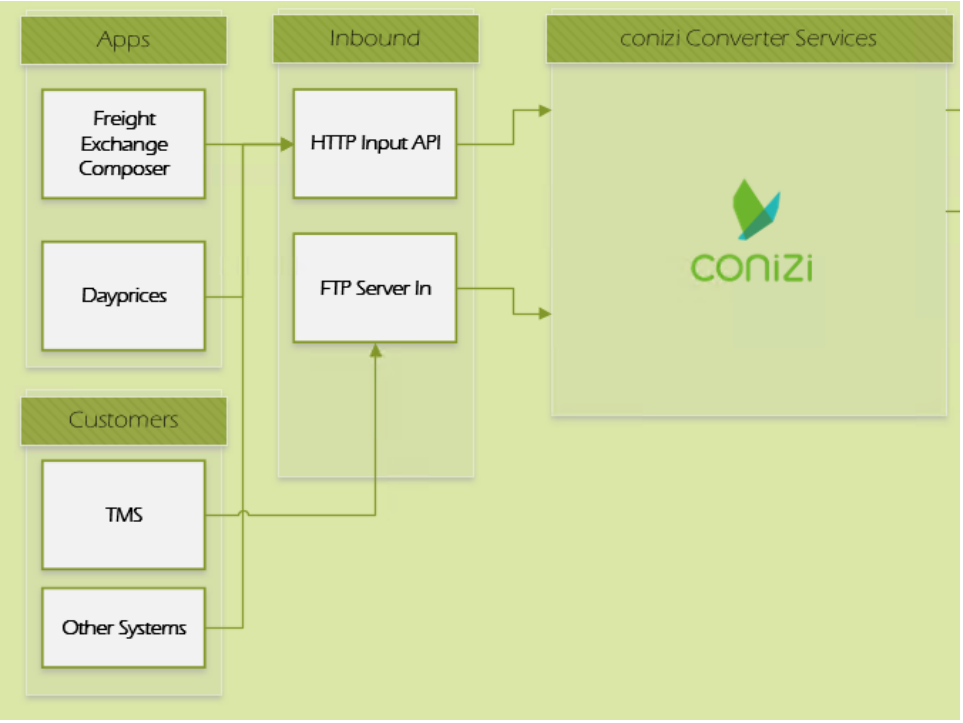
How To: Use conizi HTTP Input API

conizi Overview

The conizi HTTP Input API is one of the possible connectors into the conizi platform.

The HTTP protocol is used to open the possibility, for partners and customers, to send data to conizi in a lightweight way.



Requirements to use the API

Some requirements must be fulfilled to get access to conizi HTTP Input API

- A conizi account must be available and a tenant/site/app must be created, owned by the account or connected to the account.
- An edi connection, configured as HTTP input, for a site or app must be available.
- An API Key must be generated, for the edi connection, used for authentication against conizi.

How could the API be reached

conizi uses different environments for development, testing and production purposes

The interesting environments are Pre-production and Production

Pre-Production (for testing and quality control)

- https://preproduction.dev.conizi.io/api/input/swagger/index.html

Production (the conizi platform production environment)

- https://conizi.io/api/input/swagger/index.html

The product swagger is used to document the API for developers.

> In general, the production and pre-production environment have different versions with different functionality. Please make sure, to own an account fits your needed environment.

What are models and messages

conizi uses JSON Schema definitions to define message formats (see https://json-schema.org/). At the moment conizi uses Draft 6 to define a set of messages like tour, consignment, events...

The available models can be viewed under https://github.com/fleetboard-logistics/semantic-model as an public github.com repository.

This repository also uses different branches to fit the different environments (we use the Pre-production environment in this paper). The needed URI is https://github.com/fleetboard-logistics/semantic-model/tree/preproduction/model.

For example the consignment (https://github.com/fleetboard-logistics/semantic-model/blob/preproduction/model/transport/truck/groupage/forwarding/consignment.json) is used to send a consignment to conizi via the HTTP Input API

It is important to use the right branch for the right environment. Master and preproduction are branches for development and testing and offer in general more features and models. Production, only for tested and deployed models, run directly in conizi production environment.

Testing the API

conizi has a few tools to test the API and validate a message (we use the Pre-production environment in this paper).

As a demonstration, a basic consignment should be defined, see below. The "$schema" tag should always set to the right model definition (be aware of the branch)

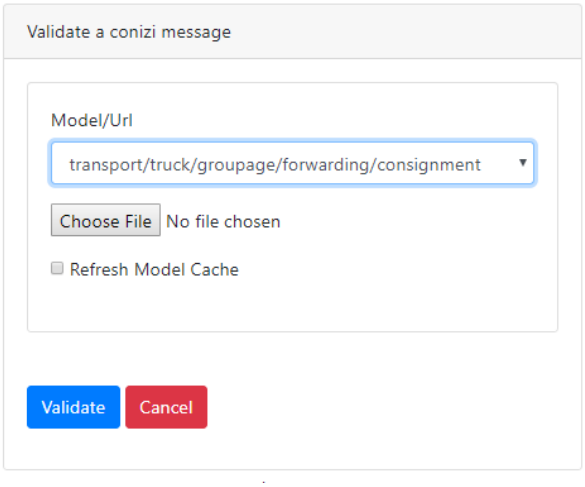
Basic consignment Code

```
{
    "$schema": "https://raw.githubusercontent.com/conizi/semantic-model/preproduction/model/transport/truck
/groupage/forwarding/consignment.json#",
    "sender": {
        "ediId": "VER4999"
    },
    "receiver": {
        "ediId": "EMP4888"
    },
    "shippingPartner": {
        "partnerId": "4999",
        "name": "Versandparter GmbH"
    },
    "receivingPartner": {
        "partnerId": "4888",
        "name": "Empfangpartner AG"
    },
    "shippingDate": "2019-01-02",
    "consignmentNoShippingPartner": "4711",
    "routing": {
        "shipper": {
            "name": "Meyer GmbH",
            "street": "Meyerstraße 10",
            "zipCode": "97424",
            "city": "Schweinfurt",
            "countryCode": "DE"
        },
        "consignee": {
            "name": "Max Mustermann",
            "street": "Musterstraße 1",
            "zipCode": "97332",
            "city": "Würzburg",
            "countryCode": "DE"
        }
    }
}
```

Validate a JSON Message

To validate a message against the conizi model, go to the validation tool under https://preproduction.dev.conizi.io/api/input/Test/Validate (for preproduction), choose the right model, select a json file and push "Validate".

The Validation result is shown, possible errors a listed.

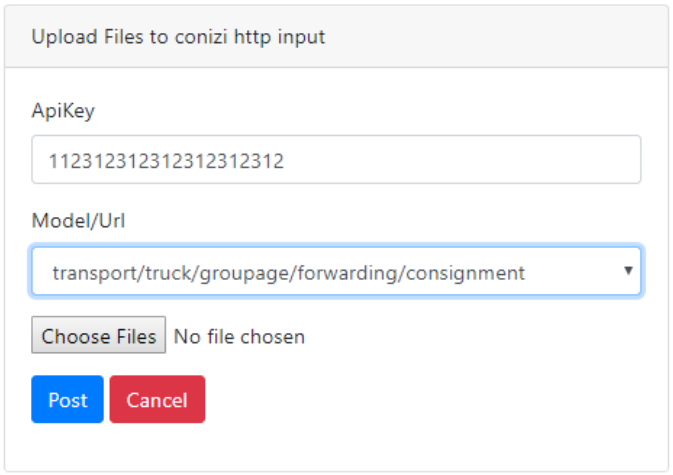


Send a message to the API

To send a message to the API, conizi makes a tester tool available under https://preproduction.dev.conizi.io/api/input/Test (for preproduction)

You should insert your valid API Key for your Site/App and to choose a model. Select a json file and push "Post". If always was right a Correlation ID is shown like "CID12312312312312233".

This ID is used to track the process in conizi. If something goes wrong, the error is shown.



If the testing tool / API reports no errors and the Correlation ID is shown, you could only be sure that the model is valid and the syntax was right. You have no guarantee if the content could by mapped or routed in a right way. conizi uses async. processing for all operations.

Develop against the API

In our examples above, we have used the Pre-Production environment, we will also do it hiere.

The base URL for the HTTP Input API is https://preproduction.dev.conizi.io/api/input/model (we need to use HTTP POST)

If a consignment should be used, the full URL should look like the following: https://preproduction.dev.conizi.io/api/input/model/**transport/truck/groupage/forwarding/consignment**

The path of the URL corresponds to the model (here consignment), so conizi now knows what kind of message should be received from you.

An example POST against the API looks like this. A custom header "x-api-key" must be set, use your available API Key from conizi, the JSON File (consignment) must be send as binary content in the body of the message.

curl POST

```
curl -X POST \
  https://preproduction.dev.conizi.io/api/input/model/transport/truck/groupage/forwarding/consignment \
  -H 'x-api-key: 112312312312312312312' \
  -H 'x-validate: True'
```

If the message was received successfully, the API responds with the Correlation ID and a HTTP Status Code 200. Otherwise the response is an error with a HTTP Status Code 400, 401 etc. Check possible status codes via our documentation https://preproduction.dev.conizi.io/api/input/swagger/index.html. If a HTTP Status Code 400 is returned, also a DTO will be generated (see the swagger link). If a validation should be executed directly on receipt of the message, the "x-validate" header must be set, see example above. On Validation Errors a DTO and HTTP Status Code 400 will be returned. For example

Response DTO

```
{
    "message": "Validation errors have occurred",
    "correlationId": "CID68839f39722f481889167427c4775ec6",
    "validationErrors": [
        "Property 'events' has not been defined and the schema does not allow additional properties. Path
'events', line 4, position 11.",
        "Required properties are missing from object: shippingDate, shippingPartner, receivingPartner. Path
'', line 3, position 1."
    ]
}
```

The direct validation works only with conizi JSON formats, other formats like FORTRAS, EDIFACT... could not be validated.