

# Expressions and Scripts in DeepSee

Version 2010.1  
17 February 2010

*Expressions and Scripts in DeepSee*  
Caché Version 2010.1 17 February 2010  
Copyright © 2010 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

<b>Preface</b> .....	<b>1</b>
<b>1 Filter Expressions</b> .....	<b>3</b>
1.1 When Filter Expressions Are Evaluated .....	3
1.2 Where Filter Expressions Are Used .....	3
1.3 User Interface for Filter Expressions .....	4
1.4 Basic Syntax for Filter Expressions .....	4
1.5 Filter Expressions Using IN .....	5
1.6 Filter Expressions Using Pattern Matching .....	6
1.7 Filter Expressions and Dates .....	6
1.7.1 Using Part of the Date .....	7
1.7.2 Using a Complete Date .....	7
1.8 Filter Expressions with Embedded Caché ObjectScript .....	7
1.8.1 Using a Runtime Variable .....	7
1.8.2 Using Session Data .....	8
1.9 Filter Expressions as Caché ObjectScript Expressions .....	8
<b>2 Caché ObjectScript Expressions</b> .....	<b>9</b>
2.1 When and Where Caché ObjectScript Expressions Are Used .....	9
2.1.1 Expressions Evaluated at Build Time .....	9
2.1.2 Expressions Evaluated at Run Time .....	10
2.1.3 Expressions Evaluated When Data Is Loaded .....	10
2.2 User Interface for Caché ObjectScript Expressions .....	11
2.3 Restricted Caché ObjectScript Expressions (Function Calls) .....	12
<b>3 Measure Formulas</b> .....	<b>13</b>
3.1 About Measure Definitions .....	13
3.2 User Interface for Measure Formulas .....	13
3.3 Basic Syntax for Measure Formulas .....	14
3.4 General Syntax for Measure Formulas .....	15
<b>4 Custom Scripts</b> .....	<b>17</b>
4.1 Where Custom Scripts Are Used .....	17
4.1.1 Scripts Executed at Run Time .....	17
4.1.2 Scripts Evaluated When Data Is Loaded .....	18
4.2 User Interface for Custom Scripts .....	18
<b>Appendix A:DeepSee Functions</b> .....	<b>19</b>
A.1 \$\$KPI Function .....	19
A.1.1 Where You Can Use \$\$KPI .....	20
A.2 \$\$VAR Function .....	20
A.2.1 Where You Can Use \$\$VAR .....	20
<b>Appendix B:Comparison of Measures, Computations, and KPIs</b> .....	<b>21</b>



# Preface

DeepSee uses formulas, expressions, and scripts of varying types. This book summarizes the cases and the syntaxes for your convenience. It contains the following sections:

- [Filter Expressions](#)
- [Caché ObjectScript Expressions](#)
- [Measure Formulas](#)
- [Caché ObjectScript Scripts](#)
- [DeepSee Functions](#)
- [Comparison of Measures, Computations, and KPIs](#)

For a detailed outline, see the [table of contents](#).

For more information, see the following books:

- [Overview of DeepSee](#), an introductory guide for all users who are interested in learning about DeepSee.
- [DeepSee Model Design Guide](#), an introductory guide for implementers and business users.
- [DeepSee Developer Tutorial](#), a tutorial for implementers who are creating DeepSee models, pivot tables, and dashboards.
- [Using the DeepSee Connector](#), a guide for implementers who are using the DeepSee Connector to import externally stored data. Note that the DeepSee Connector is available only with Ensemble.
- [Using the DeepSee Architect](#), a guide for implementers who are setting up a DeepSee model for use in the Analyzer.
- [Using the DeepSee Analyzer](#), a guide for implementers and advanced users who want to create pivot tables to embed in applications — or who simply want to explore their data.
- [Using the DeepSee Dashboard Designer](#), a guide for implementers who are using the Dashboard Designer to create dashboards.
- [DeepSee Site Configuration and Maintenance Guide](#), a guide for implementers and system administrators. This book describes how to configure and maintain a DeepSee site. It also includes a chapter on troubleshooting.
- [DeepSee User Guide](#), a user manual for your end users. This book describes how to work with deployed dashboards and pivot tables.

For general information, see the *InterSystems Documentation Guide*.



# 1

## Filter Expressions

A DeepSee *filter expression* (also called a *filter*) is a boolean expression that specifies which records to include. It contains the name of a dimension, the names of one or more members, and operators. You can use these expressions in a wide variety of contexts.

This chapter discusses the following:

- [When DeepSee evaluates filter expressions](#)
- [Where you can use filter expressions](#)
- [A look at the user interface provided for creating filter expressions](#)
- [The basic syntax](#)
- [Using the IN operator](#)
- [Using the ?= operator for pattern matching](#)
- [Filter expressions and dates](#)
- [Filter expressions with embedded Caché ObjectScript](#)

### 1.1 When Filter Expressions Are Evaluated

Filter expressions are evaluated at run time, that is, when a user displays a pivot table or dashboard.

### 1.2 Where Filter Expressions Are Used

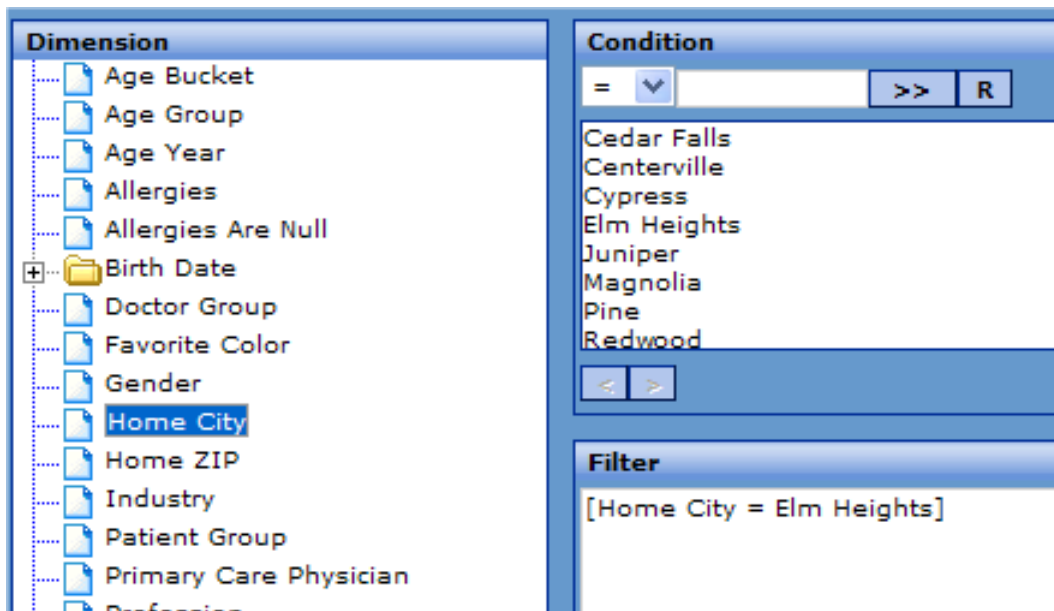
You can use filter expressions in the following contexts:

- In the Architect:
  - As an optional filter applied to a subject area (to restrict the number of rows that can be accessed).
  - In the definition of a compound member.
  - As an optional filter applied to any pivot table in a subject area when used by a specific role (a rare usage).
  - As an optional filter applied to any detail listing in a subject area when used by a specific role (a rare usage).

- In the Analyzer:
  - As an optional filter applied to a pivot table.
  - As an optional filter applied to a measure (defining a filtered measure).
  - In the definition of a compound member.
- In the KPI Editor, as an optional filter applied to a KPI.
- In the Dashboard Designer:
  - As an optional filter applied to a data element (pivot table, speedometer, or detail listing) included on a dashboard.
  - As an optional filter applied to a label or image that uses the KPI override feature.
  - As an optional filter applied to a frame that is configured to display listing fields or other source data.
  - In the definitions of the three main groups in a Venn diagram.

## 1.3 User Interface for Filter Expressions

In most cases, the Analyzer, Architect, and KPI Editor provide a dialog box like the following to help you create the filter expressions:



The Dashboard Designer uses a somewhat different dialog box, but it includes the main elements shown here.

In some places, DeepSee does not provide a dialog box to help you build the expression; instead you must type the expression directly.

## 1.4 Basic Syntax for Filter Expressions

A basic filter expression has the following syntax:

```
[DimensionName = MemberName]
```

Where:

- *DimensionName* is the name of a dimension, without quotes.
- *MemberName* is a member name, without quotes.

For example:

```
[Allergies = bee stings]
```

Notes:

- The spaces before and after the operator are required.
- The names of dimensions and members are case-sensitive.
- You can use other operators instead of the equals sign (=). The operators you can use are as follows:

Operator	Meaning
=	The filter expression is true only for records that belong to the given member.
>	The filter expression is true for records that belong to members that are sorted after the given member by default. For string values, DeepSee sorts in ascending alphanumeric order. However, for string values that contain only numbers, DeepSee sorts in ascending numeric order (when it evaluates filter expressions).
>=	The filter expression is true for records that belong to the given member and for all members that are sorted after that member by default.
<	The filter expression is true for records that belong to members that are sorted before the given member by default.
<=	The filter expression is true for records that belong to the given member and for all members that are sorted before that member by default.
IN	The filter expression is true for records that belong to any of the given members. See <a href="#">“Filter Expressions Using IN,”</a> next.
?=	The filter expression is true for records that belong to members whose name match the given pattern. See <a href="#">“Filter Expressions Using Pattern Matching.”</a>

- You can combine filter expressions by using the logical operators NOT, AND, and OR. For example:

```
[Favorite Color = Blue] OR [Favorite Color = Red] OR [Favorite Color = Yellow]
```

Use brackets to control the precedence.

- Filter expressions must always use the internal names of dimensions and members. By default, the external name of a dimension is the same as the internal name; similarly, the external name of a member is the same as the internal name. This means that if you have renamed dimensions or members, the filter editor still shows the original names.

## 1.5 Filter Expressions Using IN

A filter expression can have the following syntax:

```
[DimensionName IN String1,String2,String3,...]
```

Do not include spaces before or after the commas in this list; any such space is treated as part of a member name.

For example:

```
[Favorite Color IN Blue,Red,Yellow]
```

This expression is true for all records that belong to the `Blue`, `Red`, or `Yellow` members of the `Favorite Color` dimension.

## 1.6 Filter Expressions Using Pattern Matching

A filter expression can have the following syntax:

```
[DimensionName ?= String]
```

Where *String* is an unquoted string that includes one or more asterisk characters (\*), each of which is a wildcard that can represent any number of characters.

For example:

```
[Doctor ?= *an*]
```

This expression is true for all records where the doctor's name includes the string "an." For example:

<i>Doctor</i>	<i>Count</i>
Alexandra Alton	16
Alexandra Levinson	23
Alexandra Winters	22
Alfred Goldman	22
Alice Bachman	30
Amanda Orwell	17
Amanda Pybus	17
Andrew Ingleman	22
Angelo Ulman	13
Ashley Hanson	19

## 1.7 Filter Expressions and Dates

When you create a dimension based on a date, DeepSee also creates a set of dimension variants, which you can use separately or in combination. For example, if you create a date dimension called `Birth Date`, DeepSee also creates the variants `Birth Date Year`, `Birth Date Quarter`, `Birth Date Month`, and so on. The `Birth Date Year` variant uses only the year part of the date, `Birth Date Quarter` uses only the quarter number, and so on.

You can use any of these dimensions in filter expressions. You can use any of these dimensions as rows or columns, except for the base dimension itself (for example `Birth Date`).

## 1.7.1 Using Part of the Date

You can use date dimensions in the same way as any other dimensions, including using them in filters. For example, consider the following filter expression:

```
[Birth Date Quarter >= Q3]
```

This expression filters out records associated with quarters Q1 and Q2. For example:

<i>Birth Date Quarter</i>	Count
Q3	2,485
Q4	2,588

## 1.7.2 Using a Complete Date

When you filter by date, you often want the filter to consider a complete date rather than just an isolated segment of it. For example, you might want to filter a subject area to show only the data that falls within a particular ten-year span of time.

To do so, you use the base dimension itself. For example:

```
[Birth Date > 01/01/1980] AND [Birth Date < 01/01/1990]
```

The dates must be in the format *DD/MM/YYYY* or *DD/MM/YY*, where *DD* is a two-digit day, *MM* is a two-digit month, *YY* is a two-digit year, and *YYYY* is a four-digit year.

**Tip:** To refer to today's date in the required format, use the following expression:

```
$ZDATE($HOROLOG,4)
```

# 1.8 Filter Expressions with Embedded Caché ObjectScript

A filter expression can include an embedded Caché ObjectScript expression. The syntax is as follows:

```
[DimensionName = {COS expression}]
```

An extremely simple example is as follows:

```
[Favorite Color = {"R_"_ed"}]
```

Another extremely simple example is as follows:

```
[Favorite Color = {variable_name}]
```

## 1.8.1 Using a Runtime Variable

It is possible to pass variable name/value pairs to a dashboard when accessing it (see [Using the DeepSee Dashboard Designer](#)). DeepSee uses the name/value pair to update any variables by that name in the dashboard — including any filter expressions used in any of its elements.

In a filter expression, use the following syntax to use a variable as a member name:

```
[DimensionName = {variable_name}]
```

You can provide a default value as follows:

```
[DimensionName = {$GET(variable_name,default_value)}]
```

More generally, you can include the variable within a Caché ObjectScript expression:

```
[DimensionName = {COS expression that includes variable_name}]
```

The expression must return the name of a member of the given dimension.

## 1.8.2 Using Session Data

In a dashboard, you can refer to the current DeepSee username or the current DeepSee user role, as follows:

```
%session.Data("CurrUser")  
%session.Data("CurrRole")
```

For example, you could use the current DeepSee role to specify the filter that you apply to a subject area:

```
[Region Name = {$CASE(%session.Data("CurrRole"), "role1":"North", "role2":"South", "role3":"West", ":"East")  
}]
```

## 1.9 Filter Expressions as Caché ObjectScript Expressions

In the most general case, you can use a Caché ObjectScript expression as the entire filter expression. That is, in place of the normal filter expression, use the following:

```
{COS expression}
```

The Caché ObjectScript expression must return a valid filter expression. As an extremely simple example:

```
{"[Birth Date Year > 1970]"}
```

# 2

## Caché ObjectScript Expressions

You can use Caché ObjectScript expressions in many places throughout DeepSee. This chapter discusses the following:

- Where you can use Caché ObjectScript expressions and when they are evaluated
- A look at the user interface provided for creating these expressions
- Restricted syntax (function calls) required in parts of a dashboard

### 2.1 When and Where Caché ObjectScript Expressions Are Used

You can use Caché ObjectScript expressions in many places. Where you use the expression determines when DeepSee evaluates it.

#### 2.1.1 Expressions Evaluated at Build Time

If you include Caché ObjectScript expressions in the following locations in Architect, these expressions are evaluated when the DeepSee indicates are built or updated:

- As the source expression for a dimension or measure (the **Complex Code** option). In this case, the expression can use the variable `##this`, which refers to an instance of the BI-enabled class you started from. For example:

```
$CASE(##this.PrimaryCarePhysician.LastName,"":"No Doctor",
:##this.PrimaryCarePhysician.FirstName_"_##this.PrimaryCarePhysician.LastName)
```

- As the source expression for a detail listing field (the **Complex Code** option).
- As the values of **From** and **To**, within a range definition for a dimension.
- As the source expression for a custom aggregate (a custom measure). In this case, the expression can use the variable `Id`, which refers to an instance of base class. For example:

```
$LG(^Sample.OrdersD(Id),4)
```

## 2.1.2 Expressions Evaluated at Run Time

If you include Caché ObjectScript expressions in the following locations, these expressions are evaluated at run time — that is, when a user displays a pivot table or dashboard:

- Within any filter expression, as described in the previous chapter.
- In the Analyzer:
  - As the definition of a measure. For example:  

```
[Sales Revenue] - [Expenses]
```

See the chapter “[Measure Formulas](#).”
  - As the definition of a computation. For example:  

```
$$COL(1)+$$CELL(THISROW-1,THISCOL)
```
  - As the upper or lower value for an alert range.
- In the KPI Editor, as the definition of a KPI (typically as a combination of other KPIs). For example, the following expression returns the value of KPI 10005 divided by KPI 10001:  

```
$$KPI(10005)/$$KPI(10001)
```

For information on **\$\$KPI**, see “[DeepSee Functions](#),” later in this book.
- In the Dashboard Designer:
  - As the default value for combo boxes, list boxes, and other data controls.
  - As the value of the tooltip for an image.
  - As the default value shown in labels, button captions, and speedometer captions. Here, you can use only a specific form of Caché ObjectScript expression. See “[Restricted Caché ObjectScript Expressions \(Function Calls\)](#).”

## 2.1.3 Expressions Evaluated When Data Is Loaded

If you use the Connector, you can include Caché ObjectScript expressions in the following locations, and these expressions are evaluated when the data is loaded:

- On the **Fields Mapping** tab, as the value for a property. When it loads a record, the Connector evaluates the expression and sets the given property equal to the resulting value. To refer to the value of any field in the external data, use the following syntax:  

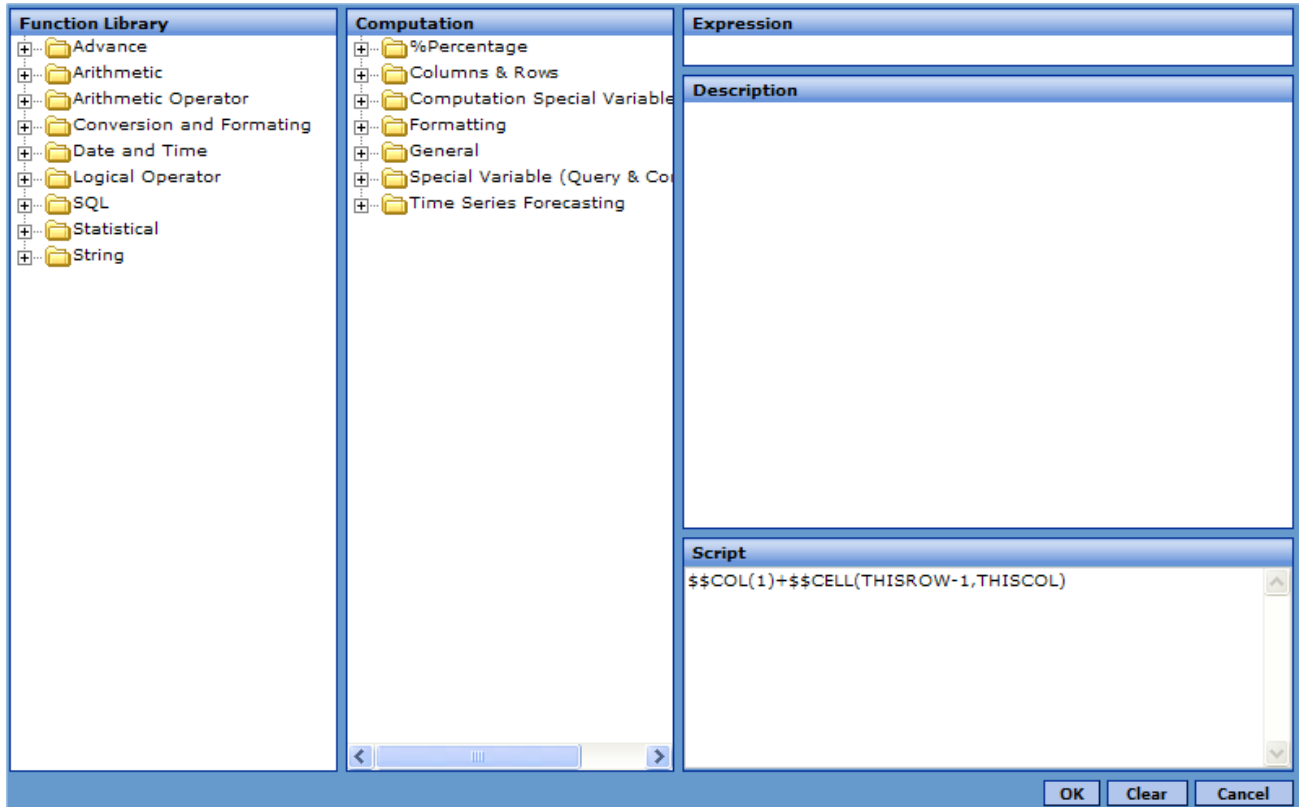
```
{fieldname}
```

For example:  

```
{FirstName} _ " " _ {LastName}
```
- On the **Cleaning Rules** tab. You can define a cleaning rule that evaluates a Caché ObjectScript expression and then, if the expression is true, logs a message and optionally rejects the row.

## 2.2 User Interface for Caché ObjectScript Expressions

In some places, DeepSee provides the following dialog box (or a variation of it) to help you create Caché ObjectScript expressions:



The **Function Library** panel displays an expandable view of functions that you can use in the expression. This includes some Caché ObjectScript functions, as well as macros that are specific to InterSystems DeepSee. The **Computation** panel displays an expandable view of macros, special variables, and template expressions that DeepSee provides. The functions, macros, and so on are all documented online within this dialog box.

In other places, however, DeepSee does not provide a dialog box to help you create expressions. Instead you must type the expression directly.

**Important:** In all places that support Caché ObjectScript expressions or statements, you can type any valid Caché ObjectScript expressions or statements, which can use any methods, routines, functions, or other elements, including your own. You are not restricted to use the elements provided within the user interface shown here. InterSystems suggests that you write your own functions wherever possible rather than use the elements that are specific to DeepSee.

## 2.3 Restricted Caché ObjectScript Expressions (Function Calls)

In labels, button captions, and speedometer captions, DeepSee supports only a very restricted form of Caché ObjectScript expression, specifically the following form:

```
$$functionname(arg1, arg2, ...)
```

The function name must start with \$\$

DeepSee provides a couple of convenient functions that you can use in these places:

- **\$\$KPI** — Refers to the value of a KPI.
- **\$\$VAR** — Refers to the value of a variable.

See the appendix “[DeepSee Functions](#).”

# 3

## Measure Formulas

This chapter discusses *measure formulas*, which define measures. It includes the following:

- [An overview of the kinds of measures](#)
- [Where you define measure formulas](#)
- [Basic syntax](#)
- [A more general view of the syntax](#)

Also see “[Comparison of Measures, Computations, and KPIs.](#)”

### 3.1 About Measure Definitions

For this discussion, it is useful to distinguish two kinds of measures: *base measures* and *calculated measures*.

- Base measures are defined in the Architect. The formula for a base measure has the following form: [MeasureName]  
Every subject area includes the default measure, `Count`, which counts the records in the base class. Each base measure is either numeric or date-typed. Other than `Count`, each base measure is based on values in the raw data, is calculated at index build time, and stored in the fact table.  
You can use `Count` in pivot tables. You can also use any other numeric base measure in pivot tables; the values are summed. You cannot use date-type base measures in pivot tables, but they are available for use in calculated measures.
- Calculated measures are defined in the Analyzer. The formula for a calculated measure can include references to base measures, and it can include information on how to aggregate values.

The values for calculated measures can be numbers, dates, or strings. Numeric values are summed by default. There is no default aggregation for dates or strings.

Calculated measures are calculated at run time, and they are not stored in the fact table.

For description of how DeepSee computes measure values, see [Overview of DeepSee](#).

### 3.2 User Interface for Measure Formulas

You create measure formulas only within the **Metrics Editor** dialog box of the DeepSee Analyzer.

The screenshot shows the Metrics Editor interface with the following components:

- Metric Name:** Patient Count
- Measure:** A list of measures including [Count], [Age], [Allergy Count], [Birth Date], [Encounter Count], and [Test Score].
- Dimension Distinct:** A list of dimension distincts including [Age.Distinct], [Age Bucket.Distinct], [Age Group.Distinct], [Age Year.Distinct], [Allergies.Distinct], [Allergies Are Null.Distinct], [Allergy Count.Distinct], [Birth Date.Distinct], and [Birth Date Day.Distinct].
- Functions:** A list of functions including .Sum, .Average, .Distinct, .Max, .Min, .FirstDate, .LastDate, and .Median.
- Formula:** A text area containing the formula [Count].
- Filter:** An empty text area for filtering.
- Display Style:** A dropdown menu set to "Number".
- Dec. Place:** An empty text field for decimal places.
- Buttons:** OK and Cancel buttons.

## 3.3 Basic Syntax for Measure Formulas

In its basic form, a measure formula has one of the following syntaxes:

```
[BaseMeasure]
[BaseMeasure.AggregatingFunction]
[DimensionName.Distinct]
```

Here:

- *BaseMeasure* is the name of a base measure. You can use the name alone if the value is numeric and you want to sum the values.
- *AggregatingFunction* is the name of an aggregating function, as listed in the lower left list in the **Metrics Editor**. This panel lists all the available aggregating functions.
  - For numeric values, you can use the following functions: .Sum, .Average, .Distinct, .Max, .Min, .Median, .StdDev, and .Variance. The default is .Sum.
  - For date values, you can use the .FirstDate and .LastDate functions. There is no default aggregation for date values.

**Note:** These functions treat null values as zeros.

- *DimensionName* is the name of a dimension.

The expression *DimensionName*.Distinct represents the count of the distinct values for that dimension. For example, Customer.Distinct represents the number of unique customers.

## 3.4 General Syntax for Measure Formulas

More generally, a measure formula is a Caché ObjectScript expression. In this expression, DeepSee interprets square brackets as a reference to a measure formula.

For example, you can use the following as a measure formula:

```
[Sales Revenue] - [Expenses]
```

The following are also valid measure formulas:

```
$h  
"hello world"
```

When you define a measure in the Analyzer, you specify its display style and (if appropriate) the number of decimal places to use. The available display styles include currency styles, date styles, and strings.



# 4

## Custom Scripts

In many locations, DeepSee supports custom scripts, which consist of one or more Caché ObjectScript statements. This chapter discusses the following:

- [Where you can use custom scripts and when they are executed](#)
- [A look at the user interface provided for creating custom scripts](#)

### 4.1 Where Custom Scripts Are Used

You can use Caché ObjectScript scripts in many places. Where you use the expression determines when DeepSee executes it.

#### 4.1.1 Scripts Executed at Run Time

If you include Caché ObjectScript scripts in the following locations, these scripts are executed at run time — that is, when a user displays a pivot table or dashboard:

- In the Analyzer, you can define the following:
  - A post-formula script, which overrides the previous value in each affected cell. You use post-formula scripts when you want to reprocess the original values, to change formatting (for example), or perhaps to simplify computation by breaking it into multiple steps.

You can write a post-formula script for computations, custom members, and scorecard rows.

In all these cases, the statements can refer to the variable *val*, which contains the initial value of the cell. Similarly, to override the value shown in the cell, the statements must set the variable *val*. For example:

```
set val=val+100
```

- A script that DeepSee executes when a user clicks a pivot table cell that has an active alert.
- A script that defines an array to use as a custom time series.

In this case, you must specify values for the *VAL* array, as in the following example:

```
For i=5:1:10 set VAL(i)=i
```

- In the Dashboard Designer, you can define the following:

- A post-loading script for a combo box, list box, or other data control.
- A post action associated with a dashboard element.

## 4.1.2 Scripts Evaluated When Data Is Loaded

If you use the Connector, you can define two cleaning scripts that are run for each row as it is being loaded:

- A script that runs before any transformations that are applied.
- A script that runs after the transformations and after the cleaning rules.

## 4.2 User Interface for Custom Scripts

In some cases, DeepSee provides a dialog box to help you create Caché ObjectScript scripts. The dialog box is similar to the one used for Caché ObjectScript expressions; see the chapter “[Caché ObjectScript Expressions](#).”

**Important:** In all places that support Caché ObjectScript expressions or statements, you can type any valid Caché ObjectScript expressions or statements, which can use any methods, routines, functions, or other elements, including your own. You are not restricted to use the elements provided within the user interface shown here. InterSystems suggests that you write your own functions wherever possible rather than use the elements that are specific to DeepSee.

# A

## DeepSee Functions

In some places, DeepSee provides a dialog box to help you create Caché ObjectScript expressions or scripts. In this dialog box, you can build expressions or scripts by double-clicking the functions, variables, and operators presented in the DeepSee function and computation library. These elements are all documented within the same dialog box.

This library consists of Caché ObjectScript functions, variables, and operators, as well as additional elements that are specific to DeepSee. The functions and variables that are specific to DeepSee are deprecated and will be replaced by Caché ObjectScript functions and variables. Two of these functions, however, are documented in this appendix.

**Important:** In all places that support Caché ObjectScript expressions or statements, you can type any valid Caché ObjectScript expressions or statements, which can use any methods, routines, functions, or other elements, including your own. You are not restricted to use the elements provided within the user interface shown here. InterSystems suggests that you write your own functions wherever possible rather than use the elements that are specific to DeepSee.

### A.1 \$\$KPI Function

This function returns the current value of a KPI, including any optional filter:

```
$$KPI(kpiid,filterexpr)
```

This function takes the following arguments:

- *kpiid* — ID of another KPI, as shown in the KPI Setup module and elsewhere in DeepSee.
- *filterexpr* — Optional quoted filter expression.

For example, the following expression returns the value of KPI 10005 divided by KPI 10001:

```
$$KPI(10005)/$$KPI(10001)
```

Suppose that KPI 10005 is based on the `Freight` measure and KPI 10001 is based on `count`. The previous expression then returns the average freight cost per order, for the entire subject area.

Consider the following variation:

```
$$KPI(10005,"[Ship Country = USA]"/$$KPI(10001,"[Ship Country = USA]"))
```

This expression returns the average freight cost per order, for orders sent to the USA.

## A.1.1 Where You Can Use \$\$KPI

You can use the **\$\$KPI** function in the definition of any of the following elements:

- KPIs. That is, you can define a KPI in terms of other KPIs.
- Measures.
- Computations.
- Speedometer captions. You can also use a KPI as the value of the speedometer. In this case, however, you choose the KPI directly and do not use the **\$\$KPI** function.
- Label captions.
- Button captions.
- Image tooltips.

If you use the **\$\$KPI** function in other locations, the function does not return the KPI value. For example, you cannot use `$$KPI(kpiid)` as the default value of a control on a dashboard.

## A.2 \$\$VAR Function

This function returns the value of a query variable, which can be either a system dashboard variable or your own custom query variable:

`$$VAR(varname)`

This function takes the following argument:

- *varname* — Name of a dashboard variable.

### A.2.1 Where You Can Use \$\$VAR

You can use the **\$\$VAR** function in the definition of any of the following elements, which do not support general Caché ObjectScript expressions:

- Speedometer captions
- Label captions
- Button captions

If you use the **\$\$VAR** function in other locations, the function does not return the variable's value.

**Note:** Do not use **\$\$VAR** in filter expressions. Instead, use the variable name within a Caché ObjectScript expression.

See the book *Using the DeepSee Dashboard Designer* for details on setting and using query variables.

# B

## Comparison of Measures, Computations, and KPIs

The following table compares measures, computations, and KPIs. Remember that base measures are defined in the Architect, and calculated measures are defined in the Analyzer.

Detail	Base Measures	Calculated Measures	Computations	KPIs
Definition can refer to source data	Yes	No	No	No
Definition can refer to a base measure	No	Yes	No	No
Definition can refer to a calculated measure	No	No	No	Yes
Definition can refer to a KPI	No	Yes	Yes	Yes
Definition can refer to other cells in a pivot table	No	No	Yes	No
Definition can include a filter	No	Yes	No	Yes
Can be used as a column in a pivot table	Yes (if numeric)	Yes	Yes	Yes*
Can be used as a row in a pivot table	No	No	Yes	No
Can be used directly in a dashboard	No	No	No	Yes
Can include display rules based on value	No	No	No	Yes

\* A KPI can be used as a column of a special type of pivot table called a *scorecard*. For information, see [Using the DeepSee Analyzer](#).

