API Response Mapping

Purpose

The purpose of API Response Mapping is to enable the automated enrichment of ticket attributes within Puzzel Case Management. This feature allows the content returned from an outbound webhook API call to be efficiently mapped to specific category or form field ticket attributes, facilitating streamlined workflow and enhanced data management. The returned response must be in JSON type in order for this feature to function correctly.

How to Use API Response Mapping

Below are the steps to create, implement, and utilise the API Response Mapping feature:

1. Creating a Response Mapping Schema

To create a new Response Mapping schema:

1. Navigate to **Settings -> Integrations -> Outbound Integrations -> Response Mappings** in your Puzzel Case Management interface.
2. Click on the option to create a new Response Mapping schema.
3. In the schema creation interface, define the mappings using JSON Path to define the link between the expected JSON response attributes and the corresponding ticket attributes (categories or form fields).

Example Response Mapping:

```
{
  "Categories": [
    {
      "Category": "Internal Customer Key",
      "JsonQuery": "$.response.base.internalCustomerKey"
    },
    {
      "Category": "Anonymous Customer Key",
      "JsonQuery": "$.response.base.anonymousCustomerKey"
    },
    {
      "Category": "Country",
      "JsonQuery": "$.response.base.country"
    }
  ],
  "Form": {
    "JsonQuery": "$.response.base.formSample"
  },
  "FormFields": [
    {
      "Field": "Free Text",
      "JsonQuery": "$.response.base.formSample..freeText"
    },
    {
      "Field": "Dropdown",
      "JsonQuery": "$.response.base.formSample..dropdown"
    },
    {
      "Field": "Nested Dropdown",
      "JsonQuery": "$.response.base.formSample..nestedDropdown.field",
      "NestedFields": [
        {
          "Field": "Free Text 1",
          "JsonQuery": "$.response.base.formSample..nestedDropdown.fields..freeTextNested1"
        },
        {
          "Field": "Free Text 2",
```

```
        "JsonQuery": "$.response.base.formSample..nestedDropdown.fields..freeTextNested2"
      },
      {
        "Field": "Nested Options",
        "JsonQuery": "$.response.base.formSample..nestedDropdown.fields..nestedOptions"
      }
    ]
  }
 ]
}
```


Here you would typically define how each attribute in your JSON response corresponds to a specific field in your ticketing system. This might include mapping a 'customer_email' from the JSON to the 'Email' field in a ticket form, or mapping a 'issue_type' to a specific category in your system.

> **Note**
>
> For individuals keen on exploring Schema validation, the schema definition is provided here for your perusal and implementation. This comprehensive guide is designed to facilitate your understanding and application of the Schema, ensuring a seamless integration process.

Response Mapping Schema breakdown

Categories

Property	Type	Description	Constraints/Notes
Category	String	Defines the category name.	Must be at least 3 characters long.
JsonQuery	String	A query string to fetch data.	Optional. Must be at least 3 characters long. Required if DefaultValue is not provided.
DefaultValue	String	Default value for the category.	Optional. Must be at least 3 characters long. Required if JsonQuery is not provided.
TransformerSteps	Array of Objects	Steps to transform data.	Refer to TransformerSteps details.

Form

Property	Type	Description	Constraints/Notes
JsonQuery	String	A query string to fetch form data.	Optional. Must be at least 3 characters long. Required if DefaultValue is not provided.

DefaultValue	String	Default value for the form.	Optional. Must be at least 3 characters long. Required if JsonQuery is not provided.
TransformerSteps	Array of Objects	Transformation steps for form data.	Refer to TransformerSteps details.

FormFields

Property	Type	Description	Constraints/Notes
Field	String	The name of the field.	Required.
JsonQuery	String	A query string to fetch field data.	Optional. Must be at least 3 characters long. Required if DefaultValue is not provided.
DefaultValue	String	Default value for the field.	Optional. Must be at least 3 characters long. Required if JsonQuery is not provided.
TransformerSteps	Array of Objects	Transformation steps for field data.	Refer to TransformerSteps details.
NestedFields	Array of Objects	Nested fields within a field.	Follows the same structure as FormFields.

TransformerSteps (Shared Structure for Transformation Logic)

Property	Type	Description	Constraints/Notes
StringAction	Object	Defines actions for string manipulation.	Optional. See below for details.
ArrayAction	Object	Defines actions for array manipulation.	Optional. See below for details.

StringAction

Property	Type	Description	Constraints/Notes

Method	String	The string manipulation method.	One of: "[]", "split", "slice", "index", "rindex".
Arguments	Array	Arguments for the method.	Items can be string or integer.

StringAction Methods

Method	Description
[]	Accesses character(s) at specified index(es). If a range is provided, returns the substring within that range.
split	Divides the string into an array of substrings based on a delimiter. The delimiter can be a string.
slice	Extracts a part of a string and returns it as a new string without modifying the original string. Can accept start and end indices.
index	Returns the index of the first occurrence of a specified substring. If not found, returns -1.
rindex	Similar to index, but searches for the substring from the end of the string, returning the index of the first occurrence found when searching backwards.

ArrayAction

Property	Type	Description	Constraints/Notes
Method	String	The array manipulation method.	One of: "[]", "first", "last", "take", "join".
Arguments	Array	Arguments for the method.	Items are strings or integers.

ArrayAction Methods

Method	Description
[]	Accesses an element in an array. When a single index is provided, returns the element at that position.

first	Returns the first element, or the first n elements, of the array. Without arguments, it returns the first element.
last	Similar to first, but returns the last element or the last n elements of the array.
take	Returns the first n elements of the array. It's similar to first with an argument, but specifically designed for taking multiple elements from the beginning.
join	Combines all elements of an array into a single string, separated by the specified delimiter. If no delimiter is provided, elements are typically joined without any separator.

Overview of Response Mapping in Webhook Processing

During the webhook sending process, the received response is compared against the pre-defined response mapping, if one is selected. This mapping directs how data from the response is extracted and stored in the corresponding categories, forms or form fields on a ticket.

Basic Mapping Example:

Consider the following response mapping configuration:

```
{
  "Categories": [
    { "Category": "CustomerID", "JsonQuery": "$.Customer.ID" },
    { "Category": "CustomerAPI", "JsonQuery": "$.Customer.href" }
  ]
}
```

When applied, this mapping instructs the system to:

1. Extract Customer.ID from the response and store it in the CustomerID category on the ticket, assuming the CustomerID category exists.
2. Extract Customer.href and store it in the CustomerAPI category on the ticket, assuming the CustomerAPI category exists.

Example Response:

```
{
  "Customer": {
    "ID": 12345,
    "href": "<https://System.com/API/V1/Customers/12345>"
  }
}
```

Result:

- CustomerID category value: 12345
- CustomerAPI category value: <https://System.com/API/V1/Customers/12345>

Handling Missing Data in the Response:

In cases where the response lacks certain data, such as the ID field, but includes the href, the system can still make use of the DefaultValue.

Advanced Mapping Example:

```
{
  "Categories": [
    { "Category": "CustomerAPI", "JsonQuery": "$.Customer.nomatch", "DefaultValue": "Rodger" }
  ]
}
```

This configuration:

1. Finds no value under $.Customer.nomatch in the response
2. Falls back to the DefaultValue provided

Result:

- CustomerAPI category value: Rodger

Advance Mapping Example - Arrays:

When we point the webhook to retrieve an array of data, maybe the<https://System.com/API/V1/Customers> endpoint, which returns a list of customers on the system.

```
{
  "Categories": [
    { "Category": "CustomerAPI", "Jsonquery": "$.Customer[N]"}
  ]
}
```

Considering the array is zero-indexed, substitute 'N' with the position in the array of the desired result. If you aim to fetch the first item, 'N' should be 0; for the second item, 'N' becomes 1, and so forth. For retrieving the last item, 'N' becomes -1.

TransformerSteps Mapping Example:

Consider the following response mapping configuration:

```
{
  "Categories": [
    {
      "Category": "CustomerID", "JsonQuery": "$.Customer.ID"
      "TransformerSteps" [
        { StringAction: { "Method": "split", Arguments: [',']}},
        { ArrayAction: { "Method": "[]", Arguments: [0]}}
      ]
    }
  ]
}
```

When applied, this mapping instructs the system to:

1. Extract Customer.ID from the response
2. Then split the response into an array with a delimeter of ,
3. Finally select the first object from within that array.

Example Response:

```
{
  "Customer": {
```



```
    "ID": "boxes,of,chocolate"
  }
}
```

Result:

- CustomerID category value: boxes

2. Editing an Existing Webhook

To apply a Response Mapping schema to an existing webhook:

1. Go to **Settings -> Integrations -> Outbound Integrations -> Webhooks**
2. Select and edit the webhook you wish to modify.
3. In the webhook settings, select one of the available response mapping schemas from a dropdown menu or similar selector. This schema will be applied to process responses when the webhook is triggered.

3. Setting up a Business Rule

To automate the triggering of the webhook:

1. Define a new business rule (this can be a productivity rule or an event rule) within the system.
2. Within the rule's action settings, specify the action to trigger a webhook and send a schema request to an external endpoint.

How It Works

When the configured webhook is triggered (via the conditions set in your business rule), the response mapping schema will automatically process the returned JSON response from the external endpoint. It will search for values contained within the JSON and map them to the predetermined form or category fields in your ticketing system. This process allows for the automatic update and enrichment of ticket attributes based on external data sources, streamlining your workflow and improving ticket information accuracy.

By integrating API Response Mapping into your workflow, you can significantly reduce manual data entry, improve response times, and ensure that ticket information is comprehensive and up-to-date.

> Note
>
> Please be aware that the 'Outbound Integrations', 'Event Rules', and 'Response Mappings' features require activation on your Puzzel Case Management instance before use. To enable these functionalities, we kindly invite you to contact our support team. Our dedicated team is ready to assist you with the activation process and ensure you have everything you need to maximise your case management system.

