

Examples of Math-Time expressions

On this page

- [Examples of Math and Time Expressions](#)
- [Math Calculus](#)
- [Date-Time Calculus](#)
- [Date-Time Calculus on Custom Schedules](#)
- [Showing Time Durations in Pretty Format](#)

Examples of Math and Time Expressions

This page presents a collection of expressions valid for the [Expression Parser](#).

Math Calculus

Expression	Returned Value	Notes
<code>max(count(subtasks(%{00041})) - 1, 0)</code> or since version 2.2.1 : <code>count(siblingSubtasks())</code>	For a sub-task, the number of sibling sub-tasks.	Function <code>max(x, y)</code> is used to avoid returning -1 when used with non-sub-task issues. %{00041} = Parent's issue key
<code>{10000} = null ? 1 : {10000} + 1</code> or since version 2.2.16 : <code>sum([{10000}]) + 1</code>	Formula to increment a numeric custom field, setting it to 1 if it's initially unset.	{10000} is the field code for a supposed numeric custom field.
<code>{10000} + {10001} + {10003}</code>	Formula for summing 3 numeric custom fields when we are certain that all 3 the fields are initialized . In case any of these fields is not initialized, an error is raised and any of the following 2 expression examples should be used.	{10000} , {10001} and {10003} are three numeric custom field.
<code>({10000} = null ? 0 : {10000}) + ({10001} = null ? 0 : {10001}) + ({10003} = null ? 0 : {10003})</code>	Formula for summing 3 numeric custom fields when some of them may be uninitialized . When any of this fields is not initialized a zero value is assumed.	{10000} , {10001} and {10003} are three numeric custom field.
<code>sum([{10000}, {10001}, {10003}])</code>	A more compact syntax for summing 3 numeric custom fields when some of them may be uninitialized . Version 2.2.16 or higher is required.	{10000} , {10001} and {10003} are three numeric custom field. This syntax is available since version 2.2.16 .

Date-Time Calculus

Expression	Returned Value	Notes
<code>{00012} - 6 * {DAY}</code>	Calculates a date 6 natural days earlier than Due Date	{00012} = Due Date
<code>addTimeSkippingWeekends({00009}, 36*{HOUR} + 45*{MINUTE}, LOCAL)</code>	Returns a date-time value equivalent to adding 36 hour and 45 minutes to date and time of issue creation , skipping the periods of time which correspond to weekend.	{00009} = Date and time of creation
<code>addTimeSkippingWeekends({00009}, 36*{HOUR} + 45*{MINUTE}, LOCAL, {FRIDAY}, {SATURDAY})</code>	Same as previous expression, but using Israeli weekend.	Israeli weekend is on Friday and Saturday.

<code>addDaysSkippingWeekends({00012}, -6, LOCAL)</code>	Calculates a date 6 work days earlier than Due Date for Jira Server's local timezone.	{00012} = Due Date Work days depend on timezone, since certain time moment maybe Sunday in certain timezones, and Monday in another ones.
<code>subtractDatesSkippingWeekends({00012}, {00057}, LOCAL)/{DAY}</code>	Returns the number of working days from Current Date and Time to Due Date , i.e., skipping weekends in Jira server's timezone.	{00012} = Due Date {00057} = Current day and time
<code>round(({00057} - {00009}) / {HOUR})</code>	Number of hours since issue creation	Function round() approximates the number of hours to the nearer integer. {00057} = Current day and time {00009} = Date and time of creation
<code>floor(({00012} - {00057}) / {DAY})</code>	Number of days to Due Date	Function floor() approximates the number of days by removing the fractional part. {00012} = Due Date {00057} = Current day and time
<code>datePart({00057}, LOCAL) + (dayOfTheWeek({00057}, LOCAL) = 7 ? 6 : 6 - dayOfTheWeek({00057}, LOCAL)) * {DAY}</code>	Returns a date value for next Friday , or for today if it's Friday	{00057} = Current day and time Example
<code>datePart({00057}, LOCAL) + (dayOfTheWeek({00057}, LOCAL) = 6 ? 7 : (dayOfTheWeek({00057}, LOCAL) = 7 ? 6 : 6 - dayOfTheWeek({00057}, LOCAL))) * {DAY}</code>	Returns a date value for next Friday , even if today is Friday.	{00057} = Current day and time Example

Date-Time Calculus on Custom Schedules

Custom Schedules are supported since version [2.2.39](#).

We use [Custom Schedules](#) when we need to do time calculations within the work-schedule of our company or organization, e.g., we want to count only the time from 8:00 to 15:00, and from 16:00 to 19:30.

Functionality provided by functions `addTimeSkippingWeekends()` and `subtractDatesSkippingWeekends()` can also be implemented using [Custom Schedules](#), and much much more.

Your Custom Schedules are defined in Jira at **Administration > Add-ons > JIRA WORKFLOW TOOLBOX > Schedules**.

Expression	Returned Value	Notes
<code>timeDifference({00012}, {00057}, "my_schedule", LOCAL)</code>	Returns the resting time to Due date within my_schedule custom schedule.	{00057} = Current day and time {00012} = Due date
<code>addTime({00057}, 24 * {HOUR}, "my_schedule", LOCAL)</code>	Returns a date-time value (i.e., an instant in time) obtained by summing 24 hours to current date-time within my_schedule custom schedule.	{00057} = Current day and time

Showing Time Durations in Pretty Format

The following examples are string expressions in **advanced parsing mode**.

Expression	Returned Value	Notes
<code>formatDuration({00057} - {00009})</code>	Calculates the time since issue creation, and shows it as a text using whole words like: 12 days 6 hours 34 minutes .	{00057} = Current day and time {00009} = Date and time of creation
<code>shortFormatDuration({00057} - {00009})</code>	Calculates the time since issue creation, and shows it as a text using abbreviations like: 12 d 6 h 34 m .	{00057} = Current day and time {00009} = Date and time of creation

formatDuration (subtractDatesSkippingWeekends({00057}, {00009}, LOCAL))	Calculates the time since issue creation skipping weekends, and shows it as a text like: 12 days 6 hours 34 minutes .	{00057} = Current day and time {00009} = Date and time of creation
formatWorkDuration ({00057} - {00009})	Calculates the time since issue creation, and shows it as text, but using the workday and workweek defined at time tracking configuration instead of 24 hours per day and 7 days per week.	Example: formatWorkDuration(24 * {HOUR} + 5 * {MINUTE}) returns " 3 days 5 minutes " when we use 8 hours per workday .
shortFormatWorkDuration ({00057} - {00009})	Similar to the previous expression but shows the result using abbreviations instead of whole words.	Example: shortFormatWorkDuration(24 * {HOUR} + 5 * {MINUTE}) returns " 3d 5m " when we use 8 hours per workday .