TTP-Dispatcher for Docker

Current Docker-Version of TTP-Dispatcher: 2023.1.4

Authors: Peter Penndorf
Version 2023.1.4 kontakt-ths@uni-greifswald.de

Contents

Introduction

The following manual or "how to" will describe the installation and usage of the TTP-Dispatcher for docker. The TTP-Dispatcher for docker is a fully configured TTP with E-PIX, gICS, gPAS and TTP-Dispatcher with a REST-Interface and all web frontends.

Version

E-PIX	gICS	gPAS	TTP-FHIR Gateway	TTP-Dispatcher	TTP-REST-Interface
2023.1.3	2023.1.4	2023.1.2	2023.1.3	2023.1.4	2023.1.4

Requirements

- **Linux-based system** with **docker** and **docker-compose** (v.2) installed
- technical knowledge

Advanced configuration

Modify the configuration of the TTP-dispatcher in order to e.g. provide services outside the virtual machine running the containers. Please change e.g. protocol, external port (default: `8080` , IP-address (default: `192.168.56.1`) or domain name of server.

Where?
`/sqls/03_init_database_dispatcher.sql`

What?
The SQL-Variable `@customVariables` can be changed. Just make sure the value string is a valid java-property-file.

Note: Please only modify every first-line underneath a comment. Other lines shall not be modified.

Demo-Config/Data

Initially the containers do not have any demo-data. If you wish to use those, you have to copy the sql-files from the demo-directory to the sql-directory before the first start or run them manually with a sql-client. The demo-SQLs also distinguish between *init* and *data*. *Init*-files only insert configurations and *data*-files insert some sample data. `Caution: The *data*-files may override existing data and are only for demonstration!` The following sections will describe the configurations.

Studies

There is a demonstrative study configured to show the basic configuration of a study in the TTP.

- **demo**
  - consent templates
    - **MDAT** (version: 1.0, mandatory)
      - modules
        - **MDAT_COLLECT** (version: 1.0)
          - policies
            - **p_storage_and_use_of_data** (version: 1.0)
  - a **consent is mandatory** for adding a patient
  - identifying fields
    - mandatory (needed for every function with identifying data like `addPatient`, `addConsentByPatient` etc.)
      - `person.firstName`
      - `person.lastName`
      - `person.gender`
      - `person.birthdate`
    - optional (are additionally displayed in `addPatient`-Form)
      - `contact.phone`
      - `contact.email`

Systems (external systems using the TTP)

- mdat (apiKey: **mdat**)
- lims (apiKey: **lims**)
- admin (apiKey: **admin**) -> can be used with every system and has all rights to do everything

The following table shows the permissions configured for each system.

Cr = create, Ca = call

functions \ apiKeys	**mdat**	**admin**	**lims**
Session	Cr	Cr	Cr
addPatient	Cr, Ca	Cr, Ca	
addConsent	Cr, Ca	Cr, Ca	
searchPatient	Cr, Ca	Cr, Ca	
addConsentByPatient	Cr, Ca	Cr, Ca	
queryPolicies		Cr, Ca	
addConsentScan		Cr, Ca	

functions \ apiKeys	mdat	admin	lims
getConsentDocument		Cr, Ca	
managePatient		Cr, Ca	
requestPSN		Cr, Ca	Cr, Ca
requestPsnByPatient		Cr, Ca	
requestPatientByIdentifier		Cr, Ca	
externalPatientMerge		Cr, Ca	
addPatientIdentifier		Cr, Ca	
confirmNotification		Cr, Ca	
getConsentStatus		Cr, Ca	
updateIDAT		Cr, Ca	

Events, Studies and Functions

Events are some kind of grouping functionality and configuration. When an external system wants to do something it 'says' what function it wants to call and why (event). Therefore, a number of combinations of functions and events are possible. Although it is possible to configure multiple studies, where each study can have its own event configuration. The following matrix shows the configuration and all combinations. The functions have the abbreviations from the interface specification.

Studies: demo = d

events	Studies	AP	AC	SP	ACP	QP	ACS	GCD	MP	RP	RPP	RPID	EPM	APID	GCS	UI
demo.search	d			x												
demo.recruitment	d	x	x													
demo.manage	d			x			x		x							
demo.new-consent	d		x		x											
demo.addScan	d						x									
demo.consentStatus	d							x							x	
demo.resolvePsn	d											x				
demo.transferData	d					x										
demo.addPatientIdentifier	d													x		
demo.requestPsn	d									x	x					
demo.updateIDAT	d															x

Web-UI for Configuration

We have started to implement a Web-UI for the configuration of the dispatcher, especially the XML-Config. This is constantly being improved and is still missing some functionality and not ready for production. Please use with caution, but feel free to test and give feedback.

The UI is available at the Web-Context `/config-web` and needs a login which is by default configured with username `admin@ttp` and password `ttp-tools`. The password can be changed in the database `gras` in table `user` with a SHA-256-Hash.

jMeter Examples

Apache jMeter is a tool to create integration tests. The jMeter-Examples are a way to visualize the communication for each function of the REST-API.

General Instructions

- download jMeter here if not already done (requires version 5.0+)
- start jMeter and load the file "./demo/ttp-dispatcher-ci_tests.jmx" from the zip-package

The jMeter-Tests are adapted to the configuration that comes with the init-demo-SQL-files. If you not used those, the requests won't work. But feel free to use the tests as template for your own configuration.

Usage

- variables and functions can be enabled/disabled as you like
- the user-defined variables are categorized as follows
  - **all Parameter** -> **adapt** `server.domain` **(domain/ip-address) and** `server.port` **to your environment** (default: `192.168.56.1`, port: `8080`)
  - **Vars & Parameter** (static) -> do not change. maybe `callback` can be changed to a service you want to get callbacks, otherwise the callbacks are visible in the logs
  - **ApiKey** -> the apiKey
  - **Patient** -> identifying data of a patient
  - **System** -> system-specific vars
- exactly one of each category must be enabled
- every function needs an existing patient except
  - `AddPatient`
- the following functions need also a valid consent for the patient to get a successful response
  - `AddConsentScan`
  - `RequestPatientByIdentifier`
  - `RequestPSN`

Validation of installation 'demo-study'

After successful installation the following web-frontends are available via browser.

- **E-PIX** -> `http://YOUR-IP:PORT/epix-web`
- **gICS** -> `http://YOUR-IP:PORT/gics-web`
- **gPAS** -> `http://YOUR-IP:PORT/gpas-web`
- **dispatcher** -> `http://YOUR-IP:PORT/ths-web/html/public/common/authenticated.xhtml`
- **dispatcher-config** -> `http://YOUR-IP:PORT/config-web`
- **tablet-consent** -> `http://YOUR-IP:PORT/ths-web/html/public/code/input.xhtml`
- **dispatcher-admin** -> `http://YOUR-IP:PORT/ths-web/html/admin/demo/demo.xhtml?apiKey=admin`

Likewise, functionalities of the **TTP-FHIR Gateway** for E-PIX, gPAS and gICS can be addressed.

Examples:

- Query of all persons (and assigned identities) of the study "demo" in JSON format via FHIR Search.

```
http://YOUR-IP:PORT/ttp-fhir/fhir/epix/Person?organization:identifier=demo&_include=Person:link&_format=json
```

- Query all Consent resources in which the policy "MDAT_wissenschaftlich_nutzen_EU_DSGVO_NIVEAU" is permitted via FHIR Search (Scope: German Medical Informatics Initiative).

```
http://YOUR-IP:PORT/ttp-fhir/fhir/gics/Consent?domain:identifier=demo&mii-provision-provision-
code=urn:oid:2.16.840.1.113883.3.1937.777.24.5.3|2.16.840.1.113883.3.1937.777.24.5.3.8&_format=json
```

Note: to use `|` in GET-Requests the property `WF_DISABLE_HTTP2=TRUE` in `ttp_commons.env` has to be set.

Version-Upgrade

Retain data

Because of the structure and functionality of the docker images, retaining the data needs a bit more effort. The following describes how to:

WARNING (not for every upgrade):

- Manually created E-PIX-Domains cannot be read by the new version, because the structure of the configuration may be different. The upgrade-script overrides only the domain(s) automatically created by the initial demo-ttp. The new configuration structure can be “copied” of those domains to adapt the manually created domain configurations.
- The dispatcher configuration will also be updated (version 1). In case this version was changed, you might want to do a backup (e.g in a version 2). After the upgrade, the version 1 will be loaded. You need to adapt your custom configuration to the new structure of version 1. Afterwards you can switch to the adapted version by setting the `dispatcher.config.current` to the new adapted version.

How-To:

1. Startup the old version
2. Connect to the database with a mysql-client
3. Run the upgrade-script found in the folder `sqls-upgrade\_\*`
4. Shutdown the containers
5. Unzip the new version in a new folder
6. Copy or move the folder `mysql_data` from the old version to the new unzipped folder
7. Startup the new version
8. Optional: adapt the custom configurations in the e-pix/dispatcher

Validation of incoming information

There is the possibility to validate incoming information like persons, contacts and consents individually. A generic approach was chosen because it may be different in each study.

Concept

ValidationRule

Is the smallest "unit" and contains the rule itself and the error message. There are the following attributes:

- **checkOgnl**: an Ognl expression that must return a boolean. If this is False, then the error message is returned. In this expression the input can be accessed with 'value'. In addition, methods of the ValidationMethods class can be used, see appendix
- **errorMessage**: ErrorMessage. If this is specified as a property, then parameters can be specified additionally. Ex: **config.common.validation.zip|10000,99998** -> the parameters are replaced as {0}...{n} in the correct order. Multilingualism would also be possible as a property. The property is retrieved from the configuration table by the dispatcher with the key **dispatcher.web.text.custom**. This must result in a valid ResourceBundle. The keys must start with the language code
- **useLanguageProperties**: must be set to **true** if error message contains a property. Default: false
-

ValidationProfile

ValidationProfiles bundle ValidationRules for an InputType/Model/Object and for specific functions. The following attributes exist:

- **inputType**: The type of model/input that the profile can validate. Currently there are only these values: **IDENTITY, CONSENT**.
- **functions**: A list of functions that can or should be applied to this profile. Whether it is actually applied is to be determined in the StudyConfig.
- **rules**: A list of ValidationRules. The order is taken into account.
-

In the StudyConfig

ValidationProfiles can be applied here. To do this, specify a list of profiles under **studyConfig->vConfig->validationProfiles**.

Example Config

```
<validationMetaConfig>
    <rules>
        <entry>
            <string>zipcode</string>
            <validationRule>
                <checkOgnl>value.contacts.size()>0 and isNumeric(value.contacts[0].zipCode) and
value.contacts[0].zipCode.length()==5</checkOgnl>
                <errorMessage>Postleitzahl muss exakt 5 Zahlen haben</errorMessage>
            </validationRule>
        </entry>
        <entry>
            <string>kvNr</string>
            <validationRule>
                <checkOgnl>value.value8 != null and value.value8.length==10</checkOgnl>
                <errorMessage>validation.personkv.empty|10</errorMessage>
                <!-- in dispatcher.web.text.custom ->
                de.validation.personkv.empty = KV-Nummer muss genau {0} Zeichen lang sein
                en.validation.personkv.empty = ...
                -->
            </validationRule>
        </entry>
    </rules>
    <profiles>
        <entry>
            <string>patientAll</string>
            <validationProfile>
                <inputType>IDENTITY</inputType>
                <functions>
                    <string>CREATE_PATIENT</string>
```

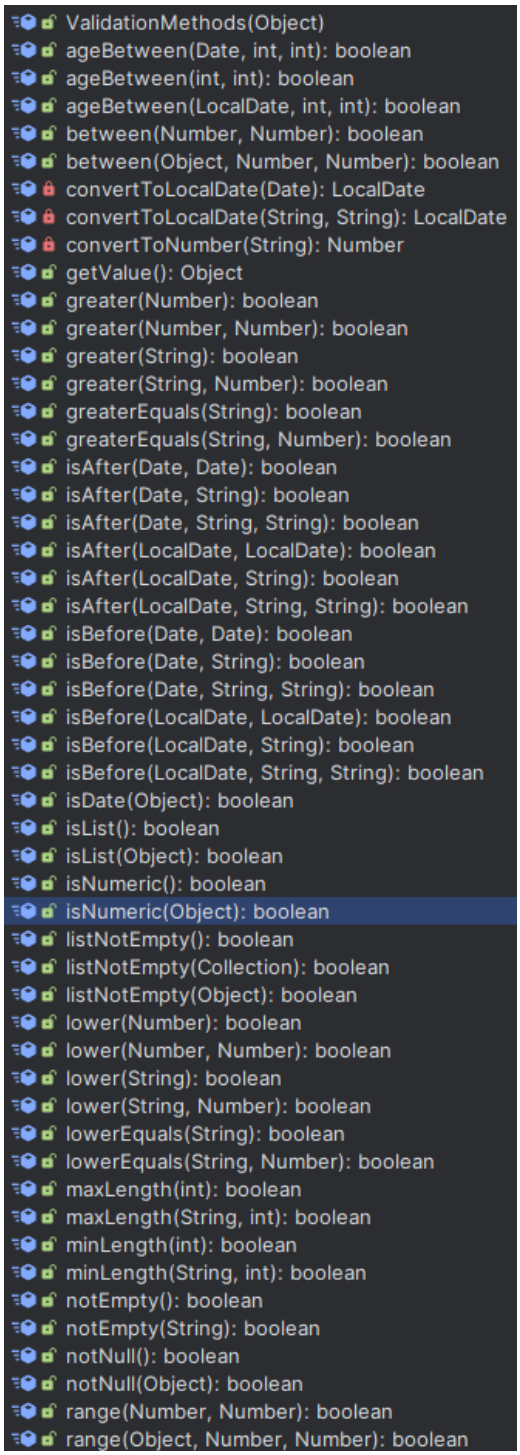
```
                </functions>
                <rules>
                    <string>zipcode</string>
                    <string>kvNr</string>
                </rules>
            </validationProfile>
        </entry>
    </profiles>
  </validationMetaConfig>

...

<studyConfigs>
        <entry>
            <string>demo</string>
            <org.icmvc.ttp.dispatcher.common.config.study.StudyConfig>
                <id>demo</id>
                <consentSignerIdTypes class="set"/>
                <vConfig>
                    <minAge>0</minAge>
                    <maxAge>999</maxAge>
                    <validationProfiles>
                        <string>patientAll</string>
                    </validationProfiles>
                </vConfig>
                ...
```

Appendix

ValidationMethods



FAQ

Is it possible to change the dispatcher-config while the container is running?

Yes, in 2 different ways.

1. via the prototyped web-ui at `/config-web` this approach does not contain all configuration possibilities yet.
2. in the database itself. Follow these instructions:

- connect to the database `ttp_dispatcher` with the user `root` and password `root` -> for all database connections see the file `./jboss/ttp.cli`
- run an update statement

```
UPDATE \`configuration\` SET \`value\`="{insert config xml}"
WHERE \`configKey\`='dispatcher.config.1';
```

- replace the value with a new configuration xml. After a maximum of 30s the config will be reloaded and should be available. If it is not valid, you will get an error in the log.
- If you changed something in the `notificationConfig`, then a restart is required.

Credits

Concept and implementation: P. Penndorf, A. Blumentritt, L. Geidel
Web-Client: P. Penndorf, A. Blumentritt
Docker: R. Schuldt, F. M. Moser

License

Contact: https://www.ths-greifswald.de/kontakt/

Supported languages

German, English