1. Introduction
This document provides additional information regarding the technical details of the P2T and T2P webservice described in the BPM 2021 resource paper [1]. This documentation is for both domain users who only want to access the publicly deployed webservice and developers who want to integrate the P2T and T2P webservice into their own application, adapt the source code and/or deploy a tailored, on-premises version of the webservice.

2. Infrastructure
The source code for both webservice is hosted on GitHub under https://github.com/tfreytag/P2T and https://github.com/tfreytag/T2P, respectively. Both applications require at least Java version 11, the Springboot framework [2], and Maven [3] as build tool. The Jenkins-based pipelines for both services build an associated Docker image and push the image to DockerHub.

![Diagram](image-url)
After completion, they can be found online under https://hub.docker.com/r/woped/process2text and https://hub.docker.com/r/woped/text2process, respectively. The docker images are tagged according to the version of the underlying Maven build file pom.xml. Information on how to use those pre-built images is provided in the following sections. Figure 1 shows an architectural overview of the build and deployment process.

3. Practical Usage and Web Frontend

Both webservices contain an simple embedded UI which can be directly accessed via the URLs and https://woped.dhbw-karlsruhe.de/p2t/ and https://woped.dhbw-karlsruhe.de/t2p/ respectively. Users can perform a transformation there without having to implement a client software. The P2T webservice provides a simple, still very basic UI with a text field where the content of the native process model language (currently BPMN or PNML) can be pasted. Clicking the “Submit” button will generate the text representation of the process model, which will be displayed in the box at the bottom (see Figure 2). Note that the output is not in plain text format but XML-tagged in order to allow re-associating text phrases with model elements on client side.

![Process2Text web interface](image)

Figure 2: Process2Text web interface
The T2P frontend provides a text input field for the process model description. Clicking the “Generate” button will call the T2P algorithms and display the generated process model in the process modelling language chosen (PNML or BPM) – see Figures 3 and 4.

4. Integration of P2T and T2P Webservices
Developers who want to call the webservices from their own code must go through an HTTP client to perform the according REST calls. There are multiple HTTP clients for various programming languages and the access to the webservices should work with any existing HTTP client. Possible HTTP software for Java can be found in [4] and [5]. Regarding the definition of the endpoints, both webservices provide an OpenAPI (former called Swagger) specification [6] that can be used to generate the client code for many programming languages available e.g. at [7]. For developers who want to write the client code themselves, the Swagger UI of the webservices can be used to gain insight into
parameters, return types and other configuration properties. The Swagger UI for the P2T and T2P webservices can be found at the URLs https://woped.dhbw-karlsruhe.de/p2t/swagger-ui/ and https://woped.dhbw-karlsruhe.de/t2p/swagger-ui/ respectively.

5. Customizing the P2T and T2P Webservices
If users find bugs or require additional features, they can raise an issue on GitHub. The developing team at the institution of the authors of this paper will then be giving their best to fix the bug or implement the new feature in time. If, however, the desired change is too complex or must be implemented very quickly, the users can either simply apply the changes themselves and create a pull-request, or they can fork the git repository and implement the code change there. If they want to deploy this version of the webservice, they must build the docker image themselves. As a critical precondition, docker must be installed on the system that is used to build the docker image and the system that is used to deploy the webservice. Installation instructions for any operating system can be found on the docker website [8].

6. Building and Deploying the P2T and T2P Webservices
In the first step, the docker image must be built. The required Docker files are in the root folder of the associated webservice code. Navigating there on the command line and executing the command

    docker build -t myWebservice:myWebserviceVersion

will create the docker image. If the image should be pushed to a container registry later, the image must be tagged according to the username and repository name of the container registry as shown in [9]. Executing the command

    docker push myUsername/myWebservice:myWebserviceVersion

will push the image to the container registry. Beforehand, `docker login` for the target container registry is required. Now, this image can be pulled from the system, where the webservice(s) should be hosted. Running the command

    docker run myWebservice:myWebserviceVersion

will pull docker image and start the webservice. Regarding the base paths and ports of the webservices, the application.yml file in either of the webservice repositories can be used to adapt those values.

7. Conclusion
This document describes how to call the two NLP webservices P2T and T2P from your own software as well as how to customize, build and deploy the webservices on your own server. For questions regarding the described procedures, do not hesitate to contact the authors via the email addresses provided above.

References


[5] https://openjdk.java.net/groups/net/httpclient/recipes.html


[8] https://docs.docker.com/get-docker

[9] https://docs.docker.com/docker-hub/repos