Working with %Status Values

Version 2017.2
2020-06-25

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

Working with %Status Values
Caché Version 2017.2 2020-06-25
Copyright © 2020 InterSystems Corporation
All rights reserved.

InterSystems, InterSystems IRIS, InterSystems Caché, InterSystems Ensemble, and InterSystems HealthShare are registered 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 Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

Working with %Status Values

Many InterSystems classes use the %Status data type class to represent status information, and their methods return a %Status value (a *status*) that represents either success or error. If the status represents an error (or multiple errors), the status value also includes information about the errors.

You can also return your own status values.

This article discusses status values and how to work with them.

1 Basics of Working with Status Values

As noted above, methods in many InterSystems classes return a status to indicate success or error. For example, the **%Save()** method in %Library.Persistent returns a status. For any such method, be sure to obtain the returned value. Then check the status and then proceed appropriately. The basic tools are as follows:

- To check whether the status represents success or error, use any of the following:
  - The `$$$ISOK` and `$$$ISERR` macros, which are defined in the include file %occStatus.inc. This include file is automatically available in all object classes.
  - The **$SYSTEM.Status.IsOK()** and **$SYSTEM.Status.IsError()** methods, which are especially convenient in the Terminal (where you cannot use macros).
- To display the error details, use **$SYSTEM.OBJ.DisplayError()** or **$SYSTEM.Status.DisplayError()**. These methods are equivalent to each other. They write output to the current device.
- To obtain a string that contains the error details, use **$SYSTEM.Status.GetErrorText()**.

1.1 Examples

For example:

```
Set object=##class(Sample.Person).%New()
Set object.Name="Smith,Janie"
Set tSC=object.%Save()
If $$$ISERR(tSC) {
  Do $SYSTEM.OBJ.DisplayError(tSC)
  Quit
}
```

Here is a partial example that shows use of **$SYSTEM.Status.GetErrorText()**:

```
If $$$ISERR(tSC) {
  // if error, log error message so users can see them
  Do ..LogMsg($System.Status.GetErrorText(tSC))
}
```

2 Multiple Errors Reported in a Status Value

If a status value represents multiple errors, the techniques give you information about only the latest. To obtain information about all the errors represented by a status value, use **$SYSTEM.Status.DecomposeStatus()**, which returns an array of the error details (by reference, as the second argument). For example:

```
Do $SYSTEM.Status.DecomposeStatus(tSC,.errorlist)
//then examine the errorlist variable
```

The variable *errorlist* is an array that contains the error information. The following shows a partial example with some artificial line breaks for readability:

```
ZWRITE errorlist
errorlist=2
errorlist(1)="ERROR #5659: Property 'Sample.Person::SSN(1@Sample.Person,ID=)' required"
errorlist(1,"caller")="%ValidateObject+9^Sample.Person.1"
errorlist(1,"code")=5659
errorlist(1,"dcode")=5659
errorlist(1,"domain")="%ObjectErrors"
errorlist(1,"namespace")="SAMPLES"
errorlist(1,"param")=1
errorlist(1,"param",1)="Sample.Person::SSN(1@Sample.Person,ID=)"
...
errorlist(2)="ERROR #7209: Datatype value '' does not match
PATTERN '3N1""-""2N1""-""4N'"_$c(13,10)_"  >
ERROR #5802: Datatype validation failed on property 'Sample.Person:SSN',
with value equal to """""
errorlist(2,"caller")="zSSNIsValid+1^Sample.Person.1"
errorlist(2,"code")=7209
...
```

If you wanted to log each error message, you could adapt the previous logging example as follows:

```
If $$$ISERR(tSC) {
  // if error, log error message so users can see them
  Do $SYSTEM.Status.DecomposeStatus(tSC,.errorlist)
  For i=1:1:errorlist {
      Do ..LogMsg(errorlist(i))
  }
}
```

3 Returning Status Values

You can also return your own custom status value. To create a status value, use the following construction:

```
$$$ERROR($$$GeneralError,"your error text here")
```

Or equivalently:

```
$SYSTEM.Status.Error($$$GeneralError,"your error text here")
```

For example:

```
quit $SYSTEM.Status.Error($$$GeneralError,"Not enough information for request")
```

To include information about additional errors, use **$SYSTEM.Status.AppendStatus()** to modify the status value. For example:

```
set tSC=$SYSTEM.Status.AppendStatus(tSCfirst,tSCsecond)
quit tSC
```

4 For More Information

For more information, see the class reference for the %SYSTEM.Status class and the %Library.Status class.