How to: Use Inline functions

Contents

Introduction

This tutorial will show examples of how to correctly use various workflow inline functions. It applies equally to Nintex Workflow 2007 and 2010, despite the screenshots showing only Nintex Workflow 2010.

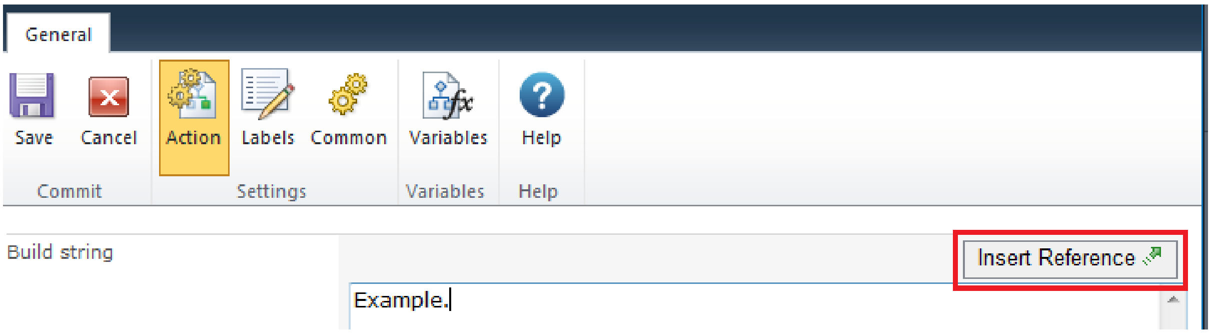
Examples will include how to correctly format inline functions, as well as explaining the use of {TextStart} and {TextEnd} to split inline function parameters.

The Nintex Workflow 2010 help files (http://connect.nintex.com/files/folders/user_guides_and_help_files_nw2010/default.aspx) contain information regarding each of the available inline functions, as well as the required input and formatting.

Inline functions can be used within any action that has an Insert Reference button.

Inline function to replace text within a workflow text variable.

The first example is based on the inline function fn-Replace().



Example.

Inline function details:

fn-Replace
Replaces a section of text in a larger string.

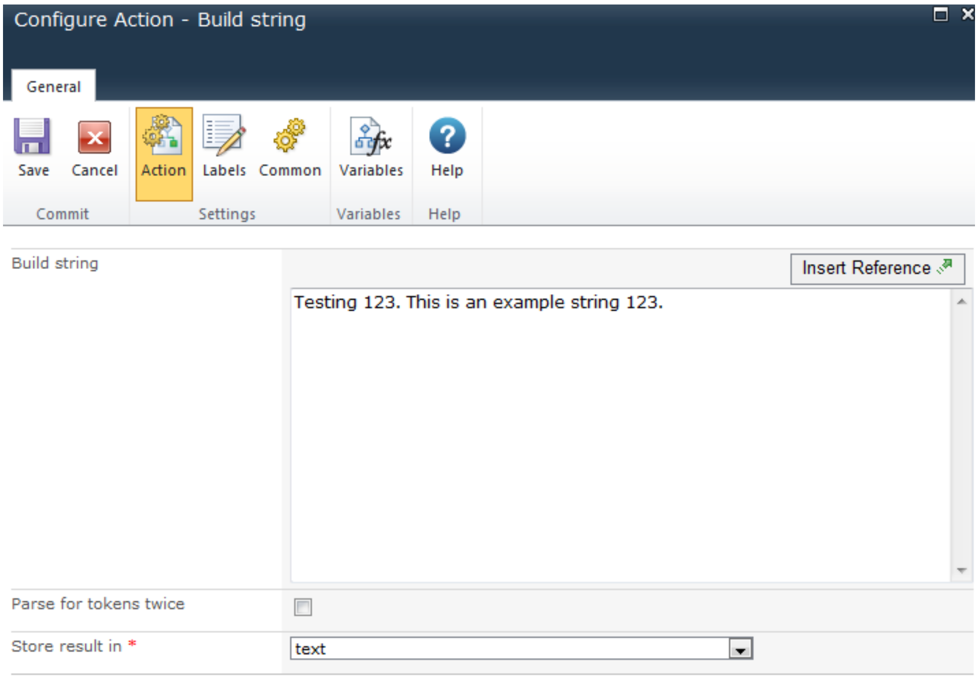
Example

fn-Replace({WorkflowVariable:Text},xxx,{ItemProperty:Title})

Arguments

Text The text to modify.
Old value The text to search for and replace.
New value The text to replace Old value with.

The first step is to create a simple text string to edit. This can be done using a ‘Build dynamic string’ action.

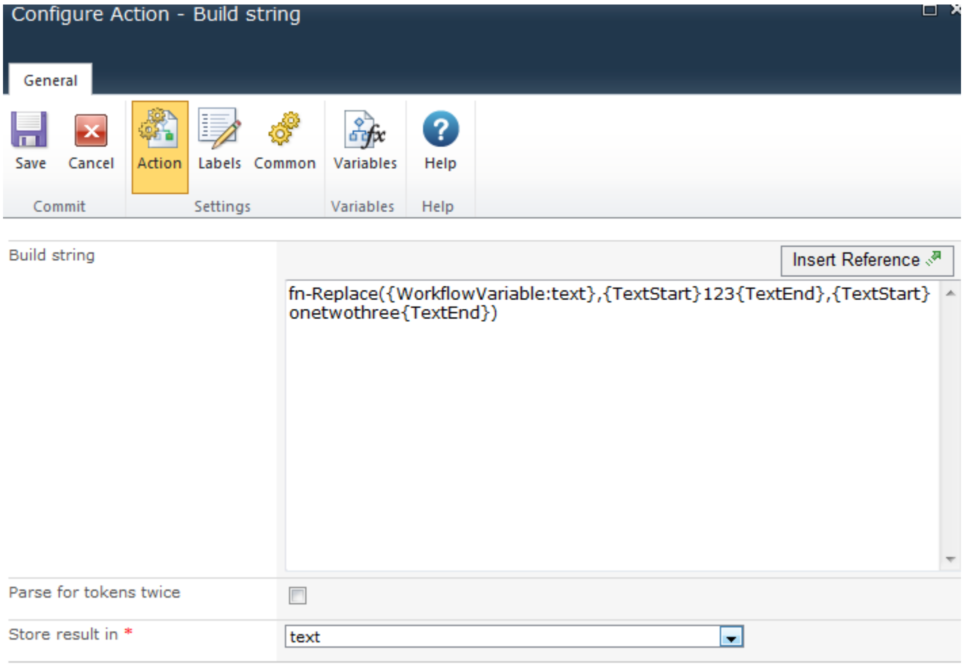


Build string: Testing 123. This is an example string 123.

Parse for tokens twice □

Store result in *: text

Another ‘Build dynamic string’ action can then be used to replace the contents of the text variable. In this case ‘123’ will be replaced with ‘onetwothree’.



Build string:

```
fn-Replace({WorkflowVariable:text},{TextStart}123{TextEnd},{TextStart}
onetwothree{TextEnd})
```

Parse for tokens twice □

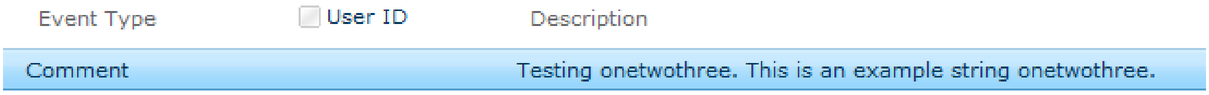
Store result in *: text

fn-Replace({WorkflowVariable:text},{TextStart}123{TextEnd},{TextStart}onetwothree{TextEnd})

The {TextStart} {TextEnd} tokens are used to ensure any values between these tokens will not interfere with the inline function syntax. For example you can use special characters between these tokens such as ‘/()@#$^ and they will correctly be replaced rather than resulting in an error when the workflow attempts to evaluate the inline function.

The text variable can then be used in any other action or logged to the history list to confirm it is correct.

Example when logging the result to the history list:



Event Type	☐ User ID	Description
Comment		Testing onetwothree. This is an example string onetwothree.

Inline function to correctly format a date/time value.

Date/Time variables are by default in the format 21/11/06 10:30:23 AM. However, often within a workflow the designer will need to know the current day or month a date value represents. The inline function fn-FormatDate() can be used to achieve this.

Inline function details:

fn-FormatDate
Represents a date time value in text of a specific format.

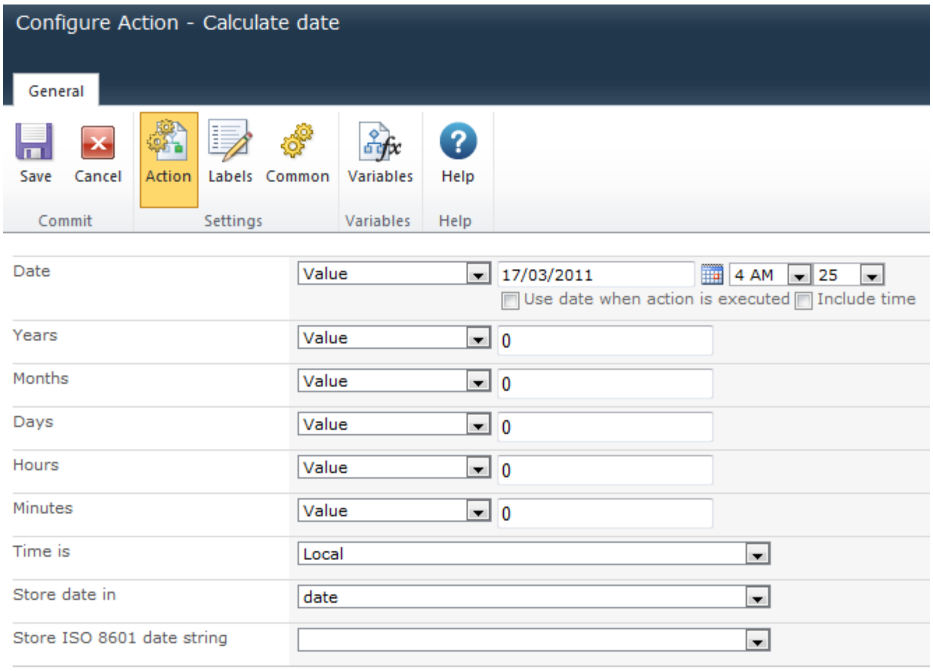
Example

fn-FormatDate({WorkflowVariable:MyDate}, d)

Arguments

Date A variable containing a date value.
Format string Text describing how the date time value should be formatted.

To format a date, the date must first be stored within a workflow date variable or referenced directly from a SharePoint date/time field. In this example, a date variable will be used and configured using the 'Calculate Date' action.



Date	Value	17/03/2011 4 AM 25 □ Use date when action is executed □ Include time
Years	Value	0
Months	Value	0
Days	Value	0
Hours	Value	0
Minutes	Value	0
Time is	Local	
Store date in	date	
Store ISO 8601 date string		

The above configuration will store the date 17/03/2011 4:25 AM into a workflow date variable named 'date'.

A ‘Build dynamic string’ action can then be used to format the date as required.

Return full month name:

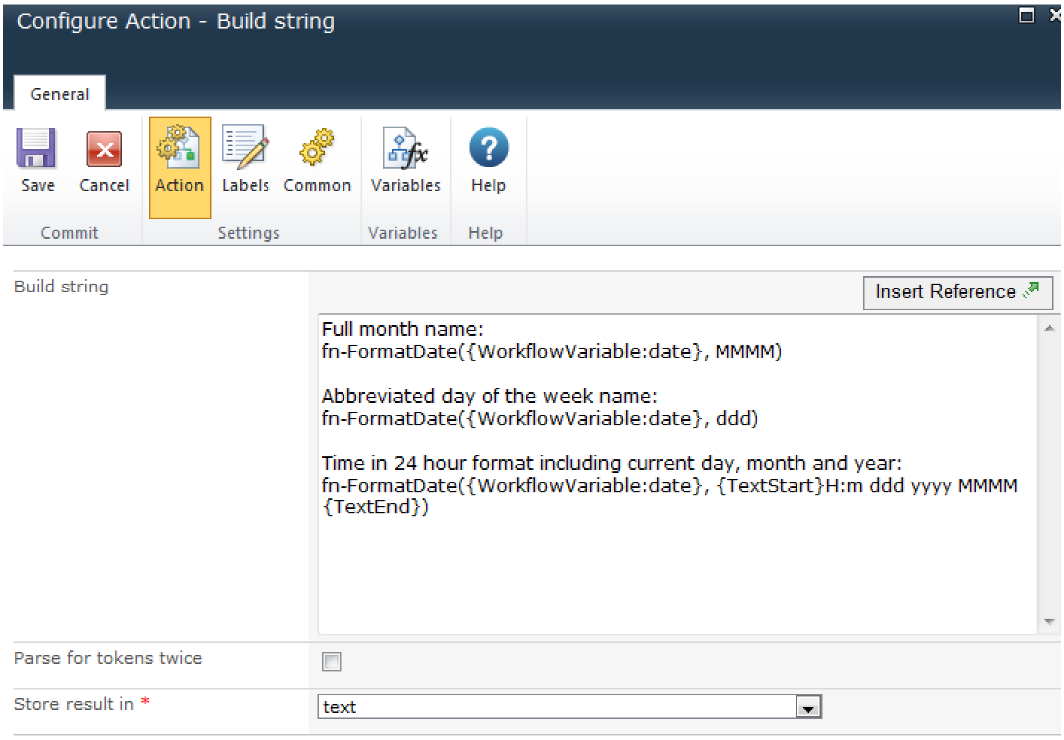
fn-FormatDate({WorkflowVariable:date}, MMMM)

Return abbreviated day of the week name:

fn-FormatDate({WorkflowVariable:date}, ddd)

Return time in 24 hour format including current day, month and year:

fn-FormatDate({WorkflowVariable:date}, {TextStart}H:m ddd yyyy MMMM{TextEnd})



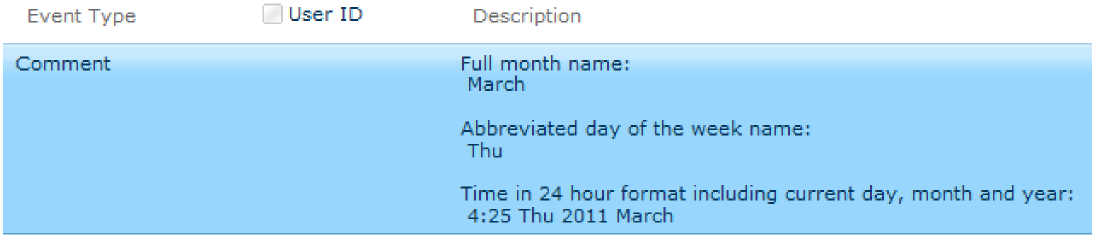
Build string

```
Full month name:
fn-FormatDate({WorkflowVariable:date}, MMMM)

Abbreviated day of the week name:
fn-FormatDate({WorkflowVariable:date}, ddd)

Time in 24 hour format including current day, month and year:
fn-FormatDate({WorkflowVariable:date}, {TextStart}H:m ddd yyyy MMMM
{TextEnd})
```

Parse for tokens twice □

Store result in * text

Example result logged to history list:



Event Type	□ User ID	Description
Comment		Full month name: March Abbreviated day of the week name: Thu Time in 24 hour format including current day, month and year: 4:25 Thu 2011 March

Using an inline function to determine the difference in days between 2 date variables

Adding to this example you can also use an inline function to determine the difference between 2 date variables.

Example:

date1 - 2/03/2011 4:25 AM
date2 - 18/03/2011 9:05 PM

Inline function details:

fn-DateDiffDays
Determines the number of days between two dates.

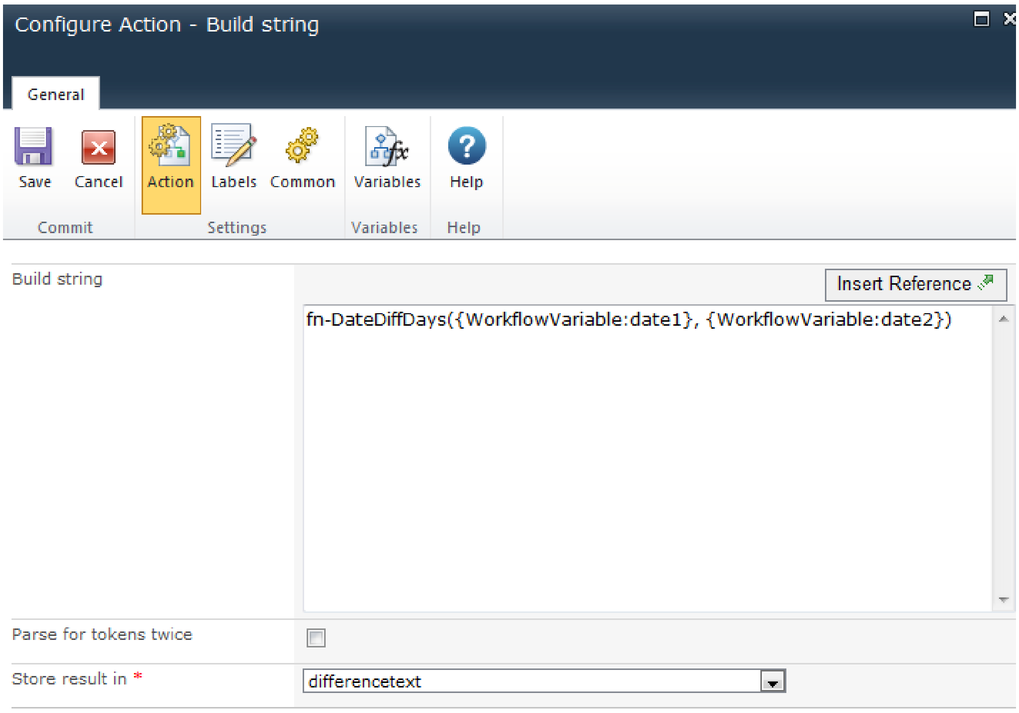
Example

fn-DateDiffDays({WorkflowVariable:StartDate}, {WorkflowVariable:EndDate})

Arguments

Start date The starting date and time to calculate the difference between.
End date The end date and time to calculate the difference between.

The following inline function will return the difference between the dates in days.



Build string

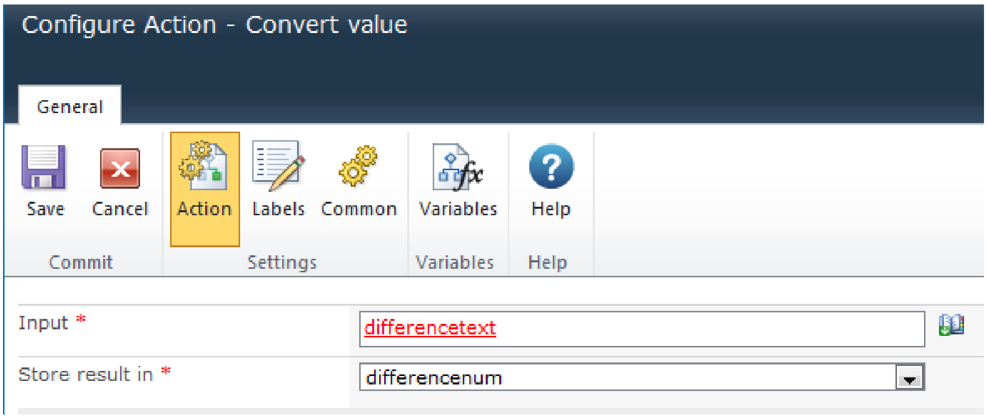
```
fn-DateDiffDays({WorkflowVariable:date1}, {WorkflowVariable:date2})
```

Parse for tokens twice □

Store result in * differencetext

fn-DateDiffDays({WorkflowVariable:date1}, {WorkflowVariable:date2})

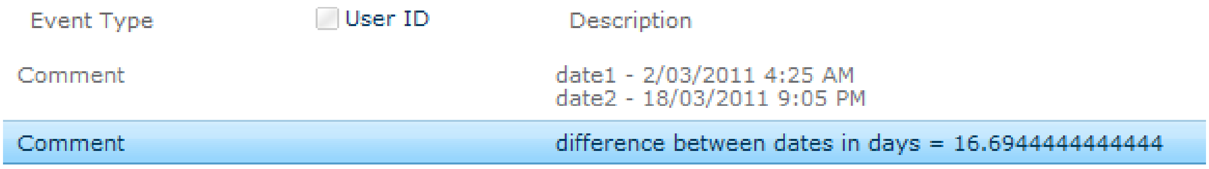
The convert value action can then be used to convert this text result to a number.



Configure Action - Convert value

Input *	differencetext
Store result in *	differencenum

Example output:



Event Type	□ User ID	Description
Comment		date1 - 2/03/2011 4:25 AM date2 - 18/03/2011 9:05 PM
Comment		difference between dates in days = 16.6944444444444