Triggers Operation

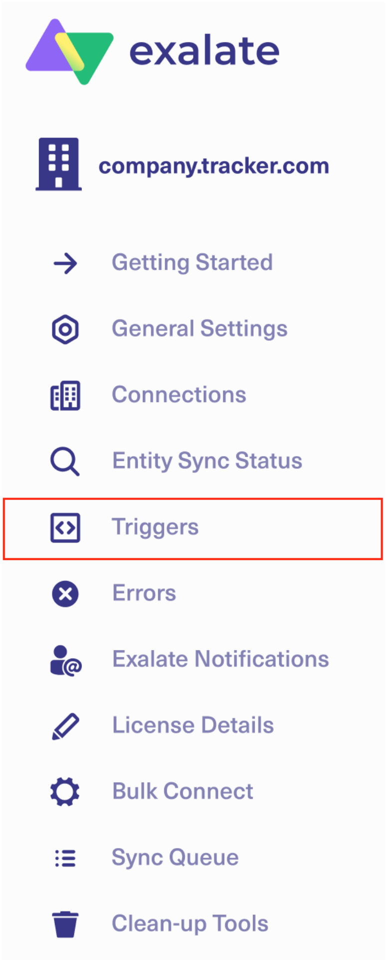
Last Modified on 02/17/2026 12:01 pm EST

Triggers in Exalate allow you to start synchronization automatically.

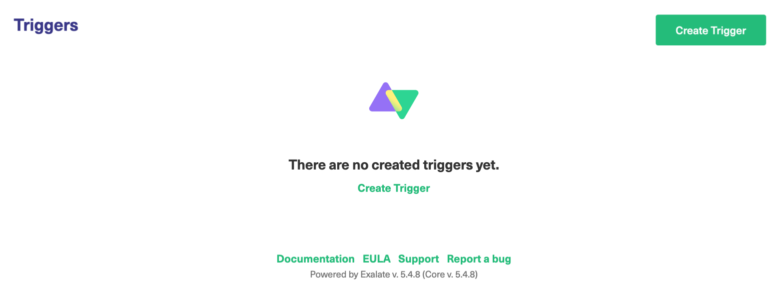
How to Create a Trigger

To create a trigger, please follow the steps:

1. Navigate to the **Triggers** tab in the Exalate admin menu



2. Click **Create Trigger**

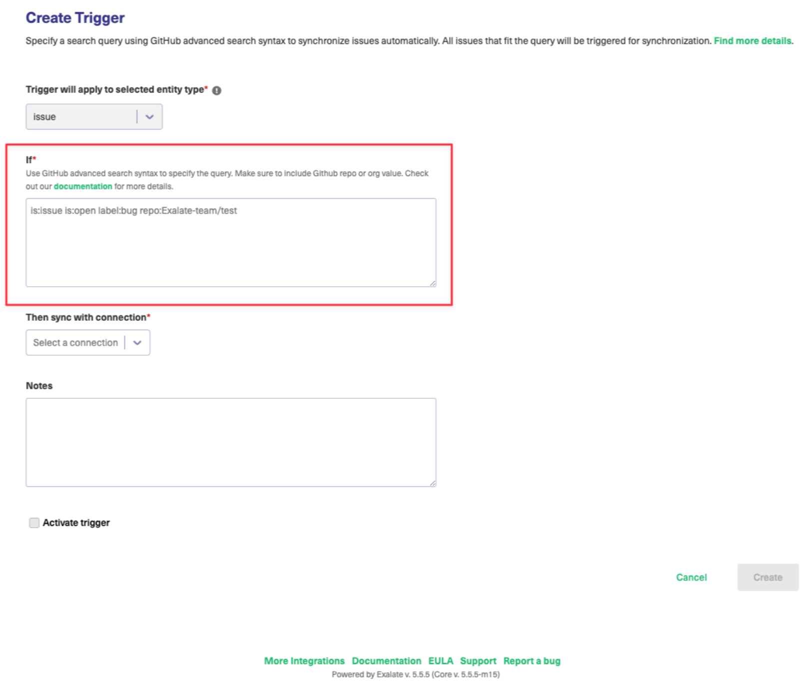


3. Select the type of entity you want to synchronize. If there is only one option available, the dropdown feature is disabled.

> **Note**: If you want to synchronize multiple entities, you need to create separate triggers for every entity type.

4. In the **If** field, enter the desired filter to decide which issues get automatically synced to the other side.

Use the platform search syntax to specify the filter query.



5. Select the Connection you want to sync with, **Then sync with the connection** list.

6. Optionally, input information to keep a reference for the trigger in the **Notes** field.

7. Check the **Activate trigger** box to activate the trigger.

8. Click **Create** to complete the process.

Note:

- To start the synchronization, you need to **Bulk Exalate** on the trigger. It helps to synchronize all issues that fit trigger search queries. For more information about this topic, please read how to bulk Exalate entities.
- To stop the synchronization, you need to **Bulk Unexalate** the trigger. It helps to stop the synchronization of all issues that fit trigger search queries. For more information

about this topic, please read how to bulk unexalate entities.

What if the connection used for a trigger has been disabled?

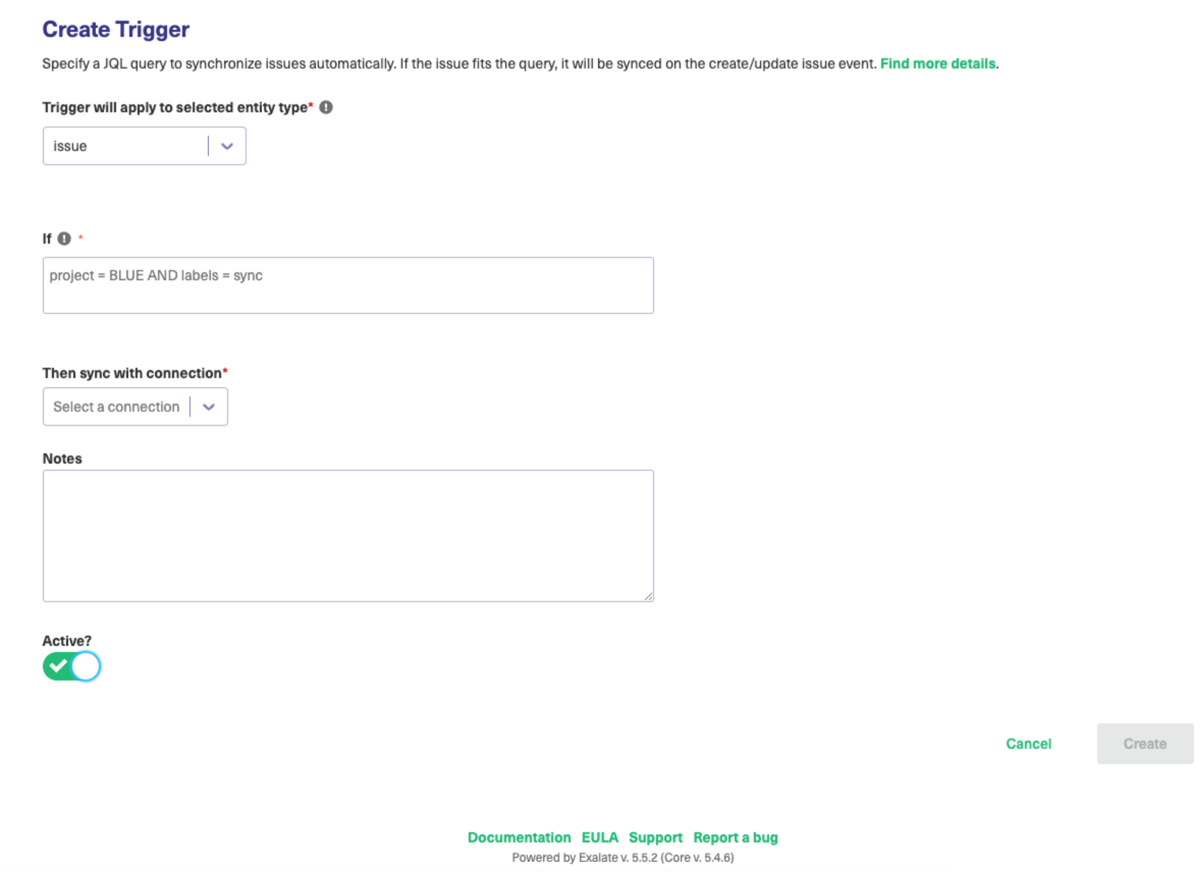
Whenever a connection used in a particular trigger is disabled, Exalate still collects all the changes that are related to issues that are under sync using that connection.

The changes are queued as sync requests. Once the connection is reenabled, sync transactions can continue from the point where they have been paused.

What if the trigger has been disabled?

Disabling the trigger makes Exalate ignore the create/update events for the issue that matches the query.

Triggers in Jira



You can start issue synchronization automatically according to a specified search query with the help of a trigger.

If the issue fits a search query it is triggered for synchronization automatically. Once someone updates the issue that is under sync, the changes are synchronized automatically.

Check the example of the Jira Query Language (JQL) trigger query below.

When you create an issue in the **project = BLUE** with **labels = sync** issue is synchronized automatically.

Also, the same trigger starts the synchronization if any issue in **project BLUE** has been updated with the **label sync.**

```
project = BLUE AND labels = sync
```

Using saved filters:

- You can use JQL saved filters in trigger configuration. Please make sure that the filter is shared with the proxy user.
- Filters can be shared with other users via user groups, projects, and project roles. They can also be shared globally. Any filter that is shared is visible to users who have the **Administer Jira** global permission.
- Use filter ID (***filter = 10500***) or name (***filter = "My filter"***) to configure a trigger

Note: **Changes due to compliance with GDPR**

JQL queries containing user identifiers such as username, display name, or email stop working starting from the 29th of March due to Atlassian's compliance with GDPR.

Update your JQL query to use account IDs instead. Find more details in the REST API migration guide and deprecation notice - userName and userKey. You can get the user account IDs with the help of JIRA Cloud REST API.

Examples:

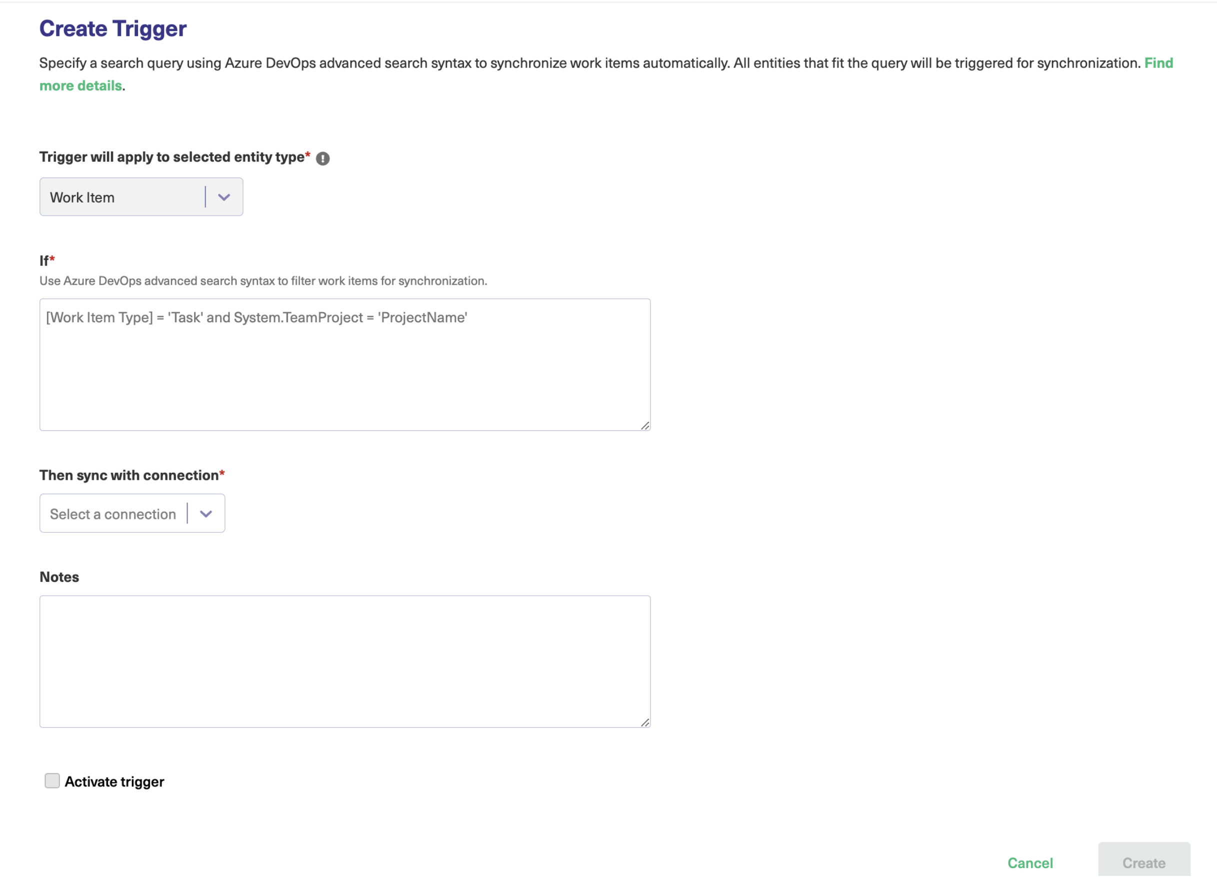
JQL with userNames

`issuetype = Bug AND assignee in (mia) AND reporter in (alana) order by lastViewed DESC`

JQL with account IDs

`issuetype = Bug AND assignee in (abcde-` `12345` `) AND reporter in (abc551-c4e99) order by lastViewed DESC`

Triggers in Azure DevOps (Server & Service)



Triggers use a WIQL (Work Item Query Language) or the Azure DevOps Search Syntax to search Azure DevOps work items.

Use the Work Item Query Language Syntax to create a search query.

You only need the conditions found in the WHERE clause of the WIQL to filter the work items. Check some examples below:

Synchronize all work items with type 'Task' from a project 'ProjectName'

```
[Work Item Type] = 'Task' AND System.TeamProject = 'ProjectName'
```

Important: Starting from Exalate for Azure DevOps version 5.4.0, make sure to indicate the project name as `System.TeamProject = 'ProjectName'`

Sync work items from project 'BatCave' with type 'Task', assignee 'Bruce Wayne', and status 'Doing'

```
[Work Item Type] = 'Task' AND
[Team Project] = 'BatCave' AND
[System.AssignedTo] = 'Bruce Wayne' AND
State = 'Doing' AND
[Area Path] = 'BatCave\Operation' AND
[System.Tags] Contains 'azure'
```

Sync all work items that have a custom field with the title 'MyCustomTextField' and value 'Sonic'

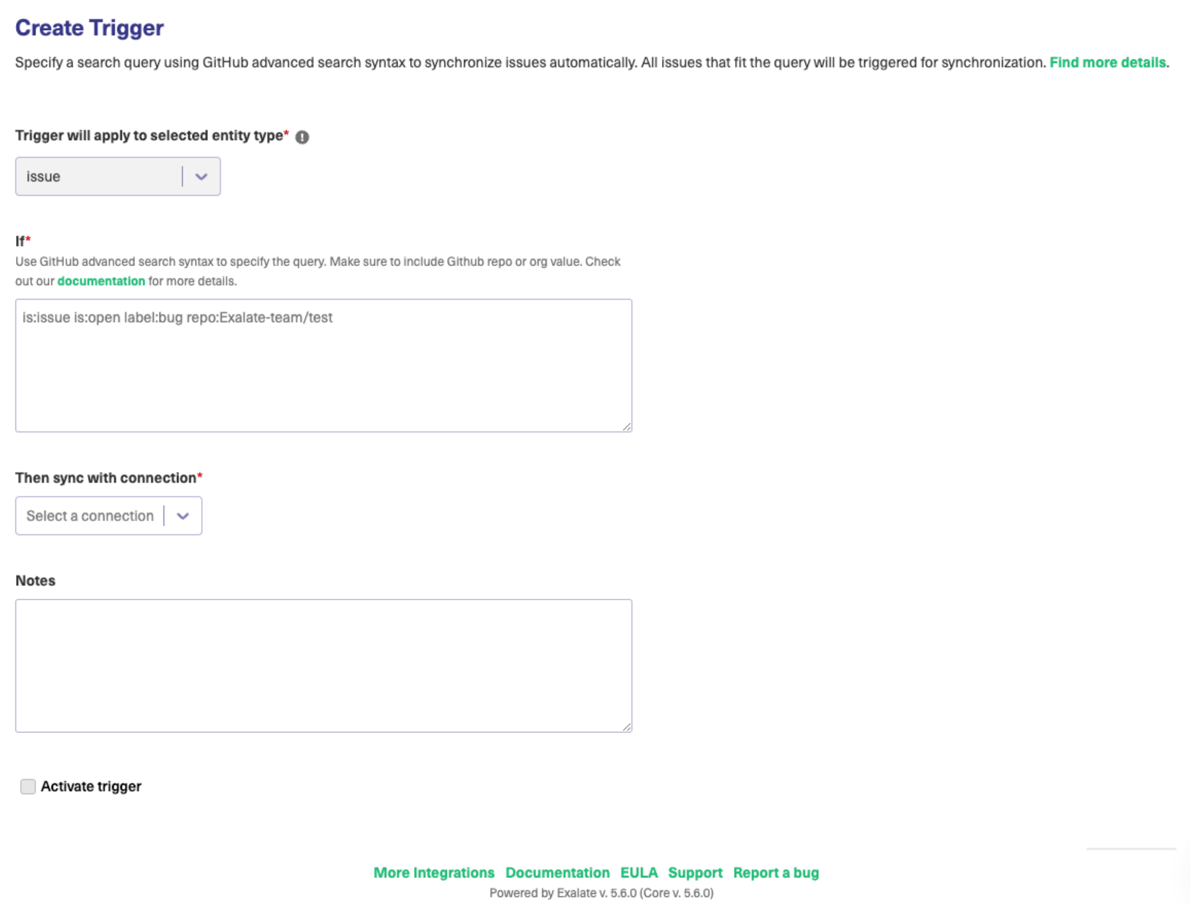
`[Custom.MyCustomTextField] = ‘Sonic’`

Sync all work items created 2 days ago

`[System.CreatedDate]` `=` `@` `today-` `2`

> **Note:** You can specify either the reference name or the friendly name. The following examples are valid WIQL syntax:
>
> - Reference name with spaces: `[System.AssignedTo]`
> - Friendly name with spaces: `[Assigned To]`
>
> Names without spaces don't require square brackets: `ID, Title, State`

Triggers in GitHub



Use the GitHub advanced search syntax to specify the query.

Warning: In order for the trigger to be saved, you MUST specify at least one repository or organization in the search query. If you do not do this, it is still possible to save the trigger, but it doesn't work. This is a known issue and we are working to address it so a warning is displayed. Here are some examples of the queries to add:

```
is:issue org:Organization
```

```
is:issue is:open label:bug repo:username/test
```

```
is:pr is:open repo:username/test
```

Below you can see some more examples to specify in the query:

Synchronize issues in status Open with label bug from the repo test

`is:open label:bug repo:username/test`

Synchronize pull requests in status Open

`is:pr is:open`

Note: If the search query contains whitespace, you need to surround it with quotation marks. Check the example below:

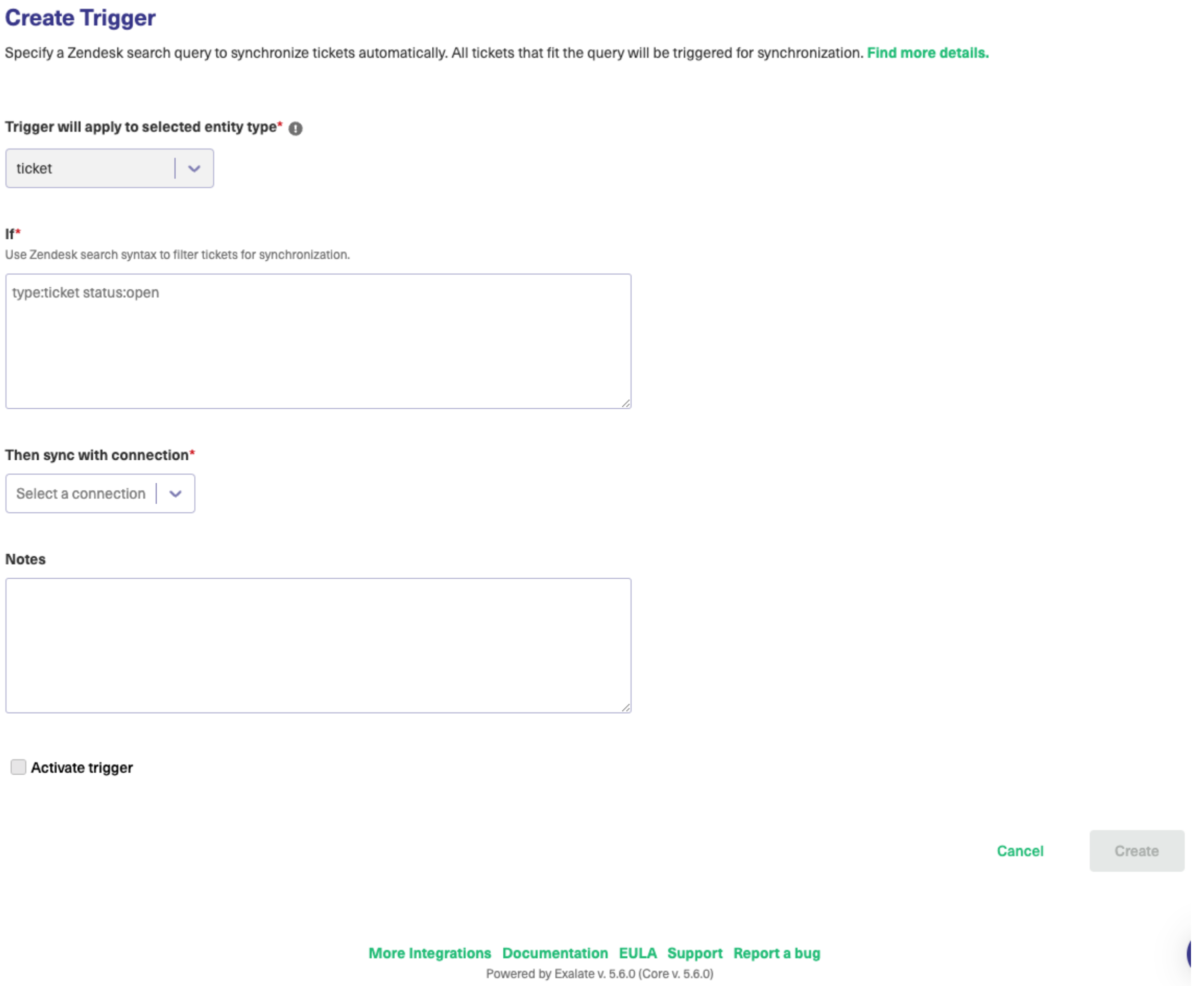
Synchronize issues with a label bug fix

```
label:"bug fix"
```

Synchronize pull requests in status in progress

```
is: "in progress"
```

Triggers in Zendesk



Use the Zendesk search syntax to specify the filter query. Below you can see an example

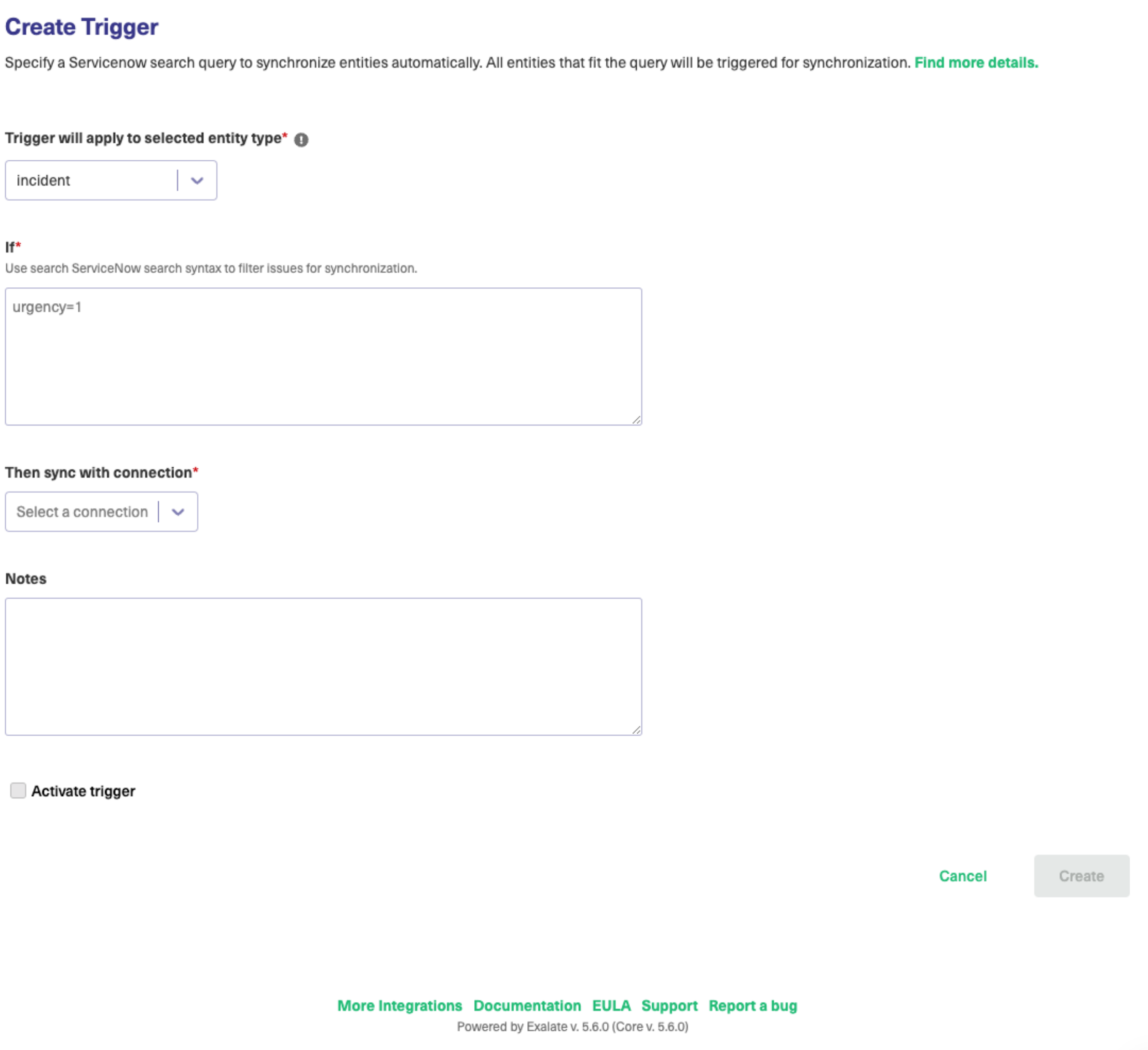
Zendesk search: sync issues of type ticket with status open

`type:ticket status:open`

Sync tickets with ticket type INCIDENT and priority Urgent

`type:ticket ticket_type:incident priority:urgent`

Triggers in ServiceNow



Use the ServiceNow search syntax to specify the filter query. Below you can see an example

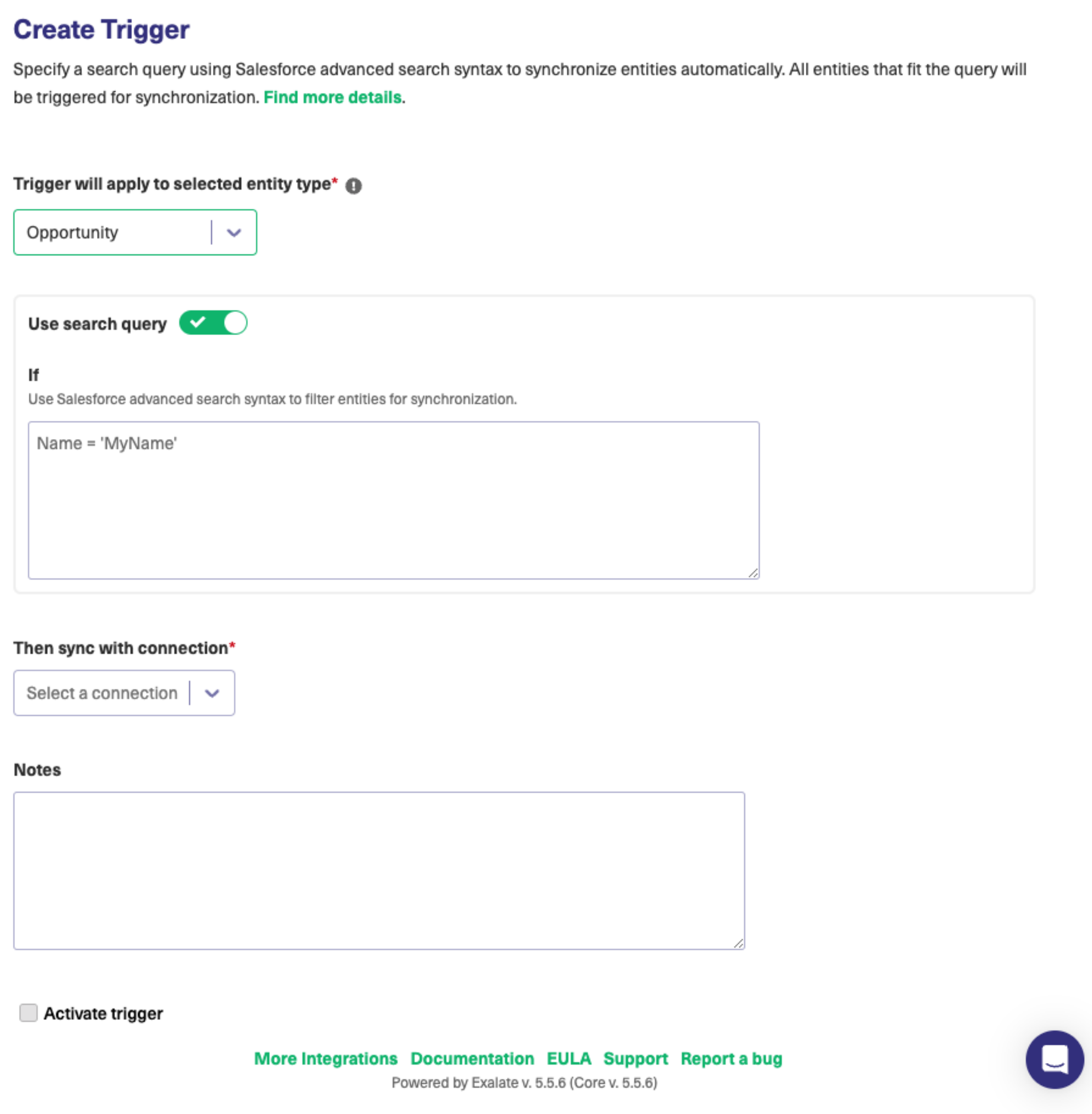
Servicenow search: sync incidents with urgency 1-High

```
urgency=1
```

Sync incidents that include "Test" in the description

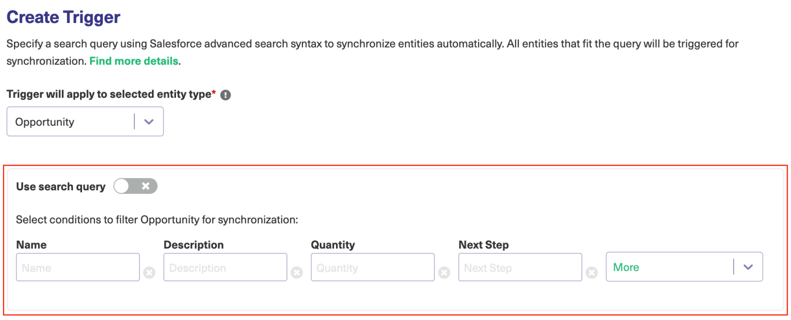
```
short_descriptionLIKETest
```

Triggers in Salesforce

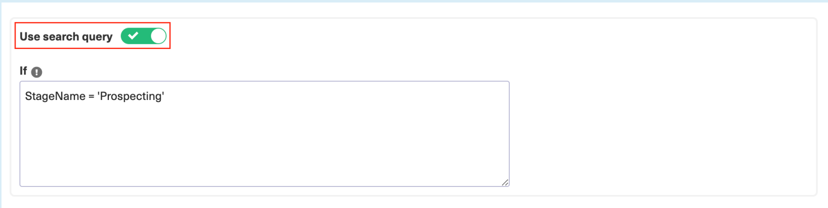


Warning: Currently, it is not possible to create a trigger using a query with text area fields. Text area fields contain text that can be longer than 4000 bytes. Unlike string fields, text area fields cannot be specified in the WHERE clause of a queryString of a query() call. For more information on the limitation, please read Field Types.

In Salesforce, you can select conditions to filter an object for synchronization. Depending on the entity type selected, you see different input fields.



Note: Alternatively, you can enable **search queries**. Use Salesforce Object Query Language to specify the filter query.



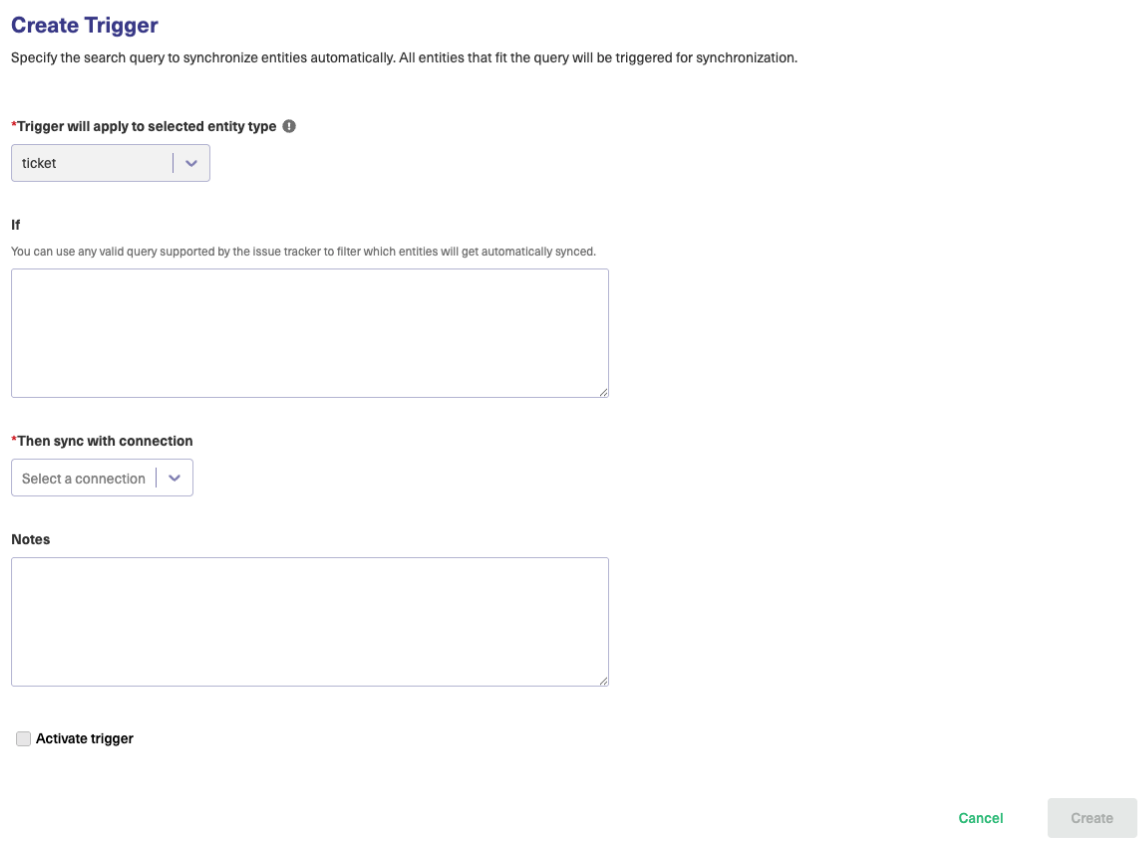
Please, read the following search query examples for Opportunity:

```
StageName= 'Prospecting' AND Name like '%demo%'
```

This search query defines the Opportunity stage as 'Prospecting' and an Opportunity name that partially matches the word 'demo'

This search query defines an Opportunity's exact name as 'demo opportunity' and its order number - 123

Triggers in Freshdesk



You can configure a trigger by specifying field values, which are separated by “AND” or commas, to filter the relevant tickets

```
filter = new_and_my_open AND requester_id = 103099112292
```

OR

```
filter = new_and_my_open, requester_id = 103099112292
```

this query filters for tickets that are new, open and assigned to the user making the query, and have a requester with the ID 103099112292.

Filters

You can use all the fields supported by the List All Tickets endpoint, e.g.:

- **new_and_my_open**: Syncs tickets that are new and assigned to the user.
- **watching**: Syncs tickets that the user is currently watching.
- **spam**: Syncs tickets marked as spam.
- **deleted**: Syncs tickets that have been deleted.

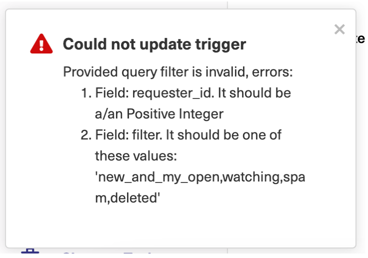
Notes

- The Freshdesk API only allows one filter value at a time. For example, you cannot combine `new_and_my_open` and `watching` in a single query.

- In the Freshdesk connector the triggers are executed by a scheduler rather than in real-time. As a result, synchronization actions may experience slight delays of approximately 30 to 40 seconds.

Error Validation

If a trigger query is not valid, an error pop-up will appear, listing the errors in the query:



Freshdesk REST API:

For further reference, you can consult the Freshdesk REST API documentation to find a complete list of supported fields and their usage in trigger queries.

[Freshdesk REST API Documentation]

Triggers in Freshservice

You can configure synchronization in FreshService using field-based filter queries with logical operators.

Syntax

When adding a trigger for FreshService, you can set the fields and values separated by logical expressions like `AND` or `OR`.

Example Filter Query:

```
priority: 1 AND status: 2 OR urgency: 3
```

Alternatively, you can use a simpler syntax with predefined filters and field values separated by `AND` or commas:

```
filter = new_and_my_open AND requester_id = 103099112292
filter = new_and_my_open, requester_id = 1070031243523
```

Supported Ticket Fields

Field	Type	Description

`workspace_id`	number	Workspace ID of the ticket
`requester_id`	number	User ID of the requester
`email`	string	Email address of the requester
`agent_id`	number	ID of the agent to whom the ticket is assigned
`group_id`	number	ID of the assigned group
`priority`	number	Priority of the ticket
`status`	number	Status of the ticket
`impact`	number	Ticket impact
`urgency`	number	Ticket urgency
`tag`	string	Associated tag
`due_by`	date	Due date (YYYY-MM-DD)
`fr_due_by`	date	First response due date
`created_at`	date	Ticket creation date (YYYY-MM-DD)

Custom Fields

Field Type	**Syntax Example**
Number	`custom_field_name: 10`
Checkbox	`custom_checkbox: true`
Dropdown	`custom_dropdown: "value"`
Date	`custom_date: "2025-06-24"`

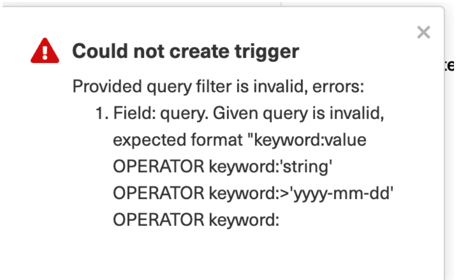
Note: Use snake_case for custom field names.

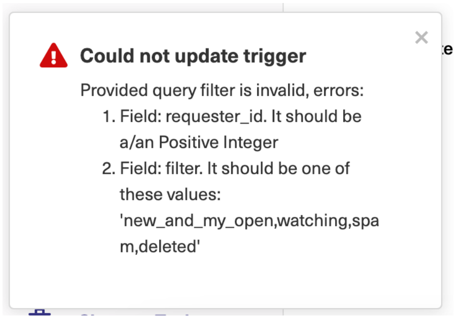
Error Validation

If your trigger query contains errors, a popup will appear listing the syntax issues. For example:

- Invalid field name
- Incorrect data type
- Misused logical operator

Screenshots of such errors help in debugging, e.g.,





How Filter Queries Differ from List Queries

Feature	**Filter Query**	**List Query**
Syntax Operator	Uses `:` (colon) for equality	Uses `=` (equals) for equality
Query Structure	Entire query in one string	Built using multiple parameters
Date Filtering	Uses `created_at:>` or `:<`	Uses `updated_since`
Logical Operators	Supports `AND`, `OR`, and `()`	Not explicitly supported
Custom Field Support	Fully supported via snake_case	Limited or unsupported

FreshService REST API Documentation

You can find the full list of supported fields and more in the FreshService REST API documentation.

Triggers in Asana

This article describes how to configure triggers in Asana to control which tasks and projects are synchronized.

Triggers in Asana: Triggers use field-based queries to filter tasks and projects. You can use project names, user emails, tags, and status types without needing Asana GIDs.

Task triggers

Filter tasks by project, assignee, tags, or completion status:

```
projects="Development"
tags="urgent"
completed=false
assignee="john.doe@example.com" AND completed=false
projects="Support" AND tags="external"
```

Project triggers

Filter projects by status, owner, or state:

```
status="at_risk"
```

Status Types: Supported project status types include at_risk, on_track, off_track, on_hold, complete, achieved, partial, missed, and dropped.

Triggers in Xurrent

This article describes how to configure triggers in Xurrent to control which requests are synchronized.

Triggers in Xurrent: Triggers use field-based queries to filter requests. In this article we will see which fields, operators we can use in triggers.

Request triggers

Filter requests by assignee, team, template, service instance, status, team or many more:

Example: High Priority Incidents Assigned to My Team

```
category=incident&impact=high,top&team=42&status=!completed
```

More details for triggers conditions and syntax: How to configure triggers in Xurrent

ON THIS PAGE

How to Create a Trigger

Triggers in Jira

Triggers in Azure DevOps (Server & Service)

Triggers in GitHub

Triggers in Zendesk

Triggers in ServiceNow

Triggers in Salesforce

Triggers in Freshdesk

Triggers in Asana

Task triggers

Project triggers

Triggers in Xurrent