



# Using the Caché %Installer Class

Version 2009.1  
30 June 2009

*Using the Caché %Installer Class*

Caché Version 2009.1 30 June 2009

Copyright © 2009 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](http://www.w3c.org). The primary document development tools were special-purpose XML-processing applications built by InterSystems using Caché and Java.



Caché WEBLINK, Distributed Cache Protocol, M/SQL, M/NET, and M/PACT are registered trademarks of InterSystems Corporation.



InterSystems Jalapeño Technology, Enterprise Cache Protocol, ECP, and InterSystems Zen 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 Customer Support**

Tel: +1 617 621-0700  
Fax: +1 617 374-9391  
Email: [support@InterSystems.com](mailto:support@InterSystems.com)

# Table of Contents

- Using the Caché %Installer Class..... 1
  - 1 Using the %Installer Class ..... 1
  - 2 %Installer Tags ..... 2
  - 3 Invoking the %Installer Class Methods ..... 7
- List of Tables
  - %Installer Tags ..... 3



# Using the Caché %Installer Class

The %Installer class, which is available in %SYS namespace, provides the ability to create a class that defines an installation manifest which can be executed via a class method. %Installer xml tags are used to define an installation manifest that generates the code to execute the manifest instructions.

This article provides information about defining namespaces, databases, and mappings (global, routine and package). Classes, routines and globals can be loaded into a namespace defined in the installation manifest. Then %Installer can compile the classes and routines making them available for execution in the namespace in which they are loaded.

**Note:** For an example of a user-defined installer class, see Sample.Installer in the SAMPLES namespace.

## 1 Using the %Installer Class

The overall structure of an %Installer class is:

```
Include %occInclude
Class MyPackage.MyInstaller
{
XData MyInstall [ XMLNamespace = INSTALLER ]
{
    <Manifest>
        <Var/>
        <If/>
        <Namespace>
            <Configuration>
                <Database/>
                <GlobalMapping/>
                <RoutineMapping/>
                <ClassMapping/>
            </Configuration>
            <If/>
            <Import/>
            <Invoke/>
        </Namespace>
    </Manifest>
}
}
ClassMethod setup(ByRef pVars, pLogLevel As %Integer, pInstaller As
%Installer.Installer) As %Status [ CodeMode = objectgenerator,
Internal, Private ]
{
    #; Let our XGL document generate code for this method.
    Quit ##class(%Installer.Manifest).%Generate(%compiledclass,
        %code, "MyInstall")
}
```

The %occlude include file, which includes methods required by the %Installer class, must be specified at the beginning of every %Installer class you create.

The name of the XData block (MyInstall) is used by the **MyPackage.MyInstaller.setup** method to generate the code for that manifest, which is passed as the third argument of the **%Installer.Manifest.%Generate** method.

**Note:** Referencing `XMLNamespace = INSTALLER` provides Studio Assist capability while authoring your %Installer class.

The outer-most xml tag, <Manifest>, contains all the information for code generation. There can be as many <Namespace> tags as needed within the <Manifest> tag to define namespaces in the manifest, but there must be at least one.

There must be one <Configuration> tag within each <Namespace>tag, and as many <Database> tags within the <Configuration> tag as needed to define the namespace. In addition, following each <Database> tag, you can add <GlobalMapping>, <RoutineMapping>, and <ClassMapping> tags as required.

Within the context of a <Namespace> tag — after the databases and their mappings have been defined — globals can be loaded, and routines and classes can be loaded and compiled using <Import> tag. Class methods can be invoked using the <Invoke> tag. Invoked class methods can execute routines and access globals that have been imported.

Optionally, you can define variables with the <Var> tag; each variable must specify a name and value. When the value for the <Var> is needed, the name is referenced by the `${NameAssigned}` syntax.

## 2 %Installer Tags

The following table describes the tags you can use in your %Installer class.

*%Installer Tags*

Tag	Description
<Manifest>	<p>Required. Contains the installation instructions (all other tags). For example:</p> <pre> &lt;Manifest&gt; &lt;Namespace ... &gt; &lt;Configuration&gt; &lt;/Configuration&gt; &lt;/Namespace&gt; &lt;/Manifest&gt; </pre>
<Var>	<p>Optional; within &lt;Manifest&gt;. Defines variables that can be used with the manifest. For example:</p> <pre> &lt;Var Name="MYAPPPDIR" Value="\$ {MYAPPPDIR} /myapp" /&gt;  &lt;Var Name="DBRESOURCE" Value=%DB_\$ {NAMESPACE} " /&gt;  &lt;Var Name="MYAPPRESOURCE" Value=%DB_MYAPP " /&gt; </pre>
<If>	<p>Optional; within &lt;Manifest&gt; or &lt;Namespace&gt;. Defines conditional. For example:</p> <pre> &lt;If Condition='\$L( "\$ {NAMESPACE} ")=0'&gt; &lt;Error Status="\$ \$\$ \$NamespaceDoesNotExist"&gt; &lt;Arg Value="\$ {NAMESPACE} " /&gt; &lt;/Error&gt; &lt;/If&gt; </pre>

Tag	Description
<Namespace>	<p>Required; within &lt;Manifest&gt;. One for each namespace. For example:</p> <pre>&lt;Namespace Name="{NAMESPACE}" Create="yes" Code="{NAMESPACE}" Data="{NAMESPACE}" &gt; &lt;Configuration&gt; &lt;Database . . . /&gt; &lt;/Configuration&gt; &lt;/Namespace&gt;</pre>
<Configuration>	<p>Required; within &lt;Namespace&gt;. Contains namespace configuration tags for databases and mapping. For example:</p> <pre>&lt;Namespace . . .&gt; &lt;Configuration&gt; &lt;Database . . . /&gt; &lt;/Configuration&gt; &lt;/Namespace&gt;</pre>



Tag	Description
<Database>	<p>Required; within &lt;Configuration&gt;. Defines one or more databases used in the namespace. For example:</p> <pre> &lt;Namespace . . .&gt; &lt;Configuration&gt; &lt;Database Name="{NAMESPACE}" Dir="{MGRDIR}/{NAMESPACE}" Create="yes" Resource="{DBRESOURCE}" PublicPermissions=" "/&gt; &lt;Database Name="MYAPP" Dir="{MYAPPPDIR}/db" Create="no" Resource="{MYAPPRESOURCE}" PublicPermissions=" "/&gt; &lt;/Configuration&gt; &lt;/Namespace&gt; </pre>
<GlobalMapping>	<p>Optional; within &lt;Configuration&gt;. Defines global mapping for the namespace. For example:</p> <pre> &lt;Namespace . . .&gt; &lt;Configuration&gt; &lt;Database . . . /&gt; &lt;Database Name="MYAPP" . . . /&gt; &lt;GlobalMapping Global="MyAppData.*" FROM="MYAPP" /&gt; &lt;GlobalMapping Global="cspRule" FROM="MYAPP" /&gt; &lt;/Configuration&gt; &lt;/Namespace&gt; </pre>

Tag	Description
<RoutineMapping>	<p>Optional; within &lt;Configuration&gt;. Defines routine mapping for the namespace. For example:</p> <pre>&lt;Namespace . . .&gt; &lt;Configuration&gt; &lt;Database . . . /&gt; &lt;Database Name="MYAPP" . . . /&gt; &lt;RoutineMapping Routines="MYAPP" Type="INC" FROM="MYAPP" /&gt; &lt;/Configuration&gt; &lt;/Namespace&gt;</pre>
<ClassMapping>	<p>Optional; within &lt;Configuration&gt;. Defines package mapping for the namespace. For example:</p> <pre>&lt;Namespace . . .&gt; &lt;Configuration&gt; &lt;Database . . . /&gt; &lt;Database Name="MYAPP" . . . /&gt; &lt;ClassMapping Package="MYAPP" FROM="MYAPP" /&gt; &lt;/Configuration&gt; &lt;/Namespace&gt;</pre>
<Import>	<p>Optional; within &lt;Namespace&gt;. Defines globals, routines, and packages to be loaded (and routines and packages to be compiled). For example:</p> <pre>&lt;Namespace . . .&gt; &lt;Configuration&gt; . . . &lt;/Configuration&gt; &lt;Import File="{MYAPPDIR}/data/Defaults.gof" /&gt; &lt;/Namespace&gt;</pre>

Tag	Description
<Invoke>	<p>Optional; within &lt;Namespace&gt;. Defines class methods to execute routines and access globals that have been imported. For example:</p> <pre> &lt;Namespace . . .&gt; &lt;Configuration&gt; . . . &lt;/Configuration&gt; &lt;Invoke Class="Sample.Installer" Method="SetupDefaults" CheckStatus="1"&gt; &lt;Arg Value="\$ {NAMESPACE}" /&gt; &lt;Arg Value="\$ {ISUPGRADE}" /&gt; &lt;/Invoke&gt; &lt;/Namespace&gt; </pre>

For a complete example of a user-defined installer class, see Sample.Installer in the SAMPLES namespace.

## 3 Invoking the %Installer Class Methods

You can invoke the %Installer class by doing either of the following:

- In the %SYS namespace, enter the following command in the Caché Terminal:

```
Do ##class(<classname>).setup(<arglist>)
```

where <classname> is the name of the %Installer class, and <arglist> specifies comma-separated named VARIABLE=<value> pairs, which are passed into the method and referenced by \$ {VARIABLE}. For example:

```
%SYS> Do ##class(MyPackage.MyInstaller).setup()
```

- Export the %Installer class as DefaultInstallerClass.xml to the same directory where the Caché install (either .msi or cinstall) is run. It is loaded and compiled, and the **setup** method is executed.

**Note:** If you use the export technique, you cannot pass arguments to the **setup** method.

