**Webhooks**

**Description**

* Web hooks provide a mechanism whereby a server-side application can notify a client-side application when a new event (that the client-side application might be interested in) has occurred on the server.
* Webhooks allow you to send information about your licensed Mendix app deployed to the Mendix Cloud to an external app or workflow. This can be used, for example, to trigger an automated CI/CD workflow when a new change is committed to the Team Server.
* Mendix provides webhooks to send project information when the following events happen to your app:
* On package upload – when a deployment package is available in the Developer Portal — this includes creating a package from the Teamserver
* Team Server push – when a new commit is pushed to the Team Server — this will only be triggered if your app is stored in a Git repository
* The webhooks contain a retry mechanism if an error response is received from the endpoint to ensure that the trigger reaches the endpoint.
* <https://www.youtube.com/watch?v=29_QCGVBISg>
* <https://codeburst.io/what-are-webhooks-b04ec2bf9ca2>
* For instance, razor pay platform provides solutions regarding payment related services that have exposed webhook events at their dashboard which can be configured for callback URL.

Once the event is configured, a callback URL would call and it carries payloads (info regarding event), which can be used in client application (Mendix app) to get the required info.

* <https://razorpay.com/e-mandate/>
* <https://razorpay.com/docs/recurring-payments/webhooks/>
* <https://razorpay.com/docs/api/recurring-payments/webhooks/#sample-payloads>

**Typical Usage Scenario**

1. Create notifications to you or anybody via email, IRC, Jabber, … ( dispute\_resolved, dispute\_won, payment\_approved, payment\_canceled, payment\_paid )
2. put the data in another app (real-time data synchronization)
3. process the data and repost it using the app's API
4. validate the data and potentially prevent it from being used by the app
5. You can use a webhook to connect a payment gateway with your email marketing software so that a user gets an email whenever a payment bounces.
6. You can also use webhooks to send information about events to external databases or data warehouses like Amazon’s Redshift or Google Big Query for further analysis.
7. You can use a webhook to connect a payment gateway with your [email marketing](https://www.getvero.com/resources/) software so that a user gets an email whenever a payment [bounces](https://www.getvero.com/resources/email-deliverability/).
8. Version control systems notify team members about a commit to a repository.
9. Monitoring systems alerting administrators about an error or unusual activity in a system
10. An e-commerce store notifying your invoicing application about a sale.
11. You can use webhooks to synchronize customer data in other applications. For example, if a user changes their email address, you can ensure the change is reflected in your CRM.
12. E-commerce stores notify merchants when a particular item is out of stock.
13. Automatically receive an email every morning about your first meeting in case you forget to check your calendar.
14. Have Instagram photos upload automatically to Twitter accounts.
15. Configure the doorbell to flash the lights when it rings.
16. Receive notification that a payment has been made and processed.

### **Prerequisites**

NA

**Features and Limitations**

We can connect team webhook with mendix and send message in the form of JSON using REST API

We can send limited message only which is available in the form of JSON.

* A webhook invocation is retried twice for three calls if a webhook is unavailable.
* A webhook must respond within 5 seconds or the webhook call is abandoned. This 5-second limitation applies to the total duration of the call and the two retries.
* Concurrent webhook requests are rate-limited at a tenant level. If webhooks are slow, performance might degrade while it waits for the rate-limit quota to be available.
* Webhook statistics are kept for 7 days.
* Notification dead-letters are limited to the following criteria:
  + Dead-letters older than 90 days are deleted.
  + If a tenant has more than 2 million dead-letters, the number of dead-letters that exceeds that quota are removed starting from oldest to newest.
  + Dead-letter reconciliations do not run for more than 2 hours.
  + Dead-letter reconciliation results are kept for 7 days.

### **Dependencies**

Firstly, We Need to generate webhook link from incoming webhook in teams and use this link in REST API to connect with the team.

Mendix Version: 9.24.0

**Installation**

1. We are Creating entities for sending our data to webhook in our domain model.

A screenshot of a computer

Description automatically generated

2. We need to generate overview pages of TeamWebhookConfiguration and Section entity and then add overview page in navigation

3.After that we need to click on new button on overview page in TeamWebhookConfiguration then this page will appear.

A screenshot of a computer

Description automatically generated

4. In the save button we are using microflow which contain submicroflow(Sub\_Payload\_TeamAPI).

A. In sub-microflow we are creating objects which is required for data.

B. In this microflow (ACT\_TeamMessageWebhoook) we are calling rest service to integrate with teams.

A diagram of a diagram

Description automatically generated

7.In ACT\_TeamMessageWebhoook microflow initially we are calling sub-microflow (Sub\_Payload\_TeamAPI). In this sub-microflow we are creating object according to the JSON code.

{

"@type": "MessageCard",

"@context": "<http://schema.org/extensions>",

"themeColor": "0076D7",

"summary": "Larry Bryant created a new task",

"sections": [

{

"activityTitle": "Larry Bryant created a new task",

"activitySubtitle": "On Project Tango",

"activityImage": "<https://teamsnodesample.azurewebsites.net/static/img/image5.png>",

"facts": [

{

"name": "Assigned to",

"value": "Unassigned"

},

{

"name": "Due date",

"value": "Mon May 01 2017 17:07:18 GMT-0700 (Pacific Daylight Time)"

},

{

"name": "Status",

"value": "Not started"

}

],

"markdown": true

}

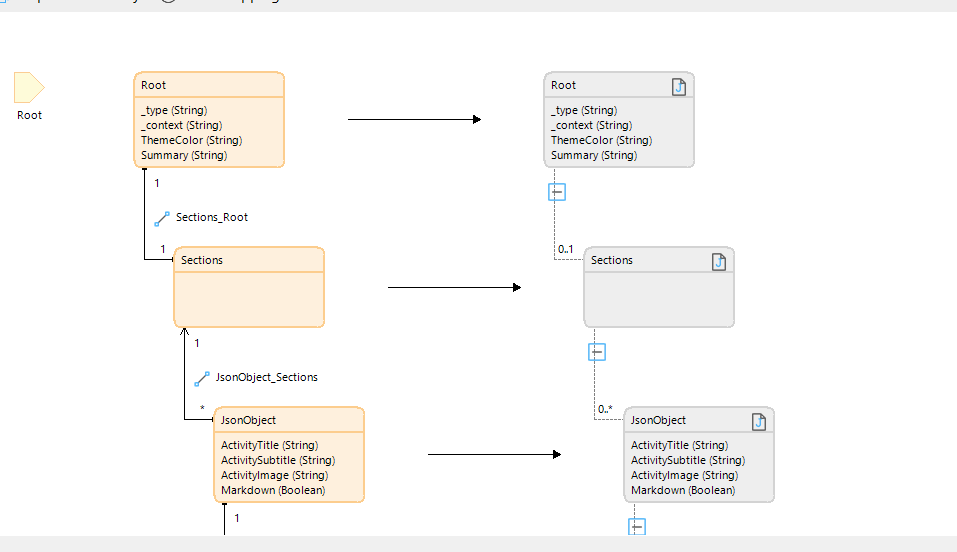
]

}

8.After creating object we are passing value of root and export mapping in Export to JSON activity in JSOn then at the last we are returning this JSON object.

Diagram

Description automatically generated



9.After sub-microflow we are calling Rest service to integrate with Teams we are passing one constant which contain the teams incoming webhook link and we are calling post service also.

Graphical user interface, text, application

Description automatically generated

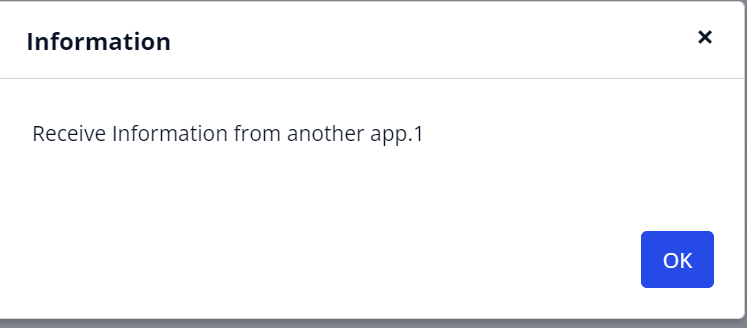
**UI ScreenShots**

1.We are passing activity title and subtitle which is available in fact object (1) and also, we are passing image address which is currently visible along with message card then at the last we are passing link which is available in the fact object (2) in microflow this is dynamic you can pass any id for each message card.

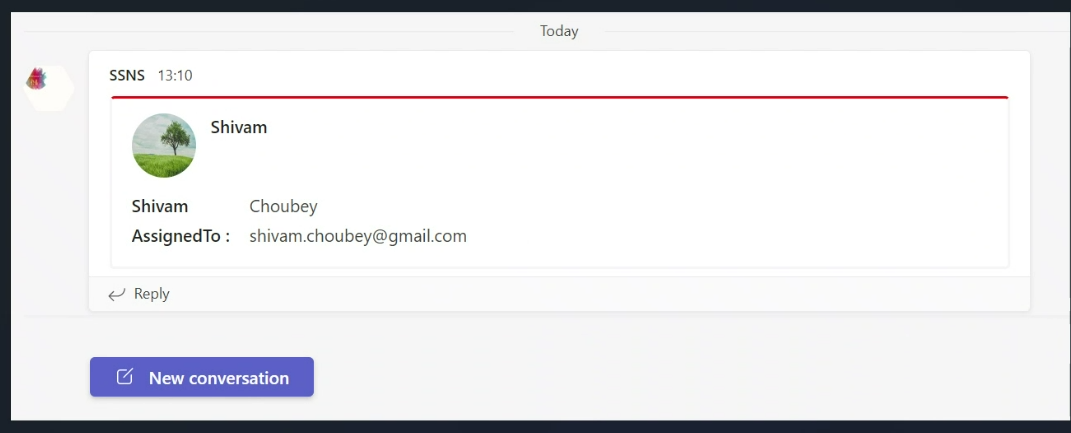
A screenshot of a computer

Description automatically generated

2.Receive Confirmation message from team:



3.Team message Received:



**Webhook Vs API**

Webhooks and APIs are both widely used across the software we use daily, and they're so similar that their individual uses can get confusing. Since a webhook is just a specific type of API, the confusion makes sense.

To summarize, webhooks enable lightweight data sharing between software when a specific action takes place, whereas APIs require user input on one end to request or modify data on the other end.

If you are looking to send a notification or update information as soon as a certain criterion is met, avoid the hassle of an API—just implement a simple webhook. If you are dealing with fluctuating data or want to modify data rather than merely push a notification, you will need to implement an API.

When choosing between the two, the most important question to ask is whether the data you want to access is constantly being updated. If it is, an API will make more sense than a webhook. If it is not, consider implementing a webhook instead.

In the end, there is no need to start a webhook vs. API debate—both are necessary for the apps we love to function.

**Configuration: NA**

**Category: Module**

**Subcategory: All**

**Visibility: Public Marketplace**

**Studio Pro Version: 9.24.0**