
TA-jira-service-desk-simple-addon Documentation

Release 1

Guilhem Marchand

Aug 01, 2021

Contents

1	Overview:	7
1.1	About	7
1.2	Compatibility	7
1.3	Support & donate	8
1.4	Download	8
2	Deployment and configuration:	9
2.1	Deployment & Upgrades	9
2.2	Configuration	10
3	User guide:	31
3.1	User guide	31
4	Troubleshoot:	69
4.1	Trouble shooting	69
5	Versions and build history:	73
5.1	Release notes	73

The Splunk Add-on for JIRA Atlassian Service Desk provides alerts action for JIRA issues creation:

- Trigger JIRA issue creation from Splunk core alerts and Enterprise Security correlation searches
- Dynamic retrieval per JIRA project for types of issues and priority
- Dynamic assignment of priority (optional)
- Dynamic and/or static assignment of summary, description, assignee and labels
- Custom fields full capabilities via the embedded custom field structure in alerts (optional)
- Deduplication feature workflow with bi-directional integration, allows detecting a duplication issue creation request, and adding new comments automatically instead of creating duplicated issues
- Attaching Splunk alert results to the JIRA issue in CSV or JSON format
- Resilient store JIRA issue creation, shall a JIRA issue fails to be created, the resilient workflow handles automatic retries with a resilient policy
- Monitoring of JIRA issue workflow via the embedded Overview dashboard and out of the box alerts
- Get any information from JIRA via the REST API custom command wrapper, generate and index to summary events or the metric store issues statistics per projects

splunk>cloud

App: JIRA Service Desk simple addon

Messages

Settings

Activity

Find

Overview - JIRA Service Desk

Search

Logging reports

Builtin alerts

Alerts

Configuration

Run a search

Overview - JIRA Service Desk

Last 24 hours

Hide Filters

56

JIRA SUCCESSFULLY CREATED

1

JIRA TEMPORARY FAILURES

0

JIRA CURRENTLY IN THE REPLAY KVSTORE

JIRA issue creation workflow:

- When a JIRA issue creation is requested, the modular alert attempts a rest call to JIRA and logs its activity in (index="_internal" OR index="cim_modactions") (source="jira_service_desk_m
- If the JIRA issue creation is successful, the keyword "JIRA Service Desk ticket successfully created" and the issue reference are returned and logged
- Should the JIRA creation fail for any reason, the keyword "JIRA Service Desk ticket creation has failed" is logged, and the issue data is stored automatically in the replay KVstore (l inputlo
- This is a temporary failure as the replay backend handles automatically failed issues stored in the KVstore, and attempts again the creation via the scheduled alert "JIRA Service Desk - Res
- The replay issue backend logs its activity in (index="_internal" OR index="cim_modactions") (source="jira_service_desk_replay_modalert.log")
- Tickets stored in the replay KVstore are attempted when the replay alert triggers (every 5 minutes), a temporary failed ticket will be attempted during a period of 3 days
- Once the ticket referenced by a uuid has reached the 3 days period, it is tagged as a permanent failure, and the alert "JIRA Service Desk - detection of permanent issue creation failure" k
- A ticket in a permanent failure state will not be attempted anymore, 7 days after its initial creation, the ticket is finally tagged for removal and will be purged automatically from the replay k
- As such, a JIRA issue that initially failed to be created is automatically retried during 3 days, and definitively purged after 7 days

status

4

2

0

09:00 Tue Apr 7 2020

10:00

11:00

12:00

13:00

14:00

15:00

16:00

17:00

18:00

19:00

20:00

21:00

22:00

23:00

00:00 Wed Apr 8

Status:

ANY

First call activity

Resilient store activity

_time

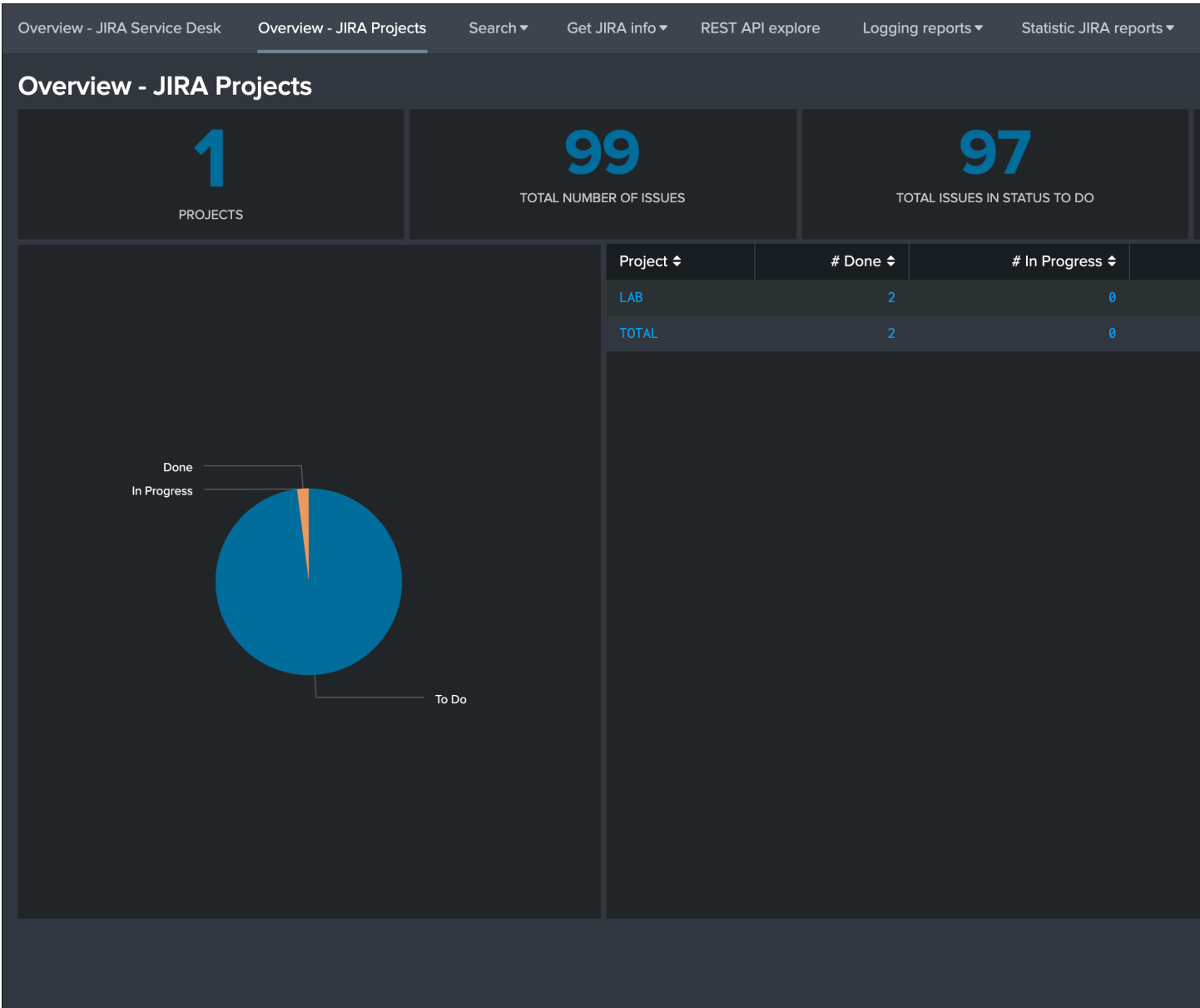
status

_raw

2020-04-08 08:03:19.149

✓

2020-04-08 07:03:19,149 INFO pid=70543 tid=MainThread file=cim_actions.py:message:424 | sendmodaction - worker="sh-i-0a589c8bdd2cb06 action_name="jira_service_desk_replay" search_name="JIRA Service Desk - Resilient store Tracker" sid="scheduler__admin_VEEtamlyYS1zZ simple-addon" user="admin" digest_mode="0" action_mode="saved" action_status="success"



Overview - JIRA Service Desk

Overview - JIRA Projects

Search ▾

Get JIRA info ▾

REST API explore

Logging reports ▾

Statistic JIRA reports ▾

Builtin alerts ▾

REST API explore

JIRA Rest API

Use the custom command `| jirarest target=<endpoint>*` to perform an HTTP rest call against any endpoint of your JIRA instance.

The default method is GET, but you can as well perform PUT, POST and DELETE calls using the `method` argument and the `json_request` if the API endpoint requires data

For API references:

[Jira Server platform REST API reference](#)

Add-on documentation page:

[REST wrapper documentation on Read the docs](#)

Try your own:

rest/api/2/myself

i	Time	Event
>	26/06/2021 10:18:33.768	<pre>{ [-] active: true applicationRoles: { [+] } avatarUrls: { [+] } deleted: false displayName: guilhem@octamis.com emailAddress: guilhem@octamis.com expand: groups,applicationRoles groups: { [+] } key: JIRAUSER10000 locale: en_US name: gmarchand self: https://localhost:8081/rest/api/2/user?username=gmarchand timeZone: Europe/London }</pre> Show as raw text

Use REST and JQL to get the total number of issues per project, per status category and calculate percentages in each status (dy

You can use the JQL language and perform any advanced query in JIRA, the following example returns the number of issues per project: `api/2/search?jql=project=<my`

_time ▾	project ▾	pct_total_done ▾	pct_total_in_progress ▾	pct_total_to_do ▾
2021-06-26 10:18:32	LAB	% 2.02	% 0.00	% 97.98

Edit Alert

When triggered



JIRA Service Desk

JIRA main fields:

Project *
required

TEST - Test



Issue Type *
required

Incident



Priority

High



Dynamic
Priority

`$result.priority$`



(Optional) Override priority using a field result, ex `$result.jira_priority$`. (case sensitive, ticket creation will fail if incorrectly defined)

Summary *
required

Test JIRA public addon

Description

`*+Alert Details+* *Description:*`

The alert condition for
'`$name$`' was triggered

Description:

(Required) Issue description, this text can include tokens based on the search results (E.g: `$result.src$`)

Assignee

(Optional) Issue assignee

Cancel

1.1 About

- Author: Guilhem Marchand, Splunk certified consultant and part of Splunk Professional Services
- First public release published in April 2020
- License: Apache License 2.0

1.2 Compatibility

1.2.1 Splunk compatibility

This application is compatible with Splunk 7.0.x and later.

1.2.2 Splunk Enterprise Security compatibility

This application has been verified with ES 5.x and 6.x.

1.2.3 Web Browser compatibility

The application can be used with any of the supported Web Browser by Splunk:

<https://docs.splunk.com/Documentation/Splunk/latest/Installation/Systemrequirements>

1.2.4 JIRA compatibility

The Add-on is compatible with:

- JIRA Server

- JIRA Cloud
- JIRA Data center

1.3 Support & donate

I am supporting my applications for free, for the good of everyone and on my own private time. As you can guess, this is a huge amount of time and efforts.

If you enjoy it, and want to support and encourage me, buy me a coffee (or a Pizza) and you will make me very happy!

This application is community supported.

1.3.1 Splunk Answers

Open a question in Splunk answers for the application:

- <https://answers.splunk.com/app/questions/4958.html>

1.3.2 Splunk community slack

Contact me on Splunk community slack, and even better, ask the community!

- <https://splunk-usergroups.slack.com>

1.3.3 Open a issue in Git

To report an issue, request a feature change or improvement, please open an issue in Github:

- <https://github.com/guilhemmarchand/TA-jira-service-desk-simple-addon/issues>

1.3.4 Email support

- guilhem.marchand@gmail.com

However, previous options are far better, and will give you all the chances to get a quick support from the community of fellow Splunkers.

1.4 Download

The Splunk application can be downloaded from:

1.4.1 Splunk base

- <https://splunkbase.splunk.com/app/4958>

1.4.2 GitHub

- <https://github.com/guilhemmarchand/TA-jira-service-desk-simple-addon>

Deployment and configuration:

2.1 Deployment & Upgrades

2.1.1 Deployment matrix

Splunk roles	required
Search head	yes
Indexer tiers	no

If Splunk search heads are running in Search Head Cluster (SHC), the Splunk application must be deployed by the SHC deployer.

2.1.2 Dependencies

There are currently no dependencies for the application.

However, if you deploy the `Splunk_SA_CIM` package, make sure you have declared the `cim_modactions` index as the Add-on logs would automatically be directed to this index if the SA CIM application is installed on the search heads.

If the `Splunk_SA_CIM` is not installed, the Add-on logs will be generated in the `_internal` index. (This is a normal behaviour for Add-on developed with the Splunk Add-on builder that provide adaptive response capabilities)

2.1.3 Initial deployment

The deployment of the Splunk application is very straight forward:

- Using the application manager in Splunk Web (Settings / Manages apps)
- Extracting the content of the `tgz` archive in the “apps” directory of Splunk

- For SHC configurations (Search Head Cluster), extract the tgz content in the SHC deployer and publish the SHC bundle

2.1.4 Upgrades

Upgrading the Splunk application is pretty much the same operation than the initial deployment.

All of TrackMe components and configuration items are upgraded resilient, in respects with Splunk configuration good practices.

2.2 Configuration

2.2.1 Configuring in Splunk Web

Usually, the configuration should be achieved via Splunk Web and the configuration UI:

Configuration page:

The screenshot shows the 'Configuration' page for the 'JIRA Service Desk' add-on in Splunk Web. The page has a top navigation bar with links: 'Overview - JIRA Service Desk', 'Overview - JIRA Projects', 'Search', 'REST API explore', and 'Configuration'. Below the navigation bar, the page title is 'Configuration' with the subtitle 'Set up your add-on'. There are three tabs: 'Proxy', 'Logging', and 'Add-on Settings', with 'Add-on Settings' being the active tab. The configuration fields are as follows:

- JIRA Service Desk URL ***: A text input field containing 'jira:8081'. Below the field, it says 'jira.atlassian.net (SSL is enforced and the URL submitted will be substituted with https://)'.
- JIRA username ***: A text input field containing 'gmarchand'. Below the field, it says 'Enter the account name to be used for the authentication'.
- JIRA password ***: A password input field with masked characters '*****'. Below the field, it says 'Enter the password token associated with this account'.
- SSL certificate validation**: A checkbox that is currently unchecked. Below it, it says 'Check this box to perform SSL certificate validation'.
- Enable passthrough mode**: A checkbox that is currently unchecked. Below it, it says 'Enable the passthrough mode, in this mode the instance will not attempt to contact the Jira instance but will write directly to the replay KVstore, see the documentation for more details'.

At the bottom of the configuration section, there is a green 'Save' button.

In a Search Head Cluster context, the generated configuration is automatically replicated across the members of the cluster.

2.2.2 Configuring via REST API

Alternatively, and this can be useful if for some reason you cannot access to the configuration UI (no end dead loop), the configuration can easily be achieved via REST calls to the Splunk API with curl.

Configuring the JIRA instance via curl

Assuming:

- JIRA instance URL: myjira.mydomain.com:8443
- JIRA login username: admin
- JIRA password: ch@ngeM3

You would run the following curl command, either locally on a search head (in SHC, this will be replicated automatically), or remotely reaching out to a search head:

```
curl -k -u admin:'ch@ngeM3' -X POST https://localhost:8089/servicesNS/nobody/TA-jira-
↪service-desk-simple-addon/TA_jira_service_desk_simple_addon_settings/additional_
↪parameters -d 'jira_url=myjira.mydomain.com:8443' -d 'jira_username=admin' -d 'jira_
↪password=ch@ngeM3'
```

You can verify your settings with a GET:

```
curl -k -u admin:'ch@ngeM3' -X GET https://localhost:8089/servicesNS/nobody/TA-jira-
↪service-desk-simple-addon/TA_jira_service_desk_simple_addon_settings/additional_
↪parameters
```

Enabling SSL validation

If you wish to enable the SSL certificate validation:

```
curl -k -u admin:'ch@ngeM3' -X POST https://localhost:8089/servicesNS/nobody/TA-jira-
↪service-desk-simple-addon/TA_jira_service_desk_simple_addon_settings/additional_
↪parameters -d 'jira_ssl_certificate_validation=1'
```

Enabling the passthrough mode

To enable the passthrough mode:

```
curl -k -u admin:'ch@ngeM3' -X POST https://localhost:8089/servicesNS/nobody/TA-jira-
↪service-desk-simple-addon/TA_jira_service_desk_simple_addon_settings/additional_
↪parameters -d 'jira_passthrough_mode=1'
```

Setting the logging mode

To enable DEBUG logging:

```
curl -k -u admin:'ch@ngeM3' -X POST https://localhost:8089/servicesNS/nobody/TA-jira-
↪service-desk-simple-addon/TA_jira_service_desk_simple_addon_settings/logging -d
↪'loglevel=DEBUG'
```

Enable and configure the proxy

Example:

```
curl -k -u admin:'ch@ngeM3' -X POST https://localhost:8089/servicesNS/nobody/TA-jira-  
→service-desk-simple-addon/TA_jira_service_desk_simple_addon_settings/proxy -d  
→'proxy_enabled=1' -d 'proxy_url=myproxy.domain.com' -d 'proxy_port=8080'
```

Additional options are:

- proxy_username (string)
- proxy_password (string)
- proxy_rdns (boolean, 0 disabled, 1 enabled)
- proxy_type (http/socks4/socks5)

2.2.3 Configuration details

Configure your JIRA instance

Enter the configuration page in the UI to setup the JIRA instance URL and credentials to be used.

The Splunk Add-on for JIRA service desk implements basic authentication as described here:

- <https://developer.atlassian.com/server/jira/platform/basic-authentication>
- <https://developer.atlassian.com/cloud/jira/service-desk/basic-auth-for-rest-apis>

The JIRA instance configuration requires:

- The JIRA URL which is https enforced, you can define the instance without the protocol like “myjira.mydomain.com” or “https://myjira.domain.com”
- The user name to be used for authentication
- The secret token defined for this user

Optionally you can request for SSL certificates validation during the REST call made to JIRA api during the issue creation, which will require the certificates of the instance to be fully valid.

Logging level

The logging level can be defined within the configuration page too, the application makes a real usage of the debug mode and will generate many more messages in debug.

In normal circumstances, the logging level should be defined to INFO, required logging level will automatically be used when any unexpected error is encountered.

Validating the connectivity

You can validate the connectivity very easily by opening any of the JIRA Get information reports, which achieve rest calls to the JIRA API to retrieve different information such as the list of projects available:

Overview - JIRA Service Desk Search Get JIRA info Logging reports Builtin alerts Alerts Configuration Run a search

JIRA Service Desk

This report exposes JIRA project

Last 5 minutes

✓ 2 events (13/04/2020 06:10:35)

2 results 20 per page

key	key_projects
SPLUNK	SPLUNK - splunk
TEST	TEST - Test

Shall the connectivity be effective and if you open the Get projects report, the list of the JIRA projects available for your JIRA instance appears in the table.

```
| jirafill opt=1 | stats count by key, key_projects
```

If the command returns the list of your JIRA projects, then the connectivity is successful:

Overview - JIRA Service Desk Search Logging reports Builtin alerts Alerts Configuration Run a search

New Search

| jirafill opt=1 | stats count by key, key_projects

✓ 2 events (11/04/2020 10:00:00.000 to 12/04/2020 10:24:23.000) No Event Sampling

Events Patterns **Statistics (2)** Visualization

20 Per Page Format Preview

key	key_projects
SPLUNK	SPLUNK - splunk
TEST	TEST - Test

You can as well simulate the creation of an alert and action the JIRA Service Desk:

- Enter a search window
- type `|makeresults`
- Click save as new alert
- Scroll down to alert actions and add the JIRA Service Desk action

Save As Alert

When triggered



JIRA Service Desk

JIRA main fields:

Project *

Select...



Issue Type *

filter



SPLUNK - splunk

Priority *

TEST - Test

Dynamic

Priority



Override priority using a field result, ex
\$result.jira_priority\$. (optional, case
sensitive, ticket creation will fail if
incorrectly defined)

Summary *

Splunk Alert: \$name\$

Description

The alert condition for
'\$name\$' was triggered.

Issue description, this text can include
tokens based on the search results (E.g:
\$result.src\$)

Assignee

Issue assignee. (optional)

Cancel

Testing access and authentication with curl:

You can as well very easily achieve a test with curl from the search head:

```
curl -k https://<jira_url>/rest/api/latest/project --user <jira_username>:<jira_
↵password>
```

Which, if successful, will return in a JSON format the list of projects available in your JIRA instance.

Using the alert action for non admin users

For non admin users to be able to use the alert action, the following role is provided out of the box:

- jira_alert_action


This role needs to be inherited for the users, or your users to be member of this role.

The role provides:

- capability list_storage_passwords
- capability list_settings
- write permission to the resilient KVstore kv_jira_failures_replay

JIRA passthrough mode**What is the JIRA passthrough?**

The passthrough has been designed for specific use cases where the Splunk main deployment is not capable of reaching directly the JIRA instance due to network and security constraints.

[Overview - JIRA Service Desk](#)[Overview - JIRA Projects](#)[Search](#) [REST API explore](#)[Conf](#)

Configuration

Set up your add-on

Proxy

Logging

Add-on Settings

JIRA Service Desk URL *

nginx:8081

jira.atlassian.net (SSL is enforced and the URL submitted will be substituted with https://)

JIRA username *

admin

Enter the account name to be used for the authentication

JIRA password *

.....

Enter the password token associated with this account

SSL certificate validation


☐

Check this box to perform SSL certificate validation

Enable passthrough mode

☒

Enable the passthrough mode, in this mode the instance will not attempt to contact the Jira instance but will write directly to the replay KVstore, see the documentation for more detail



Save

This use case is common enough for Splunk Cloud customers running JIRA on-premise, due to security considerations, it may be refused or complex to open a connectivity between Splunk Cloud and the on-premise JIRA.

Hint:

- The JIRA passthrough requires a Splunk hybrid search head connected to Splunk Cloud
 - Work with Splunk Cloud teams and Splunk Professional Services to get the setup ready
 - The final setup will allow JIRA issues creation from alerts (correlation searches in Enterprise Security) and ad-hoc adaptive response actions in incident review
 - In passthrough mode, the CSV/JSON attachment feature is not available
 - In passthrough mode, the JIRA dedup and auto-comment feature is not available
-

In a nutshell:

- The Splunk Cloud search head creates content in a local replay KVstore
- We rely on summary events to make the link between the Splunk Cloud environment and the on-premise hybrid search head
- The hybrid search performs the JIRA issue creation

Using the passthrough mode can accommodate this scenario with some additional configuration and setup, things will work as:

- The Splunk Cloud search head enabled the passthrough mode in the JIRA Add-on
- In this mode, the Add-on will not attempt to contact JIRA, instead it will insert issues to be created into the replay KVstore
- A scheduled report is created in the Splunk Cloud instance which looks at the content of the local replay KVstore and runs a collect command to generate summary events
- A scheduled report is created in the Splunk Cloud search head to purge processed issues by the hybrid search head (using logs generated by the hybrid and indexed in Splunk Cloud transparently)
- An on-premise Splunk search head is available in hybrid search mode, this means this search head can search in the Splunk Cloud indexers transparently
- The JIRA Add-on is installed in the hybrid search head, and the JIRA instance is properly configured (the hybrid search will create the issues)
- A scheduled report is created in the hybrid search head looking at the summary events, and filling its content to the local replay KVstore
- Natively, on the hybrid search head, the Add-on is looking at the replay KVstore and handles each record to be create as a new JIRA issue, and maintains (purges) the life cycle of the records upon their creation

Step 1: Get the JIRA Add-on installed


The JIRA Add-on must be installed to both the Splunk Cloud search, and the hybrid on-premise search head.

Step 2: Splunk Cloud - create a dummy configuration in the Add-on and enable the passthrough

To accept creating records in the local replay KVstore, you first need to setup a dummy connection to JIRA.

The target is not important, it will not be used as soon when the passthrough is enabled, it is required to allow the Add-on to create records in the replay KVstore.

Example:

[Overview - JIRA Service Desk](#)[Overview - JIRA Projects](#)[Search](#) [REST API explore](#)[Conf](#)

Configuration

Set up your add-on

[Proxy](#)[Logging](#)[Add-on Settings](#)

JIRA Service Desk URL *

dummy:8081

jira.atlassian.net (SSL is enforced and the URL submitted will be substituted with https://)

JIRA username *

admin

Enter the account name to be used for the authentication

JIRA password *

.....

Enter the password token associated with this account

SSL certificate validation


☐

Check this box to perform SSL certificate validation

Enable passthrough mode

☒

Enable the passthrough mode, in this mode the instance will not attempt to contact the Jira instance but will write directly to the replay KVstore, see the documentation for more detail




Save

Step 3: Hybrid - configure the JIRA connectivity

Make sure to setup properly the JIRA configuraton in the hybrid search head.

Note: do not enable the passthrough mode in the hybrid search head!

[Overview - JIRA Service Desk](#)[Overview - JIRA Projects](#)[Search](#) [REST API explore](#)[Conf](#)

Configuration

Set up your add-on

[Proxy](#)[Logging](#)[Add-on Settings](#)

JIRA Service Desk URL *

nginx:8081

jira.atlassian.net (SSL is enforced and the URL submitted will be substituted with https://)

JIRA username *

admin

Enter the account name to be used for the authentication

JIRA password *

.....

Enter the password token associated with this account

SSL certificate validation

☐

Check this box to perform SSL certificate validation

Enable passthrough mode

☐

Enable the passthrough mode, in this mode the instance will not attempt to contact the Jira instance but will write directly to the replay KVstore, see the documentation for more detail

Save

Step 4: Splunk Cloud - create lookups to populate the alert action dropdown

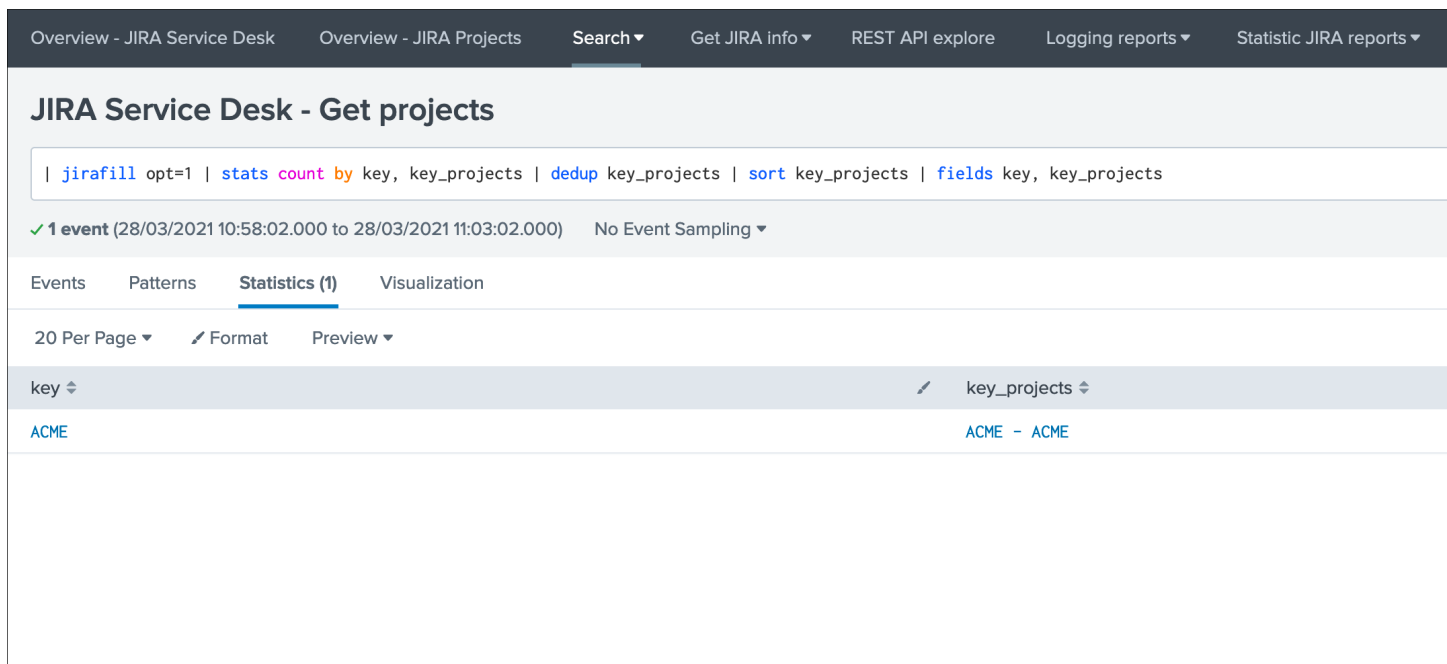
In normal circumstances, the Add-on populates the dropdown (projects, issue types, priorities) dynamically by performing REST calls to JIRA.

In our case, this will not be possible, this can be managed by running the relevant commands on the hybrid search head, extracts these as CSV files, and upload these as lookup in Splunk Cloud.

Finally, we will customise the populating macros to call these lookups rather than the jirafill custom command which normally does the rest calls.

Get JIRA projects

Run the report **JIRA Service Desk - Get projects** from the hybrid search head (in the nav menu “Get JIRA INFO”) and export as a CSV file:



The screenshot shows the Splunk Cloud interface for the report "JIRA Service Desk - Get projects". The navigation bar at the top includes links for Overview - JIRA Service Desk, Overview - JIRA Projects, Search, Get JIRA info, REST API explore, Logging reports, and Statistic JIRA reports. The report title "JIRA Service Desk - Get projects" is displayed. Below the title, the search query is shown: `| jirafill opt=1 | stats count by key, key_projects | dedup key_projects | sort key_projects | fields key, key_projects`. The report shows 1 event for the time range 28/03/2021 10:58:02.000 to 28/03/2021 11:03:02.000. The "Statistics (1)" tab is selected, showing a table with the following data:

key	key_projects
ACME	ACME - ACME

Run the report **JIRA Service Desk - Get issue types** from the hybrid search head (in the nav menu “Get JIRA INFO”) and export as a CSV file:

Overview - JIRA Service Desk Overview - JIRA Projects Search ▼ Get JIRA info ▼ REST API explore Logging reports ▼ Statistic JIRA reports ▼

JIRA Service Desk - Get issue types

This report exposes JIRA issue types available

Last 5 minutes ▼

✓ 7 events (28/03/2021 13:25:34.000 to 28/03/2021 13:30:34.000)

7 results 20 per page ▼

issues ⇅

- Bug
- Epic
- Improvement
- New Feature
- Story
- Sub-task
- Task

Run the report **JIRA Service Desk - Get issue priorities** from the hybrid search head (in the nav menu “Get JIRA INFO”) and export as a CSV file:

Overview - JIRA Service Desk Overview - JIRA Projects Search ▼ Get JIRA info ▼ REST API explore Logging reports ▼ Statistic JIRA reports ▼

JIRA Service Desk - Get issue priorities

This report exposes JIRA priorities available

Last 5 minutes ▼

✓ 5 events (28/03/2021 11:01:50.000 to 28/03/2021 11:06:50.000)

5 results 20 per page ▼

priorities ⇅

- High
- Highest
- Low
- Lowest
- Medium

Upload these lookups files in Splunk Cloud via Splunk Web, example:

Lookup table files

[Settings](#) » [Lookups](#) » Lookup table files

Showing 1-3 of 3 items

App Owner Created in the App

Path ↕	Owner ↕	App ↕
/opt/splunk/etc/apps/TA-jira-service-desk-simple-addon/lookups/jira_issue_types.csv	admin	TA-jira-service-desk-s
/opt/splunk/etc/apps/TA-jira-service-desk-simple-addon/lookups/jira_priorities.csv	admin	TA-jira-service-desk-s
/opt/splunk/etc/apps/TA-jira-service-desk-simple-addon/lookups/jira_projects.csv	admin	TA-jira-service-desk-s

Hint:

- Make sure the lookups are shared at the global level

Finally, update the populating macros to use these lookups instead:

get_jira_projects:

```
inputlookup jira_projects.csv
```

get_jira_issue_types:

```
inputlookup jira_issue_types.csv
```

get_jira_priorities:

```
inputlookup jira_priorities.csv
```

Example:

Search macros

[Settings](#) » [Advanced search](#) » Search macros

Showing 1-4 of 4 items

App Owner Created in the App

Name ↕	Definition ↕	Arguments ↕	Owner ↕	App ↕
comment()	""	text	admin	TA-jira-service-desk-simple-addon
get_jira_issue_types	inputlookup jira_issue_types.csv		admin	TA-jira-service-desk-simple-addon
get_jira_priorities	inputlookup jira_priorities.csv		admin	TA-jira-service-desk-simple-addon
get_jira_projects	inputlookup jira_projects.csv		admin	TA-jira-service-desk-simple-addon

Step 6: Splunk Cloud - disable replay out of the box reports

Using Splunk Web, disable the report JIRA Service Desk - Resilient store Tracker, this report must not be running from the Splunk Cloud search head as this job will be handled by the hybrid search head.

Step 7: Splunk Cloud - create a collect scheduled report

In the Splunk Cloud search head, create a new scheduled report in the JIRA Add-on application space, scheduled every 5 minutes with the following code:

JIRA - Collect replay KVStore:

```
| inputlookup jira_failures_replay | eval uuid=_key | eval _time=ctime
| where _time>relative_time(now(), "-5m")
| collect index=summary source=jira_replay_kvstore
```

Hint:

- This setup example uses the default summary index for the demonstration purposes, you can change this to a custom index of your choice

Example:

Edit Search

Title	JIRA - Collect replay KVStore
Description	optional
Search	<pre> inputlookup jira_failures_replay eval uuid=_key eval _time=ctime where _time>relative_time(now(), "-5m") collect index=summary source=jira_replay_kvstore</pre>
Earliest time	-5m Time specifiers: y, mon, d, h, m, s Learn More
Latest time	now Time specifiers: y, mon, d, h, m, s Learn More

Cancel

Step 8: Splunk Cloud - create a purge scheduled report

In the Splunk Cloud search head, create a new scheduled report in the JIRA Add-on application space, scheduled every 15 minutes with the following code:

JIRA - Purge processed issues in the replay KVstore:

```
| inputlookup jira_failures_replay | eval uuid=_key
| search NOT [ search (index="_internal" OR index="cim_modactions") (source="*jira_
↪service_desk_replay_modalert.log") "Purging ticket in KVstore with uuid" | table_
↪uuid ]
| eval _key=uuid
| outputlookup jira_failures_replay
```

This job will purge records in the KVstore that have been successfully proceeded by the hybrid search head, thanks to the JIRA Add-on logging capabilities which inform us about the status of issues created from the replay KVstore.

Example:

Edit Search

Title **JIRA - Purge processed issues in the replay KVstore**

Description optional

Search

```
| inputlookup jira_failures_replay | eval uuid=_key
| search NOT [ search (index="_internal" OR index="cim_modactions") (sour
    *jira_service_desk_replay_modalert.log") "Purging ticket in KVstore w
    uuid" | table uuid ]
| eval _key=uuid
| outputlookup jira_failures_replay
```

Earliest time -15m

Time specifiers: y, mon, d, h, m, s [Learn More](#)

Latest time now

Time specifiers: y, mon, d, h, m, s [Learn More](#)

Step 9 final: Hybrid search head - create a report recycling the summary events to feed the replay KVstore

Finally, create a new scheduled report in the hybrid Splunk Search head, in the JIRA Add-on application space, scheduled every 5 minutes looking at the 10 minutes of data, with the following code:

JIRA - Collect and fill the replay KVstore:

```
index=summary source=jira_replay_kvstore
| table ctime data mtime no_attempts status uuid
| eval key=uuid
| lookup jira_failures_replay _key as uuid OUTPUT _key as uuid_found
| where isnull(uuid_found) | fields - uuid_found
| outputlookup jira_failures_replay append=t key_field=key
```

Hint:

- If you used a different index in the previous step, make sure to reflect this change here

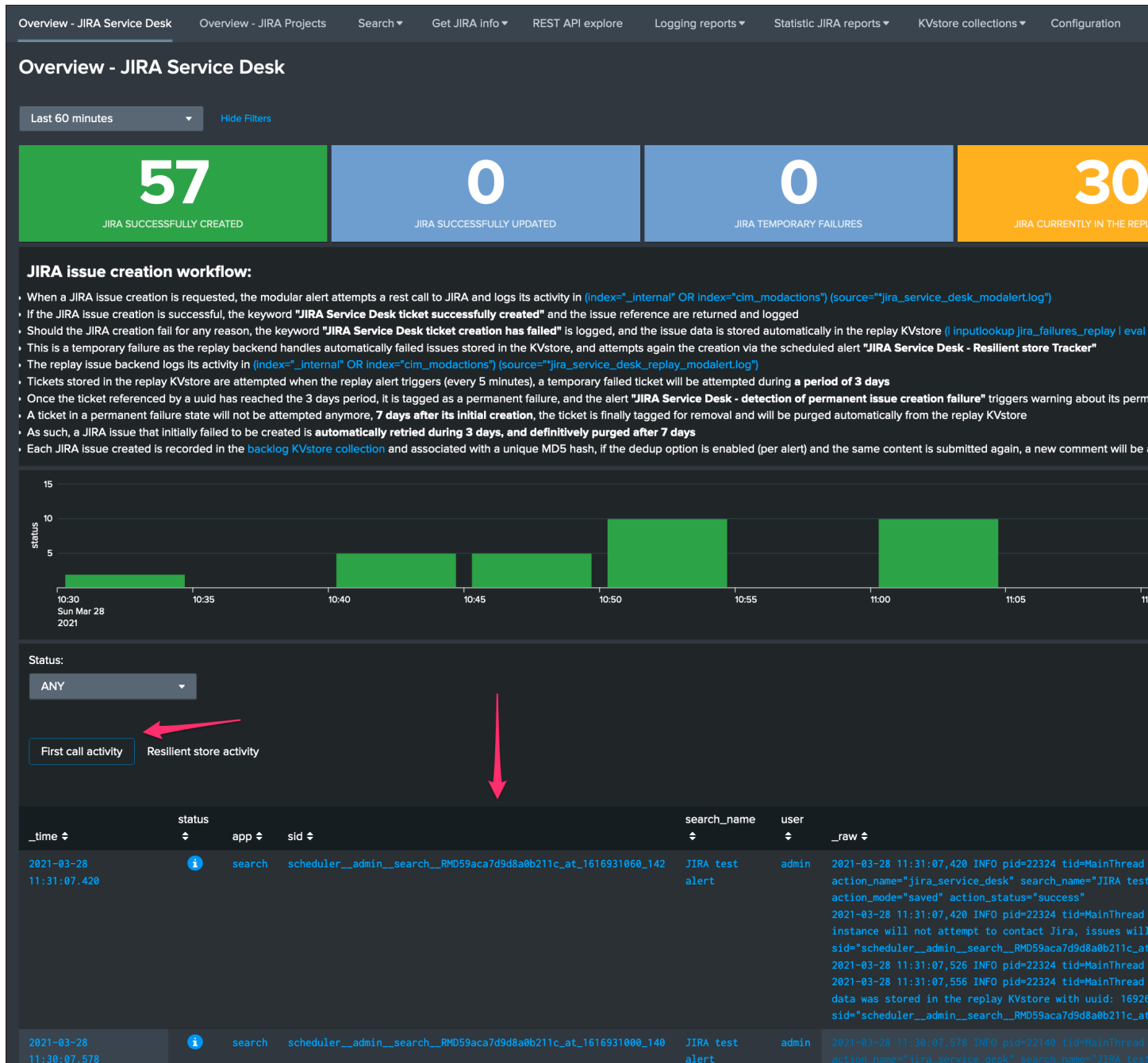
Edit Search

Title	JIRA - Collect and fill the replay KVstore
Description	optional
Search	<pre> index=summary source=jira_replay_kvstore table ctime data mtime no_attempts status uuid eval key=uuid lookup jira_failures_replay _key as uuid OUTPUT _key as uuid_found where isnull(uuid_found) fields - uuid_found outputlookup jira_failures_replay append=t key_field=key </pre>
Earliest time	-10m
	Time specifiers: y, mon, d, h, m, s Learn More
Latest time	now
	Time specifiers: y, mon, d, h, m, s Learn More

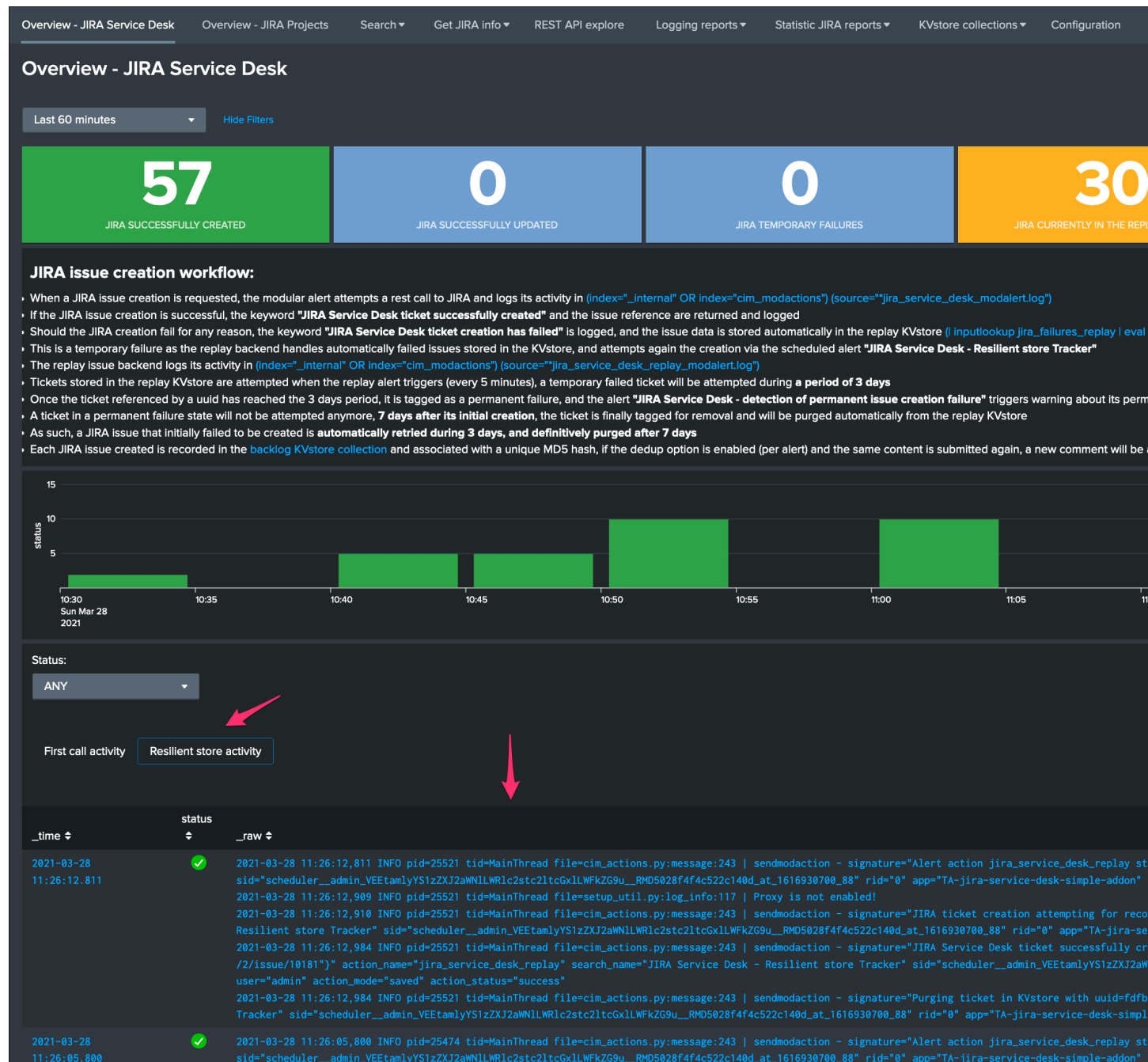
Final review

Congratulations! The step is now terminated, because logs from the execution of the Hybrid search head are made available to the Splunk Cloud search head (the hybrid forwards to the Splunk Cloud environment), the UI shows all the relevant information.

For instance, the JIRA issues “created” on the Splunk Cloud search head, will appear in the first tab and tagged as info:



The logs exposing the real creation of the issues via the replay KVstore are available in the second tab called “Resilient store activity”:



The configuration is now over and fully functional, the hybrid search will respect the normal TA workflow, issues to be created will be removed automatically from the replay KVstore upon a successful creation.

3.1 User guide

3.1.1 Using the JIRA Service Desk alert action from alerts and correlation searches

Whenever you create or configure a Splunk core alert or Enterprise Security correlation search, you can now select the JIRA Service Desk action to automatically create a new JIRA issue based on the results of a search.

Save As Alert

Settings

Title


Description


Permissions ☒ Private ☐ Shared in App


Alert type ☒ Scheduled ☐ Real-time


Run every week ▼


On at


 **Add to Triggered Alerts**
Add this alert to Triggered Alerts list

 **JIRA Service Desk**
Open an issue in JIRA Service Desk

 **Log Event**
Send log event to Splunk receiver endpoint

 **MS Teams**

 **MS teams publish to channel**
Publish a message to a Microsoft Teams channel

 **Okta Group Member Change**
Change Okta Group Membership based on inputs

The configuration of the alert is pretty straightforward and described in detail in the further sections of the above documentation.


3.1.2 Using the JIRA Service Desk alert adaptive response action from Splunk Enterprise Security

In Splunk Enterprise Security, the JIRA action can be triggered as an adaptive response action from Incident Review:

Adaptive Response Actions

Select actions to run.

+ Add New Response Action ▾

▼  JIRA Service Desk

JIRA main fields:

Project * required

TEST - Test ▾ X

Issue Type * required

Incident ▾ X

Priority

High ▾ X

Dynamic Priority

(Optional) Override priority using a field result, ex \$result.jira_priority\$. (case sensitive, ticket creation will fail if incorrectly defined)

Summary * required

IP 192.168.0.2 was detected wil

Description

The IP 192.168.0.2 was detected generating prohibited application traffic, please act accordingly.

(Required) Issue description, this text can include tokens based on the search results (E.g: \$result.src\$)

Assignee

The same options are available with the same level of features; however, tokens expansion will depend on the notable event context.

3.1.3 JIRA project



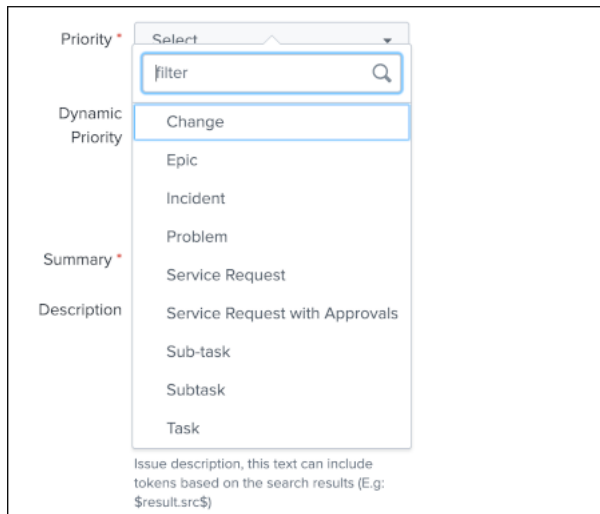
A screenshot of a web form showing a dropdown menu for selecting a JIRA project. The label 'Project *' is to the left of the dropdown. The dropdown itself has a light blue border and contains the text 'Select...' followed by a downward arrow.

Several projects might have been created in your JIRA instance; you can choose any of the projects available on per alert basis.

The list of JIRA projects made available within the configuration screen is the result of a dynamic REST call achieved against your JIRA instance anytime you access this screen, which can be reproduced manually too:

```
| jirafill opt=1 | stats count by key, key_projects
```

3.1.4 JIRA issue type

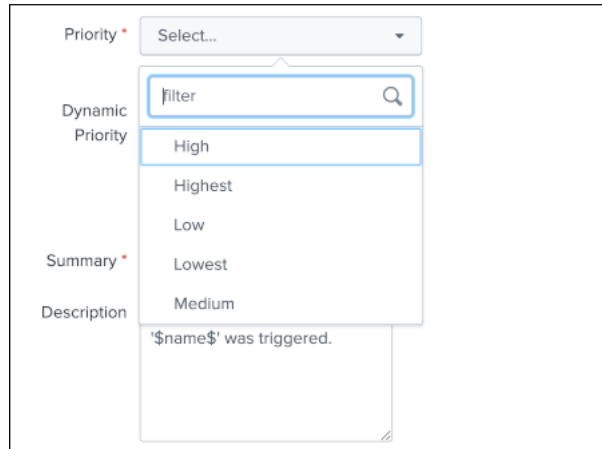


A screenshot of a web form showing a dropdown menu for selecting a JIRA issue type. The label 'Priority *' is to the left of the dropdown. The dropdown itself has a light blue border and contains the text 'Select' followed by a downward arrow. Below the dropdown, there is a search bar with the placeholder text 'filter' and a magnifying glass icon. Below the search bar, there is a list of issue types: 'Change', 'Epic', 'Incident', 'Problem', 'Service Request', 'Service Request with Approvals', 'Sub-task', 'Subtask', and 'Task'. The 'Change' option is highlighted with a blue border. Below the list, there is a small text box with the label 'Issue description, this text can include tokens based on the search results (E.g: \$result.src\$)'.

The type of issue to be created is a dynamic list provided by JIRA based on the types available for the project that has been selected, these are the result of the following command:

```
| jirafill opt=2 | stats count by issues
```

3.1.5 JIRA issue priority



The priority of the issue is dynamically retrieved from the JIRA project based on the different priorities that are made available by your JIRA screen configuration, these are the results of the following command:

```
| jirafill opt=3 | stats count by priorities
```

3.1.6 JIRA issue dynamic priority



The dynamic priority is a feature that allows you to dynamically define the priority based on the search result rather than a selected priority from the dynamic list provided by JIRA.

To use the priority of the search results, you need to define a field in your search results that exactly match the priority value expected by JIRA, which can obviously be the results of conditional operations in your SPL logic.

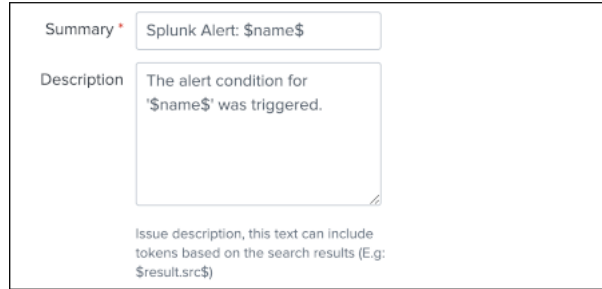
Assuming the following simplistic example in your search:

```
| eval jira_priority=case(count<10, "low", count>=10 AND count<50, "medium", count>
↪=50, "high")
```

You will define the dynamic priority to: \$result.jira_priority\$

The dynamic priority is entirely **optional** and is only used if it has been defined in the alert configuration.

3.1.7 JIRA summary and description



The screenshot shows two input fields within a form. The first field is labeled 'Summary *' and contains the text 'Splunk Alert: \$name\$'. The second field is labeled 'Description' and contains the text 'The alert condition for '\$name\$' was triggered.' Below the description field, there is a small note: 'Issue description, this text can include tokens based on the search results (E.g: \$result.src\$)'.

JIRA summary and description are the core information of a JIRA issue.

These two fields define the title of the JIRA issue, and its main content visible to your JIRA users.

Both fields will automatically handle any dynamic value that are available from the results of your search, which requires to be defined as `$result.myfield$` to be automatically translated into the relevant value.

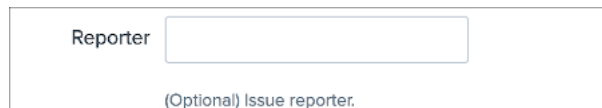
3.1.8 JIRA assignee



The screenshot shows a single input field labeled 'Assignee'. Below the field, there is a small note: 'Issue assignee. (optional)'.

The JIRA assignee field is **optional**, and can be defined to a static or a dynamic value (using a token) to automatically assign the issue to a specific JIRA user.

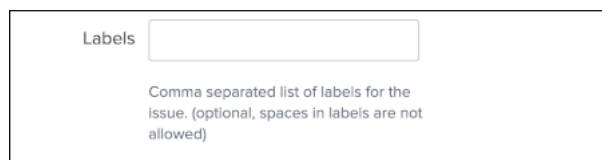
3.1.9 JIRA reporter



The screenshot shows a single input field labeled 'Reporter'. Below the field, there is a small note: '(Optional) Issue reporter.'

The JIRA reporter field is **optional**, and can be defined to a static or a dynamic value (using a token) to automatically assign the issue to a specific JIRA user.

3.1.10 JIRA labels



The screenshot shows a single input field labeled 'Labels'. Below the field, there is a small note: 'Comma separated list of labels for the issue. (optional, spaces in labels are not allowed)'.

JIRA labels is an **optional** field, which can be defined as a comma separated list of values to assign a list of labels to the JIRA issue.

3.1.11 JIRA components

Components names	<input type="text"/>
Comma separated list of component names for the issue. (optional, no space characters after commas if specifying multiple component names)	

JIRA components is an **optional** field, which can be defined as a comma separated list of values to assign a list of components to the JIRA issue. (by their names)

3.1.12 JIRA dedup behavior

Edit Alert

JIRA dedup behaviour:

Enabled

When a new JIRA issue is created, the issue key reference and it's unique md5 hash are stored in a KVstore.

If the jira dedup option is enabled and shall the same issue creation be requested again, a new comment will be added to the existing issue rather than a new issue created.

The md5 hash calculation guarantees the issue unique identification and is performed against the entire issue content. (unless jira_dedup_content is defined)

To define the content of the comment (defaults to: New alert triggered: <issue summary>), generate a field named "jira_update_comment" and this will be used as the comment automatically.

JIRA dedup excluded status categories

Done

Comma separated list of Jira status categories that will not be considered for updates, if dedup is enabled and the duplicated issue status is in one of these categories, a new issue will be created instead.

JIRA dedup content

(Optional) The default behavior is to use the full issue content to calculate the md5, by defining its content here you can limit its scope.

If this field is unset, the entire issue data is used for the deduping purposes.

Example:
Splunk Alert: \$name\$,

Cancel

The JIRA deduplication is a powerful feature that allows to automatically control the decision to create or update an issue, which relies on a bidirectional integration with JIRA.

The feature relies on 3 main options:

- JIRA dedup behaviour: this enables the dedup feature, disabled by default
- JIRA dedup excluded status categories: A comma separated list of statuses that will be considered for the decision
- JIRA dedup content: (Optional) Provides extra control on the content used to make the decision

Let's take the following example to explain how the feature works:

The following search simulates an alert triggering:

```
| makeresults
| eval user="foo@splunk.com", action="failure", reason="Authentication failed"
| eval time=strftime(_time, "%c")
```

Test JIRA - demo dedup

```
| makeresults
| eval user="foo@splunk.com", action="failure", reason="Authentication failed"
| eval time=strftime(_time, "%c")
```

✓ 1 result (24/06/2021 20:00:00.000 to 25/06/2021 20:28:53.000) No Event Sampling ▼

Events Patterns **Statistics (1)** Visualization

20 Per Page ▼ Format Preview ▼

_time ↕	action ↕	reason ↕	tin
2021-06-25 20:28:53	failure	Authentication failed	Fr

- everytime the alert triggers, the values for user, action and reason remain the same
- the time value differs every time the action triggers

Let's enable the JIRA alert action, we'll include in the description field all the fields from resulting from the alert:

Edit Alert

When triggered



JIRA Service Desk

JIRA main fields:

Project *
required

LAB - LAB

Issue Type *
required

Bug



Priority

High

Dynamic
Priority

(Optional) Override priority using a field result, ex `$result.jira_priority$`. (case sensitive, ticket creation will fail if incorrectly defined)

Summary *
requiredSplunk Alert: `$name$`

Description

The alert condition for '`$name$`' was triggered.

- user: `$result.user$`
- action: `$result.action$`
- reason: `$result.reason$`
- time: `$result.time$`

(Required) Issue description, this text can include tokens based on the search results (E.g: `$result.src$`)

Cancel

For now, we didn't enable the dedup feature, if we use the DEBUG logging mode, the logs will show the full JSON payload sent to the JIRA API in pretty print manner:

Use the navigation bar shortcut to access the logs, the final JSON is logged with a message: *json data for final rest call*

```
> 25/06/2021 2021-06-25 20:33:05,388 DEBUG pid=5759 tid=MainThread file=cim_actions.py:message:243 | sendmodaction - signature=
20:33:05.388      "fields": {
                    "project": {
                        "key": "LAB"
                    },
                    "summary": "Splunk Alert: Test JIRA - demo dedup",
                    "description": "The alert condition for 'Test JIRA - demo dedup' was triggered.\n\n user: foo@splunk.com",
                    "issuetype": {
                        "name": "Bug"
                    },
                    "priority": {
                        "name": "High"
                    }
                }
            }" action_name="jira_service_desk" search_name="Test JIRA - demo dedup" sid="scheduler__admin__search__RMD526ad4cfa87997743_at_1624653180_13" rid="0" app="search" user="admin" action_mode="saved"
Collapse
host = splunk | source = /opt/splunk/var/log/splunk/jira_service_desk_modalert.log | sourcetype = tajira:service:desk:modalert
```

Even if we didn't enable yet the feature, the Addon calculates an MD5 sum which is recorded in a KVstore collection, traces are logged about this:

```
2021-06-25 20:33:05,394 DEBUG pid=5759 tid=MainThread file=cim_actions.py:message:243 |
→| sendmodaction - signature="jira_dedup: The calculated md5 hash for this issue_
→creation request (db05a46bd3a2e6ccb57906cd749db047) was not found in the backlog_
→collection, a new issue will be created" action_name="jira_service_desk" search_
→name="Test JIRA - demo dedup" sid="scheduler__admin__search__RMD526ad4cfa87997743_
→at_1624653180_13" rid="0" app="search" user="admin" action_mode="saved"
```

The MD5 sum is calculated against the entire JSON data.

To access the KVstore collection containing these records, look at the nav menu “KVstore collections / JIRA Service Desk - Issues backlog collection”.

As every ticket corresponds to a new issue, the status is “created”.

Now, let's modify a bit the alert, we will remove the time field from the description in JIRA, and enable the dedup:

Edit Alert

When triggered



JIRA Service Desk

JIRA main fields:

Project *
required

LAB - LAB



Issue Type *
required

Bug



Priority

High



Dynamic
Priority



(Optional) Override priority using a field result, ex \$result.jira_priority\$. (case sensitive, ticket creation will fail if incorrectly defined)

Summary *
required

Splunk Alert: \$name\$

Description

The alert condition for '\$name\$' was triggered.

- user: \$result.user\$
- action: \$result.action\$
- reason: \$result.reason\$

(Required) Issue description, this text can include tokens based on the search results

Cancel

Edit Alert

JIRA dedup
behaviour:

Enabled



When a new JIRA issue is created, the issue key reference and its unique md5 hash are stored in a KVstore.

If the jira dedup option is enabled and shall the same issue creation be requested again, a new comment will be added to the existing issue rather than a new issue created.

The md5 hash calculation guarantees the issue unique identification and is performed against the entire issue content. (unless jira_dedup_content is defined)

To define the content of the comment (defaults to: New alert triggered: <issue summary>), generate a field named "jira_update_comment" and this will be used as the comment automatically.

JIRA dedup
excluded status
categories

Done

Comma separated list of Jira status categories that will not be considered for updates, if dedup is enabled and the duplicated issue status is in one of these categories, a new issue will be created instead.

JIRA dedup
content

(Optional) The default behavior is to use the full issue content to calculate the md5, by defining its content here you can limit its scope.

If this field is unset, the entire issue data is used for the deduping purposes.

Example:
Splunk Alert: \$name\$,

Cancel

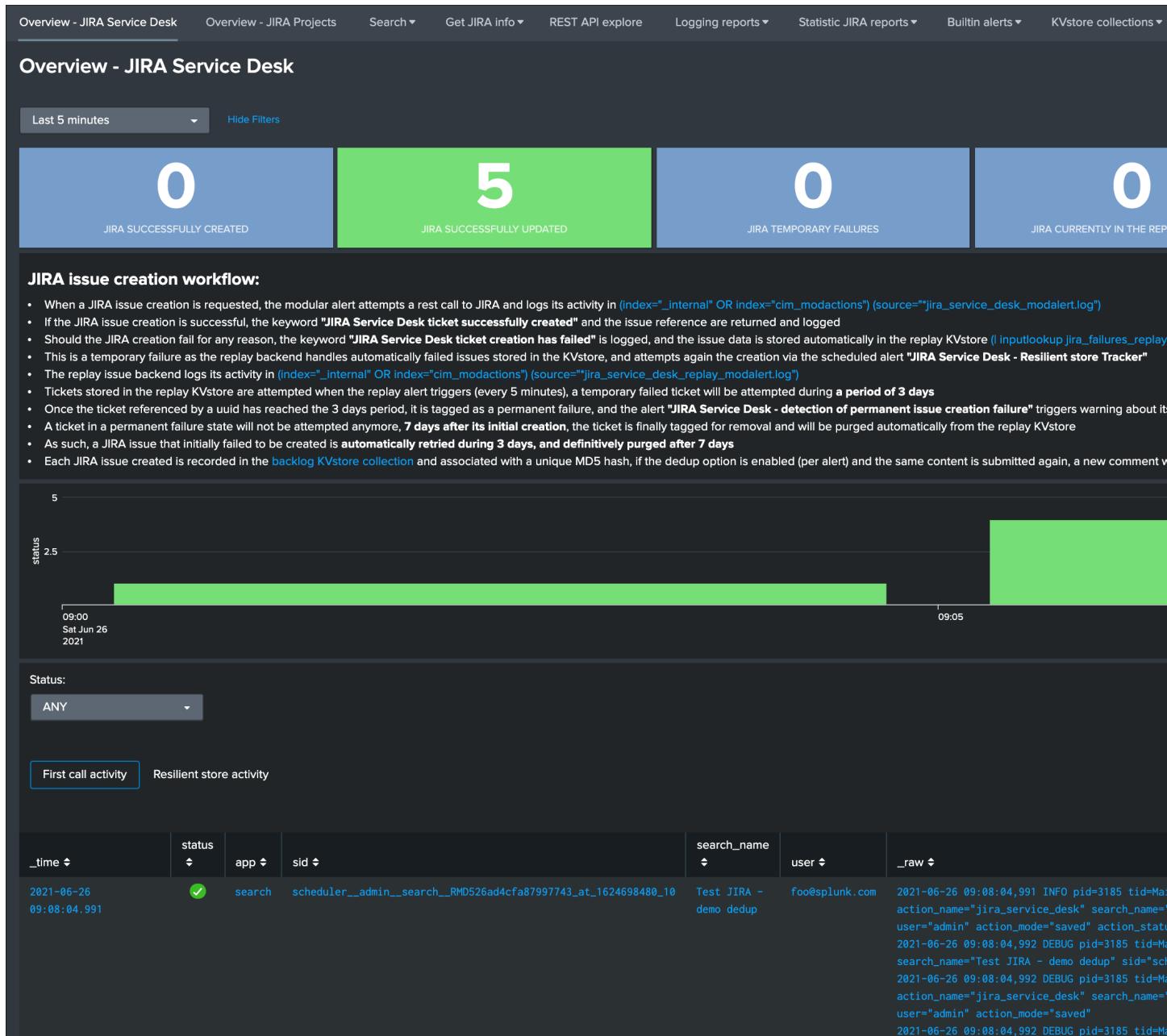
As the content of the JSON is exactly the same (we removed the time from the description), the Addon will detect it and perform an update of first created issue, adding a comment, and updating the record in the KVstore lookup:

```
2021-06-25 20:45:06,360 INFO pid=8814 tid=MainThread file=cim_actions.py:message:243
→ | sendmodaction - signature="jira_dedup: An issue with same md5 hash
→ (60727858c049e599fdb68a3cd744a911) was found in the backlog collection, as jira_
→ dedup is enabled a new comment will be added if the issue is active. (status is not_
→ resolved or any other done status), entry:={ "jira_md5" :
→ "60727858c049e599fdb68a3cd744a911", "ctime" : "1624652826.254012", "mtime" :
→ "1624652826.2540202", "status" : "created", "jira_id" : "10100", "jira_key" : "LAB-
→ 76", "jira_self" : "https://localhost:8081/rest/api/2/issue/10100", "_user" :
→ "nobody", "_key" : "60727858c049e599fdb68a3cd744a911" }" action_name="jira_service_
→ desk" search_name="Test JIRA - demo dedup" sid="scheduler__admin__search__
→ RMD526ad4cfa87997743_at_1624653900_33" rid="0" app="search" user="admin" action_
→ mode="saved" action_status="success"
```

The KVstore collection shows a status “updated” for the issue:

Overview - JIRA Service Desk Overview - JIRA Projects Search ▼ Get JIRA info ▼ REST API explore Logging reports ▼ Statistic JIRA reports ▼																																																																																																					
<h3>JIRA Service Desk - Issues backlog collection</h3> <p>This report exposes the JIRA issues backlog which contains records for every JIRA issue created by the add-on, this collection is as well used by the JIRA add-on backend for reveals an update was performed via the JIRA deduplication feature.</p> <p>Last 5 minutes ▼</p> <p>✓ 25 results (26/06/2021 09:01:50.000 to 26/06/2021 09:06:50.000)</p> <p>25 results 20 per page ▼</p> <table> <tr> <th>key ↕</th><th>ctime ↕</th><th>mtime ↕</th><th>jira_id ↕</th><th>jira_key ↕</th><th>jira_md5 ↕</th></tr> <tr> <td>84381893e99c5abba0ad786969515f75</td><td>Fri Jun 25 21:08:05 2021</td><td>Sat Jun 26 09:06:07 2021</td><td>10124</td><td>LAB-100</td><td>84381893e99c5abba0ad786969515f75</td></tr> <tr> <td>eef8ea948fca582bdfc99de30f3c945</td><td>Fri Jun 25 21:07:05 2021</td><td>Fri Jun 25 21:07:05 2021</td><td>10123</td><td>LAB-99</td><td>eef8ea948fca582bdfc99de30f3c945</td></tr> <tr> <td>8d17e9daf1b28359c2ebfa5b082e6e7b</td><td>Fri Jun 25 21:06:05 2021</td><td>Fri Jun 25 21:06:05 2021</td><td>10122</td><td>LAB-98</td><td>8d17e9daf1b28359c2ebfa5b082e6e7b</td></tr> <tr> <td>b6726bee12aee1336cc05a168e917591</td><td>Fri Jun 25 21:05:05 2021</td><td>Fri Jun 25 21:05:05 2021</td><td>10121</td><td>LAB-97</td><td>b6726bee12aee1336cc05a168e917591</td></tr> <tr> <td>4cde5b11c05904a5c1699647fe9a1249</td><td>Fri Jun 25 21:04:05 2021</td><td>Fri Jun 25 21:04:05 2021</td><td>10120</td><td>LAB-96</td><td>4cde5b11c05904a5c1699647fe9a1249</td></tr> <tr> <td>bbc4b514f589a048da9ccb0232eae21</td><td>Fri Jun 25 21:03:05 2021</td><td>Fri Jun 25 21:03:05 2021</td><td>10119</td><td>LAB-95</td><td>bbc4b514f589a048da9ccb0232eae21</td></tr> <tr> <td>d847c86a2084b12121dfa969695e1c28</td><td>Fri Jun 25 21:02:05 2021</td><td>Fri Jun 25 21:02:05 2021</td><td>10118</td><td>LAB-94</td><td>d847c86a2084b12121dfa969695e1c28</td></tr> <tr> <td>60727858c049e599fdb68a3cd744a911</td><td>Fri Jun 25 20:27:06 2021</td><td>Fri Jun 25 21:01:05 2021</td><td>10100</td><td>LAB-76</td><td>60727858c049e599fdb68a3cd744a911</td></tr> <tr> <td>efa7c0a8fda81c01f8c66ef66095f805</td><td>Fri Jun 25 20:44:06 2021</td><td>Fri Jun 25 20:44:06 2021</td><td>10117</td><td>LAB-93</td><td>efa7c0a8fda81c01f8c66ef66095f805</td></tr> <tr> <td>c779331e48490bd764fbf47c229b3f6f</td><td>Fri Jun 25 20:43:06 2021</td><td>Fri Jun 25 20:43:06 2021</td><td>10116</td><td>LAB-92</td><td>c779331e48490bd764fbf47c229b3f6f</td></tr> <tr> <td>293c764bcfc7245dbdba6f5867f588c6</td><td>Fri Jun 25 20:42:07 2021</td><td>Fri Jun 25 20:42:07 2021</td><td>10115</td><td>LAB-91</td><td>293c764bcfc7245dbdba6f5867f588c6</td></tr> <tr> <td>05bec26bd32b98a35d26618cc8865997</td><td>Fri Jun 25 20:41:06 2021</td><td>Fri Jun 25 20:41:06 2021</td><td>10114</td><td>LAB-90</td><td>05bec26bd32b98a35d26618cc8865997</td></tr> <tr> <td>0d45869dad21e6c7efff984404670e36</td><td>Fri Jun 25 20:40:06 2021</td><td>Fri Jun 25 20:40:06 2021</td><td>10113</td><td>LAB-89</td><td>0d45869dad21e6c7efff984404670e36</td></tr> <tr> <td>bfe88ac695c82f1a0216de1faa03128c</td><td>Fri Jun 25 20:39:06 2021</td><td>Fri Jun 25 20:39:06 2021</td><td>10112</td><td>LAB-88</td><td>bfe88ac695c82f1a0216de1faa03128c</td></tr> <tr> <td>65b895abf547404a50d6acb2c39bda80</td><td>Fri Jun 25 20:38:06 2021</td><td>Fri Jun 25 20:38:06 2021</td><td>10111</td><td>LAB-87</td><td>65b895abf547404a50d6acb2c39bda80</td></tr> </table>						key ↕	ctime ↕	mtime ↕	jira_id ↕	jira_key ↕	jira_md5 ↕	84381893e99c5abba0ad786969515f75	Fri Jun 25 21:08:05 2021	Sat Jun 26 09:06:07 2021	10124	LAB-100	84381893e99c5abba0ad786969515f75	eef8ea948fca582bdfc99de30f3c945	Fri Jun 25 21:07:05 2021	Fri Jun 25 21:07:05 2021	10123	LAB-99	eef8ea948fca582bdfc99de30f3c945	8d17e9daf1b28359c2ebfa5b082e6e7b	Fri Jun 25 21:06:05 2021	Fri Jun 25 21:06:05 2021	10122	LAB-98	8d17e9daf1b28359c2ebfa5b082e6e7b	b6726bee12aee1336cc05a168e917591	Fri Jun 25 21:05:05 2021	Fri Jun 25 21:05:05 2021	10121	LAB-97	b6726bee12aee1336cc05a168e917591	4cde5b11c05904a5c1699647fe9a1249	Fri Jun 25 21:04:05 2021	Fri Jun 25 21:04:05 2021	10120	LAB-96	4cde5b11c05904a5c1699647fe9a1249	bbc4b514f589a048da9ccb0232eae21	Fri Jun 25 21:03:05 2021	Fri Jun 25 21:03:05 2021	10119	LAB-95	bbc4b514f589a048da9ccb0232eae21	d847c86a2084b12121dfa969695e1c28	Fri Jun 25 21:02:05 2021	Fri Jun 25 21:02:05 2021	10118	LAB-94	d847c86a2084b12121dfa969695e1c28	60727858c049e599fdb68a3cd744a911	Fri Jun 25 20:27:06 2021	Fri Jun 25 21:01:05 2021	10100	LAB-76	60727858c049e599fdb68a3cd744a911	efa7c0a8fda81c01f8c66ef66095f805	Fri Jun 25 20:44:06 2021	Fri Jun 25 20:44:06 2021	10117	LAB-93	efa7c0a8fda81c01f8c66ef66095f805	c779331e48490bd764fbf47c229b3f6f	Fri Jun 25 20:43:06 2021	Fri Jun 25 20:43:06 2021	10116	LAB-92	c779331e48490bd764fbf47c229b3f6f	293c764bcfc7245dbdba6f5867f588c6	Fri Jun 25 20:42:07 2021	Fri Jun 25 20:42:07 2021	10115	LAB-91	293c764bcfc7245dbdba6f5867f588c6	05bec26bd32b98a35d26618cc8865997	Fri Jun 25 20:41:06 2021	Fri Jun 25 20:41:06 2021	10114	LAB-90	05bec26bd32b98a35d26618cc8865997	0d45869dad21e6c7efff984404670e36	Fri Jun 25 20:40:06 2021	Fri Jun 25 20:40:06 2021	10113	LAB-89	0d45869dad21e6c7efff984404670e36	bfe88ac695c82f1a0216de1faa03128c	Fri Jun 25 20:39:06 2021	Fri Jun 25 20:39:06 2021	10112	LAB-88	bfe88ac695c82f1a0216de1faa03128c	65b895abf547404a50d6acb2c39bda80	Fri Jun 25 20:38:06 2021	Fri Jun 25 20:38:06 2021	10111	LAB-87	65b895abf547404a50d6acb2c39bda80
key ↕	ctime ↕	mtime ↕	jira_id ↕	jira_key ↕	jira_md5 ↕																																																																																																
84381893e99c5abba0ad786969515f75	Fri Jun 25 21:08:05 2021	Sat Jun 26 09:06:07 2021	10124	LAB-100	84381893e99c5abba0ad786969515f75																																																																																																
eef8ea948fca582bdfc99de30f3c945	Fri Jun 25 21:07:05 2021	Fri Jun 25 21:07:05 2021	10123	LAB-99	eef8ea948fca582bdfc99de30f3c945																																																																																																
8d17e9daf1b28359c2ebfa5b082e6e7b	Fri Jun 25 21:06:05 2021	Fri Jun 25 21:06:05 2021	10122	LAB-98	8d17e9daf1b28359c2ebfa5b082e6e7b																																																																																																
b6726bee12aee1336cc05a168e917591	Fri Jun 25 21:05:05 2021	Fri Jun 25 21:05:05 2021	10121	LAB-97	b6726bee12aee1336cc05a168e917591																																																																																																
4cde5b11c05904a5c1699647fe9a1249	Fri Jun 25 21:04:05 2021	Fri Jun 25 21:04:05 2021	10120	LAB-96	4cde5b11c05904a5c1699647fe9a1249																																																																																																
bbc4b514f589a048da9ccb0232eae21	Fri Jun 25 21:03:05 2021	Fri Jun 25 21:03:05 2021	10119	LAB-95	bbc4b514f589a048da9ccb0232eae21																																																																																																
d847c86a2084b12121dfa969695e1c28	Fri Jun 25 21:02:05 2021	Fri Jun 25 21:02:05 2021	10118	LAB-94	d847c86a2084b12121dfa969695e1c28																																																																																																
60727858c049e599fdb68a3cd744a911	Fri Jun 25 20:27:06 2021	Fri Jun 25 21:01:05 2021	10100	LAB-76	60727858c049e599fdb68a3cd744a911																																																																																																
efa7c0a8fda81c01f8c66ef66095f805	Fri Jun 25 20:44:06 2021	Fri Jun 25 20:44:06 2021	10117	LAB-93	efa7c0a8fda81c01f8c66ef66095f805																																																																																																
c779331e48490bd764fbf47c229b3f6f	Fri Jun 25 20:43:06 2021	Fri Jun 25 20:43:06 2021	10116	LAB-92	c779331e48490bd764fbf47c229b3f6f																																																																																																
293c764bcfc7245dbdba6f5867f588c6	Fri Jun 25 20:42:07 2021	Fri Jun 25 20:42:07 2021	10115	LAB-91	293c764bcfc7245dbdba6f5867f588c6																																																																																																
05bec26bd32b98a35d26618cc8865997	Fri Jun 25 20:41:06 2021	Fri Jun 25 20:41:06 2021	10114	LAB-90	05bec26bd32b98a35d26618cc8865997																																																																																																
0d45869dad21e6c7efff984404670e36	Fri Jun 25 20:40:06 2021	Fri Jun 25 20:40:06 2021	10113	LAB-89	0d45869dad21e6c7efff984404670e36																																																																																																
bfe88ac695c82f1a0216de1faa03128c	Fri Jun 25 20:39:06 2021	Fri Jun 25 20:39:06 2021	10112	LAB-88	bfe88ac695c82f1a0216de1faa03128c																																																																																																
65b895abf547404a50d6acb2c39bda80	Fri Jun 25 20:38:06 2021	Fri Jun 25 20:38:06 2021	10111	LAB-87	65b895abf547404a50d6acb2c39bda80																																																																																																

The Addon UI shows as well that updates were performed rather than new issues creation:



The issue itself in JIRA shows new comments added everytime the alert triggered for the same content:

The screenshot shows the Jira Software interface. On the left is a sidebar with navigation options: LAB board, Backlog, Active sprints, Releases, Reports, Issues (selected), and Components. Below these are project shortcuts. The main area displays an issue titled 'Splunk Alert: Test JIRA - demo dedup' in the 'LAB' project. The issue is a 'Bug' with 'High' priority and 'None' labels. It is currently in the 'To Do' status. The description states: 'The alert condition for 'Test JIRA - demo dedup' was triggered.' followed by a list of details: user: foo@splunk.com, action: failure, reason: Authentication failed, and time: Fri Jun 25 21:08:00 2021. The activity section shows three comments from 'guilhem@octamis.com', each stating 'New alert triggered: Splunk Alert: Test JIRA - demo dedup'.

We can control the content of the comment added to the issue by creating a custom field in the resulting Splunk alert, let's modify the alert to include a new field used to control the comment:

```
| makeresults
| eval user="bar@splunk.com", action="failure", reason="Authentication failed"
| eval time=strftime(_time, "%c")
| eval jira_update_comment="The same condition was detected by Splunk for the user=" .
  user . " with action=" . action . " and reason=" . reason . ", therefore a new
  comment was added to the JIRA issue."
```

After the first issue creation, the next time the alert triggers, the Addon will use the content of the “jira_update_comment” field and use it in the comment field in JIRA:

Issue initially created:

The screenshot displays the Jira Software interface for a project named 'LAB'. The left sidebar contains navigation options: LAB board, Backlog, Active sprints, Releases, Reports, Issues (selected), and Components. Below these are project shortcuts and an 'Add link' button. The main content area shows the issue 'LAB-106' titled 'Splunk Alert: Test JIRA - demo dedup'. The issue is a 'Bug' with 'High' priority and 'None' labels. It is currently in the 'To Do' status and is 'Unresolved'. The description states: 'The alert condition for 'Test JIRA - demo dedup' was triggered.' with details: user: bar@splunk.com, action: failure, reason: Authentication failed. The activity section shows no comments yet.

Jira Software

Dashboards ▾ Projects ▾ Issues ▾ Boards ▾ Plans ▾ Create

LAB

LAB board ▾

Backlog

Active sprints

Releases

Reports

Issues

Components

PROJECT SHORTCUTS

Add a link to useful information for your whole team to see.

+ Add link

LAB / LAB-106

Splunk Alert: Test JIRA - demo dedup

Edit Comment Assign More ▾ To Do In Progress Done Admin ▾

Details

Type: Bug Status: **TO DO** (View Workfl

Priority: High Resolution: Unresolved

Labels: None

Description

The alert condition for 'Test JIRA - demo dedup' was triggered.

- user: bar@splunk.com
- action: failure
- reason: Authentication failed

Activity

All Comments Work Log History Activity

There are no comments yet on this issue.

Comment

Issue updated with our comment field:

Jira Software Dashboards ▾ Projects ▾ Issues ▾ Boards ▾ Plans ▾ **Create**

LAB

LAB board ▾

Backlog

Active sprints

Releases

Reports

Issues

Components

PROJECT SHORTCUTS

Add a link to useful information for your whole team to see.

+ Add link

LAB / LAB-106

Splunk Alert: Test JIRA - demo dedup

Edit Comment Assign More ▾ To Do In Progress Done Admin ▾

Details

Type: Bug Status: **TO DO** (View Workflows)

Priority: High Resolution: Unresolved

Labels: None

Description

The alert condition for 'Test JIRA - demo dedup' was triggered.

- user: bar@splunk.com
- action: failure
- reason: Authentication failed

Activity

All **Comments** Work Log History Activity

added a comment - 1 minute ago

The same condition was detected by Splunk for the user=bar@splunk.com with action=failure and reason=Authentication failed. A new comment was added to the JIRA issue.

added a comment - Just now

The same condition was detected by Splunk for the user=bar@splunk.com with action=failure and reason=Authentication failed. A new comment was added to the JIRA issue.

Comment

Now, let's say this issue is taken in charge in JIRA, its status is changed to Done as we think the underneath condition is fixed:

Jira Software Dashboards ▾ Projects ▾ Issues ▾ Boards ▾ Plans ▾ **Create**

LAB

LAB board ▾

Backlog

Active sprints

Releases

Reports

Issues

Components

PROJECT SHORTCUTS

Add a link to useful information for your whole team to see.

+ Add link

LAB / LAB-109

Splunk Alert: Test JIRA - demo dedup

Edit Comment Assign More ▾ To Do In Progress Done Admin ▾

Details

Type: Bug Status: **DONE** (View Workfl...

Priority: High Resolution: Done

Labels: None

Description

The alert condition for 'Test JIRA - demo dedup' was triggered.

- user: foo@splunk.com
- action: failure
- reason: Authentication failed

Activity

All Comments Work Log History Activity

added a comment - 2 minutes ago

The same condition was detected by Splunk for the user=foo@splunk.com with action=failure and reason=Auth... new comment was added to the JIRA issue.

added a comment - Just now

This issue is now fixed.

Comment

This is where the second dedup option acts, thanks to this bi-directional integration, the Addon knows that the issue was fixed and decides to open a new issue.

An INFO message is logged explaining why the Addon took this decision:

```
2021-06-26 09:42:06,237 INFO pid=13894 tid=MainThread file=cim_actions.py:message:243
→ | sendmodaction - signature="jira_dedup: The issue with key LAB-109 has the same
→ MD5 hash: 60727858c049e599fdb68a3cd744a911 and its status was set to: "Done"
→ (status category: "Done"), a new comment will not be added to an issue in this
→ status, therefore a new issue will be created." action_name="jira_service_desk"
→ search_name="Test JIRA - demo dedup" sid="scheduler_admin_search_
→ RMD526ad4cfa87997743_at_1624700520_67" rid="0" app="search" user="admin" action_
→ mode="saved" action_status="success"
```

If you have custom statuses, you can update the list of statuses to be taken into account in the alert definition, the Addon accepts a comma separated list of statuses.

Now, let's say that we need to have more information added into our JIRA ticket, some will not change if the

same alert triggers for the same condition, but others that we need such as the time field will always differ.

To achieve our goal, we will use the third option to “scope” what the Addon will use for the MD5 generation that is used to identify a duplicate issue, we will generate a specific field in the Splunk alert and recycle its value in the alert definition:

```
| makeresults
| eval user="foo@splunk.com", action="failure", reason="Authentication failed"
| eval time=strftime(_time, "%c")
| eval jira_update_comment="The same condition was detected by Splunk for the user=" .
↪ user . " with action=" . action . " and reason=" . reason . ", therefore a new_
↪ comment was added to the JIRA issue."
| eval dedup_condition = "user=" . user . "|action=" . action . "|reason=" . reason
```

Then, we modify our alert action to ask the Addon to use this token variable for the MD5 generation:

note: `$result.dedup_condition$` is how you will instruct Splunk to recycle dynamically the value of the field `dedup_condition` and pass it in the alert action.

Edit Alert

JIRA dedup
behaviour:

Enabled



When a new JIRA issue is created, the issue key reference and its unique md5 hash are stored in a KVstore.

If the jira dedup option is enabled and shall the same issue creation be requested again, a new comment will be added to the existing issue rather than a new issue created.

The md5 hash calculation guarantees the issue unique identification and is performed against the entire issue content. (unless jira_dedup_content is defined)

To define the content of the comment (defaults to: New alert triggered: <issue summary>), generate a field named "jira_update_comment" and this will be used as the comment automatically.

JIRA dedup
excluded status
categories

Done

Comma separated list of Jira status categories that will not be considered for updates, if dedup is enabled and the duplicated issue status is in one of these categories, a new issue will be created instead.

JIRA dedup
content

`$result.dedup_condition$`



(Optional) The default behavior is to use the full issue content to calculate the md5, by defining its content here you can limit its scope.

If this field is unset, the entire issue data is used for the deduping purposes.

Example:

Splunk Alert: `$name$`,

Cancel

We have now changed the way we identify what is a duplicate, and what is not, we can have fields which content will always change like our time field without breaking the dedup identification:

When the alert triggers more than once, we can see a new comment added to our issue:

The screenshot displays the Jira Software interface for a project named 'LAB'. The issue title is 'Splunk Alert: Test JIRA - demo dedup'. The issue is currently in the 'To Do' status. The description section contains the following text: 'The alert condition for 'Test JIRA - demo dedup' was triggered.' Below this, a list of details is provided: 'user: foo@splunk.com', 'action: failure', 'reason: Authentication failed', and 'time: Sat Jun 26 10:04:00 2021'. A red arrow points to the time field. The activity section shows a comment added by a user, stating: 'The same condition was detected by Splunk for the user=foo@splunk.com with action=failure and reason=Authen... new comment was added to the JIRA issue.'

The same workflow applies again, if we fix the issue the Addon will detect it and create a new ticket, if something happens to be different in the condition for the dedup identification, a new ticket will be created.

Powerful, isn't?!

Additional information about the KVstore knowledge records:

- **key** is the internal uuid of the KVstore, as well the key will be equal to the md5 hash of the first occurrence of JIRA issue created (next occurrences will have a key uuid generated automatically with no link with the md5 of the issue)
- **ctime** is the milliseconds epochtime that corresponds to the initial creation of the ticket, this value can not be changed once the record is created
- **mtime** is the milliseconds epochtime of the last modification of the record, if a comment is added to this ticket, this value corresponds to the time of that action

- **jira_md5** is the actual md5 hash for the entire JIRA issue, when the dedup option is activated for an alert, this will always be equal to the key id of the record in the KVstore
- **status** reflects the status of the issue as it is known from the add-on perspective, created means the issue was created, updated means at least one comment was made to this ticket due to dedup matching
- **jira_id** / **jira_key** / **jira_self** are JIRA information related to this ticket

JIRA Service Desk - Issues backlog collection

This report exposes the JIRA issues backlog which contains records for every JIRA issue created by the add-on, this collection is as well used by the JIRA add-on backend for revealing an update was performed via the JIRA deduplication feature.

Last 5 minutes ▾

✓ 85 results (20/06/2020 18:29:35.000 to 20/06/2020 18:34:35.000)

85 results 20 per page ▾

key ▾	ctime ▾	mtime ▾	jira_id ▾	jira_key ▾	jira_md5 ▾
04f50b829830cc0d851d55ef248770e8	Sat Jun 20 17:05:14 2020	Sat Jun 20 17:09:10 2020	98165	TEST-88166	04f50b829830cc0d851d55ef248770e8
077023cd2ccac73d916ccaf9ef8d850a	Sat Jun 20 15:05:13 2020	Sat Jun 20 15:09:10 2020	98118	TEST-88119	077023cd2ccac73d916ccaf9ef8d850a
089fdb253f6cb5bbff2b6dcfa46ab6b	Sat Jun 20 15:30:14 2020	Sat Jun 20 15:34:09 2020	98128	TEST-88129	089fdb253f6cb5bbff2b6dcfa46ab6b
1028211158441059eb15101f3b5f8213	Sat Jun 20 14:05:14 2020	Sat Jun 20 14:09:08 2020	98099	TEST-88100	1028211158441059eb15101f3b5f8213
10aa5543b8f655b93cbbc6204bb6b4df	Sat Jun 20 14:50:14 2020	Sat Jun 20 14:54:10 2020	98112	TEST-88113	10aa5543b8f655b93cbbc6204bb6b4df
1318a7ab02de00ff7f85c3f75fb5005f	Sat Jun 20 13:35:14 2020	Sat Jun 20 13:39:08 2020	98093	TEST-88094	1318a7ab02de00ff7f85c3f75fb5005f
144ac270b0a4312f6909a95a9eee6609	Sat Jun 20 14:20:19 2020	Sat Jun 20 14:24:08 2020	98102	TEST-88103	144ac270b0a4312f6909a95a9eee6609
161714bee151c34fbeb9ead7794753f	Sat Jun 20 17:15:11 2020	Sat Jun 20 17:19:09 2020	98170	TEST-88171	161714bee151c34fbeb9ead7794753f
162b0584ebe7ca581586a57a050fdc37	Sat Jun 20 15:40:14 2020	Sat Jun 20 15:44:09 2020	98132	TEST-88133	162b0584ebe7ca581586a57a050fdc37

3.1.13 JIRA attachment

Results

Enabled (JSON format) ▾

attachment: *

Enable this option to automatically attach the Splunk results as an attachment to the JIRA issue.

Attachments can be added in CSV or JSON format, however JIRA attachment preview only supports csv format.

Finally, the attachment action cannot be replayed by the resilient store, shall the JIRA creation temporary fail, the attachment will not be added if the issue creation had to be replayed due to a temporary issue.

On a per alert basis, the results from the Splunk alert that triggered can automatically be attached to the JIRA issue.

Features and limitations:

- The attachment feature is disabled by default, and needs to be enabled on a per alert basis
- The format of the results can be attached in CSV format, or JSON format
- JIRA file preview only supports the CSV format at the time of this writing

- The feature is not currently available if an HTTP proxy is used (a warning message will be emitted in logs, but the action will have not effects)
- The feature is not compatible with the resilient store, if the JIRA issue initially fails due to a temporary failure, the ticket will be created by the resilient tracker when possible but without the original attachment

When the attachment option is enabled, the following message will be logged if the attachment was successfully added to the JIRA issue, in addition with details of the ticket returned by JIRA:



```
JIRA Service Desk ticket attachment file uploaded successfully
```

File attachment in JIRA:

Note: the file name is dynamically generated, prefixed with “splunk_alert_results_” and suffixed by the relevant file extension.

Attachments

splunk_alert_results_j
dig4aed.json

 5 KB 

Activity

Show:

Comments

History

Work log

3.1.14 JIRA custom fields

JIRA custom field: (optional)

custom fields structure

✎

Optional, insert custom fields structure separated by commas, and its values: (<https://developer.atlassian.com/server/jira/platform/jira-rest-api-examples>)

The structure of the custom field value definition depends on its type which can be a text input, a dropdown or multiselect input, etc. See JIRA rest API documentation.

example:

```
"customfield_10048":
"$result.singleline_text$",
"customfield_10052": {"value":
"$result.single_choice$"},
"customfield_10053": [ {"value":
"$result.multi_choice_grp1$"}, {"value":
"$result.multi_choice_grp2$"} ]
```

JIRA custom fields are fields that can designed by your JIRA administrators to be available during the issue creation.

The Splunk Add-on for JIRA Service Desk supports any kind and any number of custom fields by allowing you to insert a custom field JSON structure in the alert configuration.

There are different types of custom fields, from a single ling text input to date and time pickers, which are described in the JIRA API documentation:

<https://developer.atlassian.com/server/jira/platform/jira-rest-api-examples>

CascadingSelectField

```
1 "customfield_10001": {"value": "green", "child": {"value": "blue"} }
```

The value associated with "name" ("green" in this example) is the parent option selected, then "blue" is the option).

DatePickerField

```
1 "customfield_10002": "2011-10-03"
```

The format is: `YYYY-MM-DD`

DateTimeField

```
1 "customfield_10003": "2011-10-19T10:29:29.908+1100"
```

This format is ISO 8601: `YYYY-MM-DDThh:mm:ss.sTZD`

FreeTextField

```
1 "customfield_10004": "Free text goes here.  Type away!"
```

GroupPicker

```
1 "customfield_10005": { "name": "jira-developers" }
```

Like users, groups are specified by name or ID.

MultiGroupPicker

```
1 "customfield_10007": [{ "name": "admins" }, { "name": "jira-developers" }, { "name":
```

Like users, groups are specified by name or ID.

MultiSelect

```
1 "customfield_10008": [ {"value": "red" }, {"value": "blue" }, {"value": "green" }]
```

MultiUserPicker

```
1 "customfield_10009": [ {"name": "charlie" }, {"name": "bioner" }, {"name": "tdurden"
```

Depending on the format of the custom field, you need to use the proper syntax, the most common are:

```
"customfield_10048": "$result.singleline_text$",
```

```
"customfield_10052": {"value": "$result.single_choice$"},
```

```
"customfield_10053": [ {"value": "$result.multi_choice_grp1$" }, {"value": "$result.  
↪multi_choice_grp2" }]
```

As usual, while you define the custom fields, you can use dynamic results from the Splunk search results by using the syntax: `$result.myfield$`

To add a list of custom fields, make sure you add a comma after each custom field, and none at the end of the JSON structure.

A full example JSON structure is provided in the alert action screen:

```
"customfield_10048": "$result.singleline_text$",  
"customfield_10052": {"value": "$result.single_choice$"},  
"customfield_10053": [ {"value": "$result.multi_choice_grp1$" }, {"value": "$result.  
↪multi_choice_grp2" }]
```

Custom fields parsing:

By default, the content of the custom fields is parsed to escape and protect any special characters that would potentially lead the JSON data not to be parsed properly.

In some circumstances, the built-in parser rules may fail to recognize an unexpected custom fields structure, the parsing can be disabled if required:

Custom fields
parsing:

Enabled



This option performs parsing of the custom fields to prevent failures that would be due to special characters in the content that would need to be protected.

However, in some cases where the sequence is not expected by the parsing function, this can lead to a json parsing failure of the custom fields structure.

When the option is set to disabled, no parsing of the custom fields will be performed and the alert needs to handle properly any special character that would be need to be escaped, as a best practice you would avoid any special characters.

How to retrieve the IDs of the custom fields configured ?

Use the built-in report and associate custom command to retrieve the list of JIRA fields information:

Overview - JIRA Service Desk Search Get JIRA info Logging reports Builtin alerts Alerts Configuration Run a search

JIRA Service Desk

This report exposes JIRA fields

Last 5 minutes

✓ 22 events (13/04/2020 06:19:2)

20 per page

i	Time	Event
>	13/04/2020 06:24:26.160	<pre>{ [-] expand: projects projects: [[+]] }</pre> Show as raw text
>	13/04/2020 06:24:26.360	<pre>{ [-] expand: projects projects: [[+]] }</pre> Show as raw text

This report achieves a REST call to JIRA to get the list of fields and their details per project and per type of issues, search for custom fields:

Select Fields

Select All Within Filter Deselect All Coverage: 1% or more ▼ customfield ✕

i	✓ ▼	Field ⇅
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10002.autoCompleteUrl
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10002.hasDefaultValue
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10002.key
✓	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10002.name
Reports Top values Events with this field Top values by time Rare values		
Organizations 8		
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10002.operations()
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10002.required
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10002.schema.custom
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10002.schema.customId
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10002.schema.items
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10002.schema.type
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10003.autoCompleteUrl
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10003.hasDefaultValue
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10003.key
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10003.name
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10003.operations()
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10003.required
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10003.schema.custom
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10003.schema.customId
>	<input type="checkbox"/>	projects().issuetypes().fields.customfield_10003.schema.items

3.1.15 JIRA REST API wrapper

A custom command is provided as a generic API wrapper which can be used to get information from JIRA by calling any REST endpoint available: By default, it uses method GET. Additional methods are supported DELETE, POST, PUT.

```
| jirarest target="<endpoint>"
```

Open the REST API dashboard to get examples of usage:

Use the custom command `| jirarest target="<endpoint>"` to perform a get call against any endpoint of your JIRA instance

For API references:

[Jira Server platform REST API reference](#)

Try your own:

rest/api/2/myself

i	Time	Event
>	19/09/2020 11:39:42.490	<pre> { [-] accountId: [REDACTED] active: true applicationRoles: { [+] } avatarUrls: { [+] } displayName: [REDACTED] emailAddress: [REDACTED] expand: groups,applicationRoles groups: { [+] } locale: en_US self: [REDACTED] timeZone: Europe/London } </pre>

Use REST and JQL to get the total number of issues per project, per status category and calculate percentages in each command):

You can use the JQL language and perform any advanced query in JIRA, the following example returns the number of issues per project: `api/2/search`

_time ↕	project ↕	pct_total_done ↕	pct_total_in_progress ↕	pct_total_to_do ↕
2020-09-19 11:39:40	SPLUNK	% 0.00	% 0.00	% 0.00
2020-09-19 11:39:40	TEST	% 0.01	% 0.00	% 99.99

The following report is provided to retrieve issues statistics per project and per status categories:

JIRA Service Desk - Issues statistics report per project

Overview - JIRA Service Desk					Search ▼	Get JIRA info ▼	REST API explore	Logging reports ▼	Statistic JIRA reports ▼	Built-in alerts ▼	KVstore
JIRA Service Desk - Issues statistics report per project											
This report exposes JIRA issues statistics per project, you can use this report with the collect or mcollect command for indexing purposes											
Last 5 minutes ▼											
✓ 1 event (19/09/2020 11:39:31.000 to 19/09/2020 11:44:31.000)											
2 results 20 per page ▼											
_time ↕		project ↕	pct_total_done ↕		pct_total_in_progress ↕		pct_total_to_do ↕				
2020-09-19 10:44:31		SPLUNK	0.00		0.00		0.00				
2020-09-19 10:44:31		TEST	0.01		0.00		99.99				

Indexing JIRA statistics for reporting purposes

If you wish to index the JIRA statistic results in Splunk for reporting purposes over time, you can easily modify or clone this report to use collect or mcollect to index these statistics:

Indexing the results to a summary report

You can use the `collect` command to automatically index the report results in a summary index of your choice, schedule this report and add a call to collect, example:

```
| collect index=summary source="JIRA - issues stats per project"
```

Overview - JIRA Service Desk Search Get JIRA info REST API explore Logging reports Statistic JIRA reports Builtin alerts KVstore

New Search

index=summary source="jira - issues stats per project"

✓ 2 events (18/09/2020 11:00:00.000 to 19/09/2020 11:51:26.000) No Event Sampling ▼

Events (2) Patterns Statistics Visualization

Format Timeline ▼ — Zoom Out + Zoom to Selection × Deselect

List ▼ Format 20 Per Page ▼

< Hide Fields	≡ All Fields	i	Time	Event
SELECTED FIELDS a host 1 a project 2 a source 1 a sourcetype 1		>	19/09/2020 11:49:54.000	09/19/2020 11:49:54 +0100, info_min_time=1600512294.000, info_max_time=1600512594.000, info_="0.00", pct_total_done="0.01", total_to_do=92845, pct_total_to_do="99.99", total_issues=92845, host = ip-10-0-0-75 project = TEST source = JIRA - issues stats per project sourcetype = stash
INTERESTING FIELDS # date_hour 1 # date_mday 1 # date_minute 1 a date_month 1 # date_second 1 a date_wday 1		>	19/09/2020 11:49:54.000	09/19/2020 11:49:54 +0100, info_min_time=1600512294.000, info_max_time=1600512594.000, info_="0.00", pct_total_done="0.00", total_to_do=0, pct_total_to_do="0.00", total_issues=0, host = ip-10-0-0-75 project = SPLUNK source = JIRA - issues stats per project sourcetype = stas

Indexing the results to a metric index

Another option is to use the `mcollect` command to automatically index these statistics as native metrics in a metric index of your choice, the following example assumes a metric index named “jira_metrics” was created, the report scheduled and the following `mcollect` command is added:

```
| eval type="jira_" | mcollect split=t prefix_field=type index=jira_metrics project
```

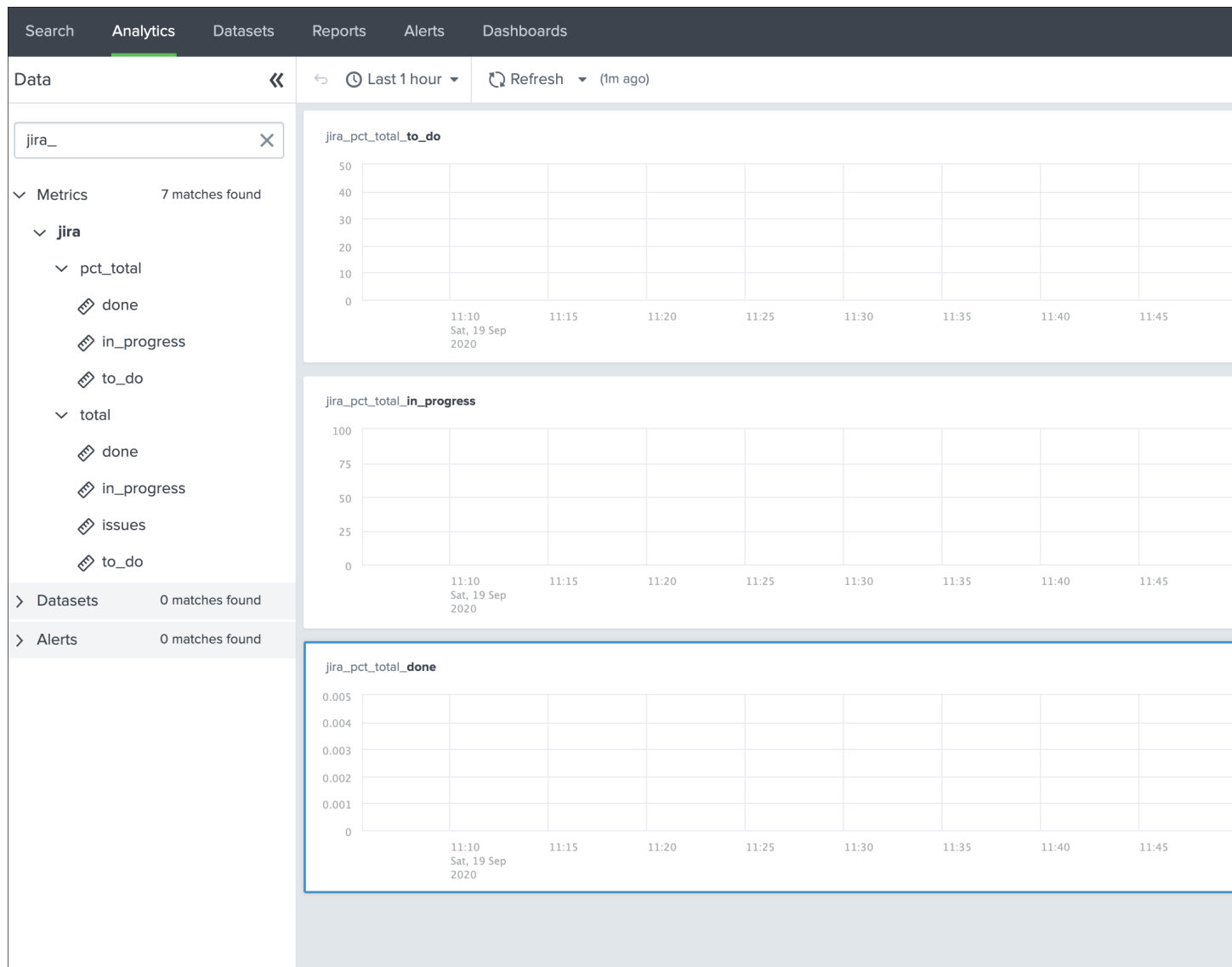
Each statistic is stored as a `metric_name` with a prefix “jira_”, while the project is stored as a dimension, you can use the `mcatalog` and `mstats` commands to use the metrics, or use the Analytics view in Splunk:

mcatalog example:

```
| mcatalog values(metric_name) values(_dims) where index=jira_metrics metric_
↪name=jira_*
```

mstats example:

```
| mstats latest(jira_pct_total_done) as pct_total_done, latest(jira_pct_total_in_
↪progress) as pct_total_in_progress, latest(jira_pct_total_to_do) as pct_total_to_do,
↪where index=jira_metrics by project span=5m
```



Additional examples for JIRA API wrapper

Method DELETE: Delete an issue

```
| jirarest target="rest/api/2/issue/{issueIdOrKey}" method=DELETE
```

Method POST: Add a comment to an issue

Example 1:

```
| jirarest target="rest/api/2/issue/{issueIdOrKey}/comment" method=POST json_request="
↳ {"body": "This is a normal comment.\""}"
```

Example 2:

```
| jirarest target="rest/api/2/issue/{issueIdOrKey}/comment" method=POST json_request="
↳{"body": "This is a comment that only administrators can see.", "visibility":
↳{"type": "role", "value": "Administrators"}}
```

Method PUT: Assign an issue

```
| jirarest target="rest/api/2/issue/{issueIdOrKey}/assignee" method=PUT json_request="
↳{"name": "harry"}
```


4.1 Trouble shooting

4.1.1 Connectivity to JIRA issues

If the connectivity to JIRA is not valid for some reasons (bad credentials, network connectivity, etc), this will result in different Python error messages when attempting to load any of the report, load the alert action page or execute an alert action.

In such as case, the easiest way is to validate the connectivity by achieving a rest cal using the curl command in CLI, ideally in any of the search head supposed to be using the alert action. (note: this step is valid for Linux only)

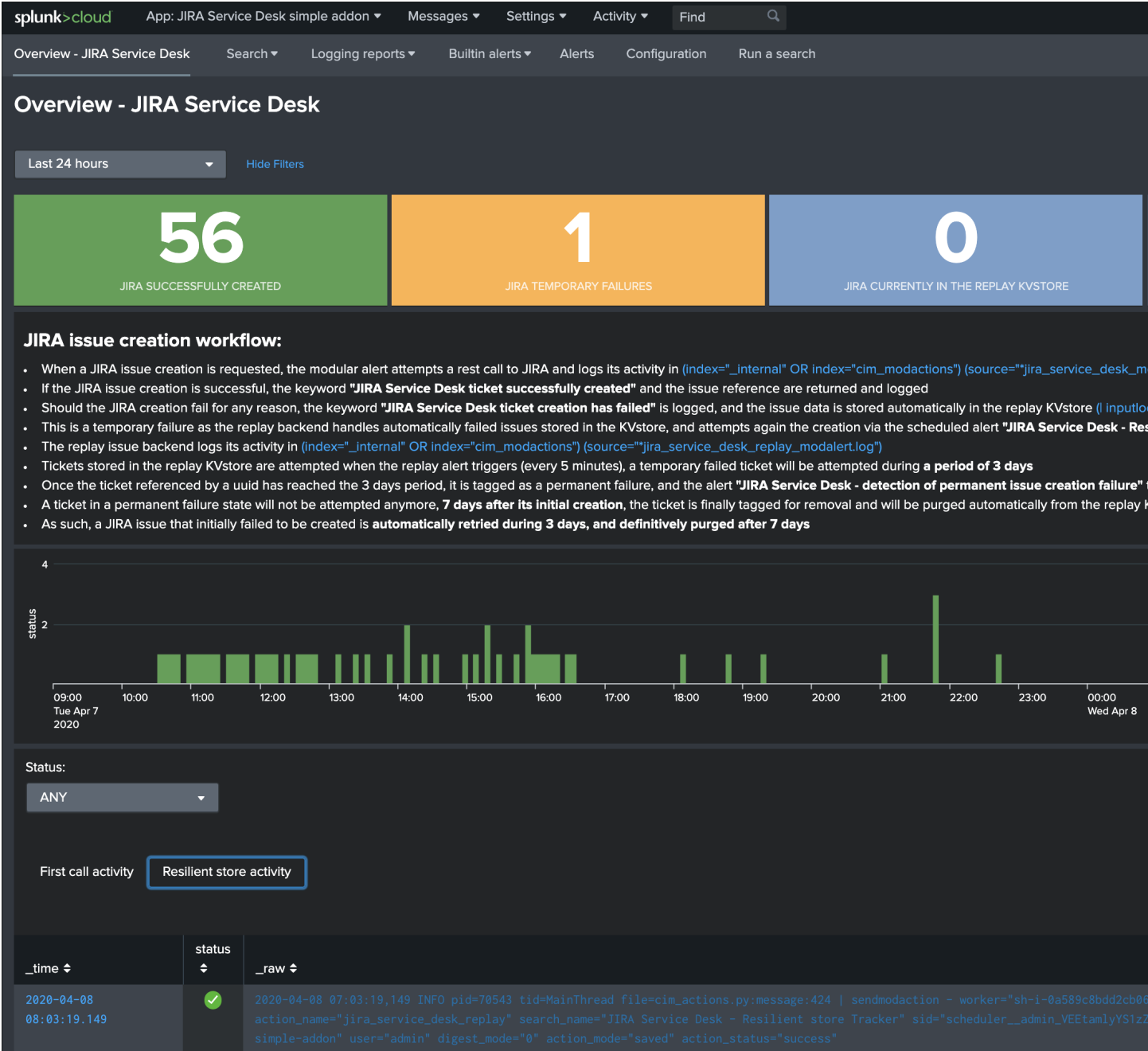
```
curl -k https://<jira_url>/rest/api/latest/project --user <jira_username>:<jira_
↪password>
```

For more information, follow these links:

- <https://developer.atlassian.com/server/jira/platform/basic-authentication>
- <https://developer.atlassian.com/cloud/jira/service-desk/basic-auth-for-rest-apis>

4.1.2 Overview dashboard and Add-on logs

The Splunk Add-on for JIRA Service Desk provides a builtin Overview dashboard that gives deep insights on the Add-on activity:



The dashboard exposes the JIRA issue workflow and direct links to access the Add-on logs.

Add-on logs for first REST call attempts

When the alert action is triggered, the Add-on records its activity in:

```
(index="_internal" OR index="cim_modactions") (source="*jira_service_desk_modalert.log")
```

When the JIRA issue is successfully achieved, the key sentence JIRA Service Desk ticket successfully created is logged.

If an error is encountered during the API call, the key sentence JIRA Service Desk ticket creation has failed is logged.

When the failure step is reached, for example if there is an issue with the credentials or reaching the JIRA instance, the workflow records the failure in a resilient store based on a KVstore lookup:

```
| inputlookup jira_failures_replay | eval uuid=_key
```

At this point, any failed call recorded in the KVstore is automatically re-attempted by the scheduled alert named: JIRA Service Desk – Resilient store Tracker

An out of box alert named JIRA Service Desk – detection of temporary issue creation failure is provided to monitor and track any JIRA failure, **the alert is by default enabled**.

Add-on logs for the resilient store feature

The resilient store feature tracks its activity in:

```
(index="_internal" OR index="cim_modactions") (source="*jira_service_desk_replay_
↪modalert.log")
```

In normal circumstances, which means there have not been recent failed attempts, there would be no activity in this logs, nor content in the KVstore.

If a record exists in the KVstore, the Add-on will re-attempt the creation every 5 minutes during 3 days per record, if it continuously failed durant that period, a key sentence permanent failure! is logged.

An out of box alert named JIRA Service Desk – detection of permanent issue creation failure is provided to monitor and track permanent JIRA failures, **the alert is by default enabled**.

After 7 days in the KVstore, a record is automatically and definitively purged.

Root cause for failures

Root causes of failures will be clearly exposes in the Add-on logs, most common causes could be:

- JIRA credential issues (verify the connectivity, see the configuration page)
- Networking issues or JIRA instance not reachable
- Content issues such as JIRA fields not available on the JIRA project (make sure these fields are associated with the right JIRA screens)
- Content issues such as JIRA field receiving an unexpected content or format (some JIRA fields such as date and date time inputs require a valid format, etc)

Shall a REST call for JIRA issue creation fail, the Add-on automatically logs full JSON data which you can use to easily review the data and trouble shoot the root causes.

Versions and build history:

5.1 Release notes

5.1.1 Version 1.0.29

- Enhancement: jirarest supports additional method for extended JIRA integration #85 (Author: Rémi Séguy)

5.1.2 Version 1.0.28

- Change: Issue #83 - Python Upgrade Readiness App complains about 'outdated Python SDK'

5.1.3 Version 1.0.27

- Fix: Issue #77 - Error reported in logs when the issue MD5 is equal, the alert continues to trigger and dedup is disabled

5.1.4 Version 1.0.26

- Feature: Issue #72 - Provides a new mode called passthrough mode, which is designed for scenarios where Splunk cannot contact the JIRA instance directly for security or restrictions purposes (such as Splunk Cloud potentially). A second Splunk instance that can connect to JIRA instance would recycle the replay KVstore content to perform the final call.
- Enhancement: Issue #73 - Provides custom search auto description (searchbnf.conf)

5.1.5 Version 1.0.25

- Change: Issue #70 - Splunk Python SDK upgrade to 1.6.15

5.1.6 Version 1.0.24

- Feature: Issue #65 - Allows defining the JIRA Issue reporter

5.1.7 Version 1.0.23

- Fix: Issue #61 - Custom commands now require Python3 mode explicitly which with AoB py3 SDK version causes error messages on the indexers #61

5.1.8 Version 1.0.22

- Fix: For Splunk Cloud vetting purposes, commands.conf needs to specify python3 explicitly

5.1.9 Version 1.0.21

- Fix: Issue #54 - Appinspect failure due to missing key in spec file
- Fix: Issue #55 - Appinspect failure in reports using the jirarest command due to checks attempting to run the reports in non JIRA connected environments, causing the map command to return an error
- Feature: Issue #56 - New Overview JIRA analytic view relying on the new jirarest command that allows live REST calls to JIRA and execution of JQL queries #56

5.1.10 Version 1.0.20

- Fix: Issue #50 - Deduplication Creating One Duplicate After Item Closed #50

5.1.11 Version 1.0.19

- Feature: Issue #33 - Exclude closed statuses from the JIRA dedup behavior, to prevent deduplicating closed issues, which list can be customised if required (defaults to Closed,Completed,Canceled)
- Feature: Issue #34 - Provides granular control against the content to be taken into account for dedup behavior and the md5 calculation used to identify duplicated tickets
- Feature: Provide a new REST API custom command wrapper to allow performing any get call against any endpoint of the JIRA API, provides a building issue statistic report that can be used with collect/mcollect to index issues statistics, provide a new dashboard exposing the wrapper usage
- Feature: Jira get field report split into two reports, one for all projects, one report providing results per project
- Fix: Issue #41 - Incident Review Manual AR Issue #41
- Fix: default.meta does not define permissions for the builtin jira_admin role for the JIRA issue backlog collection used for the dedup feature
- Change: Issue #42 - Removing Priority as a Required Input #42
- Change: Improved rendering of options and clearness for required inputs in the alert definition
- Change: Issue #16 - Deprecation of jiragetfields custom command, which is replaced with calls to the new REST wrapper jirarest

5.1.12 Version 1.0.18

- Fix: ensure aob configuration replicates in shc environment

5.1.13 Version 1.0.17

- feature: Enable / Disable custom fields structure parsing new alert option, disabling the custom fields parsing can be useful when the backend fails to parse properly a custom fields structure that is not expected

5.1.14 Version 1.0.16

- fix: Splunk Cloud vetting refused due to a remaining https protocol check in jiragetfields.py, checking if the URI contains https rather than starts with https

5.1.15 Version 1.0.15

- fix: Splunk Cloud vetting refused due to https protocol verification checking if the URI contains https rather than starts with https
- fix: JIRA dedup feature might under some systems be generating a different hash for the same issue due to a different order of the json data after json load operation in Python, perform the md5 calculation before calling json load

5.1.16 Version 1.0.14

- fix: remove the automatic addition of the result link in the description field as it systematically creates a different JIRA content, which creates confusion with the dedup JIRA option
- fix: change in configuration app the sentence “JIRA token password” to “JIRA password” to avoid confusion between basic authentication and OAuth2 which isn’t used by the Add-on
- fix: in some custom configuration, the custom command jiragetfields would not return the expected results, the type of issue is removed from the rest call to retrieve all fields information on a per project basis instead

5.1.17 Version 1.0.12

- Feature: Issue #18 - New option on a per alert basis allows automatically attaching Splunk alert results to the JIRA issue in CSV or JSON format
- Feature: Issue #18 - Add by default in the description field the result link token call

5.1.18 Version 1.0.11

- Feature: Issue #12 - New JIRA deduplication feature workflow allows handling automatically on a per alert basis updating JIRA issues by the addition of a comment (that can be controlled) to the original issue, instead of creating duplicated JIRA issues
- Feature: Issue #15 - Adding support for components definition on a per alert basis, components can now be defined by their name in a comma separated format within alerts
- Feature: Upgrade of Jinja2 2.11.2 libraries to address vulnerabilities reported during Splunk Cloud app vetting process

- Feature: Upgrade of PyYAML 5.3.1 libraries to address vulnerabilities reported during Splunk Cloud app vetting process
- Feature: Upgrade of httpLib2-0.18.1 libraries to address vulnerabilities reported during Splunk Cloud app vetting process
- Feature: Upgrade of urllib3-1.25.9 libraries to address vulnerabilities reported during Splunk Cloud app vetting process

5.1.19 Version 1.0.10

- Fix: Issue #9 - Parsing failure in custom field section with non standard fields in between square brackets

5.1.20 Version 1.0.9

- Fix: Issue #11 - SSL verification disablement is not honoured properly and remains active even if the checkbox is not checked
- Change: app.manifest schema upgrade to 2.0.0 to ease Cloud automated deployments

5.1.21 Version 1.0.8

- Fix: Allows defining non custom fields in the custom section, such as builtin non standard fields (Components) that would have been made required by JIRA admins

5.1.22 Version 1.0.7

- Fix: Default timed out value during REST calls are too short and might lead to false positive failures and duplicated creation of JIRA issues

5.1.23 Version 1.0.6

- Change: For Splunk Cloud vetting purposes, explicit Python3 mode in restmap.conf handler

5.1.24 Version 1.0.5

- Fix: Provide an embedded role jira_alert_action that can be inherited for non admin users to be allowed to fire the action and work with the resilient store feature

5.1.25 Version 1.0.4

- Feature: resilient store improvements, catch all failures and exceptions during issue creation attempts
- Fix: minor fix in resilient store table
- Fix: remove redundant alert link in nav bar

5.1.26 Version 1.0.3

- Fix Issue #2: Avoids error messages on indexers in distributed mode to report error messages on jirafill and jiragetfields custom commands due to enabled distributed mode
- Fix Issue #2: Avoids error messages reported during execution of jirafill and jiragetfields custom commands related to insecure HTTP calls with urllib3

5.1.27 Version 1.0.2

- Feature: Support for Web Proxy
- Feature: Full support for Python 3 (migration to newer Add-on builder libs, embedded custom commands)
- Fix: Support defining the JIRA instance URL with or without [https://](#)
- Fix: Potential creation failure with number type custom fields
- Fix: Metadata avoid sharing alerts, reports and views at global level
- Fix: Help block appears right shifted within Enterprise Security correlation search editor, but centered properly in Splunk core alert editor

5.1.28 Version 1.0.1

- unpublished

5.1.29 Version 1.0.0

- initial and first public release