JWT Release Notes 2.2.7

[2016-02-01] **Released Jira Workflow Toolbox 2.2.7**

New Features

- Added 2 new functions to the parser:

FUNCTION	RETURNED VALUE
usersWhoTransitioned(*string* **origin_status**, *string* **destination_status**) : *string list*	returns a *string list* with the names of the users who transitioned current issue from **origin_status** to **destination_status**, order ascending by time. An empty string as argument is interpreted as *any status*. Example: `last(usersWhoTransitioned("Open", "In Progress"))` returns the name of the user who executed transition "*Start Progress*" more recently. Assign issue to last user who executed a certain transition in the workflow
usersWhoTransitioned(*string* **origin_status**, *string* **destination_status**, *string* **issue_key**) : *string list*	returns a *string list* with the names of the users who transitioned current issue from **origin_status** to **destination_status**, order ascending by time. An empty string as argument is interpreted as *any status*. Example: `count(usersWhoTransitioned("Open", "In Progress", %{00041}))` returns the number of times transition "*Start Progress*" has been executed in parent issue.
timesOfTransition(*string* **origin_status**, *string* **destination_status**) : *string list*	Returns a *number list* with the times when current issue was transitioned from **origin_status** to **destination_status**, order ascending by time. An empty string as argument is interpreted as *any status*. Example: `last(timesOfTransition("", "Resolved"))` returns the most recent time when the issue was resolved.
timesOfTransition(*string* **origin_status**, *string* **destination_status**, *string* **issue_key**) : *string list*	Returns a *number list* with the times when issue with key **issue_key** was transitioned from **origin_status** to **destination_status**, order ascending by time. An empty string as argument is interpreted as *any status*. Example: `first(usersWhoTransitioned("Closed", "", %{00041}))` returns the first time when parent issue was reopened.
addTimeSkippingWeekends(*number* **t**, *number* **timeToBeAdded**, **time_zone**) : *number*	Adds **timeToBeAdded** to **t** with the difference that weekends don't count in the sum, e.g., if **t** represents a date-time which coincides with a Saturday, adding *timeToBeAdded = 2 * {HOUR}* will return a date-time for next Monday at 02:00. Use negative values at **timeToBeAdded** for subtracting time from **t**.
addTimeSkippingWeekends(*number* **t**, *number* **timeToBeAdded**, **time_zone**, *number* **beginning_of_weekend**, *number* **end_of_weekend**) : *number*	Same as previous function, but with a custom defined weekend. Arguments **beginning_of_weekend** and **end_of_weekend** take values *{MONDAY}, {TUESDAY} ... {SUNDAY}*. Example of usage for adding 12 hours to *Current date and time* using Israeli weekend: `addTimeSkippingWeekends({00057}, 12 * {HOUR}, LOCAL, {FRIDAY}, {SATURDAY})`, being `{00057}` field code for *Current date and time*.
addDaysSkippingWeekends(*number* **t**, *number* **n**, **time_zone**, *number* **beginning_of_weekend**, *number* **end_of_weekend**) : *number*	Same as previous function, but with a custom defined weekend. Arguments **beginning_of_weekend** and **end_of_weekend** take values *{MONDAY}, {TUESDAY} ... {SUNDAY}*. Example of usage for adding 10 workdays to *Due date* using Israeli weekend: `addDaysSkippingWeekends({00012}, 10, LOCAL, {FRIDAY}, {SATURDAY})`, being `{00012}` field code for *Due date*.
subtractDatesSkippingWeekends(*number* **minuend_date**, *number* **subtrahend_date**, **time_zone**, *number* **beginning_of_weekend**, *number* **end_of_weekend**) : *number*	Same as previous function, but with a custom defined weekend. Arguments **beginning_of_weekend** and **end_of_weekend** take values *{MONDAY}, {TUESDAY} ... {SUNDAY}*. Example of usage calculating the worktime from *Creation* to *Resolution* using Israeli weekend: `subtractDatesSkippingWeekends({00112}, {00009}, LOCAL, {FRIDAY}, {SATURDAY})`, being `{00112}` field code for *Resolution date and time*, and `{00009}` field code for *Creation date and time*.
acos(*number* **x**) : *number*	Returns the arc cosine of **x**; the returned angle is in the range 0.0 through pi.
asin(*number* **x**) : *number*	Returns the arc sine of **x**; the returned angle is in the range 0.0 through pi.
atan(*number* **x**) : *number*	Returns the arc tangent of **x**; the returned angle is in the range 0.0 through pi.
cbrt(*number* **x**) : *number*	Returns the cube root of **x**.
cos(*number* **x**) : *number*	Returns the trigonometric cosine of angle **x** expressed in radians.
cosh(*number* **x**) : *number*	Returns the hyperbolic cosine of **x**.
log(*number* **x**) : *number*	Returns the natural logarithm (base e) of **x**.
log10(*number* **x**) : *number*	Returns the base 10 logarithm of **x**.
modulus(*number* **dividend**, *number* **divisor**) : *number*	Returns `dividend - (divisor * floor(dividend / divisor))`.
sin(*number* **x**) : *number*	Returns the trigonometric sine of angle **x** expressed in radians.
sinh(*number* **x**) : *number*	Returns the hyperbolic sine of **x**.
tan(*number* **x**) : *number*	Returns the trigonometric tangent of angle **x** expressed in radians.
tanh(*number* **x**) : *number*	Returns the hyperbolic tangent of **x**.
toDegrees(*number* **x**) : *number*	Converts an angle **x** measured in radians to an approximately equivalent angle measured in degrees.
toRadians(*number* **x**) : *number*	Converts an angle **x** measured in degrees to an approximately equivalent angle measured in radians.
filledInTransitionScreen(%{nnnnn}) : *boolean*	Returns *true* only if selected field has an actual value in current transition's screen. Example: `filledInTransitionScreen**(%{00003})` returns *true* only if the field *Assignee* was present in the transition screen and contained a value at the moment of submitting the form.

- Issue #236
- Issue #220
- Issue #195
- Issue #223

Improvements

- Minor UI improvements.

Bug Fixes

- Virtual field **New labels** failed when it was written with a string containing blank space separated labels. It tried to create a single label containing spaces, which is forbidden in Jira.
- Virtual field **Date and time of resolution** was not returning a value under certain circumstances.
- **Issue #231** - Error in description of function **indexOf**()