CiSender V2.2 manual

Program Version: V2.2

Modified: 2021-12-09

Table of contents

Document changes....3
I. Preface....4
II. CiSender working modes....4
III. Installation....5
III.1 Distribution format....5
III.2 Prerequisites....5
III.3 Installation steps....5
III.3.1 Accessing Charlie-India System production system....6
IV. Application configuration....6
IV.1 Credential configuration....6
IV.2 Optional configuration parameters....6
IV.2.1 ImageSending mode....7
IV.3 Java SSL configuration....7
IV.3.1 Run time Java SSL configuration variables....8
IV.3.2 External tools to handling of SSL certificates....8
IV.4 Command line parameters....9
IV.5 Testing of configuration....9
V. Running of application....10
V.1 Starting the application....10
V.2 Application locking....10
V.2.1 Lock timeout feature....10
V.3 Generating XmlParameters structure....12
V.4 Generating DeliveryList structure....13
V.5 DocProcessor failure notification....14
V.5.1 CISender properties....14
V.5.2 EventLogger mail sending....14
VI. Housekeeping....16
VI.1 Backup and restore....16
VI.2 Reprocessing of failed files....16
VI.3 Purging of obsolete data....16
VII. Log files....17
VII.1 Log of results....17
VII.2 Application logs....18
VII.3 Logging SMTP Notifications....18
Appendix A.) ChangeLog of current version....20
Appendix B.) Distributed files and their descriptions....21
Appendix C.) Used OpenSource components and licensing....22
Appendix D.) Sample of CiSender.properties config....23
Appendix E.) In case of program errors....24

Document changes

Document version	Date	Author	Summary of change
V2.0	2021-11-24	János TŐBERLING	Initial version of manual
V2.1	2021-11-28	János TŐBERLING	Capability catching and sending microflow errors Adding filename parameter to XML trafos Extend certificate configuration
V2.2	2021-12-09	János TŐBERLING	Adding feature pdf FilenName replace pattern.

I. Preface

CiSender is a standalone application that is able to connect to Charlia-India standard SOAP interface (MotorSoapV2) – and issue XML files from a directory.

II. CiSender working modes

The CiSender application is an asynchronous polling-sending application.

The application has different modes (Basic processing, Token-testing[1],) – this document describes the basic working mode.

In basic mode the application scans an input directory for new incoming XML files – containing invoice data.
If an XML file fount it is transmitted to Charlie-India System. 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 program is recommended to invoke after the invoice generation and/or scheduled periodically from the scheduler of hosting OS (Windows scheduler / Unix cron).

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

III. Installation

III.1 Distribution format

The CiSender application is a Java JAR application packed with supplemental directories and files into a ZIP archive file[2].

III.2 Prerequisites

- Java JRE 1.11 (preferred OpenJDK 13 or later)[3]
- 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.)
- Charlie-India System Technical user, Relationship and Api key credentials

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/CiSender.properties)
5. Optional: Configure further parameters (eg: proxy, non-standard directories, logging etc.)
6. Optional: Configure Charlie-India System production system url
7. Test configuration

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

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

III.3.1 Accessing Charlie-India System production system

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

- ./resources/CiSender.properties – the serviceurl parameter needs to be changed:
serviceurl=https://ci2.partnerhub.hu/motor_soapv2

IV. Application configuration

IV.1 Credential configuration

Before run credential information needs to be supplied into ./resources/CiSender.properties file. The following data is required from Charlie-India System authentication:

- Technical username
- Technical user password
- ApiKey
- Relationship ID

Please refer to *Appendix D.) Sample of CISender.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/CISender.properties file:

- HTTP REST endpoint (currently only Charlie-India System test system REST endpoint available)
- HTTP Proxy configuration
- Referred document type
- Called microflow
- Default folder access path configuration
- Image sending mode
- Calling Parameters specification

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

IV.2.1 ImageSending mode

The application can send images during the request.

The image sending is controlled in ./resources/CISender.properties file by imagemode parameter. If the “imagemode=true” the application is searching a PDF file in the configured input directory according to the following logic:

- By default the image filename is derivated from the xml filename by adding “.pdf” ending to the name of the XML. (Eg: ‘input.xml’ → ‘input.xml.pdf’)

- If ‘imgfilenamereplacepattern’ and ‘imgfilenamereplacement’ properties parameters are defined the filename is processed through a regexp filter and matches replaced. Eg:

imgfilenamereplacepattern=\.xml
imgfilenamereplacement=\.pdf
‘input.xml’ → ‘input.pdf’

If image specified file not found the XML is moved to the error folder and application run terminates. **As an existing XML file starts the transmission the image file needs to be put first into input directory followed by the belonging XML file.**

Image data is handed over in transmission by the xmlparameters structure. For details please refer to: *V.3 Generating XmlParameters structure.*

IV.3 Java SSL configuration

To protect transmission channels Charlie-India System can apply SSL Layer on the SOAP and/or SMTPs interface.

Charlie-India System certificates can be store on operating system level Java keystores or program level keystore (./resources/cacerts.jks). If Charlie-India System 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**.

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

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

Upon Charlie-India System certificates are changing the update of keystore entries might be nescessary.

Example scripts provided as ./resources/1_downloadcert_civ4.sh and ./resources/2_downloadcert_cmdlineparamserver.sh where the required steps can be found upon Charlie-India System certificate changes and update required on keystore. (Certificate is expected to change during Charlie-India System software lifecycle.)

IV.3.1 Run time Java SSL configuration variables

To succesfully connect an SSL endpoints (SOAP, SMTPS) the following Java runtime parameters might be configured:

Parameter	**Description**
javax.net.ssl.trustStore	*Default*: ./resources/cacerts.jks Points to Java trustSrore file holding all remote server certificates.
javax.net.ssl.trustStorePassword	*Default*: changeit Cleartext password protecting keystore file. Provided at keystore generatrion.
javax.net.ssl.trustStoreType	*Default*: PKCS12 [4] Type of trustStrore. Possible values: JKS, PKCS12

Runtime parameters can be setup in execution command line in like the following example:

```
#!/bin/bash
java -jar \
      -Djavax.net.ssl.trustStore=./resources/cacerts.jks \
      -Djavax.net.ssl.trustStorePassword=changeit \
      -Djavax.net.ssl.trustStoreType=PKCS12 \
      -Dlog4j.configurationFile=./resources/log4j2.xml \
      -jar cisender.jar -kt 2 $*
```

IV.3.2 External tools to handling of SSL certificates

OpenSource *openssl* and Java *keytool* programs are usefull to generate and manage trustStore certificates. The file resources/1_downloadcert-cmdlineparamserver.sh provides the following steps to regenrate trusStore

```
#!/bin/bash

echo -n | openssl s_client -connect $1:443 | sed -ne '/-BEGIN CERTIFICATE-/,/-
END CERTIFICATE-/p' > ./$1.pem
```

4 Default trusStoreType was changed by Oracle from JKS → PKCS12 by Java version 9.

```
openssl x509 -in ./$1.pem -text -noout | grep "Subject:"


rm ./cacerts.jks


keytool -import -trustcacerts -keystore ./cacerts.jks -storepass changeit -
noprompt -alias $1.cert -file ./$1.pem
keytool -list -keystore ./cacerts.jks -storepass changeit -noprompt
```

IV.4 Command line parameters

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

```
java -jar CiSender.jar -h
```

The available command line parameters are the following:

```
 -c,--config <arg>                Properties file location (Default:
                                  ./resources/CiSender.properties)
 -d,--disablepropertiesfile       Disable load of CISender.properties
 -f,--xmlfile <arg>               Input XML file path. Implies
                                  scanfoldermode=false.
 -h,--help                        Command line help
 -k,--lockfile <arg>              Lockfile location (Default
                                  ./resources/CiSender.lock)
 -kt,--locktimeout <arg>          Lockfile timeout (Default: none)
 -p,--parameterxml <arg>         Parameters XML file path (Default:
                                 _empty_)
 -t,--test                       SOAP_Hello Test to provider
 -u,--serviceurl <arg>           Service URL (Default:
                                 https://civ4/motor_restv2
 -v,--version                    Version info
 -X,--disablesslcncheck          Disable SSL CN check. THINK TWICE!

Debug options:
 -Dorg.slf4j.simpleLogger.defaultLogLevel=DEBUG   -Set loglevel
```

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:

```
[2021-11-24 17:19:41:066] [INFO] [CISender:564] - CISender v2.0 - (c) PartnerHub 2016-
2021.
[2021-11-24 17:19:41:082] [INFO] [CISender:217] - use -h or --help for usage
[2021-11-24 17:19:41:717] [INFO] [ReflectionServiceFactoryBean:436] - Creating Service
{https://civ4/motor_soapv2}Motor_soapv2 from class
com.partnerhub.cisender.MotorSoapv2Soap
[2021-11-24 17:19:42:771] [ERROR] [MotorSoapv2:135] - ERROR: Unable to set TLS
Parameters!: null
```

```
[2021-11-24 17:19:43:943] [INFO] [CISender:613] - SOAP Hello Response: Hello, CISender
v2.0 - (c) PartnerHub 2016-2021. - V2!
[2021-11-24 17:19:43:943] [INFO] [CISender:634] - Application terminates succesfully.
```

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 cisender.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/CiSender.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/CiSender.lock
> date
2018. okt. 18., csütörtök, 09:44:55 CEST
> ./runme.sh
[2018-10-18 09:44:59:383] [INFO] [CiSender:801] - CiSender 2018. V1.1.7
[2018-10-18 09:44:59:390] [INFO] [CiSender:415] - use -h or --help for usage
[2018-10-18 09:44:59:628] [ERROR] [CiSender:370] - ERROR: LockFile exisits - cannot run: ./resources/CiSender.lock
Exception in thread "main" java.lang.IllegalStateException: ERROR: LockFile exisits - cannot run: ./resources/CiSender.lock
        at com.Charlie-India Systemonline.CiSender.applock(CiSender.java:371)
        at com.Charlie-India Systemonline.CiSender.main(CiSender.java:807)


> ./runme.sh  -kt 1
[2018-10-18 09:45:03:026] [INFO] [CiSender:801] - CiSender 2018. V1.1.7
[2018-10-18 09:45:03:033] [INFO] [CiSender:415] - use -h or --help for usage
[2018-10-18 09:45:03:034] [INFO] [CiSender:441] - LockFile Timeout (secs): 1
[2018-10-18 09:45:03:280] [INFO] [CiSender:362] - Lock timeout is greater than expected - IGNORING OLD LOCK, touching lockfile!
[2018-10-18 09:45:03:281] [INFO] [CiSender:827] - Scanning for .xml and .XML files in /home/jtoberling/workspace_java/CiSender/input including those in subdirectories
[2018-10-18 09:45:03:282] [INFO] [CiSender:838] - Number of files found: 0
[2018-10-18 09:45:03:282] [INFO] [CiSender:743] - Checking pending queries...
[2018-10-18 09:45:03:285] [INFO] [CiSender:752] - Number of queries to run in this loop: 0 (max: 100), MaxQueryAttempts: 300
[2018-10-18 09:45:03:316] [INFO] [CiSender:884] - Application terminates.
>
```

V.3 Generating XmlParameters structure

The CISender application can add xmlparameters structure to Charlie-India Soap calls.

- By default the following structure is sent as xmlparameter data:

“<parameters><parameter><filename>*INPUT_FILENAME*[5]</filename></parameters”

- If “parameterxml” file is specified in CISender.properties the xmlparamter data is read from that file.

- In case image (PDF) file is also found and enabled the following structure is generated:
(BASE64IMAGEDATA is the Base64 encoded content of the image file)

“<parameters><parameter><image>BASE64IMAGEDATA</image><filename>*INPUT_FILENAME*</filename></parameters”

- If “parameterstrafo” fille is specified in CISender.properties file than the base XML imput file will be XML transformed by the file specified in “parameterstrafo” path. In such case “param_image” XML parameter is set to the content of BASE64IMAGEDATA.

The following example shows a parameter configuration for parameterstrafo file:

```
<xsl:stylesheet version="1.0"  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:param name="param_image" />
        <xsl:param name="param_filename" />
        <xsl:template match="/">
            <parameters>
                    <parameter>
                            <image><xsl:value-of select="$param_image"/></image>
                            <filename><xsl:value-of select="$param_filename"/></filename>
                    </parameter>
            </parameters>
        </xsl:template>
</xsl:stylesheet>
```

5 The filename of the actually processed xml file.

V.4 Generating DeliveryList structure

he CISender application can add deliverylist structure to Charlie-India Soap calls.

- By default the following structure is sent as deliverylist data:

“</deliverylist>”

- If “deliverylistxml” file is specified in CISender.properties the deliverylist data is read from that file.

- If “deliverylisttrafo” fille is specified in CISender.properties file than the base XML imput file will be XML transformed by the file specified in “deliverylisttrafo” path.

The following example shows a parameter configuration for deliverylisttrafo file:

```
<xsl:stylesheet version="1.0"  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:param name="param_filename" />
        <xsl:template match="/">
            <deliverylist>
                    <delivery>
                              <mode>MAIL_PSEUDOSOAP</mode>
                              <source>result</source>
                              <address><xsl:value-of
select="substring(./Invoice/AccountingCustomerParty/Party/PartyTaxScheme[./TaxScheme/ID='VAT']/
CompanyID,1,8)" />@example.com</address>
                    </delivery>
            </deliverylist>
        </xsl:template>
</xsl:stylesheet>
```

V.5 DocProcessor failure notification

Unsuccesfull notification can be captured by the logging subsystem. Failed events can be also sent by email via the Log4j2 subsystem.

If the DocProcessor function call exits with failure (success=false) the return structure could contain two optional fields:

- errorcode = A Charlie-India system assigned numeric value of the problem

- errorobj = A Charlie-India system assigned error structure .

If CISender application has an errorcode-to-errrormessage mapping structure stored in *resources/microflowerrors.properties* file. If error detected CISender is trying to map the errorcode to an error message – and if CISender configuration enables logs the error accordingly.

V.5.1 CISender properties

The *resources/CISender.properties* file has two parameters to controll notification sending:

- Property “notificationerrors=true” enables the notification handling functionality.

- Property “notificationpattern=.*” holds a regexp pattern to filter logged failures. By default “.*” pattern catches all problems.

V.5.2 EventLogger mail sending

The following Log4j2 (*resources/log4j2.xml*) settings enable sending DocProcessor failures to email addresses:

1.) Create SMTP appender entry for LogMail

```
…
<Appenders>

        <SMTP name="DocProcessorErrorMail"
            to="smtpout@toberling.eu"
            from="sender@test.charlie-india.com"
            smtpHost="smtp-server.test.charlie-india.com"
            smtpPort="465"
            bufferSize="1"
            smtpUsername="smtpoutuser@test.charlie-india.com"
            smtpPassword="Password____"
            smtpProtocol="smtps"
            smtpDebug="false"
            ignoreExceptions="false"
            subject="CI MailWarning">
```

```
            <RegexFilter regex="LogMail: .*" useRawMsg="true" onMatch="ACCEPT" onMismatch="DENY"/>
            <PatternLayout pattern="[%d{yyyy-MM-dd HH:mm:ss:SSS}] %m%n" />
        </SMTP>
...
```

2.) Enable Logger for DocProcessorErrorMapper classs:

```
...
<Loggers>
        <Logger name="com.partnerhub.cisender.DocProcessorErrorMapper"
        level="info" additivity="false">
                <AppenderRef ref="DocProcessorErrorMail" />
        </Logger>
```

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 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.3 Purging of obsolete data

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

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

Purging old data is out of scope of application run.

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[6].

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.
Filed separator: , (comma)
Text delimiter: “ (doublequotes)
Record separator: newline

An example of output can be seen below:

```
2021-11-24,17:39:50.719,/home/jtoberling/workspace_java/cisender/input/pp.xml,SUCCESS,"DocProcessorResult: succes: true DocProcessor(
DocType: _core_party , Method: store_data , Id:9, DocLen:4457 bytes )
Processingtime: 767.0259475708 msec
"
2021-11-24,17:40:42.847,/home/jtoberling/workspace_java/cisender/input/pp-bad.xml,FAILURE,"DocProcessorResult:      succes:      false
DocProcessor( DocType: _core_party , Method: store_data , Id:9, DocLen:1345 bytes )
Processingtime: 359.66300964355 msec
",
```

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	./input/ example_1.xml	Path of processed input file, or AUTOQUERY in case of automatic status query issued
4	SUCCESS	Error code – only if applicable
5	DocProcessorResult: succes: true ….	Charlie-India generated DocProcessor debug message

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

VII.2 Application logs

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

```
[2021-11-24 17:48:00:660] [INFO] [CISender:570] - CISender v2.0 - (c) PartnerHub 2016-2021.
[2021-11-24 17:48:00:672] [INFO] [CISender:217] - use -h or --help for usage
[2021-11-24 17:48:00:674] [INFO] [CISender:243] - LockFile Timeout (secs): 2
[2021-11-24 17:48:00:820] [WARN] [CISender:270] - WARNING: DISABLING SSL CN CHECK! Possible security failure!
[2021-11-24 17:48:01:409] [INFO] [ReflectionServiceFactoryBean:436] - Creating Service
{https://ci2.partnerhub.hu/motor_soapv2}Motor_soapv2 from class com.partnerhub.cisender.MotorSoapv2Soap
[2021-11-24 17:48:02:439] [ERROR] [MotorSoapv2:135] - ERROR: Unable to set TLS Parameters!: null
[2021-11-24 17:48:02:448] [INFO] [CISender:624] - Scanning for .xml and .XML files in
/home/jtoberling/workspace_java/cisender/input including those in subdirectories
[2021-11-24 17:48:02:473] [INFO] [CISender:628] - - processing scanned file:
/home/jtoberling/workspace_java/cisender/input/pp-bad.xml
[2021-11-24 17:48:04:309] [INFO] [CISender:538] - DocProcessorResult: succes: false DocProcessor( DocType:
_core_party , Method: store_data , Id:9, DocLen:1345 bytes )
Processingtime: 308.38489532471 msec

[2021-11-24 17:48:04:312] [INFO] [CISender:481] - File is moved to: ./error/ERROR_20211124_174804_pp-bad.xml
[2021-11-24 17:48:04:363] [INFO] [ResultLogger:34] - /home/jtoberling/workspace_java/cisender/input/pp-
bad.xml,FAILURE,"DocProcessorResult: succes: false DocProcessor( DocType: _core_party , Method: store_data , Id:9,
DocLen:1345 bytes )
Processingtime: 308.38489532471 msec
"
[2021-11-24 17:48:04:364] [INFO] [CISender:640] - Application terminates succesfully.
```

VII.3 Logging SMTP Notifications

The Log4j2 subsystem is capable of sending email notification upon on specific log entries. The following ./resources/log4j2.xml entries are showing a possible configuration for SMTP alerts:

1.) In Appenders section add SMTP appender. Also configure a RegExp filter to catch error patterns.

```
   <Appenders>

...

        <SMTP name="eMail"
           to="smtpout@example.eu"
             from="notify@charlie-india.example.eu"
             smtpHost="mailserver.example.eu"
             smtpPort="465"
             bufferSize="1"
             smtpUsername="smtpoutuser@exammple.eu"
             smtpPassword="<Password>"
             smtpProtocol="smtps"
             smtpDebug="false"
             ignoreExceptions="false"
             subject="Subject of failure">
             <RegexFilter regex=".*RegExp Pattern To Catch by SMTP.*" useRawMsg="true"
onMatch="ACCEPT"
                 onMismatch="DENY"/>
        </SMTP>

…
```

2.) In Loggers section add SMTP appender to the list of loggers.

```
    <Loggers>
        <Root level="INFO">
            <AppenderRef ref="eMail" />
…
```

Please note in case of connecting SMTPS the appropriate server trustChain might be required to registered in the applicable runtime trustStore. Please refer to *IV.3.1 Run time Java SSL configuration variables.*

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](https://keepachangelog.com/en/1.0.0/),
and this project adheres to [Semantic Versioning](https://semver.org/spec/v2.0.0.html).


## [1.0] - 2016-04-11
- Initial release

## [2.0] - 2021-11-23
- CiSender release

## [2.1] - 2021-11-28
- Capability catching and sending microflow errors
- Adding filename parameter to XML trafos

## [2.1.1] - 2021-12-06
- Fixing success return evaluation

## [2.2] - 2021-12-09
- Adding feature pdf FileName replace pattern
```

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 Charlie-India System 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	CiSender application ChangeLog
./README.md	CiSender application basic Readme file
./cisender.jar	CiSender JAR application binary
./resources/1_downloadcert_civ4.sh	Unix shell script example to update keystore – Charlie-India System Test system
./resources/2_downloadcert_cmdlineparamserver.sh	Unix shell script example to update keystore from server provided in command line par
./resources/civ4.pem	PEM certificate of Charlie-India System test system
./resources/cacerts.jks	Java keystore holding Charlie-India System test system certificate
./resources/CISender.properties	CiSender main configuration file
./resources/log4j2.xml	CiSender log4j configuration file
./resources/paramtrafo.xslt	Default Parameterstrafo transformation file
./resources/deliverylist.xslt	Example DeliveryList transformation file
./CiSender_manual_V*.pdf	CiSender Manual
./resources/microflowerrors.properties	CI default errorcode mappings

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/
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 CiSender.properties config

```
# ------------------------------------------------------------ #
#
#  Change this section as of Charlie-India System registration data
#
# Service Endpoint URL
serviceurl=https://ci2.partnerhub.hu/motor_soapv2

# CI Username
username=nimda

# CI Encoded password
password=<B64ENCODEDPASSWORD>=

# CI SOAP identification data
apikey=ABCDKEY
rel_id=999

# DocProcessor parameters
doctype=_core_data
method=send_document

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

# Scan folder mode
scanfolder=true

# Scan images with name '<input.xml>.pdf' - put them to /parameters/image on sending
imagemode=false
#imgfilenamereplacepattern=\.xml
#imgfilenamereplacement=\.pdf

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

#parameterxml=xmlparam.file
parameterstrafo=./resources/paramtrafo.xslt

#deliverylistxml=deliverylist.xml.file
deliverylisttrafo=./resources/deliverylist.xslt

## Do we need to send DocProcessor errors
notificationerrors=true

## RegeExp pattern to catch notification errorcodes to send in mail
notificationpattern=.*
```

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 Charlie-India System 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)
- Collect last referring application logs / submit to support.