ADOBE CAMPAIGN

Integration with Slack – Custom Activity

Abstract

This custom activity enables Adobe Campaign Classic to send real-time Slack alerts, reducing reliance on traditional email notifications. It supports personalized payloads and flexible routing through external account configuration

David Garcia
martech.network

Introduction

Adobe Campaign is a powerful orchestration platform, but it lacks a native way to send real-time alerts to third-party collaboration tools. One of the most widely used tools in the industry is Slack, a platform designed for instant communication, team collaboration, and streamlined workflows across channels.

Typically, Adobe Campaign relies on email for operational notifications. However, this approach can quickly become overwhelming as most solutions generate hundreds of automated email alerts, making it easy for critical messages to get lost or ignored.

This “Proof of concept” was created to solve that problem. By building a Custom Slack Activity for Adobe Campaign, I offer a modern alternative that delivers alerts directly to Slack channels, right where teams are already working. This approach helps reduce email clutter and ensures that important updates are seen and acted on in real time.

Prerequisites

I. **Slack App**
To interact with Slack, it is mandatory to create and configure a simple Slack app bot through the Slack API portal, it only takes 5 minutes. For a step-by-step guide, refer to the following YouTube setup video, ensure the bot has both **chat:write** and **chat:write.public** scopes enabled, usually this setup should be completed by your Slack workspace administrator or you can create a free slack workspace to get you started.

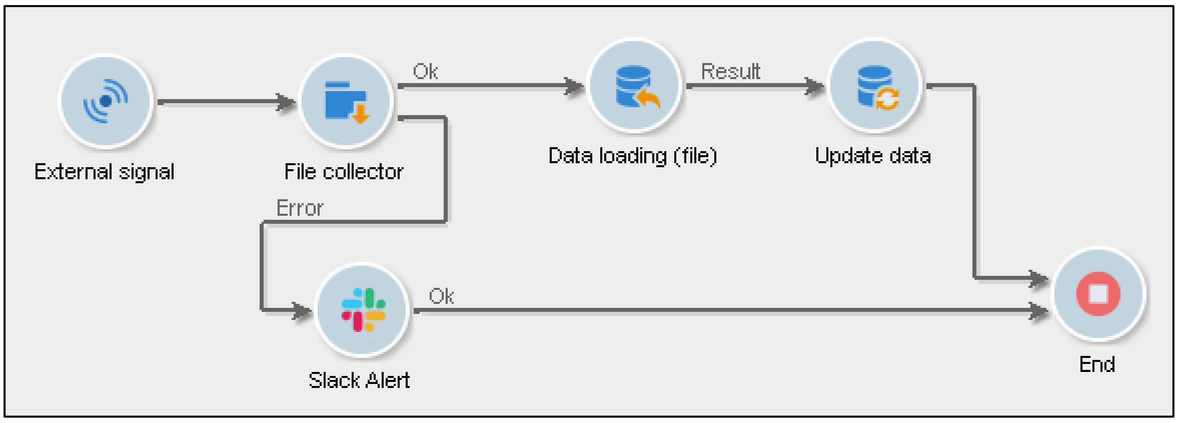
II. **Adobe Campaign**
Ensure you have development or administrative access to Adobe Campaign, as you’ll need the ability to create and modify key components such as schemas, input forms, and other critical assets required for integration.

III. **Postman (or others)**
Before implementing your custom JSON templates or integrating business logic into Adobe Campaign, it's essential to test your payloads using an API tool like Postman. This ensures your requests are well-structured and Slack-ready before deploying them in production. If you're familiar with API testing, you'll feel right at home here.

Overview of the Custom Activity

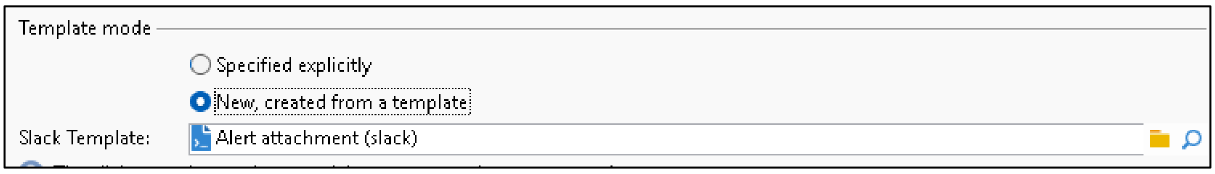
This custom activity provides a real-time alerting alternative to the traditional email based approach, alerts are routed directly into Slack where teams are more likely to see and act on them promptly.

While this tutorial focuses on integrating with Slack, the same development initiative can be employed with other platforms with API support. Tools like, Teams, Discord, Nagios, Prometheus, or Splunk are all within reach, making the solution versatile and future friendly.



Features

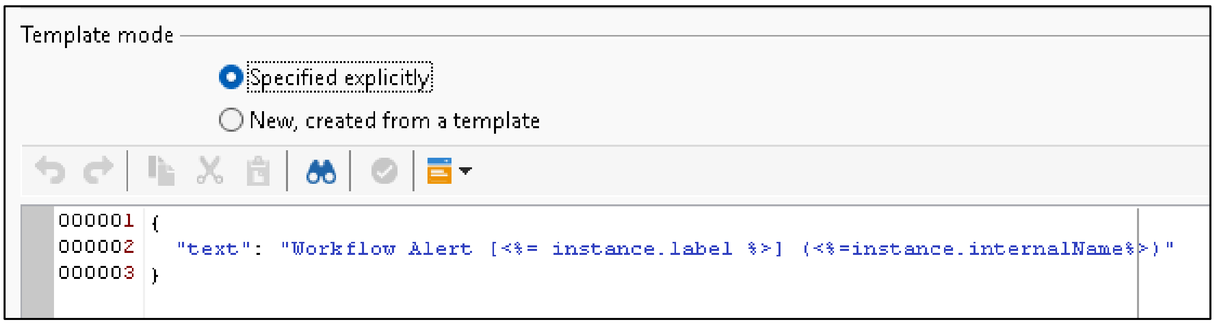
- **Templates or explicit** – You can define the message payload directly using inline JSON, or reference a reusable template stored in a JavaScript schema. To help you get started, a few sample templates will be provided.



Template mode

Specified explicitly [✓ New, created from a template]

Slack Template: Alert attachment (slack)

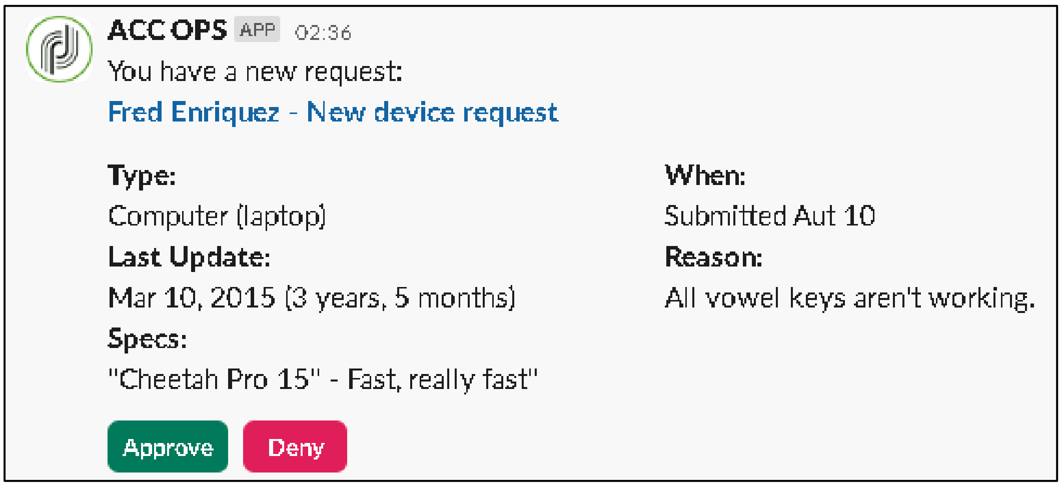


Template mode

[✓ Specified explicitly] New, created from a template

```
{
  "text": "Workflow Alert [<%= instance.label %>] (<%=instance.internalName%>)"
}
```

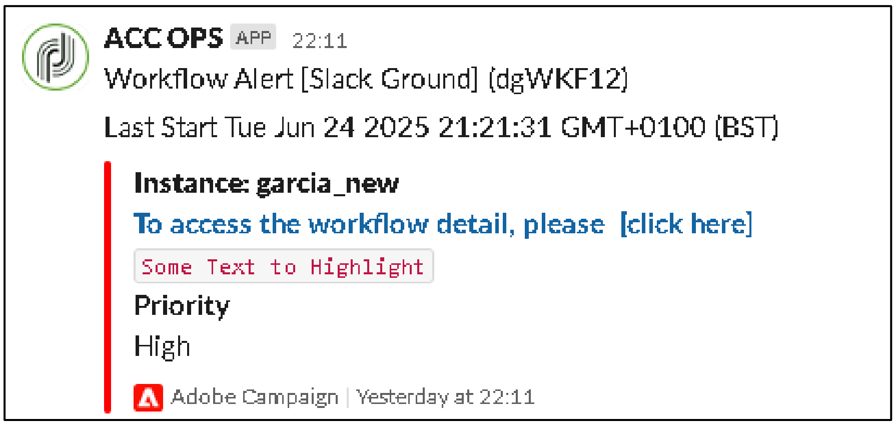
Template example #1 – Think campaign approval or workflow restart; you could enhance the payload to send actionable commands back to Adobe Campaign. Similar to how email approvals work, you can mimic that behaviour in Slack by embedding the approval or action links directly into designated buttons.


ACC OPS APP 02:36
You have a new request:
Fred Enriquez - New device request

Type:
Computer (laptop)
Last Update:
Mar 10, 2015 (3 years, 5 months)
Specs:
"Cheetah Pro 15" - Fast, really fast"

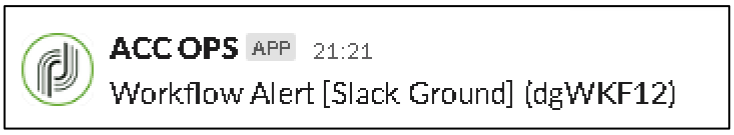
When:
Submitted Aut 10
Reason:
All vowel keys aren't working.

Template example #2 – The following JSON template is my personal favourite and is what I would recommend to be used for failure alerts, it includes a link to the actual workflow, exactly like the email alternative.

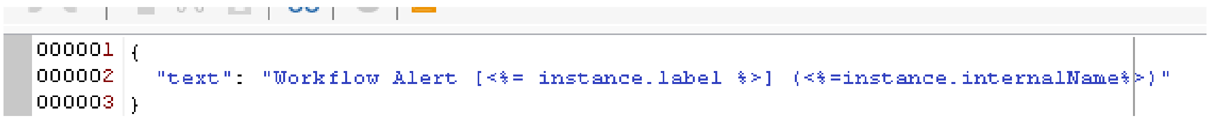

ACC OPS APP 22:11
Workflow Alert [Slack Ground] (dgWKF12)
Last Start Tue Jun 24 2025 21:21:31 GMT+0100 (BST)

Instance: garcia_new
To access the workflow detail, please [click here]
`Some Text to Highlight`
Priority
High
Adobe Campaign | Yesterday at 22:11

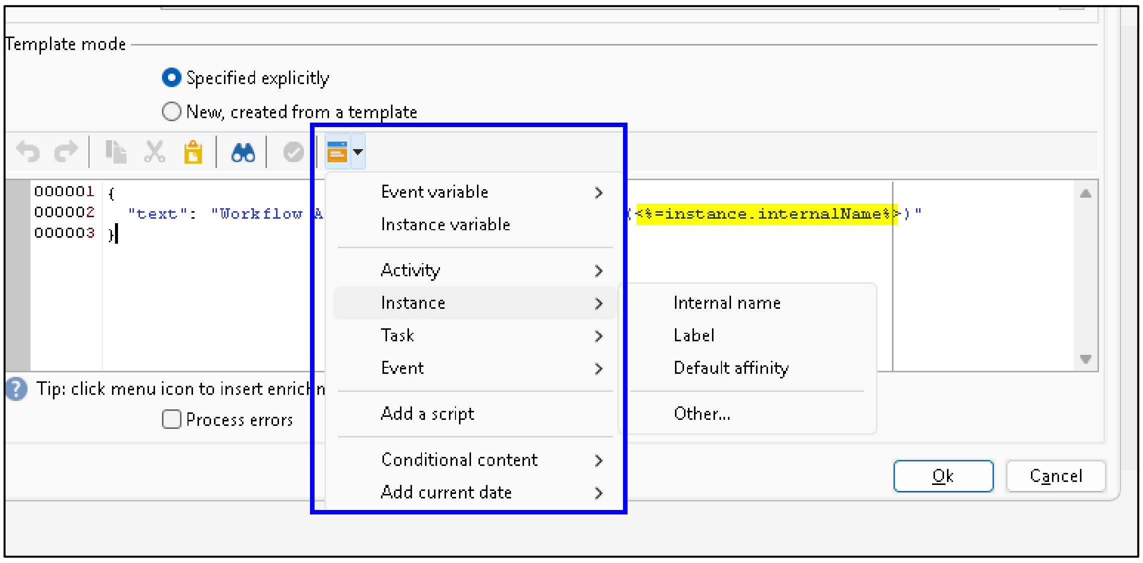
Template example #3- showcasing a template in its simplest text form with some personalization


ACC OPS APP 21:21
Workflow Alert [Slack Ground] (dgWKF12)

Customize the JSON payload using the "Specified explicitly" option to match your requirements.


```
{
   "text": "Workflow Alert [<%= instance.label %>] (<%=instance.internalName%>)"
}
```

- **Personalization**- is supported through instance variables or any other type of variable available in the workflow. The built-in JavaScript logic parses these placeholders and dynamically replaces them, allowing you to tailor each alert message to its context.



Template mode

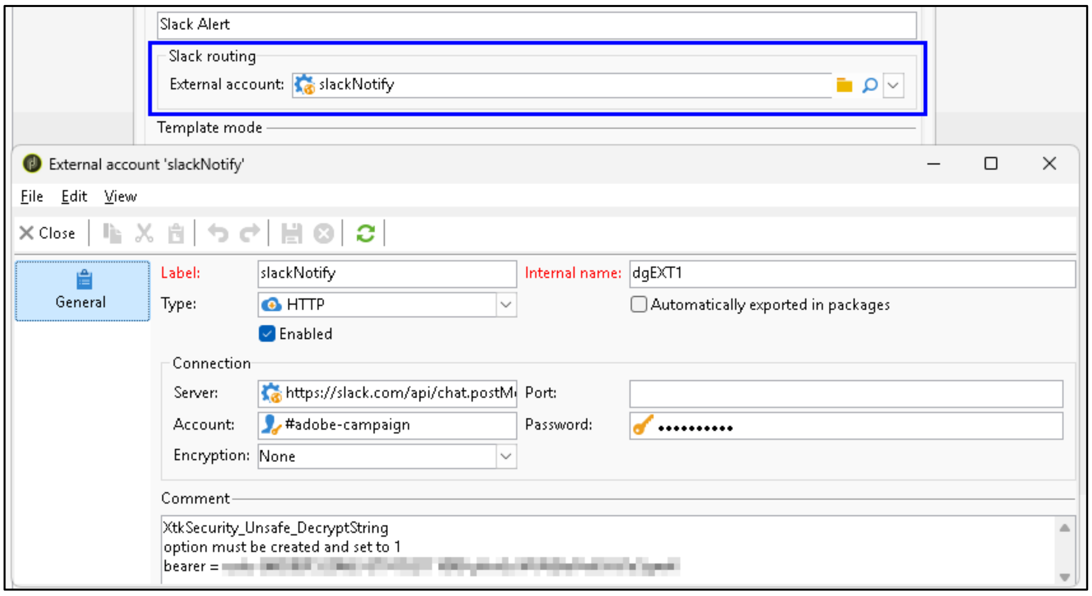
[✓ Specified explicitly] New, created from a template

```
{
  "text": "Workflow &                          (<%=instance.internalName%>)"
}
```

Tip: click menu icon to insert enrich

☐ Process errors

- **External Account (routing)**- was integrated to simplify credential management. You can configure multiple Slack external accounts, each mapped to a specific target channel. This setup allows you to route alerts to different Slack spaces/channels depending on your operational needs. For added flexibility, you can extend the activity to support runtime channel overrides; I'll leave that enhancement up to you.



Slack Alert

Slack routing

External account: slackNotify

Template mode

External account 'slackNotify'

Label:	slackNotify	Internal name:	dgEXT1
Type:	HTTP	☐ Automatically exported in packages	
	✓ Enabled		

Connection

Server:	https://slack.com/api/chat.postM	Port:	
Account:	#adobe-campaign	Password:	••••••••••
Encryption:	None		

Comment

XtkSecurity_Unsafe_DecryptString
option must be created and set to 1
bearer = [redacted]

Setup instructions

Please head over to our GitHub repo to download assets or code referenced in this document - https://github.com/martech-network/slack-activity

ServerConf

Locate your backend "ServerConf.xml" and find **"<urlPermission"** opening tag, you must whitelist slack's domain to allow campaign to make outbound request.

```
<url dnsSuffix="slack.com" urlRegEx="https://.*"/>
```

Schema

Begin by creating an extension of the **xtk:workflow** schema and insert the following xml code.

1. Replace "dg" namespace to yours.
2. Upload canvas logo `slackalert50px.png`
   Available at https://github.com/martech-network/slack-activity/blob/main/slackalert50px.png

```
  <element desc="Slack Alert Definitions" img="dg:slackalert50px.png"
label="Slack Alert"
           labelSingular="Workflow" name="slackAlert"
template="xtk:workflow:activity">

    <attribute default="'dg:slackalert50px.png'" name="img"/>
    <attribute default="'dg:slackAlert.js'" name="library"/>
    <attribute name="label" type="string"/>
    <attribute name="mode" type="integer"/>
    <element label="Slack Template" name="slackTemplate"
target="xtk:javascript"
             type="link"/>
    <element label="External account" name="extAccount"
target="nms:extAccount" type="link"/>
    <element desc="Slack Template JSON" edit="jsedit" name="script"
type="memo" xml="true">
      <default>'{
  "text": "Workflow Alert [&lt;%= instance.label %&gt;]
(&lt;%=instance.internalName%&gt;)"
}'</default>
    </element>
    <element label="Temporary schema linked to the activity"
name="extension" type="ANY"/>
    <element name="transitions" template="xtk:workflow:doneAndError"/>
  </element>
```

```
<element img="xtk:workflow.png" label="workflow" name="workflow">
  <element name="activities" xml="true">
    <element name="slackAlert" ref="slackAlert" unbound="true"/>
  </element>
</element>
```

Input form

This will control the behaviour, look and feel of the activity.

1. Upload activity menu icon `mini-slackalert.png`
2. Navigate to `"Administration/Configuration/Input forms/"` and edit **xtk:workflow**
3. Find the following line
   `<input img="nms:miniatures/mini-alert.png" xpath="alert"/>`
   in new line add;
   `<input img="nms:mini-slackalert.png" label="Slack alert" xpath="slackAlert"/>`

4. To insert our custom input code, find the following text `<!--[of]:jsx-->` and find its closing tag `<!--[cf]-->` and under a new line add the following code.

   GitHub code https://github.com/martech-network/slack-activity/blob/main/xtk%3Aworkflow

```
  <!--[of]:slackAlert-->
        <form label="Template Configuration" name="slackAlert"
  type="notebook">
          <enter>
            <if expr="![@extAccount-id]">
              <set value="0" xpath="@mode"/>
            </if>
          </enter>
          <container img="nms:miniatures/mini-page.png">
            <input nolabel="true" xpath="@label"/>

            <container label="Slack routing" type="frame">

              <input img="nms:extAccount.png" noAutoComplete="true"
  xpath="extAccount">
                <sysFilter>
                  <condition expr="@messageType = 0 and @type=5"/>
                </sysFilter>
              </input>
            </container>
            <static label="Template mode" type="separator"/>
            <input checkedValue="0" label="Specified explicitly"
  type="RadioButton"
                   xpath="@mode"/>
            <input checkedValue="1" label="New, created from a template"
  type="RadioButton"
                   xpath="@mode"/>

            <container type="visibleGroup" visibleIf="@mode==1">
              <input img="xtk:form/script.png" noAutoComplete="true"
  xpath="slackTemplate">
                <sysFilter>
                  <condition expr="@namespace = 'slack'"/>
                </sysFilter>
              </input>
            </container>

            <container type="visibleGroup" visibleIf="@mode==0">
              <input fullToolbar="true" nolabel="true" xpath="script">
                <container>
                  <input menuId="workflowMenuBuilder"
  type="customizeBtn" xpath="/ignored/customizeBtnOut"/>
                </container>
              </input>
            </container>

            <static type="help">Tip: click menu icon to insert
  enrichment tags &lt;= instance.example &gt;</static>

            <input label="Process errors"
  xpath="transitions/error[@name='error']/@enabled"/>
          </container>
          <container ref="xtk:workflow:lib/tabs/advancedActivityPage"/>
          <leave>
            <check expr="[@extAccount-id]!=0">
              <error>You must specify Slack account</error>
            </check>
            <if expr="@mode=1">
              <check expr="[@slackTemplate-name]!=''">
                <error>You must specify JSON template</error>
              </check>
            </if>
            <set expr="'nms:extAccount:'+[extAccount/@name]"
  xpath="extAccount/@name"/>
          </leave>
        </form>
  <!--[cf]-->
```

JavaScript

The backbone of our custom activity is managed by two js libraries.

slackAlert.js - is used to interpret content and configuration aspect of the activity such as json payload tag personalization, external account and template.

slackNotify.js – is a helper referenced in slackAlert.js which manages the connectivity aspect to Slack's API.

1. Create the following files under `“Administration/Configuration/JavaScript codes/”`
- https://github.com/martech-network/slack-activity/blob/main/dg%3AslackAlert.js
- https://github.com/martech-network/slack-activity/blob/main/dg%3AslackNotify.js

slackAlert.js

```
// Load Slack API helper library
loadLibrary("dg:slackNotify.js");

var _Obj_;
var Obj;
var payload;
var slackTemplate;

if (activity.mode == 1 && activity.slackTemplate_name) {
  var slackTemplate =
xtk.javascript.load(activity.slackTemplate_namespace+":"+activity.slackTe
mplate_name).data;
} else if (activity.mode == 0) {
  var slackTemplate = activity.script;
}

// Remove single-line comments outside of payload
var _Obj_ = slackTemplate.replace(/^\/\/.*$/gmi, "");

//replace placeholders dynamically
var Obj   = _Obj_.replace(/\<\%\=(.+?)\%\>/gmi, function (matched) {
    try {
        return Function('return ' + matched.replace(/[^a-zA-Z0-9."()
_]/g, ''))();
    } catch (e) {
        logWarning("Placeholder parsing error: " + e);
        return matched; // Keep original placeholder if there's an error
    }
});

logInfo(activity.slackTemplate_name) //troubleshooting
logInfo(activity.mode) //troubleshooting

// Validate JSON before parsing to avoid errors

try {
    payload = JSON.parse(Obj);
    payload.channel = extAccount() ? extAccount().account : "default-
channel"; // Safe handling, define a fallback slack channel

} catch (e) {
    logWarning("JSON Parsing Error: " + e);
    payload = {}; // Prevent execution failure
}

// Slack alert execution function
function slackAlert_call() {}

// Retry mechanism for Slack API call
function slackAlert_recall() {
    try {
        slackNotify(JSON.stringify(payload));
        task.postEvent(task.doneTransition());
        task.setCompleted();
        return 0;
    } catch (e) {
        logWarning("Slack API Call Failed: " + e);
        task.postEvent(task.errorTransition());
    }
}
```

slackNotify.js

```
// Function to retrieve external account details
function extAccount() {

    if(activity.extAccount_id == 0){
      logError("Slack external account not configured in alert.")

      return null
      } else {
        try {
            return nms.extAccount.load(activity.extAccount_id);
        } catch (e) {
            logWarning("Failed to load external account: " + e);
            return null; // Prevent execution failure
        }
    }
}
```

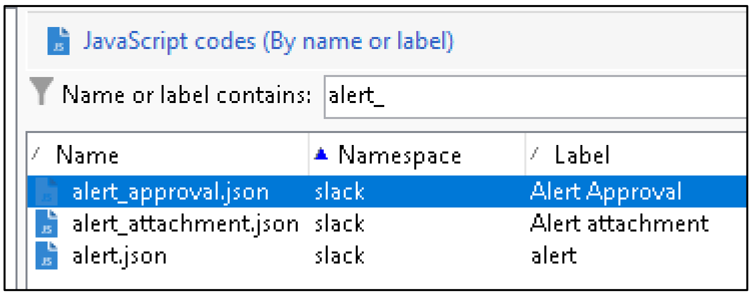
```
// Slack API request function
function slackNotify(payload) {

    var account = extAccount();
    if (!account) {
        logError("External account missing, unable to send Slack 
notification.");
        return;
    }

    var req = new HttpClientRequest(account.server);
    var bearer = decryptString(account.password); // Requires 
XtkSecurity_Unsafe_DecryptString option

    req.header["Authorization"] = "Bearer " + bearer;
    req.header["Content-Type"] = "application/json; charset=utf-8";
    req.method = "POST";
    req.body = payload;

    try {
        req.connect();
        req.execute();

        // Log response for debugging
        logInfo("Slack API Response Code: " + req.response.code);
        if (req.response.code !== 200) {
            throw("Slack API response: " + req.response.code);
        }
    } catch (exception) {
        logError("Slack API exception: " + exception);
    } finally {
        req.disconnect();
    }
}
```

JSON Templates

Our custom activity has been developed with templating in mind, allowing you to create your own JSON templates which can be configured in the activity. These templates are created under the JavaScript schema with a specific namespace (slack), our input form has a sysfilter which will only list scripts created under this namespace.

```
<sysFilter><condition expr="@namespace = 'slack'"/></sysFilter>
```

Keep in mind file naming conventions.

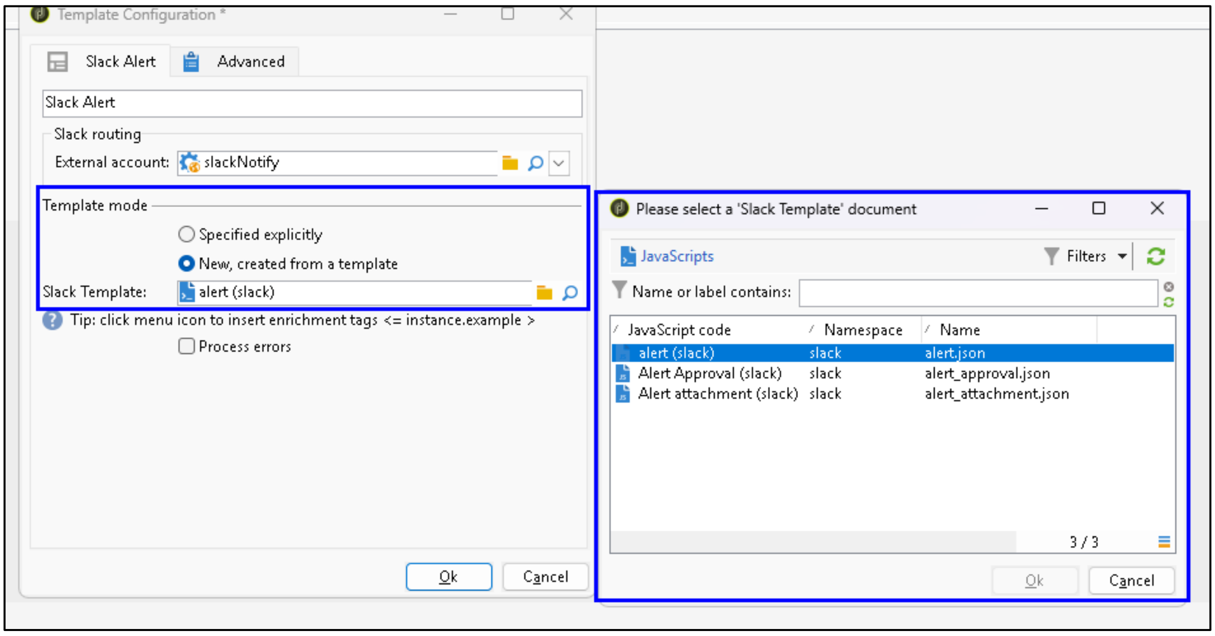


JavaScript codes (By name or label)

Name or label contains: alert_

Name	Namespace	Label
alert_approval.json	slack	Alert Approval
alert_attachment.json	slack	Alert attachment
alert.json	slack	alert

Name	Namespace	Desc
alert_approval.json	slack	Alert Approval
alert_attachment.json	slack	Alert attachment
alert.json	slack	alert



Template Configuration *

Slack Alert | Advanced

Slack Alert

Slack routing

External account: slackNotify

Template mode

( ) Specified explicitly

(● New, created from a template)

Slack Template: alert (slack)

Tip: click menu icon to insert enrichment tags <= instance.example >

☐ Process errors

Ok | Cancel

Please select a 'Slack Template' document

JavaScripts — Filters

Name or label contains:

JavaScript code	Namespace	Name
alert (slack)	slack	alert.json
Alert Approval (slack)	slack	alert_approval.json
Alert attachment (slack)	slack	alert_attachment.json

3 / 3

Ok | Cancel

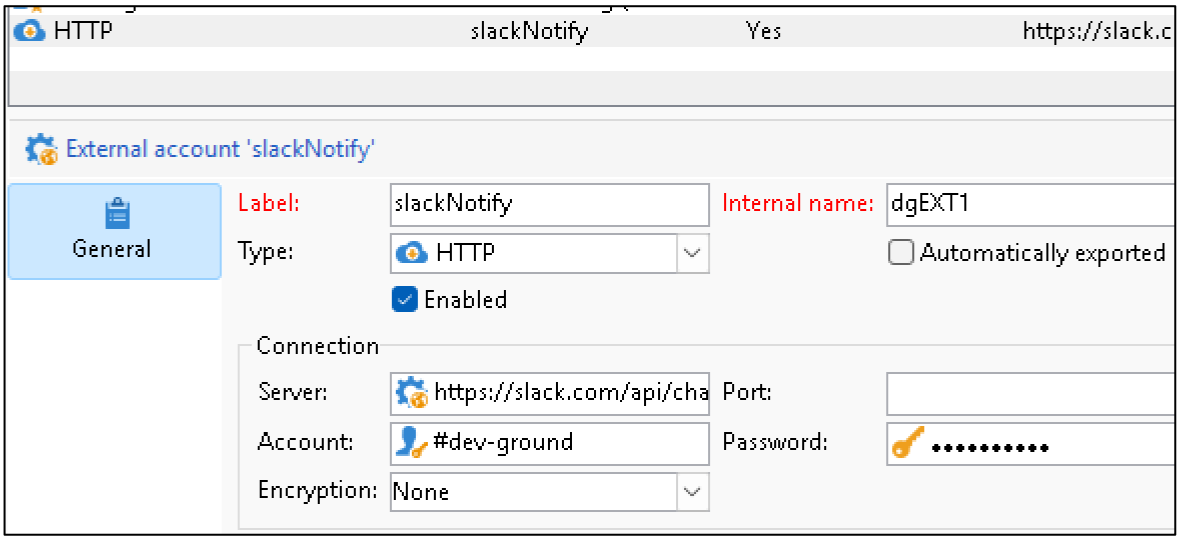
Platform options

Our JavaScript requires to decrypt the password (bearer token) stored under external account which also contains details of Slack's API endpoint. To be able to decrypt you must enable or create a hidden platform option; check for the existence of **XtkSecurity_Unsafe_DecryptString** option, if not created and set the value to 1. For more information, here is the link to documentation https://experienceleague.adobe.com/en/docs/campaign-classic/using/installing-campaign-classic/appendices/configuring-campaign-options.

External account

You can setup multiple connectors/external account each pointing to different Slack channels and/or workspaces.

Server: https://slack.com/api/chat.postMessage
Account: is your slack channel
Password: is the token generated when the app was created



HTTP slackNotify Yes https://slack.c

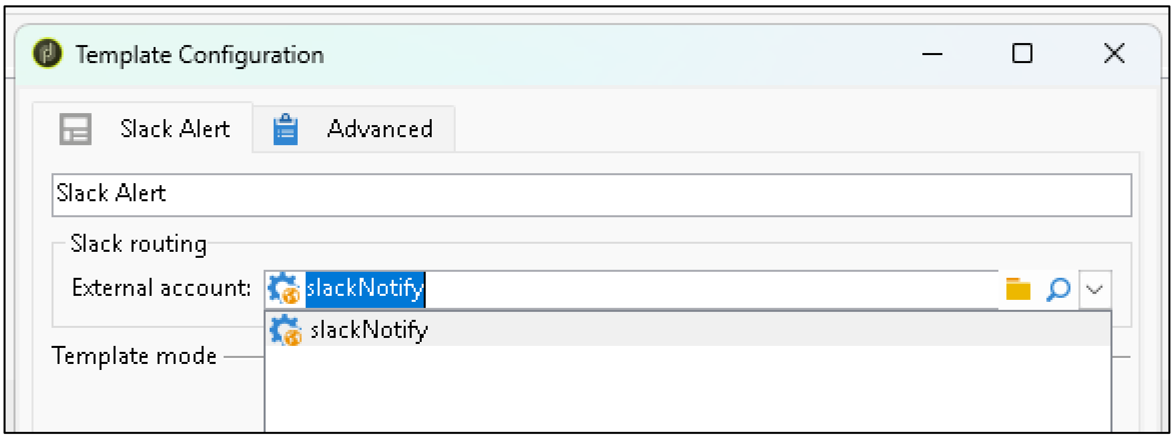
External account 'slackNotify'

General

Label:	slackNotify	Internal name:	dgEXT1
Type:	HTTP		☐ Automatically exported
	✓ Enabled		

Connection

Server:	https://slack.com/api/cha	Port:	
Account:	#dev-ground	Password:	••••••••••
Encryption:	None		



Template Configuration

Slack Alert | Advanced

Slack Alert

Slack routing

External account: slackNotify

slackNotify

Template mode

Improvements

The current state of the custom activity is fully functional and should be enough to get you started, however, you may tweak and improve it as per your business requirements, below are a few I thought would be nice to have.

1. **Runtime channel override**: could allow you to still use an external account whilst allowing you to configure a channel within the payload or by adding a new field in the input form/schema on the fly.
2. **Default external account**: rather than having to configure the slack account every time you add the slack activity to the workflow canvas, perhaps you could improve its functionality by setting up a default external account. This may incur creating a platform option and having the primary key of the default external account, your JavaScript code will then look for the existence of this option and whether it has a value corresponding to an external account, the input form should somehow also be able to determine if this option exist and has a value to automatically set it in the custom activity.
3. **Notify specific users:** by granting the app additional scopes, in specific those that allow you to DM a user, you could enhance the activity to also DM specific users rather than slack channels, you will need to also add these options within the custom activity input form.

That's it for now, enjoy

Share your improvements over in our exclusive Slack channel

Join us on https://martech.network

GitHub repo - https://github.com/martech-network/slack-activity

LinkedIn https://www.linkedin.com/in/david-garcia-uk/

PS. There will be a new tutorial on how to automatically trigger workflow failures to Slack using Error management (Notification of the workflow supervisor (notifySupervisor) soon.