Arazzo 1.0.1 Cheat Sheet

bump.sh

Why Arazzo

Arazzo describes business workflows that orchestrate several API calls. While OpenAPI defines an API's structure, Arazzo defines how multiple operations are chained together to achieve a specific goal.

Document structure

An Arazzo document is a JSON or YAML file containing the following root elements:

```
arazzo: 1.0.1              # The spec version
info: {}                   # General workflow info
sourceDescriptions: {}     # OpenAPI or Arazzo sources
workflows: {}              # List of workflows
components: {}             # Reusable components
```

General information

```
info:
  title: Holiday booking
  summary: Can be used to plan your next holiday...
  description: A booking is made with these 4 st...
  version: 1.2.34
```

Sources

Import OpenAPI and other Arazzo definitions required to run workflows defined in the document.

```
sourceDescriptions:
- name: trainBooking
  url: https://example.com/openapi.yaml
  type: openapi
- name: holidayRentalSelection
  url: ./sources/arazzo.yaml
  type: arazzo
```

Sources can be imported from a remote URL or a filesystem path.

Reuse elements

Avoid duplicating elements by defining reusable components:

```
components:
  parameters: {}
  successActions: {}
  ...
```

Use your components with the reference keyword:

```
reference: $components.successActions.notify
```

Workflow structure

Workflows are composed of user inputs, steps (API calls), and outputs.

```
workflows:
- workflowId: book-a-train-ticket
  summary: Book a train ticket
  description: Search, select and pay for a train...
  inputs: {}
  steps:{}
  outputs: {}
```

Inputs

Global inputs described in JSON Schema.

```
inputs:
  type: object
  properties:
    destination:
      type: string
  required: # Inputs required by steps
  - destination
```

Steps

Call an API or another workflow defined in `sourceDescriptions`:

- `operationId`: ID of an operation
- `operationPath`: JSONPath (if no operationId)
- `workflowId`: ID of a workflow you want to run as a step

```
steps:
- stepId: search-train-trip
  description: Search for a train trip
  operationId: searchTrip
  parameters:
  - name: language
    in: query
    value: en
  body:
    contentType: application/json
    payload:
      destination: $inputs.destination
  successCriteria:
  - condition: $statusCode == 200
  onSuccess: {}
  onFailure: {}
  outputs: {}
```

Outputs

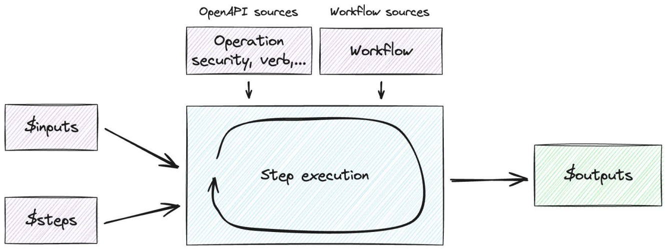
Result of a step or a workflow.

```
outputs:
  trips: $response.body
```

Runtime

Step execution

The tool runs your workflow step by combining data from your Arazzo file and your OpenAPI sources. It generates and executes the API request, then returns the step's outputs so they can be reused by the following steps.



Runtime expressions

Retrieves and transfers data between different steps' inputs and outputs using a dot-based/JSONPath syntax:

`$inputs.destination`, `$method`, `$request.body#/date`, `$steps.search-train-trip.outputs.trips`, `$response.body.trips`, `$statusCode`, `$url`, ...

Full list: spec.openapis.org/arazzo/latest.html#runtime-expressions

Success and failure

`successCriteria` determines whether the overall result is a success or a failure.

`onSuccess` and `onFailure` determine the next action based on the result of the current step.

```
successCriteria:
- condition: $statusCode == 200
onSuccess:
- name: no-results
  type: end # goto|retry|end
  criteria:
  - context: $response.body
    type: jsonpath # jsonpath|regex|simple|xpath
    condition: $[?count(@.trips) == 0]
onFailure:
- name: no-results-yet
  type: retry
  criteria:
  - condition: $statusCode == 202
```

You can define `successAction` and `failureAction` at the workflow level. If defined, they apply to every step.