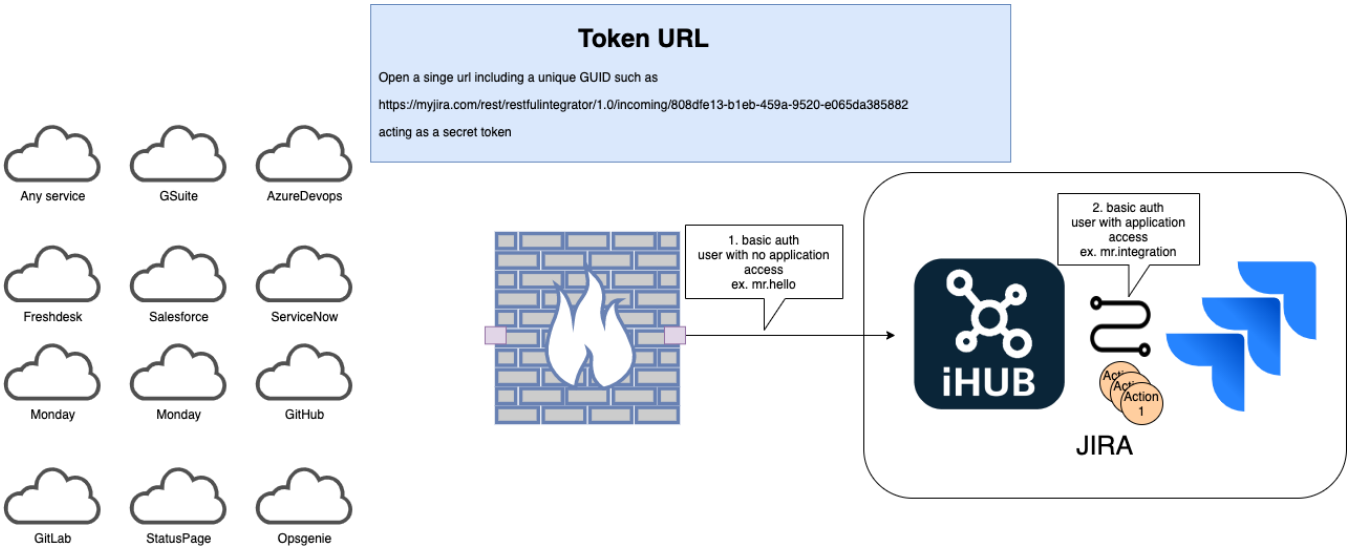
Build Integrations behind Firewall



The problem

Today many JIRA servers are located inside the companies firewall, which makes it hard to do two way integrations. In many cases you don´t want to expose the whole JIRA instance to the internet. Since JIRA default only handles Basic Authentications and might be seen as to unsecure.

This causes integrations to only be setup using a one way flow. There is normally a huge value by doing this, but the value can be even higher if allowing the other system to talk back to JIRA. This article shows how you can do this in a secured way.

The solution

iHUB Incoming rules! By utilizing the incoming rule engine in iHUB you will be able to setup a specific URL that the internet can access, and that URL can be secured by first of all using token urls. That is basically a url constructed using a GUI or Token.

The Token URL

A token URL can be formatted in many ways examples:

- https://myjira.com/rest/restfulintegrator/1.0/incoming/808dfe13-b1eb-459a-9520-e065da385882
- https://myjira.com/rest/restfulintegrator/1.0/incoming?token=808dfe16543ewsxcyuftdtr4567865da385882

It is essentially a unique URL that is super hard to guess or to find out.

The firewall

On the firewall, add a rule that only allows this URL to be accessed from the outside, so whenever a service send a request it goes to that URL.

That URL can then be connected to one or more iHUB incoming rule(s).

The Rule Engine user

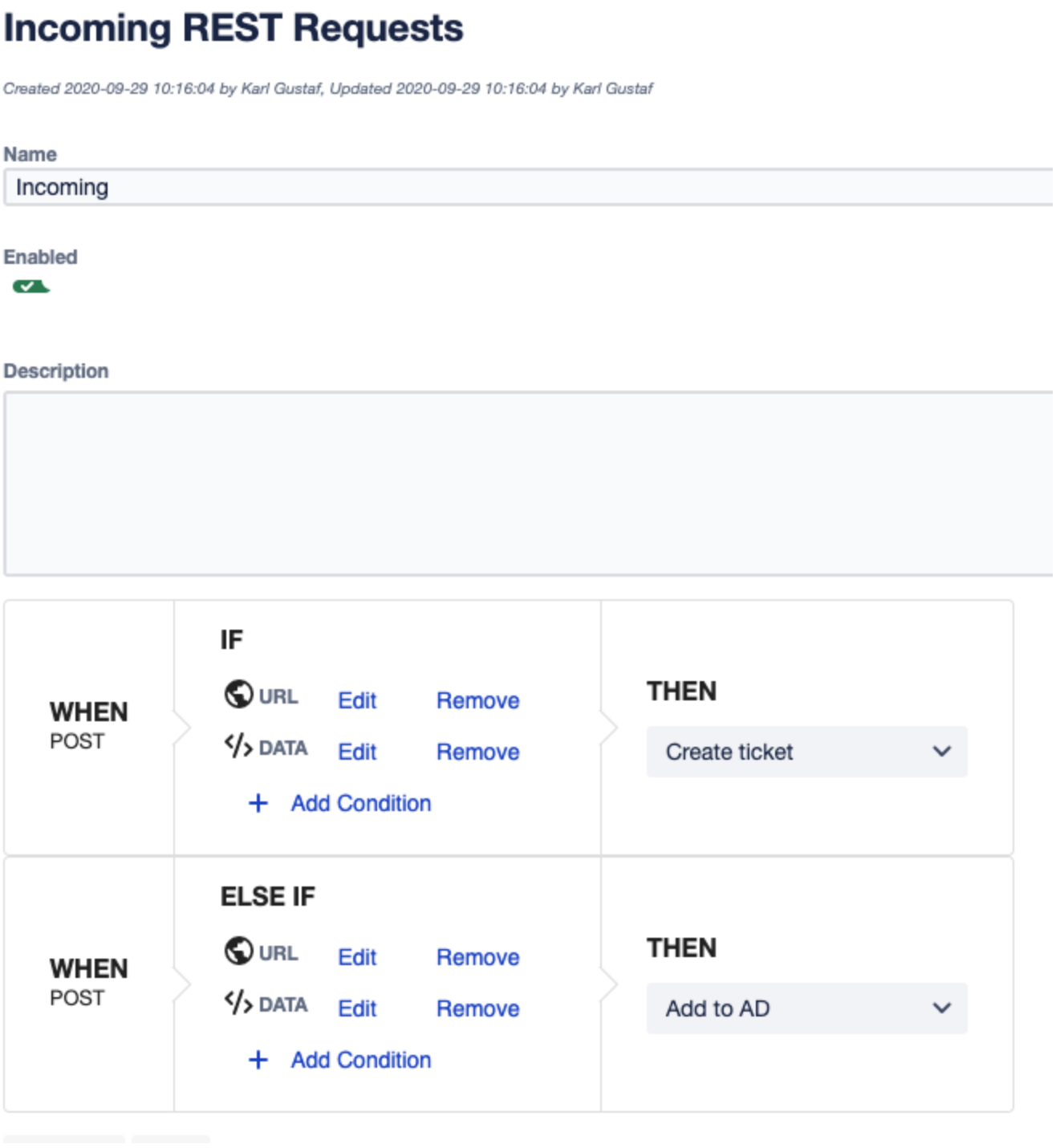
Second part of the security is that you setup a user with no Application Access in JIRA, meaning that user can never directly do anything in JIRA. Let´s call this mr.hello, that user in JIRA does not require any license either.

mr.hello role is to simply login to iHUBs incoming rule and pass the data to the rule engine.

The iHUB Incoming rules

Once the call has passed the firewall and login the rule engine takes over and validates on any of these three things;

1. Correct incoming URL is called mapped to any rule - Always used for security reasons but not required for internal services.
2. Correct user calls the rule
3. Data condition - does the rule has the right data



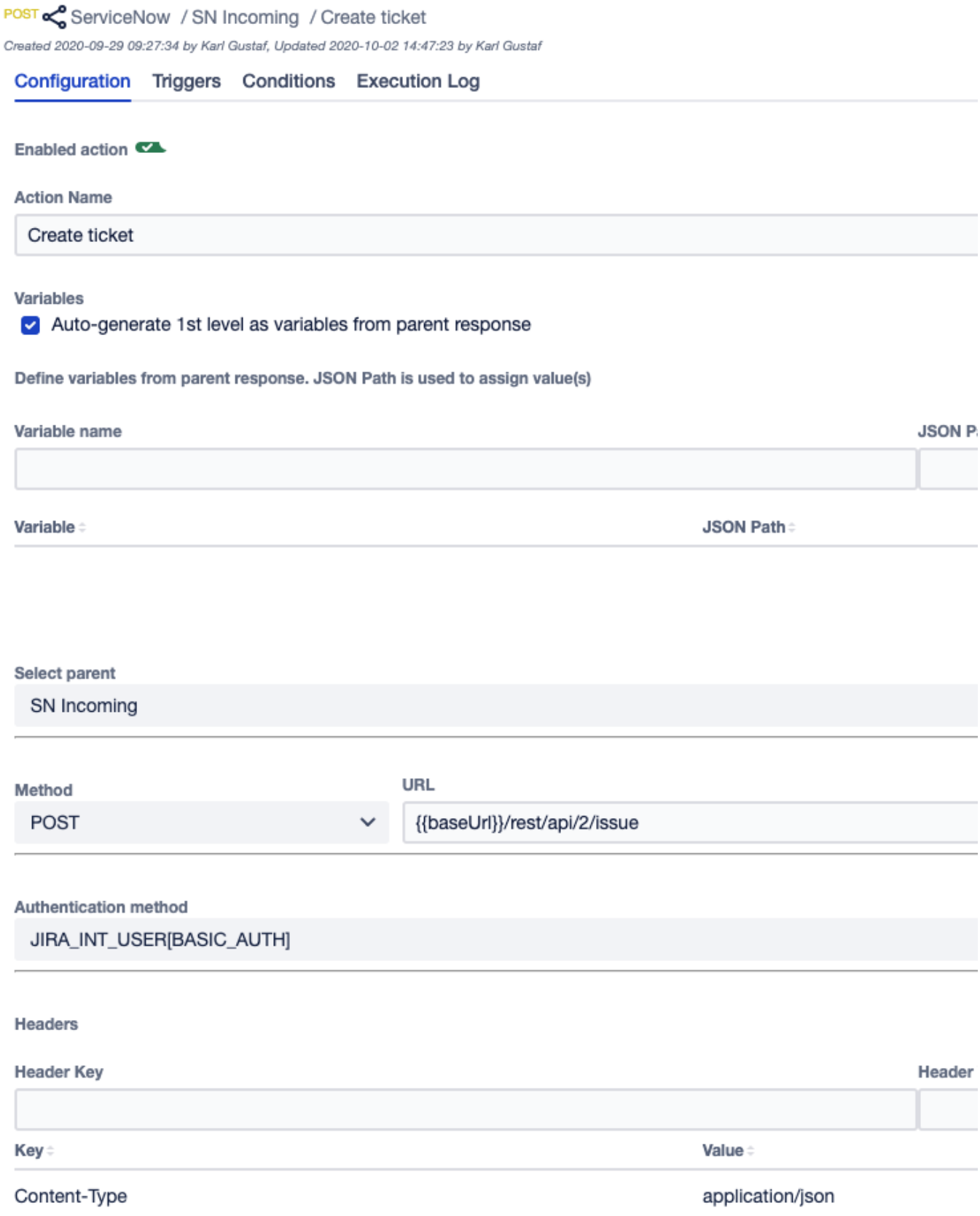
The rule engine maps the request to an action

The Action

The action is the actual work that needs to be done. It will call any REST service, but it most cases it will call the JIRA REST API to perform updates.

So for example an Incident might be triggered in ServiceNow and sent through to an action, the action then takes the data and map that to a JIRA issue and creates that issue.

Here is the second authentication, where the triggered call is always run by a action user, in this example below the JIRA_INT_USER performs the call towards Jira and has the Application Access and roles needed in the project(s) that this integration relates to.


POST ServiceNow / SN Incoming / Create ticket
Created 2020-09-29 09:27:34 by Karl Gustaf, Updated 2020-10-02 14:47:23 by Karl Gustaf
Configuration
Triggers
Conditions
Execution Log
Enabled action
Action Name
Create ticket
Variables
Auto-generate 1st level as variables from parent response
Define variables from parent response. JSON Path is used to assign value(s)
Variable name
JSON P
Variable
JSON Path
Select parent
SN Incoming
Method
POST
URL
{{baseUrl}}/rest/api/2/issue
Authentication method
JIRA_INT_USER[BASIC_AUTH]
Headers
Header Key
Header
Key
Value
Content-Type
application/json

Summary

To summarize, open a single url in the firewall that only allow traffic to iHUBs rule engine. Create a user that can login to the rule engine and trigger actions based on the conditions. Setup an action that will be triggered by a rule. Let the action do the integration work mapping data and creating tickets.