



# Using the Caché \$ZOBJxxx Intrinsic Functions

Version 5.2  
01 September 2006

*Using the Caché \$ZOBJxxx Intrinsic Functions*  
Caché Version 5.2 01 September 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](http://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 registered trademarks of InterSystems Corporation.



The Ensemble product and its logos are registered 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](mailto:support@InterSystems.com)

# Table of Contents

- Using the Caché \$ZOBJxxx Intrinsic Functions..... 1**
- \$ZOBJCLASSMETHOD Intrinsic Function ..... 1
- \$ZOBJMETHOD Intrinsic Function ..... 3
- \$ZOBJPROPERTY Intrinsic Function ..... 4



# Using the Caché \$ZOBJxxx Intrinsic Functions

This document describes Caché-supplied (intrinsic) functions that can be used to generalize processing of objects by allowing a reference to a class and one of its methods or properties to be computed at runtime.

This article describes the following intrinsic functions:

- [\\$ZOBJCLASSMETHOD](#)
- [\\$ZOBJMETHOD](#)
- [\\$ZOBJPROPERTY](#)

For more information on ObjectScript functions generally, see the section “[Functions](#)” in the [Caché ObjectScript Guide](#).

The function names are shown here in all uppercase letters, but they are, in fact, case-insensitive.

## \$ZOBJCLASSMETHOD Intrinsic Function

---

```
$ZOBJCLASSMETHOD (Classname, Methodname, Arg1, Arg2, Arg3, ... )
```

### *Parameters*

Classname	The name of an existing, accessible class.
Methodname	The name of a classmethod in the designated class.
Arg1, Arg2, Arg3, ...	Values to be substituted for the arguments of the classmethod when it is invoked.

## Description

This function permits a Caché ObjectScript program to invoke an arbitrary classmethod in an arbitrary class. Both the class name and the method name may be computed at runtime or supplied as string constants.

If the method takes arguments, they are supplied by the list of arguments that follow the *methodname*. A maximum of 255 argument values may be passed to the method.

## Parameters

### *Classname*

An expression which evaluates to a string. The content of the string must match exactly the name of an existing, accessible, previously-compiled class. In the case of references to Caché classes, the name may be either in its canonical form (%Library.String), or its abbreviated form (%String).

### *Methodname*

An expression which evaluates to a string. The value of the string must match the name of an existing ClassMethod in the class identified by *Classname*.

### *Arg1, Arg2, Arg3, ...*

A series of expressions that will be substituted sequentially for the arguments to the designated method. The values of the expressions can be of any type. It is the responsibility of the implementor to make sure that the type of the supplied expressions match what the method expects, and have values within the bounds declared.

If the specified method expects no arguments then no arguments beyond the *MethodName* need be given in the function invocation. If the method requires arguments, the rules that govern what must be supplied are those of the target method.

## Remarks

The invocation of **\$ZOBJCLASSMETHOD** as a function or a procedure will determine the invocation of the target method. See the examples for more detail.

An attempt to invoke a method which is non-existent, or one which is not declared to be a ClassMethod, will result in a <METHOD DOES NOT EXIST> error.

## Examples

The following example shows **\$ZOBJCLASSMETHOD** used as a function:

```

set ClassName = "%Dictionary.ClassDefinition"
set ClassmethodName = "NormalizeClassname"
set SingleArgument = "%String"
write $ZOBJCLASSMETHOD(ClassName, ClassmethodName, SingleArgument), !

```

## \$ZOBJMETHOD Intrinsic Function

```
$ZOBJMETHOD (Instance, Methodname, Arg1, Arg2, Arg3... )
```

### Parameters

Instance	An instance of the class containing the method name.
Methodname	The name of a method in the designated class.
Arg1, Arg2, Arg3, ...	Values to be substituted for the arguments of the methodname when it is invoked.

### Description

This function permits a Caché ObjectScript program to call an arbitrary method in an existing instance of some class. Since the first argument must be a reference to an object, it is computed at execution time. The method name may be computed at runtime or supplied as a string literal. If the method takes arguments, they are supplied from the list of arguments that follow the methodname. A maximum of 255 argument values may be passed to the method. If the method requires arguments, the rules that govern what must be supplied are those of the target method.

### Parameters

#### *Instance*

An expression which evaluates to an object reference. The value of the expression must be that of an in-memory instance of the desired class.

#### *Methodname*

An expression which evaluates to a string. The value of the string must exactly match the name of an existing method in the instance of the class given as the first argument.

#### *Arg1, Arg2, Arg3, ...*

A series of expressions that will be substituted in turn for the arguments to the designated method. The values of the expressions can be of any type. It is the responsibility of the implementor to make sure that the supplied expressions both match in type and have values with the bounds that the method expects.

If the specified method expects no arguments then nothing beyond *Classname* and *MethodName* need be used in the function invocation. If the method requires arguments, the rules that govern what must be supplied are those of the target method.

## Remarks

The invocation of **\$ZOBJMETHOD** as a function or a procedure will determine the invocation of the target method. See the examples for more detail.

When used within one method of a class instance to refer to another method of that instance, the **\$ZOBJMETHOD** may omit *Instance*. The comma that would normally follow *Instance* is still required, however.

An attempt to invoke a method which is non-existent, or one which is declared to be a *ClassMethod*, will result in a <METHOD DOES NOT EXIST> error.

## Examples

The following example shows **\$ZOBJMETHOD** used as a function:

```
set ListOfStuff = ##class(%Library.ListOfDataTypes).%New()
for i = "First", "Second", "Third", "Fourth"
{
    do ListOfStuff.Insert((i _ "-Element"))
}
set MethodName = "Count"
set Elements = $ZOBJMETHOD(ListOfStuff, MethodName)
write "Elements: ", Elements, !
set i = $RANDOM(Elements) + 1
write "Element #", i, " = ", $ZOBJMETHOD(ListOfStuff, "GetAt", i), !
```

# \$ZOBJPROPERTY Intrinsic Function

---

```
$ZOBJPROPERTY (Instance, Propertyname, Index1, Index2, Index3... )
```

## Parameters

Instance	An instance of the class containing the method name.
Propertyname	The name of a property in the designated class.
Index1, index2, index3, ...	If the property named as the second argument is multidimensional, then the index<n> are treated as indexes into the array that constitutes the property.

## Description

This function permits a Caché ObjectScript program to select the value of an arbitrary property in an existing instance of some class. Since the first argument must be an instance of a class, it is computed at execution time. The property name may be computed at runtime or supplied as a string literal. The contents of the string must match exactly the name of a property declared in the class.

If the property is declared to be multidimensional, then the arguments after the *propertyname* are treated as indexes into a multidimensional array. A maximum of 255 argument values may be used for the index.

The **\$ZOBJPROPERTY** may also appear on the left side of an assignment. In this instance, it provides the location to which a value is assigned.

## Parameters

### *Instance*

An expression which evaluates to an *oref*. The value of the expression must be that of an in-memory instance of the desired class.

### *Propertyname*

An expression which evaluates to a string. The value of the string must match the name of an existing property defined in the class identified by *Classname*.

### *Arg1, Arg2, Arg3, ...*

If *Propertyname* is a multidimensional value, then this series of expressions will be treated as indexes into the array represented by the property.

If the specified property is not multidimensional, the presence of extra arguments causes an error at runtime.

## Remarks

When **\$ZOBJPROPERTY** appears to the left of an assignment operator, it is the location being set. When it appears to the right, it is the value being used in the calculation.

When used within a method to refer to a property of the current instance, the **\$ZOBJPROPERTY** may omit *Instance*. The comma that would normally follow *Instance* is still required, however.

An attempt to get (set) a multidimensional value from (into) a property which is not declared to be multidimensional will result in a <FUNCTION> error.

## Examples

The following example shows **\$ZOBJPROPERTY** used as a function:

```
set TestName = "%Library.File"
set ClassDef = ##class(%Library.ClassDefinition).%OpenId(TestName)
for i = "Name", "Super", "Persistent", "Final"
{
    write i, ": ", $ZOBJPROPERTY(ClassDef, i), !
}
}
```

This one shows its use on both sides of an assignment operator:

```
set TestFile = ##class(%Library.File).%New("AFile")
write "Initial file name: ", $ZOBJPROPERTY(TestFile, "Name"), !
set $ZOBJPROPERTY(TestFile, "Name") = $ZOBJPROPERTY(TestFile, "Name")
    _ "Renamed"
write "File name afterward: ", $ZOBJPROPERTY(TestFile, "Name"), !
```