Understanding Check Expressions in Deskpro

Kim - 2025-09-09 - 댓글 (0) - Using Deskpro

What are Check Expressions?

Check expressions are a type of condition you can add to triggers in Deskpro. They use expression language to evaluate ticket data and the event context, which allows you to create rules that go beyond what the standard trigger criteria can do.

For example, check expressions allow you to:

- **Reference objects on the ticket:** e.g.

  `ticket.getLastReply().getPerson().is_agent == false` checks the Reply object on the ticket, looks at the Person object who sent it, and confirms whether that person is not an agent (meaning the reply came from a user).

- **Compare values across multiple fields:** e.g.

  `ticket.getCustomDataForField(248).value > ticket.getCustomDataForField(249).value` to see if one custom field is greater than another. (Only works on date and number fields)

- **Use functions to retrieve details about the person:** e.g.

  `ticket.agent.id === context.getPersonContext().id` to check if the agent assigned to the ticket is the one who performed the action.

- **Create conditions based on time or event context:** e.g.

  `ticket.getLastReply().getDateCreated().getTimestamp() > 1703376000 and ticket.getLastReply().getDateCreated().getTimestamp() < 1704153600` to only run a trigger within a certain date. (Such as Christmas, in this example)

They follow Symfony's expression language syntax.

Why Use Check Expressions Instead of Regular Criteria?

Regular trigger criteria are simple and user-friendly. They cover common scenarios, such as checking if a ticket belongs to a department or if its status has changed.

Check expressions come into play when:

- You need **comparisons across fields**, not just a single field.

- You want to run triggers **within a date or time range**.

- You need to reference the **event performer** or **context of the action**.
- You need more flexibility than the standard drop-down criteria provide.

In short: standard trigger criteria handle most situations, while check expressions let you build more precise or complex rules.

Scope and Limitations

- Check expressions can only reference **objects directly linked to the ticket** (such as the ticket itself, replies, or the agent/user context).
- They cannot query data outside the ticket's scope (for example, you can't directly pull unrelated organisation details unless tied to the ticket).
- You build check expressions using the ticket-specific context, such as the ticket's department, the ID of the person who last replied, or functions to look up and compare information related to the ticket.

Examples of Check Expressions

Expression / Method	Purpose	Example	What it does
`ticket.getCustomDataForField(ID).value`	Accesses the value of a custom field by its numeric ID.	`ticket.getCustomDataForField(248).value > ticket.getCustomDataForField(249).value`	Runs only if the value of field 248 is greater than the value of field 249. Works only with number or date fields.
`ticket.getLastReply().getDateCreated().getTimestamp()`	Returns the Unix timestamp of the last reply.	`ticket.getLastReply().getDateCreated().getTimestamp() > 1703376000 and ticket.getLastReply().getDateCreated().getTimestamp() < 1704153600`	Ensures the trigger fires only if the last reply was created between two dates (e.g. during Christmas).
`ticket.agent.id` with `context.getPersonContext().id`	Compares the assigned agent with the person who performed the event.	`ticket.agent.id === context.getPersonContext().id`	Runs only if the agent assigned to the ticket is the same person who performed the action.
`context.getEmailContext()`	Fetches the email context (only valid if `hasEmailContext()` is true).	`hasEmailContext() and context.getEmailContext()`	Runs only when there is an email context available, such as when the event is triggered by an incoming email.
`context.getEventType()`	Returns the type of event that triggered the expression.	`context.getEventType() == "ticket_created"`	Runs only when the event was a ticket being created.
`context.getEventPerformer()`	Returns the person object for whoever performed the event.	`context.getEventPerformer().id == ticket.agent.id`	Runs only if the agent assigned to the ticket is the same person who performed the event.

`context.getEventMethod()`	Returns how the event happened (e.g. web, email, API).	`context.getEventMethod() == "email"`	Runs only if the ticket event was performed via email.
`ticket.getCustomDataForField(ID).getData()`	Gets the stored value for a custom field, including null/empty checks.	`(ticket.getCustomDataForField(416) !== null && ticket.getCustomDataForField(416).getData() !== "") && (ticket.getCustomDataForField(415) !== null && ticket.getCustomDataForField(415).getData() !== "") && (ticket.getCustomDataForField(416).getData() === ticket.getCustomDataForField(415).getData())`	Runs only if both custom fields 416 and 415 are filled in and their values are identical.

Context Functions

These helper functions let you reference different aspects of the ticket event:

- `context.getPersonContext()` → returns the person linked to the action (e.g. the agent or user).
- `context.getEmailContext()` → used with `hasEmailContext`, gives details about the email involved.
- `context.getEventType()` → checks the type of event that triggered the rule.
- `context.getEventPerformer()` → identifies who performed the event.
- `context.getEventMethod()` → details the method of the event.

Summary

Check expressions are a powerful way to extend triggers when standard criteria aren’t enough. They let you:

- Compare field values.
- Restrict triggers by time.
- Work with event and person context.