InvoiceSenderV3 manual

Program Version: V3.5.0
Document Version: V3.5.0
Modified: 2021-03-09

Table of contents

Document changes....3
I. Preface....4
I.1 Overview of basic flow of information....5
II. InvoiceSender working modes....6
III. Installation....7
III.1 Distribution format....7
III.2 Prerequisites....7
III.2.1 Required Java version since 29th April 2019....7
III.2.2 Required Java version since 1th April 2020....8
III.3 Installation steps....8
III.3.1 Accessing NAV production system....9
IV. Application configuration....9
IV.1 Credential configuration....9
IV.2 Optional configuration parameters....9
IV.3 Java SSL configuration....10
IV.4 Command line parameters....10
IV.5 Testing of configuration....11
V. Running of application....12
V.1 Starting the application....12
V.2 Application locking....12
V.2.1 Lock timeout feature....12
V.3 Transaction Query mode....13
VI. Housekeeping....15
VI.1 Backup and restore....15
VI.2 Internal database maintenance....15
VI.3 Reprocessing of failed files....15
VI.4 Purging of obsolete data....15
VI.5 Maintenance of Java SSL certificates....15
VII. Log files....16
VII.1 Log of results....16
VII.2 Application logs....17
VII.3 Output response structure....17
VIII. Known issues....18
VIII.1 Queries remain in ‘PROCESSING’ state....18
VIII.2 Transmission: “SSLHandshakeException: Received fatal alert: handshake_failure”....18
VIII.3 Transmission: “NoSuchAlgorithmException: SHA3-512 MessageDigest not available”. 19
Appendix A.) ChangeLog of current version....20
Appendix A.1) Update steps V1.1.3 → V1.1.4, V1.1.5....21
Appendix A.2) Update steps V1.1.5 → V1.1.6....22
Appendix A.2) Update steps V1.1.6 → V1.1.7....22
Appendix A.3) Update steps V1.1.6, V1.1.7 → V1.1.8....22
Appendix A.4) Update steps V1.1.8 → V2.0.0....23
Appendix A.5) Update steps V1.1.[0,1] → V2.0.2....23
Appendix A.6) Update steps V2.0.[0,1] → V3.0.0....23
Appendix B.) Distributed files and their descriptions....24

Appendix C.) Used OpenSource components and licensing.....25
Appendix D.) Sample of InvoiceSender.properties config.....26
Appendix E.) In case of program errors.....28

Document changes

Program Version	Document version	Date	Author	Summary of change
V1.1.2	V1.0	2018-05-18	János TŐBERLING	Initial version of manual
V1.1.2	V1.1	2018-05-18	János TŐBERLING	Locking, housekeeping and reprocessing description, typo fixing.
V1.1.3	V1.2	2018-05-27	János TŐBERLING	V0.13 Api Changes – compressedContent required tag
V1.1.4	V1.3	2018-06-26	János TŐBERLING	Error Handling changes, PROD system setup preparation, Software tag changes
V1.1.4	V1.4	2018-06-27	János TŐBERLING	Java version on rume.sh, Production validation set to false
V1.1.5	V1.5	2018-06-27	János TŐBERLING	MaxQueryAttempts feature
V1.1.6	V1.6	2018-07-19	János TŐBERLING	Adding annulment feature Fixing logs/ResultLog.csv logging.
V1.1.7	V1.7	2018-10-18	János TŐBERLING	Reading input XML files in alphabetical order Adding locktimeout feature
V1.1.8	V1.8	2019-04-19	János TŐBERLING	NAV API v1.1 changes NAV TLSv1.2 changes
V2.0.0	V2.0.0	2019-12-22	János TŐBERLING	NAV API V2.0 fork H2 DB Request structure change
V2.0.0	V2.0.1	2020-03-04	János TŐBERLING	Changed Java requirements
V2.0.2	V2.0.2	2020-08-31	János TŐBERLING	Query transaction feature
V3.0.0	V3.0.0	2020-11-29	János TŐBERLING	Implement NAV API V3.0
V3.5.0	V3.5.0	2021-03-09	János TŐBERLING	Support electronic Invoicing by NAV API V3.0 <electronicInvoiceHash>

I. Preface

From 2018. July 1st Hungarian Tax Authority (NAV) requests that all the invoices reaching 100.000 HUF VAT (ÁFA) and referring to companies (none for individual persons) have to send to the tax authority immediately during invoice generation process.
NAV defined an interface – on where invoice data can be send over Internet.
The interface itself consists of 2 layers:
– Data layer is to contain invoice data in XML format (Data exchange)
– API layer is to transfer embedded Data XML over the internet (Communication layer)

InvoiceSender application is created to handle NAV's API layer communication.

As from 2nd May 2019 NAV is forcing V1.1 application protocol in their production system – the previous V1.0 transmissions will be rejected.[1]

Along with theV1.1 protocol change the NAV is forcing TLSv1.2 HTTPS submission since 29th April 2019. [2] TLSv1.2 has Java version prerequisites – please check chapter "*III.2.1 Required Java version since 29th April 2019*". In case the system's default version is not supported alternate Java runtime required to be installed. In case of connection issues please refer to chapter "*VIII.2 "SSLHandshakeException: Received fatal alert: handshake_failure" during transmission*".

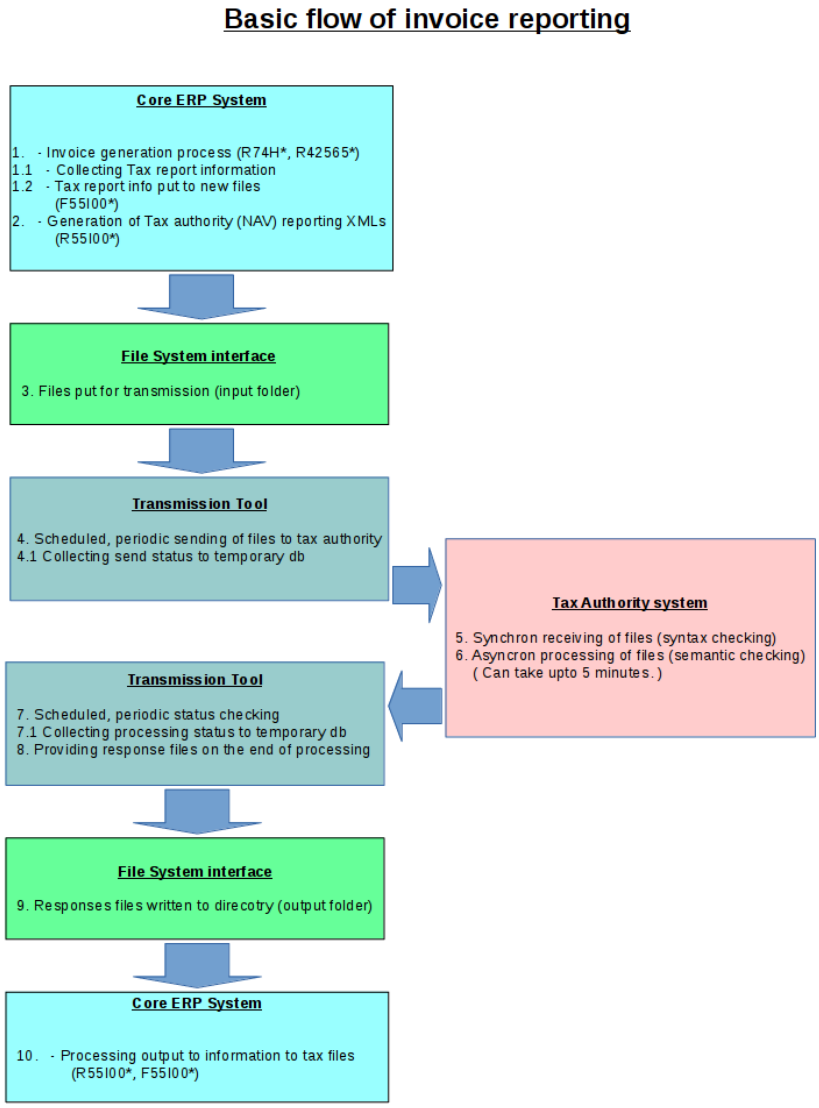
As from 1st April 2020 NAV is forcing V2.0 application protocol in their production system – the previous V1.1 transmissions will be rejected.[3]

As from 1st January 2021 NAV is opening V3.0 interface on production system. The V2.0 will be available by end of March. From 1st April V3.0 data and API interfaces are mandatory.

1 Please refer for V1.1 changes to: https://onlineszamla.nav.gov.hu/dokumentaciok
2 Please refer to TLSv1.2 changes to: https://onlineszamla.nav.gov.hu/home / News 2019-03-12
3 Please refer for V2.0 changes to: https://onlineszamla-test.nav.gov.hu/api/files/container/download/Online%20Szamla_Interfesz%20specifik%C3%A1ci%C3%B3_EN_v2.0.pdf

I.1 Overview of basic flow of information

The following figure shows how the basic flow and communication steps between different systems and components during the reporting activity. The InvoiceSender application takes apart in the flow referred as the Transmission Tool.



I.2 Support of electronic invoicing

As described in "*National Taxand Customs Administration / The NAV Online InvoiceSystemThe NAV Online Invoice SystemInvoice Data Reporting, Description of the REST API Interface & Developer's Documentation*" V3.0 03.12.2020:

"…
2. 6 Electronic invoicing support
...

2.6.1 Archiving

Uncertainties in interpreting the regulations related to the digital archiving of electronic invoices and the adequacy of the solutions used in practice could be identified as a significant obstacle to the spread of electronic invoicing.

ITM Decree 1/2018. (VI. 29.) introduced the possibility that the archiving of electronic invoices could be done via a hash code built into the data report, thereby protecting the data content of the electronic invoice, in addition to other methods listed in this legislation. Thus, the Online Invoice System provides the following solution:

During the invoicing process, an electronic invoice document (for example, a PDF file without an electronic signature or timestamp) is created on the seller's side. The seller's system generates a hash code for the file, using one of the hashing algorithms indicated in this specification. This hexadecimal number is communicated to NAV as part of the invoice data report (electronicInvoiceHash element in invoiceApi.xsd).

The electronic invoice issued on the seller's side and the electronic invoice received from the seller on the buyer's side must be stored on a suitable storage medium. If it is necessary to prove the authenticity and integrity of the electronic invoice (which is a file, specifically a PDF file in the example given) during a tax audit, the appropriate method would be comparing the hash value generated by the algorithm indicated by the seller in the data report with the hash value generated by the algorithm for the file stored by the seller/buyer. Therefore, if the hash value generated by the algorithm during verification is the same as the one provided by the seller when issuing the invoice, the authenticity of the electronic invoice as required by the VAT Act (actually issued by the seller) and its integrity (the data content not modified since its creation) can be considered verified.

For electronic invoice data report, creating and including the hash value in the data report itself is not mandatory. This is an opportunity to effectively fulfill the digital archiving obligation associated with an electronic invoice.
…"

The InvoiceSender application supports the above method by the following steps:

- If EInvoiceing is switched on either in command line or by properties parameter during sending the application checks if "**ELECTRONIC**" mode provided on **invoiceAppearance**

xml tag. If invoiceApprearance is not electronic than the sending continues on “normal” mode.

- In case of electronic invoiceAppearance the application searches for PDF image files.
  - The search is done on folder “**pdfsfolder**” specified in properties file.
  - The search filename is generated from XML filename by replacing “.xml” extension with “.PDF” and “.pdf”.
  - If the file not found the XML input is not processed. Error log entry created, input xml untouched – according to application parameters next run of the application tries sending again.
- If image file found than application generates SHA3-512 hash code for the PDF file (invoiceHash later).
- The invoiceHash is being send to NAV API during xml transmission.
- In “./processed/” folder “processed_<inputxml>.hash” file is generated containing invoiceHash.
- The application response file is also extended by 2 csv columns:
  - eInvoice: “true” / “false”
  - invocieHash

II. InvoiceSender working modes

The InvoiceSender application is an asynchronous polling-sending-receiving application.

The application has different modes (Basic processing, Token-testing[4], Single file processing) – this document describes the basic working mode.

In basic mode the application scans an input directory for new incoming XML files – containing NAV invoice data.
If an XML file fount it is transmitted to NAV. Upon successful transmission the file is moved to the processed folder. In case of error occurred the file is moved to an error folder. In both case the filename is prefixed with a time stamp of the processing time.
The application records the fact of sending to an internal database. Once the sending phase finished the program queries back the NAV processing state of the pending transactions.
The processing and querying of files are tracked on a CSV file in log folder as well on a response file in output folder.

The program is recommended to invoke after the invoice generation and/or scheduled periodically from the scheduler of hosting OS (Windows scheduler / Unix cron).

4 For Token-testing mode please refer to chapter: *IV.5 Testing of configuration*

III. Installation

III.1 Distribution format

The InvoiceSender application is a Java JAR application packed with supplemental directories and files into a ZIP archive file[5].

III.2 Prerequisites

- Java JRE 1.9 (preferred OpenJDK 13 or later)[6]
- Windows or Linux environment
- Read/Write access to directories (./input, ./output, ./processed, ./errors, ./logs)
- Read access to directories (./resources)
- Internet connectivity / Java SSL configuration
- Optionally: OS Scheduler (Windows scheduler or Cron subsystem.)
- Hungarian tax authority registration / Nav Online szamla technical user credentials

III.2.1 Required Java version since 29th April 2019

Obsolate – please refer to III.2.2 Required Java version since 01th April 2020

NAV forces TLSv1.2 HTTPS communication from 29th April 2019. The following tables shows the required Java versions supporting the encryption:[7]

Version	Release date	End of Free Public Updates	Extended Support Until	Last public Update Version	TLS v1.2 Support
J2SE 1.4	2002 Feb	2008 Oct	2013 Feb	1.4.2 u30	-
J2SE 5.0	2005 Sep	2009 Nov	2015 Apr	1.5.0 u22	-
Java SE 6	2006 Dec	2013 Apr	2018 Dec	6 u45	since u111*
Java SE 7	2011 Jul	2015 Apt	2022 Jul	7 u80	since u95*
Java SE 8 (LTS)	2014 Mar	**2018 Jan for Oracle (commercial)** 2020 Dec **for Oracle (personal use)**	2025 Mar	8 u241	Yes

5 The detailed description of the distribution package is available at: Appendix B.)

6 In case multiple Java versions installed PATH and JAVA_HOME environment variables can be set accordingly in runme.bat or runme.sh.

7 Source: https://en.wikipedia.org/wiki/Java_version_history

Java SE 9	2017 Sep	2018 Mar for OpenJDK	N/A	9.0.4	Yes
Java SE 10	2018 Mar	2018 Sep for OpenJDK	N/A	10.0.2	Yes
Java SE 11 (LTS)	2018 Sep	2022 Sep for AdoptOpenJDK	2026 Sep	11.0.6	Yes
Java SE 12	2019 Mar	2019 Sep for OpenJDK	N/A	12.0.2	Yes
Java SE 13	2019 Sep	2020 Mar for OpenJDK	N/A	13.0.2	Yes

* Commercial updates only

III.2.2 Required Java version since 1th April 2020

NAV data transmission protocol required SHA3-512 hashing on V2 protocol communication.

Java versions prior to Java9 are lack of the required crypto.

As a requirement a supported, up-to-date Java Runtime (JRE) required to be installed for running the application.
Please refer to

III.3 Installation steps

1. Unzip distribution archive to a specific location.
2. Provide RW access to required directories.
3. Provide Execute access to shell scripts/cmd files.
4. Configure mandatory credential parameters (./resources/InvoiceSender.properties)
5. Optional: Configure further parameters (eg: proxy, non-standard directories, logging etc.)
6. Optional: Configure NAV production system serviceurl
7. Test configuration

III.3.1 Accessing NAV production system

The application distribution package is configured for NAV's Test environment. To switch to NAV Production system please apply the following changes:

- ./resources/InvoiceSender.properties – the serviceurl parameter needs to be changed:
serviceurl=https://api.onlineszamla.nav.gov.hu/
- ./resources/InvoiceSender.properties – Make sure that local XML validation is turned off.

...

```
# Validation action(To enable set it to: true)
validatexml=false
```

IV. Application configuration

IV.1 Credential configuration

Before run credential information needs to be supplied into ./resources/InvoiceSender.properties file. The following data is required from NAV authentication:

- Hungarain tax registration number of the company (only first 8 characters)
- NAV side generated technical username
- NAV side provided technical password
- NAV side generated XMLSignKey
- NAV side generated Exchange key

Please refer to *Appendix D.) Sample of InvoiceSender.properties config* for further information.

IV.2 Optional configuration parameters

Besides the mandatory credential configuration the following group of parameters can be tuned in ./resources/InvoiceSender.properties file:

- HTTP REST endpoint (currently only NAV test system REST endpoint available)
- HTTP Proxy configuration
- Internal database configuration
- Sending software configuration
- Default folder access path configuration
- Validate/Transmit mode configuration
- Input filepattern configuration

Please refer to *Appendix D.) Sample of InvoiceSender.properties config* for further information.

IV.3 Java SSL configuration

To protect transmission channel NAV applied SSL Layer on the REST interface. The application can use the OS's default or an application configured Java keystore to accept NAV's certificate.

On ./runme.sh & ./runme.bat files Java parameters are configured to use ./resources/cacerts.jks keystore (currently holding NAV Test System certificate and NAV Production System certificates).

Alternatively NAV SSL certificates can be added to the keystore of the operation system – please refer to OpSys & Java user documentations.

Upon NAV certificates are changing the update of keystore entries might be nescessary. Example scripts provided as ./resources/1_downloadcert_teszt.sh and ./resources/2_downloadcert_prod.sh where the required steps can be found upon NAV certificate changes and update required on keystore. (Certificate is expected to change during NAV software lifecycle.)

IV.4 Command line parameters

Command line parameters of the program can be invoked by issuing:

```
java -jar InvoiceSender.jar -h
```

The available command line parameters are the following:

```
 -c,--config <arg>                Properties file location (Default:
                                  ./resources/InvoiceSender.properties)
 -d,--disablepropertiesfile       Disable load of InvoiceSender.properties
 -db,--database <arg>             Database JDBC location. Default: empty.
                                  Possible 'h2:filename'
 -dbdc,--dbtabledontcreate        DB disbale automatic [REQUESTS] table
                                  creation. True if specified. Defaults:
                                  false
 -dbpw,--dbpwd <arg>              Database password. Default: ''
 -dbq,--dbautoquery <arg>         DB automatic query runs. [true, false]
                                  Requires db. Default: true
 -dbu,--dbuser <arg>              Database user. Default: sa
 -e,--einvoiceing                 Handle electronic invoicing
 -f,--xmlfile <arg>               Input XML file path. Implies
                                  scanfoldermode=false.
 -h,--help                        Command line help
 -k,--lockfile <arg>              Lockfile location (Default
                                  ./resources/InvoiceSender.lock)
 -kt,--locktimeout <arg>          Lockfile timeout (Default: none)
 -l,--dontvalidate                Disable XML Validation (overwrites
                                  properties settings)
 -lqi,--logqueryinfo              Logs AutoQuery info (Number of pending,
                                  max attempt, oldest entry) to application
                                  log at INFO level. Default: false
 -p,--httpuseproxy                Use http proxy
```

```
 -q,--maxqueryattempts <arg>     Maximum number of query attempts.
                                 Default: 300 (eg: 25 hours * 12 queries/hour)
 -ph,--httpproxyhost <arg>       Proxy host. Default: localhost
 -pp,--httpproxyport <arg>       Proxy port. Default: 3128
 -pt,--httpproxyprotocol <arg>   Proxy protocol. Default: http Possible:
                                 http, https
 -r,--filesperrun <arg>          Number of files to process per run.
                                 Default: 100
 -rfo,--resultfinalstateonly     Query answers provided only for ABORT or
                                 DONE state.(Default: false, eg: PROCESSED
                                 included)
 -s,--dontsend                   Disable XML NAV Transfer (overwrites
                                 properties settings)
 -t,--tokentest                  Test mode. Get NAV Exchange token.
 -tq,--transactionsquery <arg>   If set than 'Transaction Query mode'
                                 active. Parameter for NAV transaction ID.
 -trno,--transactionrespno <arg> Parameter for response file in
                                 'Transaction Query mode'. Default: empty – don
                                 not write response file.
 -u,--serviceurl <arg>           Service URL (Default:
                                 https://api-test.onlineszamla.nav.gov.hu/
 -v,--version                    Version info
 -x,--httpuseproxyuser           Use http with proxy user (requires -p
                                 also)
 -xp,--httpproxypassword <arg>   Proxy user password
 -xu,--httpproxyuser <arg>       Proxy user
```

IV.5 Testing of configuration

Testing of setup and credential configuration is possible via the -t command line parameter.
A successful check should give similar output:

```
[2018-06-26 12:58:24:657] [INFO] [InvoiceSender:744] - InvoiceSender 2018. V1.1.5
[2018-06-26 12:58:24:665] [INFO] [InvoiceSender:384] - use -h or --help for usage
[2018-06-26 12:58:24:665] [INFO] [InvoiceSender:445] - Token test mode selected.
[2018-06-26 12:58:24:668] [INFO] [InvoiceSender:269] - Don't move on technical exception:
true
[2018-06-26 12:58:24:875] [INFO] [DatabaseHandler:145] - DB connected: h2:invoices_2018
[2018-06-26 12:58:24:877] [INFO] [DatabaseHandler:176] - Database structure initialized
[2018-06-26 12:58:24:878] [DEBUG] [InvoiceSender:351] - Application run is locked with
filelock: ./resources/InvoiceSender.lock
[2018-06-26 12:58:24:878] [INFO] [InvoiceSender:754] - Acquire NAV exchange token…
[2018-06-26 12:58:43:724] [INFO] [InvoiceSender:758] - NAV Exchangetoken succesfully
acquired: 40fd854c-5daa-4310-9864-60833892373a27HD7NP1OK5O
[2018-06-26 12:58:43:735] [INFO] [InvoiceSender:696] - Checking pending queries..
```

V. Running of application

V.1 Starting the application

Running of application is possible as standard runnable jar files. The ./runme.bat and ./runme.sh files are provided to help issuing the application.

Example distributed runme.sh:

```
#!/bin/bash

java -Djavax.net.ssl.trustStore=./resources/cacerts.jks \
  -Dlog4j.configurationFile=./resources/log4j2.xml -jar InvoiceSender.jar $*
```

The following Java configuration defines recommended:

```
- javax.net.ssl.trustStore – Define default SSL keystore for the application
- log4j.configurationFile  - Define default Log4j2 configuration file
```

V.2 Application locking

During the program run the application writes ./resources/InvoiceSender.lock file (path can be overwritten from command-line parameter -k) to prevent concurrent execution.

In case of abnormal program termination lock file needs to be removed manually before next program invocation!

V.2.1 Lock timeout feature

As from version V1.7 lock timeout feature can be switched on to prevent long-time stale locks hanging out operation. With switch the feature on “–locktimeout <integer_secs>” (or “-kt <integer_secs>”) lock timeout value can be specified. If the feature is switched on than if lock file exists the program run examines the last modification time of the lockfile – in case if it’s older than the specified time the lockfile is ignored and “taken over”.

The execution flow is shown in the below log samples whereon 2nd run 1 sec timeout sepcified:

```
> touch resources/InvoiceSender.lock
> date
2018. okt. 18., csütörtök, 09:44:55 CEST
> ./runme.sh
[2018-10-18 09:44:59:383] [INFO] [InvoiceSender:801] - InvoiceSender 2018. V1.1.7
```

```
[2018-10-18 09:44:59:390] [INFO] [InvoiceSender:415] - use -h or --help for usage
[2018-10-18 09:44:59:394] [INFO] [InvoiceSender:283] - Don't move on technical exception: true
[2018-10-18 09:44:59:626] [INFO] [DatabaseHandler:148] - DB connected: h2:invoices_2018
[2018-10-18 09:44:59:627] [INFO] [DatabaseHandler:179] - Database structure initialized
[2018-10-18 09:44:59:628] [ERROR] [InvoiceSender:370] - ERROR: LockFile exisits - cannot run: ./resources/InvoiceSender.lock
Exception in thread "main" java.lang.IllegalStateException: ERROR: LockFile exisits - cannot run: ./resources/InvoiceSender.lock
        at com.navonline.InvoiceSender.applock(InvoiceSender.java:371)
        at com.navonline.InvoiceSender.main(InvoiceSender.java:807)


> ./runme.sh  -kt 1
[2018-10-18 09:45:03:026] [INFO] [InvoiceSender:801] - InvoiceSender 2018. V1.1.7
[2018-10-18 09:45:03:033] [INFO] [InvoiceSender:415] - use -h or --help for usage
[2018-10-18 09:45:03:034] [INFO] [InvoiceSender:441] - LockFile Timeout (secs): 1
[2018-10-18 09:45:03:037] [INFO] [InvoiceSender:283] - Don't move on technical exception: true
[2018-10-18 09:45:03:278] [INFO] [DatabaseHandler:148] - DB connected: h2:invoices_2018
[2018-10-18 09:45:03:280] [INFO] [DatabaseHandler:179] - Database structure initialized
[2018-10-18 09:45:03:280] [INFO] [InvoiceSender:362] - Lock timeout is greater than expected - IGNORING OLD LOCK, touching lockfile!
[2018-10-18 09:45:03:281] [INFO] [InvoiceSender:827] - Scanning for .xml and .XML files in /home/jtoberling/workspace_java/InvoiceSender/input including those in subdirectories
[2018-10-18 09:45:03:282] [INFO] [InvoiceSender:838] - Number of files found: 0
[2018-10-18 09:45:03:282] [INFO] [InvoiceSender:743] - Checking pending queries...
[2018-10-18 09:45:03:285] [INFO] [InvoiceSender:752] - Number of queries to run in this loop: 0 (max:  100), MaxQueryAttempts: 300
[2018-10-18 09:45:03:316] [INFO] [InvoiceSender:884] - Application terminates.
>
```

V.3 Transaction Query mode

Under special circumstances (eg: preceding communication problems on NAV's side) there is possibility in the application to query specific transaction that was previously not recorded in sender side internal H2 database.

This mode is called "Transaction Query Mode" and can be controlled by two command-line parameters:

-tq,--transactionsquery <arg> If set than 'Transaction Query mode' active. Parameter for NAV transaction ID.

-trno,--transactionrespno <arg> Optional parameter for response file in 'Transaction Query mode'. Default: empty - don not write response file. (Requires -tq parameter to be present!)

If Transaction Query Mode is activated by -tq parameter no other modes like Token Test, Sending / Autoquery will be running.

During program execution the application issues status query to NAV's API. The queried transaction id does not need to be previously recorded in the internal DB – any NAV recorded transactions can be queried.

The results of the query will be available in the application log at INFO level.

If -trno parameter provided than the results will be also written in ResultLog format to the configured output directory. (By default: output/<trno>.resp)

Example of run:

```
java \
              -Djavax.net.ssl.trustStore=./resources/cacerts.jks \
       -Dlog4j.configurationFile=./resources/log4j2.xml \
       -jar target/InvoiceSenderV2.jar \
       -tq 3107D1AC1J9C3X25 -trno abc123

[2020-08-31 20:16:24:642] [INFO] [InvoiceSender:856] - InvoiceSender 2018-2019. V2.0.2
[2020-08-31 20:16:24:643] [INFO] [InvoiceSender:197] - Forcing https.protocols=TLSv1.2
[2020-08-31 20:16:24:849] [INFO] [InvoiceSender:436] - use -h or --help for usage
[2020-08-31 20:16:24:850] [INFO] [InvoiceSender:462] - LockFile Timeout (secs): 7200
[2020-08-31 20:16:24:851] [INFO] [InvoiceSender:520] - Transaction query mode active:
3107D1AC1J9C3X25
[2020-08-31 20:16:24:851] [INFO] [InvoiceSender:524] - Transaction response invocie Number: abc123
[2020-08-31 20:16:24:854] [INFO] [InvoiceSender:302] - Don't move on technical exception: true
[2020-08-31 20:16:25:065] [INFO] [DatabaseHandler:149] - DB connected: h2:invoicesV2_2020
[2020-08-31 20:16:25:067] [INFO] [DatabaseHandler:180] - Database structure initialized
[2020-08-31 20:16:25:069] [INFO] [InvoiceSender:866] - Transaction query mode...
[2020-08-31 20:16:25:069] [INFO] [InvoiceSender:638] - Query transactionId: 3107D1AC1J9C3X25
[2020-08-31 20:16:25:950] [INFO] [InvoiceSender:870] - QueryResult:
{validationErrorCode=SUPPLIER_CUSTOMER_MATCH_NAME, method=QUERY,
requestId=XX20200831201625072H74pCMM1eWf, funcCode=DONE, errorCode=, validationResultCode=WARN,
message=Eladó és vevő neve nem lehet azonos., invoiceStatus=DONE, transactionId=, schemaMessage=: }
[2020-08-31 20:16:25:956] [INFO] [InvoiceSender:894] - Result put to file: abc123
[2020-08-31 20:16:25:970] [INFO] [InvoiceSender:973] - Application terminates.
```

VI. Housekeeping

VI.1 Backup and restore

Application binary and all related application data can be backed up and restored by archiving the root directory of deploy.

Please make sure that during backup and restore operations the application is not running and scheduler is switched off.

VI.2 Internal database maintenance

By default the application is using an embedded H2 Database engine to track and query NAV transmissions. By default the database resides in a single file – if not exists it is created automatically. The database file location can be specified in ./resources/InvoiceSender.properties file – it is recommended to change the filename annually.

The application is using standard JDBC connector and syntax – connecting to different databases might be possible but out of scope of this document.

VI.3 Reprocessing of failed files

In case of technical failures (eg: no internet connectivity) syntactically good XML files in ./error directory can be issued again for transmission by moving over to ./input directory. Processed filenames will be prefixed again with time stamp. If ./output files required to be in specific name to further consumed by eg: ERP system please rename XML files before retransmit.

VI.4 Purging of obsolete data

To purge obsolete data client purging/archiving rules should apply on following points:

./processed folder – Contains NAV sent XML files.
./errors folder – Contains XML files failed to send to NAV.
./logs folder – Contains application and result logs
./output folder – Contains NAV status responses.

Purging old data is out of scope of application run.

VI.5 Maintenance of Java SSL certificates

NAV HTTPS certificates can be store on operating system level Java keystores or program level keystore (./resources/cacerts.jks). If NAV changes certificates and program terminates due to SSL

connection issues update of keys might be required. Local certificate maintenance is out of scope of program tasks and belongs locale IT operations.

VII. Log files

The application has two type of log files:

- Results of transmissions and status querying is put to logs/ResultLog.csv file.
- Generic application log file

The log files are created by applying the standard Log4j2 subsystem – fine tuning is possible as Log4j2 allows it. Please refer to Log4j documentation[8].

Both ResultLog and ApplicationLog files are configured to be daily-rolling, compressed archive logs.

VII.1 Log of results

The application logs the results of transmits to ./logs/ResultLog.csv file.

An example of output can be seen below:

```
2018-05-16,12:03:20.178,T2019042603,CREATE,SUCCESS,OK,XX20180516120319831XXV2GXx19AU,25UQ6J8GLXGA,,./input/example_1.xml,
2018-05-16,12:03:21.831,,QUERY,FAILURE,ABORTED,XX20180516120320197tjk2aQXUfLk,25UQ6J8GLXGA,AUTOQUERY,INVOICE_NUMBER_NOT_UNIQUE
2018-05-16,12:03:55.320,T2019051601,CREATE,SUCCESS,OK,XX20180516120354967ffCtsFfrNet,25UQ7ACSHM51,
./input/example_1.xml,
2018-05-16,12:03:55.671,,QUERY,FAILURE,PROCESSING,XX20180516120355349RfwjXc78N4,25UQ7ACSHM51,AUTOQUERY,
2018-05-16,12:04:32.002,,QUERY,SUCCESS,DONE,XX20180516120430558JAWUA1jFbpk,25UQ7ACSHM51,AUTOQUERY,
```

The structure of the file is the following in columns:

#	Example	Info
1	2018-05-16	Date of event YYYY-MM-DD
2	12:03:20.178	Time of event HH:MM:SS.mmm
3	T2019042603	Invoice Nr.
4	CREATE	NAV Api operation mode
5	SUCCESS	Transmission result
6	OK	NAV state
7	XX20180516120319831XXV2GXx19AU	Generated request ID
8	25UQ6J8GLXGA	NAV Side generated transaction id

8 https://logging.apache.org/log4j/2.x/manual/index.html

#	Example	Info
9	./input/ example_1.xml	Path of processed input file, or AUTOQUERY in case of automatic status query issued
10	INVOICE_NUMBER_NOT_UNIQUE	Error code – only if applicable
11	true	Invoice is electronic. true/false according to XML's invoiceAppearance==’ELECTRONIC’
12	Base64 string	On electronic invoices SHA3-512 hash of PDF file

VII.2 Application logs

Application logs can be found in ./logs/InvoiceSender.log files.
Example of application log:

```
[2018-06-26 13:00:04:641] [INFO] [InvoiceSender:744] - InvoiceSender 2018. V1.1.4
[2018-06-26 13:00:04:649] [INFO] [InvoiceSender:384] - use -h or --help for usage
[2018-06-26 13:00:04:652] [INFO] [InvoiceSender:269] - Don't move on technical exception: true
[2018-06-26 13:00:04:863] [INFO] [DatabaseHandler:145] - DB connected: h2:invoices_2018
[2018-06-26 13:00:04:865] [INFO] [DatabaseHandler:176] - Database structure initialized
[2018-06-26 13:00:04:866] [DEBUG] [InvoiceSender:351] - Application run is locked with filelock:
./resources/InvoiceSender.lock
[2018-06-26 13:00:04:867] [INFO] [InvoiceSender:771] - Scanning for .xml and .XML files in
/home/jtoberling/workspace_java/InvoiceSender/input including those in subdirectories
[2018-06-26 13:00:04:879] [INFO] [InvoiceSender:696] - Checking pending queries...
```

VII.3 Output response structure

If configured appropriately upon NAV asynchronous processing ended the application provided response file in ./output directory.

The filename is the base filename provided in ./input directory plus ‘.resp.txt’ ending.
(Eg: input.xml → input.xml.resp.txt)

An example response structure is the following:
25UQ7ACSHM51,DONE,

Whereon the comma separated columns stands for:
- 1. Original NAV assigned transaction ID,
- 2. NAV processing status code. DONE or ABORTED typically
- 3. Error code – only if applicable

VIII. Known issues

VIII.1 Queries remain in ‘PROCESSING’ state

The program maintains an internal database to track pending queries automatically. The automatic query is issued on every program run on pending requests. The number of entries to query is determined by "filesperrun” property parameter.

According to NAV the processing of transaction on NAV’s side should happen in 5 minutes – however upon technical problems on NAV’s system several hour processing time was experienced already.

The default number of autoqueries determines that how many time a pending transaction is queried. From program version V1.1.5 the automatic queries are happening in 300 times (eg: every 5 mins for 25 hours in case of automatic 5 minute schedule).

The number currently pending transactions can be viewed by specifying the -lqi command line parameter[9]. Example of application log should contain:

```
[2018-06-27 22:22:20:339] [INFO] [DatabaseHandler:356] - Count of pending queries: 1
[2018-06-27 22:22:20:340] [INFO] [DatabaseHandler:357] - Max of query attempts: 17
[2018-06-27 22:22:20:340] [INFO] [DatabaseHandler:358] - Oldest entry to query: 2018-06-27 20:39:34.951
```

The maximum query attempt number can be overwritten by -q command line or “maxqueriattempts” properties parameter.

VIII.2 Transmission: “SSLHandshakeException: Received fatal alert: handshake_failure”

WARNING: OBSOLATE – PLEASE REFER TO VIII.3!

From 16th April 2019 in NAV’s test system and from 29th April 2019 in NAV’s production system NAV is forcing TLSv1.2 protocol.

If “SSLHandshakeException: Received fatal alert: handshake_failure” is raising in log files Java update is required (please refer to chapter “*III.2.1 Required Java version since 29th April 2019*” for more info).

9 In case of several DB entries collecting pending query information can be time consuming – consider not adding this command line parameter to to regular program invocation.

Please upgrade running Java runtime to a supported version.
If upgrade is not possible (eg: JD Application server is using system default Java) alternative Java runtime needs to be deployed and set up.

Sample steps for deploying alternative JRE:

1. Download supported JRE from Oracle: https://www.java.com/en/download
   (eg: jre-8u211-linux-i586.tar.gz)
2. Extract downloaded JRE to a system directory. Eg: *//usr/lib/jvm/<your_java_version>/ (eg: /usr/lib/jvm/jre1.8.0_211)*
3. Setup runme.bat or runme.sh to run the appropriate java from the new path.

```
#!/bin/bash
export JAVA_HOME=/usr/lib/jvm/jre1.8.0_211/
/usr/lib/jvm/jre1.8.0_211/bin/java -Djavax.net.ssl.trustStore=./resources/cacerts.jks -
Dlog4j.configurationFile=./resources/log4j2.xml -jar InvoiceSender.jar $*
```

VIII.3 Transmission: “NoSuchAlgorithmException: SHA3-512 MessageDigest not available”

As from NAV protocol version V2 (mandatory in production system since 1st April 2020) SHA3-512 hashing algorithms is required on NAV API protocol during transmissions.

Java version since Java 9 are supporting SHA3-512

Please upgrade running Java runtime to a supported version.
(Minimum version: Java9, recommended version: OpenJDK 13.)
If upgrade is not possible (eg: JD Application server is using system default Java) alternative Java runtime needs to be deployed and set up.

Sample steps for deploying alternative JDK/JRE:

1. Download supported JDK/JRE from OracleJDK: https://jdk.java.net/13/
   (eg: openjdk-13.0.2_linux-x64_bin.tar.gz)
2. Extract downloaded JDK/JRE to a system directory. Eg: *//usr/lib/jvm/<your_java_version>/ (eg: /usr/lib/jvm/jdk-13.0.2/)*
3. Setup runme.bat or runme.sh to run the appropriate java from the new path.

```
#!/bin/bash
export JAVA_HOME=/usr/lib/jvm/jdk-13.0.2/
/usr/lib/jvm/jdk-13.0.2/bin/java -Djavax.net.ssl.trustStore=./resources/cacerts.jks -
Dlog4j.configurationFile=./resources/log4j2.xml -jar InvoiceSender.jar $*
```

Appendix A.) ChangeLog of current version

```
# Changelog
All notable changes to this project will be documented in this file.

The format is based on [Keep a Changelog](http://keepachangelog.com/en/1.0.0/)
and this project adheres to [Semantic Versioning](http://semver.org/spec/v2.0.0.html).

## [1.0.0] - 2018-02-24
- Initial version

## [1.0.1] - 2018-03-10
- Fix NAV HTTP1.0 header bug
- Adding resources/InvoiceSender.properties singletokenmode
- Fix software header SoftwareDevTaxNumber typo

## [1.0.2] - 2018-04-11
- Requestheader timestamp with milliseconds

## [1.0.3] - 2018-04-20
- Downgrading to support Java 1.6.
- Adding proxy support
- Adding runme.sh / runme.bat
- Removing singletokenmode parameter from properties file
- Adding support for Query transactions


## [1.1.0] - 2018-04-21
- Adding database functionality over JDBC
- Adding H2 db
- Adding response functionality - results to ./output folder
- Adding token test mode

## [1.1.1] - 2018-04-26
- Adding output\<origfilename>.resp CSV file

## [1.1.2] - 2018-04-27
-  Adding resultfinalstateonly parameter. Only writes DONE or ABORTED state query
responses.

## [1.1.3] - 2018-05-27
- Adding Api v0.13 invoiceOperation/compressedContent=false tag

## [1.1.4] – 2018-06-26
- Adding property dontmovetechnicalexception=true to InvoiceSender.property file.
- Forcing software tag elements
- Configuration for GoLive production parameters

## [1.1.5] - 2018-06-27
- Increase maxqueryattempts default to 300
- Adding -q, -lqi command line and maxqueryattempts property parameter

## [1.1.6] - 2018-07-19
- Adding annulment function
- Fixing ./logs/ResultLog.csv logging

## [1.1.7] - 2018-10-18
- Read input XML files in alphabetical order
- Adding locktimeout feature
```

```
## [1.1.8] - 2019-04-18
- NAV protocol V1.1
- Forcing TLSv1.2

## [2.0.0] – 2019-12-22
- Fork from V1.1 for NAV 2.0 API support

## [2.0.2] – 2020-08-31
- TransactionQueryMode feature

## [3.0.0] - 2020-11-29
- NAV V3.0 API


## [3.0.1] - 2020-12-14
- NAV V3.0 API SHA-512 hash CrytoType change

## [3.0.2] - 2021-01-03
- Sha3-512 algorithm fallback fix on old Java versions

## [3.5.0] - 2021-03-09
- eInvoicing: SHA3-512 hash generation and sending on PDF files 'ELECTRONIC'
```

Appendix A.1) Update steps V1.1.3 → V1.1.4, V1.1.5

The following steps required to upgrade from V1.1.3 to V1.1.4:

1. Recommended: Backup current directory
2. Replace ./InvoiceSender.jar file from distribution zip file.
3. Replace ./resources/cacerts.jks from distribution zip file.
4. Extract ./resources/2_downloadcert_prod.sh from distribution zip file.
5. Extract ./resources/invoiceData_20180601.xsd from distribution zip file.
6. Modify ./resources/InvoiceSender.properties:
    1. Add “dontmovetechnicalexception=false” to a new line.
    2. Replace line “invoiceDataValidator=./resources/invoiceData_20180601.xsd”
    3. Remove Software identification parameters. (Obsolete entries – softwareid, softwarename, softwareoperation, softwareversion is hardcoded, softwaredevname, softwaredevcontact, softwarecountrycode, softwaredevtaxnumber is removed).
    4. Optional: For production access modify line to production api url: “serviceurl=https://api.onlineszamla.nav.gov.hu/”
7. Test configuration (Hint: Please use -t command line parameter to check tokenGet function from NAV.)

Appendix A.2) Update steps V1.1.5 → V1.1.6

The following steps required to upgrade from V1.1.5 to V1.1.6:

1. Recommended: Backup current directory
2. Replace ./InvoiceSender.jar file from distribution zip file.
3. Replace ./resources/log4j2.xml from distribution zip file.

Appendix A.2) Update steps V1.1.6 → V1.1.7

No mandatory upgrade steps required.

Optionally “locktimeout” feature can be activated – please refer documentation above.

To activate the feature runme.sh or runme.bat needs to be modified:

Example rumme.sh:

java -Djavax.net.ssl.trustStore=./resources/cacerts.jks -Dlog4j.configurationFile=./resources/log4j2.xml -jar InvoiceSender.jar **-kt 7200** $*

Appendix A.3) Update steps V1.1.6, V1.1.7 → V1.1.8

The following steps required to upgrade from V1.1.6 or V1.1.7 to V1.1.8:

1. Recommended: Backup current directory
2. Replace ./InvoiceSender.jar file from distribution zip file.

Appendix A.4) Update steps V1.1.8 → V2.0.0

The following steps required to upgrade from V1.1.8 to V2.0.0:

1. Recommended: Backup current directory
2. Change “database” entry in ./resources/InvoiceSender.properties to a new file.
   Example:

   database=h2:invoicesV2_2019
3. Extract ./InvoiceSenderV2.jar file from distribution zip file.
4. Replace ./runme.sh and/or ./runme.bat from distribution zip file.

Appendix A.5) Update steps V1.1.[0,1] → V2.0.2

1. Recommended: Backup current directory
2. Extract ./InvoiceSenderV2.jar file from distribution zip file.

Appendix A.6) Update steps V2.0.[0,1] → V3.0.0

1. Recommended: Backup current directory
2. Extract ./InvoiceSenderV3.jar file from distribution zip file.
3. Amend runme.sh or runme.bat to point on InvoiceSenderV3.jar file.

Appendix A.7) Update steps V3.0.[*] → V3.5.0

1. Recommended: Backup current directory
2. Extract ./InvoiceSenderV3.jar file from distribution zip file.
3. Move and backup *.h2 database file (***The new version has changed internal db structure***!)
4. Change InvoiceSender.properties with the following parameters:
   - add: “einvoiceing=true”
   - add: “pdfsfolder=/u01/app/jde/IF/szamla/”
     (Amend path as necessary!)
   - change: “database=h2:invoicesV35_2021”
5. Amend runme.sh or runme.bat to point on InvoiceSenderV3.jar file.

Appendix B.) Distributed files and their descriptions

File	Description
./resources	Directory containing configuration files
./input	Directory containing input XML files to process
./processed	Directory containing successfully processed/transmitted files
./error	Directory containing unsuccessfully processed/transmitted files
./output	Directory containing NAV response files
./examples	Directory containing example data XMLs (can be removed)
./logs	Directory containing logs
./runme.bat	Example batch file for running under Windows environment
./runme.sh	Example batch file for running under Unix environment
./CHANGELOG.md	InvoiceSender application ChangeLog
./README.md	InvoiceSender application basic Readme file
./examples/example_bad.xml	NAV Data XML (bad)
./examples/example_QUERY.xml	NAV Query file example
./examples/example.xml	NAV Data XML (good)
./invoicesV2_2019.h2.db	Example H2 database (To be removed before first run!)
./InvoiceSenderV3.jar	InvoiceSender JAR application binary
./logs/InvoiceSender-2018-04-20.log.gz	Example archived application log
./logs/InvoiceSender.log	Example application log
./logs/ResultLog.csv	Example result log
./processed/processed_20180321_132315_example.xml	Example processed Data XML
./resources/1_downloadcert_teszt.sh	Unix shell script example to update keystore – NAV Test system
./resources/2_downloadcert_prod.sh	Unix shell script example to update keystore – NAV Production system
./resources/api-test.pem	PEM certificate of NAV test system
./resources/cacerts.jks	Java keystore holding NAV test system certificate
./resources/invoiceApi_20180111.xsd	NAV issued API XSD
./resources/invoiceData_20180601.xsd	NAV issued Data XSD
./resources/InvoiceSender.properties	InvoiceSender main configuration file

File	Description
`./resources/log4j2.xml`	InvoiceSender log4j configuration file

Appendix C.) Used OpenSource components and licensing

Component name	License	Further info
Log4j2	Apache License, Version 2.0	https://logging.apache.org/log4j/2.x/license.html
Apache commons	Apache License, Version 2.0	http://www.apache.org/licenses/
H2 Database engine	MPL2.0 / EPL 1.0	http://www.h2database.com/html/license.html
Junit	EPL 1.0	https://junit.org/junit4/license.html
Apache Directory Studio	Apache License, Version 2.0	http://www.apache.org/licenses/
Apache HTTP Components	Apache License, Version 2.0	http://www.apache.org/licenses/

Appendix D.) Sample of InvoiceSender.properties config

```
# ------------------------------------------------------------ #
#
#  Change this section as of NAV registration data
#
servicelogin=<NAV_technical_user_goes_here>
servicepassword=<NAV_technical_userpassword_goes_here>
taxnumber=12345678
xmlsignkey=<NAV_xmlsignkey_goes_here>
exchangekey=<NAV_exchangekey_goes_here>
#

# eInvoiceing parameters
einvoiceing=true
pdfsfolder=/u01/app/jde/IF/szamla/


# ------------------------------------------------------------ #
#
#  Change this section to modify DB properties

# JDBC database access specification.
# H2 DB JDBC driver included. Pls. refer: http://www.h2database.com
# Missing entry means disabled DB functionality! (Also AutoQuery!)
database=h2:invoicesV35_2021
dbuser=sa
dbpwd=
dbtabledontcreate=false
dbautoquery=true
#maxqueryattempts=300


# ------------------------------------------------------------ #
#
#  Change this section in case HTTP Proxy required
#

httpuseproxy=false
httpproxyprotocol=http
httpproxyhost=localhost
httpproxyport=3128
# Basic Authentication is supported
httpuseproxyuser=true
httpproxyuser=user
httpproxypassword=Password123


# ------------------------------------------------------------ #
#
# Custom Interface identification - on the begin of RequestID
#
# Interface ID - 2 chars string
interfaceid=XX
# ------------------------------------------------------------ #

# Nav Teszt URL
serviceurl=https://api-test.onlineszamla.nav.gov.hu/

# Nav Prod URL
#serviceurl=https://api.onlineszamla.nav.gov.hu/
```

```
# ------------------------------------------------------------ #
#
# Function control

# Scan folder mode (To enable set it to: true)
scanfolder=true

# Scan folder mode=true, search *.xml files in this directory
inputfolder=./input/


# Write queriy results to response folder
queryresponse=true

# Write only DONE or ABORTED state response files
resultfinalstateonly=true

# After end of transaction should we move the file?
movetofolders=true
dontmovetechnicalexception=true
processedfolder=./processed/
errorfolder=./error/
responsefolder=./output/

# Validation action(To enable set it to: true)
validatexml=false
invoiceDataValidator=./resources/invoiceData_20180601.xsd

# Transmit action(To enable set it to: true)
transmitxml=true

# File patterns for operations
modifyfilepattern=MODIFY_.*\.xml
stornofilepattern=STORNO_.*\.xml
queryfilepattern=QUERY_.*\.xml
annulfilepattern=ANNUL_.*\.xml
```

Appendix E.) In case of program errors

If program misbehavior occurs please do the following steps:

- Check is application lock exists. Remove it if necessary. (Please refer to: Application locking)
- Check network connectivity to NAV system.
- Perform if application can get token (Please refer to: Testing of configuration)
- Check application logs. Increase log level to DEBUG in ./resources/log4j2.xml if necessary. (Please refer to: Application logs)
- In case internal H2 database is used H2 file can be monitored with h2 jar.
  (Hint: http://h2database.com/html/quickstart.html)
- Collect last referring application logs / submit to support.