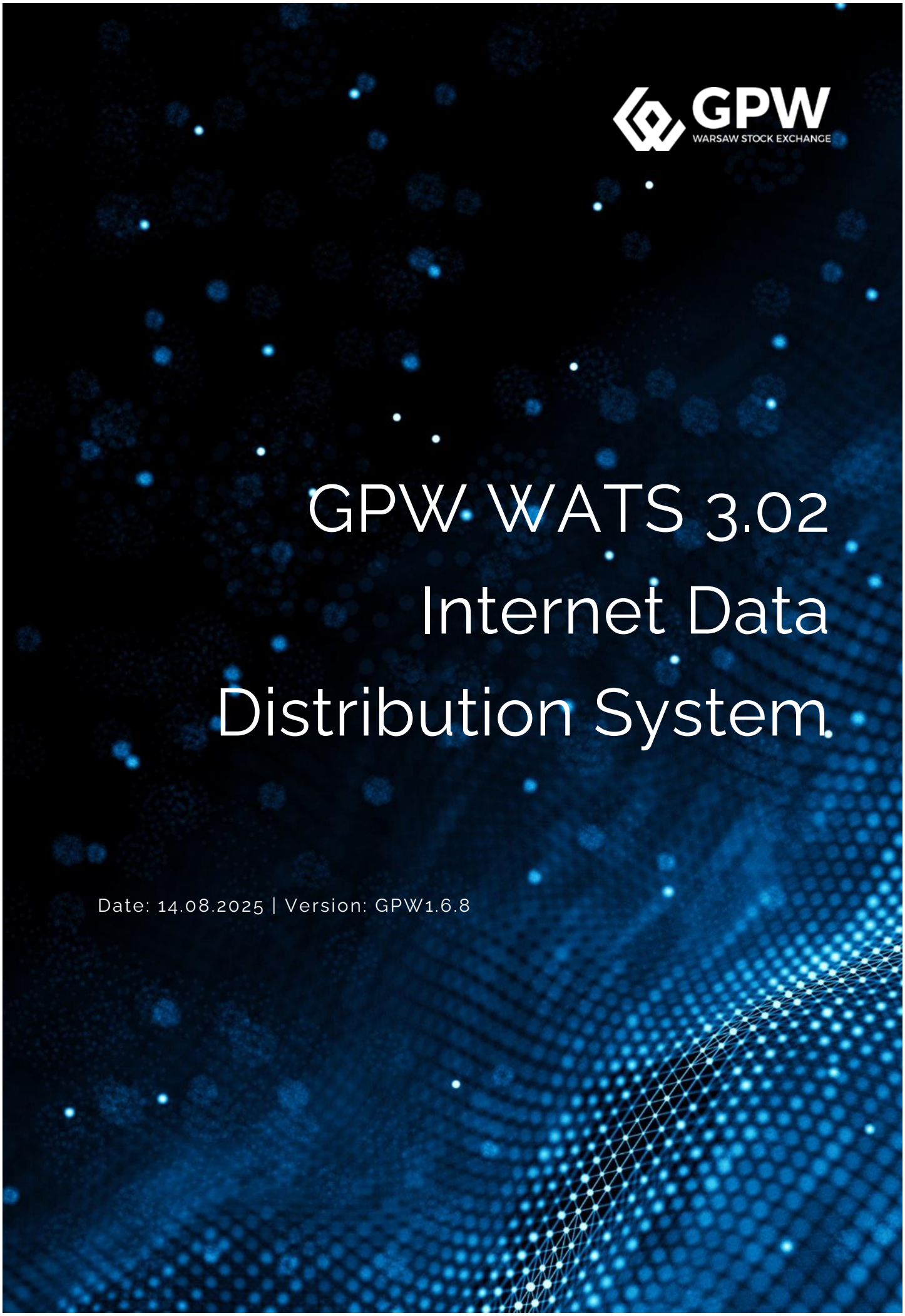

GPW WATS 3.02 Internet Data Distribution System

Date: 14.08.2025 | Version: GPW1.6.8

CONTENTS

1. DISCLAIMER

This document is for information purposes only and does not form any part of contractual documentation.

Reasonable care has been taken to ensure details contained within are accurate and not misleading at the time of publication. Warsaw Stock Exchange is not responsible for any errors or omissions contained in this document.

Warsaw Stock Exchange reserves the right to treat information contained in this document subject to later change without prior notice.

This document contains confidential information to Warsaw Stock Exchange and may not be reproduced, disclosed, or used in whole or part, in any manner, without prior written consent from the owner of this document. Information included in this document shall be maintained and exercised with adequate security measures necessary to protect confidential information from unauthorized access or disclosure.

In case of sections of documentation at a **High** level work progress according to the current version of *GPW WATS Advancement of Documentation*, Warsaw Stock Exchange will endeavor to limit changes to these sections of documents to those related to:

1. correcting errors in the documentation or in the software;
2. clarification of the documentation content or removing ambiguity;
3. implementation of approved change requests or;
4. regulatory changes.

2. PREFACE

This section describes the basic information about GPW WATS Internet Data Distribution System. You can learn about the target audience, the document purpose and all the necessary documents you should read in relation to this Specification.

2.1. TARGET AUDIENCE

This document is dedicated to entities, such as Independent Software Vendors who produce software integrated with GPW.

2.2. DOCUMENT'S PURPOSE

This document describes how to obtain mTLS certificate and interact with our Rest and Stream APIs.

2.3. ASSOCIATED DOCUMENTS

Please check the following documents to learn about the construction of Trading System.

- GPW WATS 1.01 Trading System.

Please check the documentation of the trading protocols supported by GPW WATS.

- GPW WATS 2.01 Native Order Gateway Specification (this document),
- GPW WATS 2.02 FIX Order Gateway Specification.

Please check the description of the communication with Data Distribution Service.

- GPW WATS 3.01 Market Data Protocol.

Please check the description of the communication with Internet Data Distribution System.

- **GPW WATS 3.02 Internet Data Distribution System** (this document),
- GPW WATS 3.03 Streaming Messages for IDDS,
- GPW WATS 3.04 Rest API Messages for IDDS.

Please check the additional documentation, which explains other services provided within GPW WATS.

- GPW WATS 4.01 Drop Copy Gateway,
- GPW WATS 4.02 Post Trade Gateway,
- GPW WATS 5.01 Risk Management Gateway.

Please check the additional documentation describing the following:

- GPW WATS 2.03 Rejection Codes,
- GPW WATS 2.04 BenDec Message Definition Format,
- GPW WATS 4.03 Contract Notes,
- GPW WATS 6.01 Connectivity,

- GPW WATS 6.02 (ENG) Short Code Record Keeping,
- GPW WATS 6.02 (PL) Mapowanie Short Code,
- GPW WATS 6.03 Short-Long Mapper User Guide.

It is recommended to read ***GPW WATS 1.01 Trading System*** document first.

3. Document History

Version	Date	Description
0.61	29.02.2024	Initial publication of the document.
0.62	25.03.2024	Publication of v0.62. No changes in the document.
1.0	30.04.2024	Publication of v1.0. No changes in the document.
1.1	28.06.2024	Indexes topic have been added: • indexes • indexes_delayed The following topics have been removed: • bbo_indexes • bbo_indexes_delayed • fd_indexes
1.1.2	9.08.2024	Publication of v1.1.2. No changes in the document.
1.2	18.09.2024	Publication of v1.2. No changes in the document.
1.3	17.10.2024	**Rest API** Example response in 5.1 Using Rest API has been updated. Deleted endpoint: /api/v1/reference/{date}/markets/{mic}/kids **Streaming** Deleted topics fd_refdata_nextsession
1.4	6.12.2024	Unpublished version. No changes in the document.
1.5	3.02.2025	Publication of v1.5. No changes in the document.
1.5.4	30.04.2025	Publication of v1.5.4. No changes in the document.
1.6	26.05.2025	Publication of v1.6. No changes in the document.
1.6.5	18.06.2025	Publication of v1.6.5. No changes in the document.
1.6.6	10.07.2025	Publication of v1.6.6. No changes in the document.
1.6.7	7.08.2025	Topics have been renamed. Authentication method has been changed (mTLS).
1.6.8	14.08.2025	Publication of v1.6.8. No changes in the document.

4. Overview of IDDS

IDDS is an Internet Data Distribution System. It is an alternative data distribution channel to GPW WATS DDS service that operates online.

4.1. Connectivity

This system enables to access data by the use of two protocols:

- Rest API,
- Streaming API (Kafka protocol).

4.1.1. AUTHENTICATION

Applications connects to IDDS (both REST services and Kafka) only using mTLS authentication (the application must present a trusted certificate). Steps to follow:

a) Certificate Signing Request:

Create the configuration file cert.cnf as shown below:

```
[req]
default_days = 738
prompt = no
default_md = sha256
req_extensions = req_ext
distinguished_name = dn
[dn]
C=PL
ST=Mazowieckie
L=Warszawa
O=GPW
OU=IT
emailAddress=user@company.com
CN = <company>-idds@<environment>
[req_ext]
keyUsage = nonRepudiation, digitalSignature, keyEnciphermentkeyUsage
extendedKeyUsage = clientAuth
```

Generate private key and csr file:

```
openssl req -newkey ec:<(openssl genpkey -genparam -algorithm ec -pkeyopt
ec_paramgen_curve:P-384) -keyout priv.key -out klient.csr -config
cert.cnf
```

b) Customer provides only CSR (without priv.key) to sign by CA GKGPW (via Jira ticket).

c) GPW provides signed certificate and necessary CA certificates.

5. Introduction to REST

Rest API shares data using HTTP protocol secured with SSL. To obtain data you need to use your mTLS certificate (to see how to generate your certificate, check *4.1.1 Authentication*) and an URL of the data package you would like to access. The data received with Rest API is especially useful while Reference Data is required, as you need to repeat your request to obtain the most up to date data.

AVAILABLE URLS:

Reference Data

- /api/v1/reference/{date}/market-structures,
- /api/v1/reference/{date}/calendar,
- /api/v1/reference/{date}/markets/{mic}/instruments/list,
- /api/v1/reference/{date}/markets/{mic}/instruments,
- /api/v1/reference/{date}/instruments,
- /api/v1/reference/{date}/indexes,
- /api/v1/reference/{date}/markets/{mic}/accrued-interest-tables,

Market Data

- /api/v1/snapshot/instrument/{id},
- /api/v1/snapshot/index/{id},
- /api/v1/summary/instrument,
- /api/v1/summary/index.

Common

- /api/v1/schema/avro.

For the complete instructions and description of the received data for each URL, check *GPW WATS 16.03 Rest Messages for IDDS*.

5.1. Using Rest API

Request with certificate:

Example request

```
curl -X GET "https://[host]:[port]/[rest endpoint]" \
     --cert client-cert.pem \
     --key client-key.pem \
     --cacert ca-cert.pem
```

Example response

When you make a successful GET request to the /api/v1/reference/{date}/market-structures endpoint, the response includes a collection of MarketStructureMsg objects. Here's an example of what the response might look like:


```
[
  {
    "id": 12345,
    "parentId": 0,
    "mic": "XABC",
    "name": "Primary Market",
    "marketModelType": "CONTINUOUS_TRADING"
  },
  {
    "id": 12346,
    "parentId": 12345,
    "mic": "XABC",
    "name": "Secondary Market",
    "marketModelType": "AUCTION"
  }
]
```

6. Introduction to Streaming

Streaming API shares data using Kafka protocol. Clients are authenticated using mTLS. You receive live data as long as you are subscribed to one on the following data topics.

To connect, you have to configure your Kafka Client via client.properties.

6.1. Topics

Common:

eUAT	PROD-Bis	PROD (prePROD)
mt-idds-avro-schema-e01	mt-idds-avro-schema-b01	mt-idds-avro-schema-p01
mt-idds-fd-refdata-e01	mt-idds-fd-refdata-b01	mt-idds-fd-refdata-p01
mt-idds-fd-refdata-nextsession-e01	mt-idds-fd-refdata-nextsession-b01	mt-idds-fd-refdata-nextsession-p01

Full Data Live:

eUAT	PROD-Bis	PROD (prePROD)
mt-idds-fd-bonds-e01	mt-idds-fd-bonds-b01	mt-idds-fd-bonds-p01
mt-idds-fd-derivatives-e01	mt-idds-fd-derivatives-b01	mt-idds-fd-derivatives-p01
mt-idds-fd-equities-e01	mt-idds-fd-equities-b01	mt-idds-fd-equities-p01
mt-idds-fd-structures-e01	mt-idds-fd-structures-b01	mt-idds-fd-structures-p01

BBO Live:

eUAT	PROD-Bis	PROD (prePROD)
mt-idds-bbo-bonds-e01	mt-idds-bbo-bonds-b01	mt-idds-bbo-bonds-p01
mt-idds-bbo-derivatives-e01	mt-idds-bbo-derivatives-b01	mt-idds-bbo-derivatives-p01
mt-idds-bbo-equities-e01	mt-idds-bbo-equities-b01	mt-idds-bbo-equities-p01
mt-idds-bbo-structures-e01	mt-idds-bbo-structures-b01	mt-idds-bbo-structures-p01

BBO Delayed:

eUAT	PROD-Bis	PROD (prePROD)
mt-idds-bbo-bonds-delayed-e01	mt-idds-bbo-bonds-delayed-b01	mt-idds-bbo-bonds-delayed-p01
mt-idds-bbo-derivatives-delayed-e01	mt-idds-bbo-derivatives-delayed-b01	mt-idds-bbo-derivatives-delayed-p01
mt-idds-bbo-equities-delayed-e01	mt-idds-bbo-equities-delayed-b01	mt-idds-bbo-equities-delayed-p01
mt-idds-bbo-structures-delayed-e01	mt-idds-bbo-structures-delayed-b01	mt-idds-bbo-structures-delayed-p01

eUAT	PROD-Bis	PROD (prePROD)
mt-idds-indexes-e01	mt-idds-indexes-b01	mt-idds-indexes-p01
mt-idds-indexes-delayed-e01	mt-idds-indexes-delayed-b01	mt-idds-indexes-delayed-p01

Indexes:

eUAT	PROD-Bis	PROD (prePROD)
mt-idds-indexes-e01	mt-idds-indexes-b01	mt-idds-indexes-p01
mt-idds-indexes-delayed-e01	mt-idds-indexes-delayed-b01	mt-idds-indexes-delayed-p01

6.2. Client Configuration

The example below illustrates how to configure a Kafka client to establish a connection with IDDS. It is based on the console client distributed with Kafka.

Client properties file

Instructions for generating the keystore and truststore, along with additional examples, are provided in the examples.

```
security.protocol=SSL
ssl.keystore.location=keystore.p12
ssl.keystore.password=<keystore_password>
ssl.truststore.location=truststore.p12
ssl.truststore.password=<truststore_password>
```

Receive Avro Schema

```
bin/kafka-console-consumer.sh --bootstrap-server [kafka_broker_host]:[port] --
topic mt-avro_schema-e01 --from-beginning --consumer.config client.properties
```

6.2.1. READ TOPIC WITH AVRO DESERIALIZATION

Then you are successfully connected, authenticated, authorized and received avro schema. Now you can consume data topic and deserialize it.

6.2.2. EXAMPLES

How to generate truststore:

```
openssl pkcs12 -in gkgpw-rootca.pem -in gk-gpwCA1.pem -export -out
truststore.p12 -nokeys -password pass:<truststore_password>
```

How to generate keystore:

```
openssl pkcs12 -in user.pem -inkey user.key -export -out keystore.p12 -
password pass:<keystore_password>
```

Samples:

An example application for retrieving the list of topics from a Kafka cluster. Implementations of this functionality are provided using Shell, Python, and Groovy.

To establish a connection with Kafka using mutual TLS (mTLS), a configuration file (client.properties) is required. This file must be supplied as a parameter (-F) in kcat command executions.

Shell

```
#!/bin/bash

kcat -b [kafka_broker_host]:[port] -F client.properties -L | grep mt- | cut -
f2 -d\"|sort
```

Python

```
#!/usr/bin/python

from kafka import KafkaConsumer

consumer = KafkaConsumer(bootstrap_servers="[kafka_broker_host]:[port]",
        security_protocol='SSL',
        ssl_certfile='client.pem',
        ssl_keyfile='client-key-crypted.pem',
        ssl_password='client-key-password.')

for t in sorted(consumer.topics()):
    print (t)

consumer.close()
```

Groovy

```
///usr/bin/env jbang "$0" "$@" ; exit $?
//DEPS org.apache.kafka:kafka-clients:4.0.0
import org.apache.kafka.clients.consumer.KafkaConsumer

KafkaConsumer<String, String> consumer = new KafkaConsumer<>([
        'bootstrap.servers': ="[kafka_broker_host]:[port]',
        'key.deserializer':
'org.apache.kafka.common.serialization.StringDeserializer',
        'value.deserializer':
'org.apache.kafka.common.serialization.StringDeserializer',
        'security.protocol': 'SSL',
        'ssl.keystore.location': 'keystore.p12',
        'ssl.keystore.password': 'keystore_password',
])

consumer.listTopics().sort { a, b -> a.key <=> b.key }.each { t -> println
t.key }

consumer.close()
```