debugging

How do I start the debugger?

You can connect the debugger to a target process in of the following ways:

- Define a “debugging target” (name of program or routine to debug) for the current project using the **Settings** command within the **Project** menu and then use the **Go** command within the **Debug** menu to start the target program and connect to its server process.
- Use the **Attach** command within the **Debug** menu to choose from a list of running processes on a Caché server and connect to it.

For more details refer to the [Debugging](#) chapter in the *Using Caché Studio* guide.

How can I debug a class?

At this time, Studio does not support class-level debugging so you have to use a few tricks to debug classes.

1. Make sure that you set the **Keep Generated Source Code** option before you compile your class. This option is located in the Options dialog on the Classes tab. You can invoke this dialog using the **Options** command within the **Tools** menu.
2. View the INT code generated for your class using the **View Other** command in the **View** menu (available when the current window contains a class definition).
3. Set a breakpoint at the desired location in the INT code by pressing the **F9** (toggle breakpoint) key on the desired source line.
4. Set a debugging target to specify where you want the debugger to begin code execution. You can set this within the Project Settings dialog. To invoke this dialog, use the **Settings** command within the **Project** menu.

Hint: you can enter the name of a class method here along with arguments:

```
##class(MyApp.MyClass).MyMethod("This is a test")
```

5. Start the debugger using the **Go** command in the **Debug** menu.

Can I watch variables?

Yes. While debugging type in the name of the variable (or an expression) in the left-hand column of the Studio Watch Window. Each time the debugger pauses, the variable or expression will be reevaluated.

Editing

What do the different colors in the editor mean?

The Studio uses different colors to display the various syntax elements of a given language.

Why is there a wavy, red line underneath my code?

The wavy, red line indicates that the underlined code (or possibly code before it) contains syntax errors.

Can I change the colors in the editor?

Yes. You can change the colors used for the various syntax elements as follows:

1. Invoke the Options dialog using the **Options** command in the **Tools** menu and display the Editor tab.
2. Choose a language from the list on the left.
3. Choose the desired syntax element (comment, variable, etc.)—the list of available items depends on the selected language.
4. Select a color and press **OK**.

Does Studio support Kanji characters?

Yes. Studio has complete support for UNICODE and Kanji characters.

Does Studio support Hebrew characters?

Yes. The Studio Editor supports Hebrew characters as well as bidirectional editing.

Importing Files

Can I import class definitions or routines from external files?

Yes. Use the **Import** command in the **Tools** menu.

What is the difference between Local and Remote files?

Studio is a client-server application; the Studio itself runs on a client system and talks to a server. The server can either be on the same machine or on a remote machine. The Studio uses the terms “Local” and “Remote” to refer to operating system files (such as when you are importing or exporting) that are stored on the client and server systems, respectively.

If both the client and server are on the same system then there is no difference between Local and Remote.

Printing

Can I print from Studio?

Yes. The Studio supports both printing as well as print preview. Both are available from the **File** menu.

Templates

What is a Template?

Templates are a mechanism for creating user-defined Studio add-ins.

A template is a small program that injects a useful code fragment into the current document at the current cursor point. Templates can use Caché Server Pages to present a sophisticated user interface within a pop-up browser hosted by Studio.

To find out more, refer to the [Templates](#) chapter in the *Using Caché Studio* guide.

Is there a list of available Templates?

Yes. You can either invoke the Templates menu (using the **Templates** command in the **Tools** menu) or look at the list in the [Templates](#) chapter in the *Using Caché Studio* guide.

Can I create a new Template?

Yes. You can use Studio to create new Templates. Refer to the [Templates](#) chapter in the *Using Caché Studio* guide.

Multiuser Support

Does Studio support development by multiple users?

Yes. You can do this in several ways:

- Set up a common Caché server system and have all developers store their code on it.
- Use local Caché servers (on the developer's system) and store source code in a source control system as exported XML files.

What happens if I try to open a class (or routine) that someone else is editing?

Studio will display a dialog stating that the class (or routine) is in use by someone else and ask you if you want to open it in read-only mode.

What if someone wants to edit a super class of a class that I am working on?

Studio does not prevent another developer from modifying the super class of a class you are working on.

While Studio could take out locks on all subclasses whenever a class is opened for editing, in practice this would be annoying and unwieldy. Instead, a development needs to work out rules and procedures for defining and modifying super classes. This is similar to how development teams in other languages (say Java) usually work with class definitions in source control systems.

Classes

How do I create a new class?

Use the **New** command in the **File** menu and ask for a new Class Definition. This invokes the New Class Wizard.

For more details, see the [Class Definitions](#) chapter in the *Using Caché Studio* guide.

How can I see the source code generated for my class?

Yes. You can see all the source code generated by the Class Compiler using the **View Other** command in the **View** menu (available when the current window contains a class definition).

Make sure that you set the **Keep Generated Source Code** option before you compile your class. This option is located in the Options dialog on the Classes tab. You can invoke this dialog using the **Options** command within the **Tools** menu.

When I try to compile my class, the Studio says it is up to date and does not need to be compiled. Can I force a compile to happen?

Yes. Turn off the **Do not compile up-to-date items** option. This option is located in the Options dialog on the Classes tab. You can invoke this dialog using the **Options** command within the **Tools** menu.

Routines

How do I create an INT routine?

Create a new Caché ObjectScript routine using the **New** command in the **File** menu and then save the new routine using a name with a .INT extension. You can create an include (.INC) file in the same fashion.

SQL

How do I define an SQL View?

Studio does not include a mechanism for defining SQL views. To do this, as well as other SQL tasks, use the Caché System Management Portal.

Source Control

Does Studio integrate with external Source Control systems?

Yes. The procedure is:

1. Create a subclass of the system-supplied class `%Studio.SourceControl.Base` where you implement its methods to interact with your source-control system. The class that you create is called from Studio in response to particular events and then performs the actions that you have specified.
2. In the System Management Portal, go to the **Studio Source Control Settings** page (**[Home]** > **[Configuration]** > **[Studio Source Control Settings]**), select the site-specific source-control class from the list, and click **OK**.

At this point, Studio has been configured to interact with the source-control system. When Studio attempts to open a document, prior to opening it, the **OnBeforeLoad** method of your source-control class is invoked; typically, this method checks the timestamp on a file representation of the document and, if it is newer in the file, the method calls a `Caché` function to import this into the current namespace. This makes sure that the user is seeing the most up-to-date version of the file.

If the user modifies the file and saves it, then Studio calls the **OnAfterSave** method of the source-control class, which typically exports this document to the filesystem. (This takes keeps these files in sync with the routines, classes, etc. that are in `Caché`.)

When the user attempts to modify a document Studio attempts to get a lock on it, which also triggers a call to a source control method **GetStatus**. If the file is locked in source control, Studio can then ask if the user wants to check it out; this triggers a call to the **CheckOut** method which performs the actions required the item out.

The `%Studio.SourceControl.Base` and its parent, `%Studio.Extension.Base`, provide a set of methods that allow you to create interactions between Studio and your source-control system that are as simple or complex as you choose.

Can I create my own hooks?

Yes. You can define hooks—code that is executed whenever items are saved to or loaded from the server. For details refer to the [Source Control Hooks](#) section in the *Using Caché Studio* guide.

Compatibility

Can I use this version of Studio with older versions of Caché?

In order to support class editing and debugging, the new Studio requires server features that are only available in Caché v5.0 and above.

Will Studio be compatible with future versions?

Yes. The new version of Studio is designed to be forwards and backwards compatible starting with Caché v5.0.

Can I run Studio on Linux?

The Studio client only runs on Windows platforms and will not run on Linux. You can use a Windows-client to talk to a Linux server. You can also use a partition manager, such as VMWARE, to run both Windows and Linux partitions on your development system and run Studio in the Windows partition with Caché running in the Linux partition. The only trick is to configure your networking so that the Windows partition can talk to the Linux partition via TCP/IP.

Can I use Studio with a UNIX or OpenVMS server?

Yes. The new version of Studio can work with any server as long as it is running Caché v5.0 or higher.

Studio Implementation

Why doesn't Studio use the licensed components of Microsoft Visual Studio?

There are several reasons why we built Caché Studio from the “ground up” instead of licensing or extending Visual Studio:

- The Caché Studio editor uses advanced parsing technology not available within the Microsoft Studio framework.
- Microsoft cannot guarantee the compatibility of future versions of Visual Studio.

Why wasn't the Studio interface developed using Java?

At this time, the only way to get acceptable performance for the Studio editor is to use direct calls to the Windows API. While there are syntax-coloring editors developed using Java they do not offer the sophisticated multi-language parsing used by Studio and they typically require very high performance computers for decent performance.