Condition User Guide

Contents

Introduction

Condition is a powerful expression system which can realize many types of complicated conditions.

The condition is an expression of returning Boolean value, true or false. The Boolean value will decide if the function will take effect. This means that only when a condition returns as true, then a predefined column or view permission is enabled. Otherwise, the product will not function. Note, if an error occurs in the condition, then the returned value will be false.

How to use condition editor

The condition can be edited in Basic or Advanced modes.

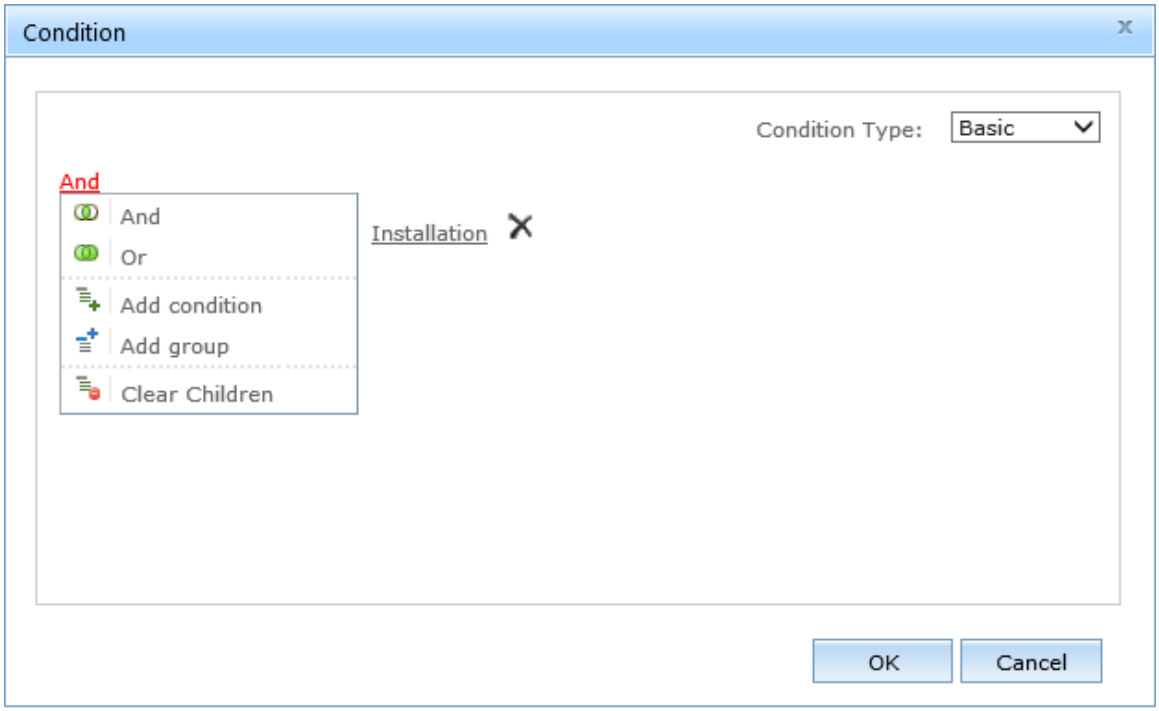
Basic mode: launch the editor in basic mode, then simply use the drop-down menu or elements.

Advanced mode: launch the editor in advanced mode and enter the expression manually. You can create expression using the predefined variables and functions.

Expressions in basic mode can saved after being converted to advanced mode.

Basic mode

Open the condition editor, and set Condition Type as Basic, then you can add and edit expressions using the predefined columns and operators.



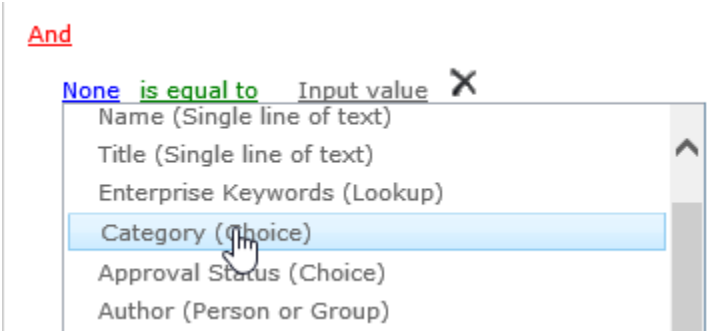
The elements available in Basic mode are shown below.

Condition Type	Choose the condition mode, Basic or Advanced.
AND	Performs a logical conjunction on two expressions.
OR	Performs a logical disjunction on two Boolean expressions.
Add condition	Add an expression in the editor.
Add group	Add a group of expressions joined by AND or OR operator.
Clear Children	Remove the expressions under one logic operator.
✕	Remove the expression.

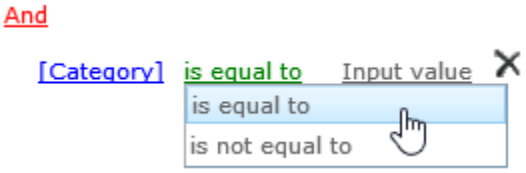
To create an expression.

a. Click **Add condition** first.

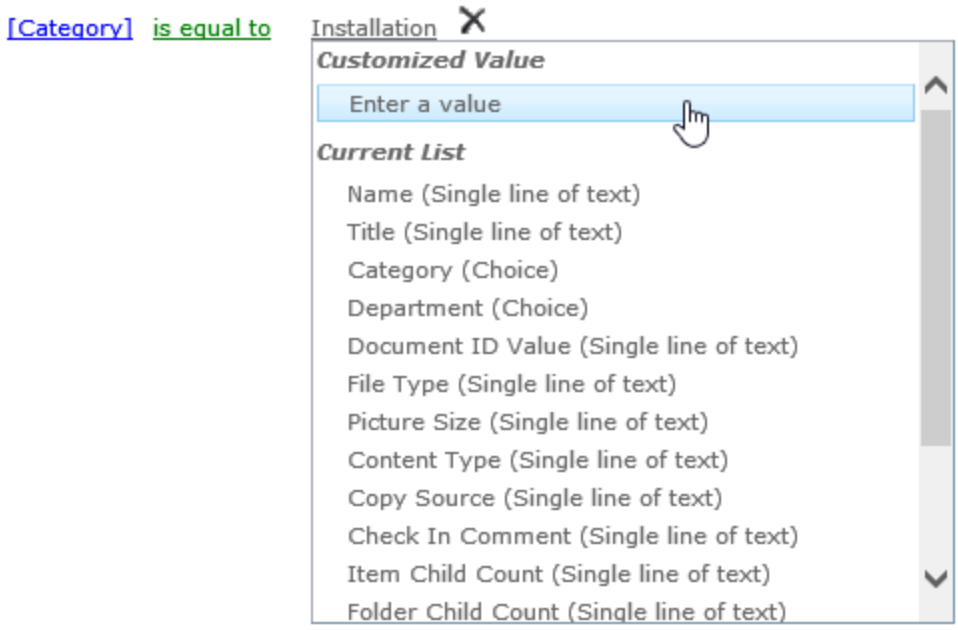
b. Select one SharePoint column from the drop-down list.



c. Select one predefined operator from the drop-down list.



d. Specify the input value.

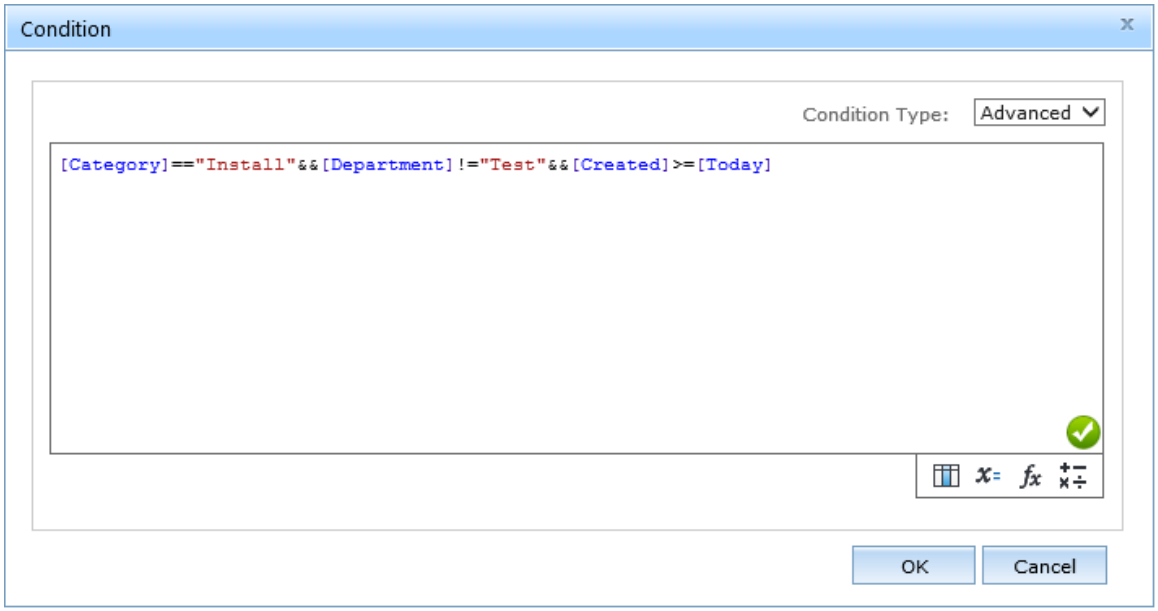


You can enter the value or select a SharePoint column.

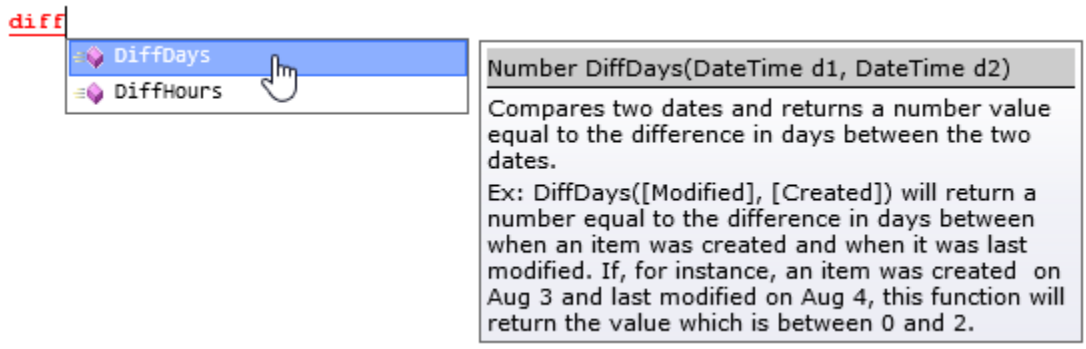
e. To add more expressions, click the **Add** or **Or** operator, then add an expression based on the above steps.

Advanced mode

To create an expression in Advanced mode, set the Condition Type as Advanced. Then you can just select and insert the predefined functions, operators and constants from the drop-down list, or enter the expression manually.



The auto-complete feature is provided when you insert a function, which helps you easily and quickly add expressions.

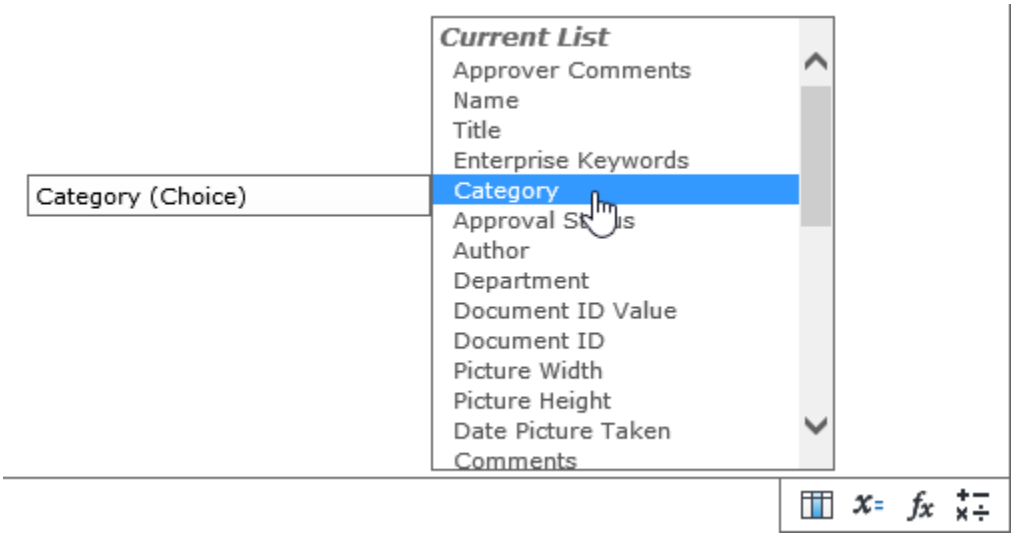


Elements in the advanced mode.

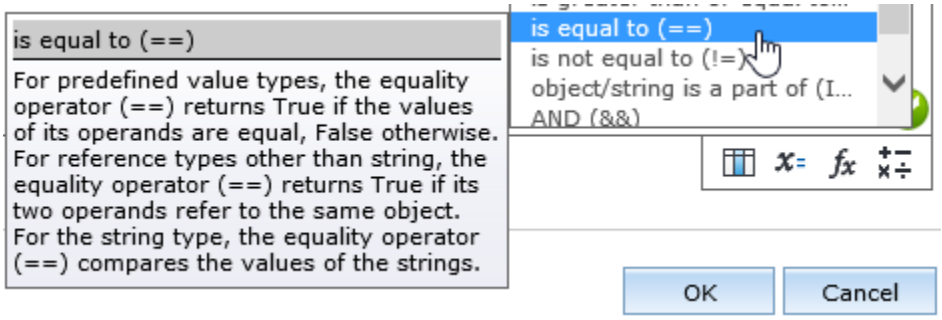
	SharePoint column, includes all SharePoint columns in current list or library.
	Constant
	Function
	Operator
	Indicator which indicate if the expression is valid. Indicate expression is valid. Indicate expression is invalid.

To add an expression.

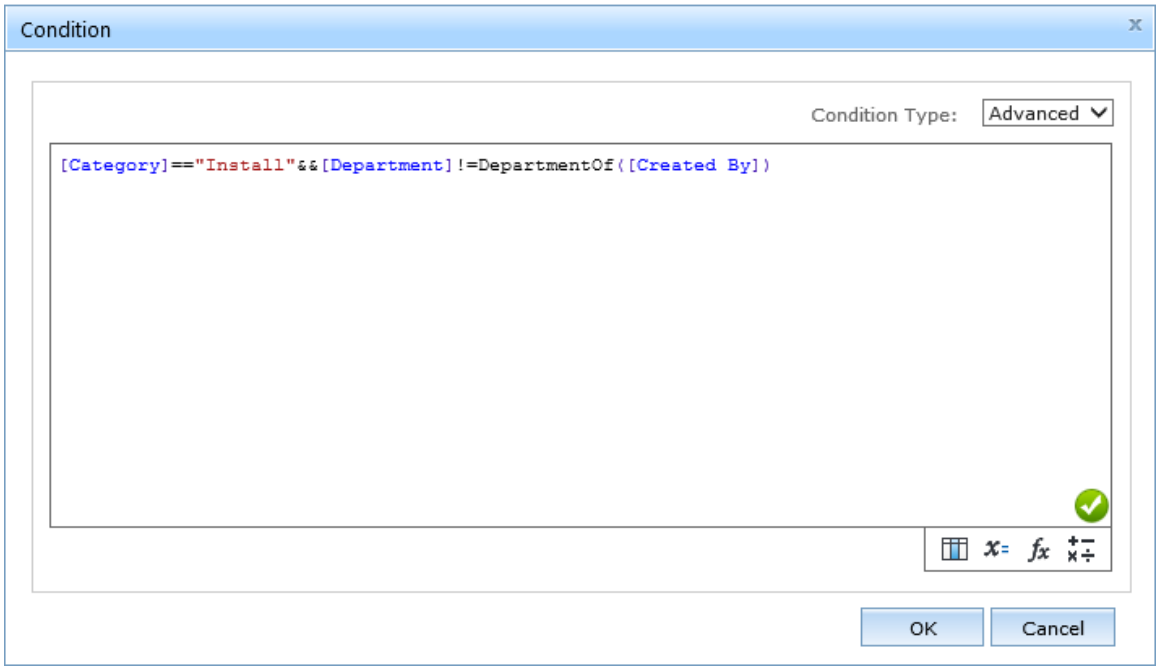
a. In the editor area, enter an operand first, such as a column or constant.



b. Then enter or select an operator.



c. Enter an operand, such as a column or constant.



d. After expression finished, wait to see if the expression is validated.

   An invalid expression cannot be saved successfully.

Insert Columns

You can insert a column in condition using the "Column" drop-down list. And you can also enter the column name manually.

For the current list column, you can type the column name with brackets, such as [Title].

For the column from another list, the name must be separated by a dot , such as [(List).column].

In the list, columns may contain some special characters, and they will converted based on the condition rules.

The following table is the conversion rules.

Special characters	Convert to
\	\\

(	\(
)	\)
"	\"

Expressions

About an Expression

The condition is an expression which will return True or False.

And, an expression is composed of operands and operators.

Operands

An operand is an entity on which an operator acts. An operand can be any column name, constant, value, or a sub-expression.

Operators

Operators that can be used in an expression are contained below.

Numeric Operators

The numeric operators only can be used in advanced mode.

Operators	Description	Apply to
+	plus	Advanced mode
-	minus	Advanced mode
*****	Multiplied by	Advanced mode
/	Devided by	Advanced mode
%	remainder	Advanced mode

+ operator rules

+	Text	Integer	Number	Boolean	DateTime	User
Text	Yes	No	No	No	No	No
Integer	No	Yes	Yes	No	No	No
Number	No	Yes	Yes	No	No	No
Boolean	No	No	No	No	No	No
DateTime	No	No	No	No	No	No

User	No	No	No	No	No	No

- (binary), *, /, % operator rules

-, *, /	**Text**	**Integer**	**Number**	**Boolean**	**DateTime**	**User**
Text	No	No	No	No	No	No
Integer	No	Yes	Yes	No	No	No
Number	No	Yes	Yes	No	No	No
Boolean	No	No	No	No	No	No
DateTime	No	No	No	No	No	No
User	No	No	No	No	No	No

- (unary) operator rules

%	**Text**	**Integer**	**Number**	**Boolean**	**DateTime**	**User**
	No	Yes	Yes	No	No	No

Logic and Boolean Operators

In this chapter, we will introduce Logic and Boolean operators included in the condition and operation rules.

Operators	**Description**	**Apply to**
!	Is not	Advanced mode
<	Is less than	Advanced, Basic mode
<=	Is less than or equal to	Advanced, Basic mode
>	Is greater than	Advanced, Basic mode
>=	Is greater than or equal to	Advanced, Basic mode
==	Is equal to	Advanced, Basic mode
!=	Is not equal to	Advanced, Basic mode
&&	AND	Advanced, Basic mode
||	OR	Advanced, Basic mode
IN	Object/string is a part of	Advanced, Basic mode
Begin with		Basic condition

! operator rule

!	Text	Integer	Number	Boolean	DateTime	User
	No	No	No	Yes	No	No

&& operator rule

&&, ||	Text	Integer	Number	Boolean	DateTime	User
Text	No	No	No	No	No	No
Integer	No	No	No	No	No	No
Number	No	No	No	No	No	No
Boolean	No	No	No	Yes	No	No
DateTime	No	No	No	No	No	No
User	No	No	No	No	No	No

<, <=, >, >= operators rule

<, <=, >, >=	Text	Integer	Number	Boolean	DateTime	User
Text	No	No	No	No	No	No
Integer	No	Yes	Yes	No	No	No
Number	No	Yes	Yes	No	No	No
Boolean	No	No	No	No	No	No
DateTime	No	No	No	No	Yes	No
User	No	No	No	No	No	No

==, != operators rule

==, !=	Text	Integer	Number	Boolean	DateTime	User
Text	Yes	No	No	No	No	No
Integer	No	Yes	Yes	No	No	No
Number	No	Yes	Yes	No	No	No
Boolean	No	No	No	Yes	No	No
DateTime	No	No	No	No	Yes	No
User	No	No	No	No	No	Yes

Operation Precedence

Precedence rules determine the order in which operations are performed within expressions. High precedence operations are performed before lower precedence operations.

This list indicates the precedence of operators from highest to lowest:

Priority	Operator
1	( )
2	!
3	* / %
4	+ -
5	< > <= >=
6	== !=
7	&&
8	||

Data types

There are 6 data types within Conditions, and SharePoint columns and constants will be mapped to them:

Text Data Type

Text Data Type allows the storage of characters, including spaces, punctuation marks and symbols and is ideal for use in storing names and sentences. A Text value must be enclosed with double quotes ("").

For example:

"Hello, world!"

Integer Data Type

Integer Data Type defines a number that does not require the storage of a decimal part. This data type represents signed numbers with values ranging from negative 2147483647 through positive 2147483647.

For example:

-204

248

Decimal Data Type

Decimal Data Type defines a number that can contain a decimal part.

For example:

248.123

Boolean Data Type

Boolean Data Type represents a Boolean value. It can only store TRUE or FALSE values.

For example:

True

False

DateTime Data Type

DateTime Data Type stores an instance of time expressed as a date and time of day. The supported range of this data type is from 1900-01-01 00:00:00 to 8900-12-31 23:59:59.

For example:

2013-01-01 00:00:00

User Data Type

User Data Type represents a SharePoint user or group value.

For example:

Hans Zermo

Within Condition, only Decimal and Integer can be mutually converted automatically.

The decimal fraction of a value will be rounded down directly when converting Decimal to Integer, while Integer will be converted to a float-point type directly when converting Integer to Decimal.

The following functions can be used to convert data types:

ToBoolean

ToDateTime

ToPeople

ToInt

ToNumber

ToText

Columns

Columns represent SharePoint columns, which will be replaced by the actual values of the column during expression calculation.

Each Column has a SharePoint type, such as a single line of text, a number, currency, etc., and they map another type in expression. The following table displays the relationship between SharePoint column type and condition data type.

Column type	**Data type**	**Note**
Choice	Text	
Single line of text	Text	
Multiple lines of text	Text	
Number	Decimal	If the number is shown as percentage, it will be changed to its true value. E.g., 10% will be 0.1.
Currency	Decimal	
Data and Time	DateTime	
Lookup	Text, Decimal or DateTime	It depends on the column type you look up.
Yes/No	Boolean	
Person or Group	User	
Calculated	Text, Decimal or DateTime	It depends on the calculated column type.
Hyperlink or Picture	Text	
External Data	Text	
Managed Metadata	Text	
ID	Integer	

Version	Decimal	
Attachment	Boolean	True represents there is an attachment in the item; otherwise, false.
Content Type	Text	The content type name.
Workflow Status	Integer	Each workflow status is represented by a corresponding number: Not Started = 0 Failed On Start = 1 In Progress = 2 Error Occurred = 3 Canceled = 4 Completed = 5 Failed On Start (Retrying) = 6 Error Occurred (Retrying) = 7 Canceled = 15 Approved = 16 Rejected = 17
Approval Status	Integer	Each approval status is represented by a corresponding number: Approved=0 Rejected=1 Pending=2 Draft=3 Scheduled=4
Folder Child Count	Integer	
Item Child Count	Integer	
Approver Comments	Text	
Check Out To	User	
Check In Comments	Text	
Copy Source	Text	
File Size	Text	
Workflow Name	N/A	
Name	Text	
Recurrency	Boolean	
Post	Text	
Post by	User	

Picture Size	Text	
Picture Height	Text	
Picture Width	Text	
UDC Purpose	Text	
Connection Type	Text	

Following table indicates data types of SharePoint special columns.

Column type	**Data type**
Full HTML content with formatting and constraints for publishing	Text
Image with formatting and constraints for publishing	Text
Hyperlink with formatting and constraints for publishing	Text

And, each BoostSolutions column maps these data types.

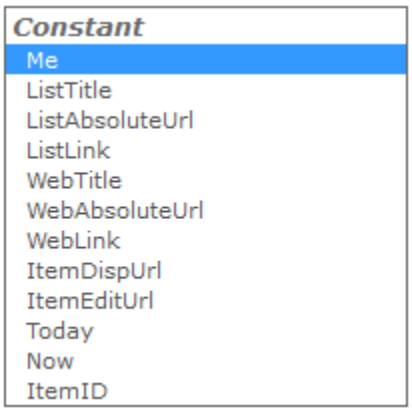
Column type	**Return Data type**	**Note**
Cross-Site Lookup	Text	Same as SharePoint lookup column.
Cascaded Lookup	Text	Same as SharePoint lookup column.
Discussion Column	Text	
Choice indicator	Text	Same as SharePoint choice column.
Progress monitor	Decimal	Same as SharePoint number column.

You can insert the columns from the drop-down list provided in Condition Editor, or can type the column name manually (For details, see Insert Columns).

Constant

In order for customization, Condition provides more variables beyond SharePoint columns. You can inset constants like SharePoint columns.

The constants can only be used in advanced mode, except the Today constant which can be used in basic and advanced mode.



Constant	Definition	Data Type
Me	The current logged in user.	User
ListTitle	The title of the current list or document library.	Text
ListAbsoluteUrl	The absolute url of the current list or document library.	Text
ListLink	The link of the current list or document library.	Text
WebTitle	The site title which the list or document library is located.	Text
WebAbsoluteUrl	The sites absolute url which the list or document library is located.	Text
WebLink	The site link which the list or document library is located.	Text
ItemDispUrl	The item display form url.	Text
ItemEditUrl	The item edit form url.	Text
Today	The current date.	DateTime
Now	The current date and time.	DateTime
ItemID	The item ID.	Text

Functions

Function: The supplement of operators, offering more rich functions.

The functions are categorized by their functionality.

- Logic functions
- Convert functions
- DateTime functions
- Text functions
- User Functions
- Math functions

Logic Functions

If Function

Checks the logical condition and return one value if true and another value if false.

Syntax

```
If (Boolean B, Type v1, Type v2)
```

Parameters

B

Type: Boolean

An expression, condition or Boolean value will return True or False.

v1

Type: Text, Integer, Decimal, Boolean, DateTime, User

The first value to return.

v2

Type: Text, Integer, Decimal, Boolean, DateTime, User

The second value to return.

Returns

Type: Text, Integer, Decimal, Boolean, DateTime, User

Returns v1 if the condition is met, otherwise, returns v2.

Example

(If([Priority]=="(1) High", 5, 2)) > DiffDays([Due Date],[Today]) means that this function will return 5 if the difference in days between today and the due date is smaller than the value returned by the if condition. If the priority is "(1) High", the value returned is 5, otherwise it is 2.

IsNull Function

Returns Boolean value whether an object is null or valid. The object could be any data type.

Syntax

```
IsNull(Object o)
```

Parameters

o:

Type: Text, Integer, Decimal, Boolean, DateTime, User

An object to be checked for NULL.

Returns

Type: Boolean

Returns *True* if the object is null, otherwise, returns *False* if the object contains data.

Example

IsNull([ID]), the function would return False. Because any ID column in SharePoint item cannot be null.

IfError Function

Checks if the first parameter meet an error, and returns the corresponding value.

Syntax

```
IfError(Type, Type)
```

Parameters

Type (first)

Expression

Type (second)

Type: Text, Integer, Decimal, Boolean, DateTime, User

The value will be returned if expression occurs an error.

Returns

Type: Text, Integer, Decimal, Boolean, DateTime, User

Return the specified value if the expression evaluates to an error; otherwise, return the value of the expression itself.

Example

If(ToDateTime("2011/12/19")>ToDateTime("Due Date"), False) will returns False, because the expression met an error that text cannot be converted to a DateTime type.

Contains Function

Determines whether the second value is contained in the first value.

Syntax

```
Contains(Type, Type)
```

Parameters

Type: Text, User

The text or user used to be compare. The first parameter and second should be same type.

Returns

Type: Boolean

Returns True is the second parameter is contained in first one, otherwise, returns False.

Example

Contain([Attendees], ToPeople("SharePoint\Tom")). Suppose the [Attendees] contain an AD group Product Team and a user Jerry and Tom is member of the Product Team. This function will return True despite the Tom is not equal to Jerry or Product Team.

Contain("SharePoint", "Share") will return *True* because the "Share" is contained in "SharePoint".

Data Type Conversion Functions

ToBoolean

Converts value to Boolean.

Syntax

```
Boolean(Type)
```

Parameters

Type

Type: Text, Integer, Decimal

The value to convert.

Returns

Type: Boolean

For Text, the “True” and “Yes” text will be converted to *True*. Any other value is *False*.

For Integer, returns False if the Integer value is 0; otherwise, returns True.

For Decimal, returns False if the Decimal value is 0; otherwise, returns True.

Other type, converts to *False* if the value is null, otherwise converts to *True*.

Example

ToBoolean (10) returns True.

ToDateTime

Converts the specified string representation of a date and time to an equivalent date and time value.

Syntax

```
ToDateTime(Text s)
```

Parameters

s

Type: Text

A string to convert.

Returns

Type: DateTime

The date and time equivalent of the specified string.

Example

ToDateTime("9/8/2009") will returns the date Sept 8, 2009.

ToPeople

Converts Text to User.

Syntax

```
ToPeople(Text s)
```

Parameters

s

Type: Text

A string to convert.

Returns

Type: User

The user equivalent of the specified string.

Example

[Created By] == ToPeople("Tom") will check if the item was created by Tom. In this expression, "Tom" is a string and [Created By] (in this case) must be a user or group.

ToInt

Converts the specified string representation of a number to an equivalent integer value.

Syntax

```
ToInt(Text s)
```

Parameters

s

Type: Text

A string that contains the number to convert.

Returns

Type: Integer

An integer that is equivalent to the number in string.

Example

ToInt("10") returns integer 10.

ToNumber

Converts the specified string representation of a number to an equivalent decimal number

Syntax

```
ToNumber(Text s)
```

Parameters

s

Type: Text

A string that contains the number to convert.

Returns

Type: Decimal

A decimal number that is equivalent to the number in string.

Example

ToNumber('1210.73') returns the number 1210.73

ToText

Converts any type value to Text.

Syntax

```
ToText(Type)
```

Parameters

Type: DateTime, Boolean, Decimal, Integer, User

A value to set.

Returns

Type: Text

A string.

For User type, converts to full login name.

For Boolean type, converts True to “True”, convert False to “False”.

For DateTime type, converts to text in the format of the current region configuration.

Example

ToText(10) returns 10.

Date and Time Functions

AddDays Function

Adds the specified number of days to one date and time.

Syntax

```
AddDays(DateTime d, Number n)
```

Parameters

d

Type: DateTime

A specified date and time.

n

Type: Integer

A number of whole days. The value can be negative or positive.

Returns

Type: DateTime

A new date and time that adds the specified number of days.

Example

AddDays([Today], 4) would add 4 days to the current date, meaning if today's date is Oct 14, the function would return Oct 18.

AddHours Function

Adds the specified number of hours to a date and time.

Syntax

```
AddHours(DateTime d, Number n)
```

Parameters

d

Type: DateTime

A specified date and time.

n

Type: Integer

A number of whole hours. The value can be negative or positive.

Returns

Type: DateTime

A new date and time that adds the specified number of hours.

Example

AddHours([Now], 2) would return a time 2 hours after the current time. If the current time is 9:53 am, this function would return 11:53 am.

AddMonths Function

Adds the specified number of months to a date and time.

Syntax

```
AddMonths(DateTime d, Number n)
```

Parameters

d

Type: DateTime

A specified date and time.

n

Type: Integer

A number of whole months. The value can be negative or positive.

Returns

Type: DateTime

A new date and time that adds the specified number of months.

Example

AddMonths([Modified], 2) would return a date 2 months after the modified date. If the modified date is Oct 10, this function would return Dec 10.

AddYears Function

Adds the specified number of years to a date and time.

Syntax

```
AddYears(DateTime d, Number n)
```

Parameters

d

Type: DateTime

A specified date and time.

n

Type: Integer

A number of whole years. The value can be negative or positive.

Returns

Type: DateTime

A new date and time that adds the specified number of years.

Example

AddYears([Today], 1) would return a new date that is one year after today. If today's date is May 1, 2007, the function would return May 1, 2008.

Day Function

Returns the number of days in the specified month and year.

Syntax

```
Day(DateTime d)
```

Parameters

d

Type: DateTime

A specified date and time.

Returns

Type: Integer

The number of days in month for the specified date and time.

The number is given as an integer ranging from 1 to 31.

Example

Day([Today]) would return the number corresponding to today's date, meaning that if today is July 4, 1996 the function would return the value 4.

DiffDays Function

Compares two dates and returns a number value equal to the difference in days between the two dates.

Syntax

```
Day(DateTime d1, DateTime d2)
```

Parameters

d1

Type: DateTime

A specified date to compare.

d2

Type: DateTime

A specified date to compare.

Returns

Type: Integer

An integer equals to the difference in days between two dates.

Example

DiffDays([Modified], [Created]) would return the difference in days between when an item was created and when it was last modified. If an item was created on Aug 3 and last modified on Aug 4, this function would return 1.

DiffHours Function

Compares two time values and returns a number value equal to the difference in hours between the two time values.

Syntax

```
Day(DateTime d1, DateTime d2)
```

Parameters

d1

Type: DateTime

A specified time to compare.

d2

Type: DateTime

A specified time to compare.

Returns

Type: Integer

An integer equals to the difference in hours between two times.

Example

DiffHours([Modified], [Created]) would return the difference in hours between when an item was created and when it was last modified. If an item was created on Aug 3 at 8:00 am and last modified on Aug 3 at 10:00 am, the result is 2.

GetDate Function

Retrieves the date from a date and time value.

Syntax

```
GetDate(DateTime d)
```

Parameters

d

Type: DateTime

A specified date and time.

Returns

Type: DateTime

The sequential serial number that represents a particular date.

Example

GetDate([Now]) would return today's date, if it is now Jan 1, 2006 11:00 am, this function will return 1/1/2006.

GetTime Function

Retrieves the time from a date and time value.

Syntax

```
GetTime(DateTime d)
```

Parameters

d

Type: DateTime

A specified date and time.

Returns

Type: DateTime

A value indicating the time of datetime value.

Example

GetTime([Now]) would return the current time, if it is now Jan 1, 2006 11:00 am, this function would return 0001-1-11 11 am.

Hour Function

Returns a number that represents the hour from a datetime value.

Syntax

```
Hour(DateTime d)
```

Parameters

d

Type: DateTime

A specified date and time.

Returns

Type: Integer

An integer that represents the hour from a datetime value, ranging from 0 (12:00 A.M.) to 23 (11:00 P.M.)

Example

Hour([Now]) means that if now is 2012/12/19 17:24:30, the function would return the value 17.

Weekday Function

Returns a number representing the day of the week.

Syntax

```
Weekday (DateTime d)
```

Parameters

d

Type: DateTime

A specified date and time.

Returns

Type: Integer

An integer represents the day of the week, ranging from 0 (Sunday) to 6 (Saturday).

Example

Weekday([Today]) would return the number corresponding to the current day of the week, meaning that if today is Thursday, the function would return the value 4.

Year Function

Returns the year of datetime value.

Syntax

```
Year (DateTime d)
```

Parameters

d

Type: DateTime

A specified date and time.

Returns

Type: Integer

An integer represents the year of the datetime value, ranging from 1 to 9999.

Example

Year([Created]) would return the number of the created year, if created date is Oct 17, 2009, the function will return 2009.

Month Function

Returns the month of a date represented by a serial number.

Syntax

```
Month (DateTime d)
```

Parameters

d

Type: DateTime

A specified date and time.

Returns

Type: Integer

An integer represents the month of the datetime value, ranging from 1 (January) to 12 (December).

Example

Month([Created]) would return the number corresponding of the created month, meaning that if created date is Oct 17, 2009, the function would return the value 10.

IsLastMonthDay Function

Determines if the date is the last day in that month.

Syntax

```
IsLastMonthDay (DateTime)
```

Parameters

Type: DateTime

A specified date to determine.

Returns

Type: Boolean

Returns True if the date is the last day of month; otherwise, returns False.

Example

IsLastMonthDay(ToDateTime("2011/12/31")) would return True, because the 2011/12/31 is the last day in December.

WeekOfMonth

Return the week of month.

Syntax

```
WeekOfMonth (DateTime)
```

Parameters

Type: DateTime

A specified date.

Returns

Type: Integer

An integer indicates the week of month.

Example

WeekOfMonth(ToDateTime("2011/12/20")) will return 4, because the 2011/12/20 is in the fourth week in the December.

WeekOfYear

Return the week of year.

Syntax

```
WeekOfYear (DateTime)
```

Parameters

Type: DateTime

A specified date.

Returns

Type: Integer

An integer indicates the week of year.

Example

WeekOfYear(ToDateTime("2011/2/2")) will return 5, because the 2011/2/2 is the fifth week in the 2012.

DateEqual

Determines if the two datetime values are same.

Syntax

```
DateEqual (DateTime, DateTime)
```

Parameters

Type: DateTime

Two specified dates to compare.

Returns

Type: Boolean

Returns True if two datetime values are same; otherwise, returns False.

Example

DateEqual(ToDateTime("2011/12/1 03:22:05"), ToDateTime("2011/12/1 20:45:08")) will return True, because they are in same day.

BirthdayEqual

Determines whether two datetime instances have the same year and month value.

Syntax

BirthdayEqual (DateTime, DateTime)

Parameters

Type: DateTime

Two specified dates to compare.

Returns

Type: Boolean

Return True if the year and month are both same; otherwise, returns False.

Example

BirthdayEqual(ToDateTime("1984/12/1"), ToDateTime("2012/12/1")) will return true, because they are all in 1 December.

Text Functions

IndexOf

Searches for the specified text and returns the zero-based index of it if it exists.

Syntax

```
IndexOf (Text s1, Text s2)
```

Parameters

s1

Type: Text

From which to search.

s2

Type: Text

The string to seek.

Returns

Type: Integer

Returns the zero-based position in text where search text can be found. Returns –1 if search is not found or if search is empty.

Example

IndexOf(“First name”, “i”) will return 1.

SubString

Returns a sub-string of *t* beginning at *start* zero-based position and with *length* characters.

Syntax

```
Substring (Text s, Integer start, Integer count)
```

Parameters

s

Type: Text

From which to search.

start

Type: Integer

The index of the start of the substring.

count

Type: Integer

The number of characters in the substring, it is optional.

Returns

Type: Text

A string equivalent to the substring of length *count* that begins at *start* in the text, or Empty if start is equal to the length of this text and length is zero.

Example

Substring(SharePoint, 5, IndexOf(“First name”, “s”)) will return ePo.

LengthOf

Gets the count of characters in text.

Syntax

```
LengthOf (Text)
```

Parameters

Type: Text

String to be counted.

Returns

Type: Integer

An integer that specifies the length of the string.

Example

LengthOf("SharePoint") will return integer 10.

ToLower

Converts text to the equivalent lowercase text.

Syntax

```
ToLower (Text s)
```

Parameters

s

Type: Text

String to be converted.

Returns

Type: Text

A string in lowercase.

Example

ToLower([Title]) would convert all letters in the Title column to lowercase. If the title is "Company Name", this function would return "company name".

ToUpper

Converts text to the equivalent uppercase text.

Syntax

```
ToUpper (Text s)
```

Parameters

s

Type: Text

String to be converted.

Returns

Type: Text

A string in uppercase.

Example

ToUpper ([Title]) would convert all letters in the Title column to lowercase. If the title is "Company Name", this function would return "COMPANY NAME".

User information Functions

NameOf

Gets the name of a specified SharePoint user.

Syntax

```
NameOf (User)
```

Parameters

Type: User

A user object that represents the user.

A specified SharePoint user.

Returns

Type: Text

Returns the name as a string.

Example

NameOf([Created by]) will return the name of who created the item, such as “Hans”.

ManagerOf

Gets the manager of a SharePoint user based on the Active Directory settings.

Syntax

```
ManagerOf (User)
```

Parameters

Type: User

A specified SharePoint user. Or the column and constant represents the User type.

Returns

Type: User

Returns the manager as a user.

Example

If user Henry's manager is Hans in active directory settings. Then ManagerOf(ToPeople("Henry")) will return user Hans.

AccountOf

Gets the name of a SharePoint user.

Syntax

```
AccountOf (User)
```

Parameters

Type: User

A specified SharePoint user. Or the column and constant represents the User type.

Returns

Type: Text

Returns the name as a string.

Example

AccountOf([Modified by]) will return the name of who modified the item, such as "Hans".

WorkEmailOf

Gets the work e-mail of a SharePoint user.

Syntax

```
WorkEmailOf (User)
```

Parameters

Type: User

A specified SharePoint user. Or the column and constant represents the User type.

Returns

Type: Text

Returns the work e-mail as a string.

Example

AccountOf(ToPeople("henry")) will return Henry's work e-mail, such as "henry@losting.com".

MobilePhoneOf

Gets the mobile phone number from a SharePoint user.

Syntax

```
MobilePhoneOf (User)
```

Parameters address

Type: User

A specified SharePoint user. Or the column and constant represents the User type.

Returns

Type: Text

Returns the mobile phone number as a string.

Example

MobilePhoneOf([Created by]) will return mobile phone number of who created the item.

SIPAddressOf

Gets the Session Initiation Protocol (SIP) address of a SharePoint user.

Syntax

```
SIPAddressOf (User)
```

Parameters

Type: User

A specified SharePoint user. Or the column and constant represents the User type.

Returns

Type: Text

A string that contains the SIP address of the user.

DepartmentOf

Gets the department information of a SharePoint user.

Syntax

```
DepartmentOf (User)
```

Parameters

Type: User

A specified SharePoint user. Or the column and constant represents the User type.

Returns

Type: Text

A string that contains department name.

Example

DepartmentOf(ToPeople("henry")) will return Henry's department name, such as "Marketing".

JobTitleOf

Gets the job title of a SharePoint user.

Syntax

```
JobTitleOf(User)
```

Parameters

Type: User

A specified SharePoint user. Or the column and constant represents the User type.

Returns

Type: Text

A string that contains the job title.

Example

JobTitleOf(ToPeople("henry")) will return Henry's job title, such as "Manager".

FirstNameOf

Gets the first name of a SharePoint user.

Syntax

```
FirstNameOf(User)
```

Parameters

Type: User

A specified SharePoint user. Or the column and constant represents the User type.

Returns

Type: Text

Returns the user's first name as a string.

Example

FirstNameOf([Created by]) will return the first name of who created the item.

LastNameOf

Gets the last name of a SharePoint user.

Syntax

```
LastNameOf(User)
```

Parameters

Type: User

A specified SharePoint user. Or the column and constant represents the User type.

Returns

Type: Text

Returns the user's last name as a string.

Example

FirstNameOf([Created by]) will return the last name of who created the item.

WorkPhoneOf

Gets the work phone number of a SharePoint user.

Syntax

```
WorkPhoneOf(User)
```

Parameters address

Type: User

A specified SharePoint user. Or the column and constant represents the User type.

Returns

Type: Text

Returns the work phone number as a string.

Example

WorkPhoneOf([Created by]) will return work phone number of who created the item.

UserIDOf

Gets the user ID of a SharePoint user.

Syntax

```
UserIDOf(User)
```

Parameters address

Type: User

A specified SharePoint user. Or the column and constant represents the User type.

Returns

Type: Text

Returns the user ID as a string.

Example

UserIDOf(ToPeople("Henry")) will return the ID of user Henry, such as "25".

UserNameOf

Gets the user name of a SharePoint user.

Syntax

```
UserNameOf(User)
```

Parameters address

Type: User

A specified SharePoint user. Or the column and constant represents the User type.

Returns

Type: Text

Returns the user name as a string.

Example

UserNameOf([Created by]) will return the user name of who created the item.

UserCreatedOf

Get the created time of a SharePoint user.

Syntax

```
UserCreatedOf(User)
```

Parameters address

Type: User

A specified SharePoint user. Or the column and constant represents the User type.

Returns

Type: DateTime

The date and time when the user is added on SharePoint.

UserModifedOf

Get the modified time of a SharePoint user.

Syntax

```
UserModifiedOf(User)
```

Parameters address

Type: User

A specified SharePoint user. Or the column and constant represents the User type.

Returns

Type: DateTime

The date and time when the user information was last modified.

Mathematic Functions

Abs

Returns the absolute value of a specified value.

Syntax

```
Abs (Integer n)
```

or

```
Abs (Decimal n)
```

Parameters

n

Type: Integer, Decimal

Value to be converted.

Returns

Type: Integer, Decimal

The absolute value.

Example

Abs(-3) would return the value 3.