BUSINESS CONNECTOR - EXECUTOR

INTEGRATION SPECIFICATION

DOCUMENT CONTROL

Document revision history

Version	Date	Author	Comments
1.2.0	28/01/2014	Development team	Some package names changed (be.connecor doesn't exist anymore) New structure of business connector explained
1.3.0	20/05/2014	Development team	MSCOR exception on .Net 4.0 fixed
1.3.1	12/06/2014	Development team	Using eHealth 3.3.1 version of technical connector Using new IKVM FileLock added on offline queue Wsconsent example added in commandline
1.4.0	25/06/2014	Development team	Added setSystemProperty method to CommonIntegrationModule. Business change, unsealing changed. Added isTest to comWrapper interface (ChapterIV).
1.4.1	16/10/2014	Development team	Register/Update/Revoke is successfully added to offline queue when no session is available. Connector can start with empty productfilter folder/file. Changed chapter IV in properties file. truststore.jks added to p12 folder. Message queue will unlock locked files older than 2 minutes when starting a new session.
1.4.2	31/10/2014	Development team	truststore.jks properties added to the connector-client.properties file. Bugfix register Bvac. Bugfix encoding.
1.4.3	06/10/2015	Development team	MyCareNet Insurability v2
1.5.0	08/12/2015	Development team	- Migration to latest version of eHealth Technical Connector (3.4.0-beta-9). - Migration to latest version of Chapter IV and WSconsent webservices (compatible with Technical Connector 3.4.0-beta-9). - Integration Orchestrated MyCareNet Insurability v2 consultation. - Security modifications. - Minor modifications to COM-interface (based on input ISV). - Addition of self-signed Government CA 2011 certificate to caCertificateKeystore.jks. - Bugfix connector not correctly loading truststore.jks. - Cosmetic changes to connector-client.properties file.

DOCUMENT CONTROL

1.5.1	15/04/2016	Development team	- Migration to latest version of eHealth Technical Connector (3.4.0-beta-10) - Migration to latest version of Chapter IV and WSconsent webservices (compatible with Technical Connector 3.4.0-beta-10). - Revision logging levels. - Addition of new logging parameter: “MaxBackupIndex”. - Revision Queue behavior. - Additional validation of single messages (BVAC and GFD-DPP).
1.6.0	12/10/2016	HealthConnect	- Java to version 1.7 - Single Message to version 2.0.3 - Ehealth connectors to version 3.10.4 (including MyCareNet Insurability BCP functionality) - Addition of 2 new functions: Archive Prescription and Archive Prescription Comment
1.6.1	21/12/2016	HealthConnect	- all services: fixed problem loading truststore in .NET implementation - recip-e: critical bugfix when retrieving prescription of some dentist prescriptions - GFD-DPP: GetData call no longer gets stuck when KGSS returns SOA-error. - RAOTD: fixed problem with DGUID and SGUID being the same - RAOTD: fixed bug not correctly setting correct time in RID.xml file - RAOTD: fixed timestamp problem - Kmehr to version 1.19 - Statusmessage adaptations
1.6.1P	18/07/2018	HealthConnect	Patch for version 1.6.1: - New endpoint STS - SSL-configuration changes services.ehealth.fgov.be (TLSv1.2)
1.6.1P2	20/08/2018	HealthConnect	Patch for version 1.6.1P: - Bugfix create session with eID - Configuration new timestamp certificate production

TABLE OF CONTENT

TABLE OF CONTENT **4**

TABLE OF FIGURES **7**

LIST OF TABLES **7**

1 EXTERNAL REFERENCES **8**

1 INTRODUCTION **9**

1.1 OBJECTIVES *9*

1.2 INTENDED AUDIENCE *9*

1.3 SCOPE *9*

1.4 ASSUMPTIONS *9*

2 COMMON **10**

2.1 BUSINESS CONNECTOR STRUCTURE *10*

2.1.1 CONFIGURATION 10

2.2 CERTIFICATE PROCEDURES *17*

2.2.1 ENVIRONMENTS 17

2.2.2 SOFTWARE REQUIREMENTS 17

2.2.3 INITIATE CERTIFICATE CREATION 18

2.2.4 COMPLETE CERTIFICATE CREATION 20

2.3 AUTHENTICATION & AUTHORIZATION *21*

2.3.1 DEFINITION 21

2.3.2 SESSION CREATION 21

2.3.3 SESSION CREATION FALLBACK 22

2.3.4 SESSION STOP 22

2.3.5 ACCESS TO KEYSTORE 22

2.4 SERVICE INVENTORY *23*

2.4.1 COMMON INTEGRATION MODULE 23

3 RECIP-E **30**

3.1 INTRODUCTION *30*

3.1.1 RECIP-E SOLUTION 30

3.1.2 DESCRIPTION AND PURPOSE OF THE INTEGRATION SPECIFICATIONS 31

3.1.3 DOCUMENT SCOPE 31

3.2 RESPONSIBILITIES *31*

3.2.1 ENCODING 31

3.2.2 IDENTIFICATION OF THE PATIENT 31

3.2.3 PRESCRIPTION FORMAT 31

3.2.4 FEEDBACK FORMAT 36
3.2.5 BARCODE SPECIFICATION 36
3.2.6 ARCHIVING OF PRESCRIPTIONS 37
3.2.7 PERFORMANCE METRICS UPLOAD 37
3.3 INTEGRATION 38
3.3.1 OVERVIEW 38
3.3.2 INTEGRATION DETAIL 39
3.3.3 SERVICE INVENTORY 42
4 GFDDPP 51
4.1 REQUIREMENTS FOR INTEGRATION 51
4.1.1 OPERATION SUPPORT 51
4.1.2 DATA SUPPORT 51
4.2 CONNECTOR MODULE DESIGN 52
4.2.1 APPROACH 53
4.2.2 JAVA 53
4.2.3 DLL 53
4.3 OPERATIONS AND BUSINESS FLOW 55
4.3.1 SESSION MANAGEMENT 55
4.3.2 REFERENCE FLOW 57
4.3.3 INTERFACES & OPERATIONS 59
4.3.4 ERROR CODES 64
4.4 SINGLE MESSAGE FORMAT 64
4.4.1 DELIVERED MEDICATION ENTITY 65
4.4.2 MEDICATION DELIVERY EVENT 67
4.4.3 ASSURPHARMA – BVAC EVENT 68
4.4.4 STATUS ENTITY 69
4.4.5 OTHER TYPES OF DATA 69
4.4.6 MESSAGE STRUCTURES FOR EACH OPERATION 70
5 APPENDIX: EXECUTOR PACKAGE CONTENT 74
6 APPENDIX: RELEASE NOTES 75
6.1 SDK VERSION 0.9.5T (GFD-DPP) 75
6.2 SDK VERSION 0.9.6T (GFD-DPP) 75
6.3 SDK VERSION 0.9.7T (GFD-DPP) 75
6.4 SDK VERSION 0.9.8T (MANDATORY) (GFD-DPP) 75
6.5 SDK VERSION 0.9.9T (GFD-DPP) 76
6.6 SDK VERSION 1.0.0 76
6.7 SDK VERSION 1.1.0 77
6.8 SDK VERSION 1.1.1 77
6.9 SDK VERSION 1.1.2 77

6.10 SDK VERSION 1.2.0 77
6.11 SDK VERSION 1.3.0 77
6.12 SDK VERSION 1.3.1 78
6.13 SDK VERSION 1.4.0 78
6.14 SDK VERSION 1.4.1 78
6.15 SDK VERSION 1.4.2 78
6.16 SDK VERSION 1.4.3 78
6.17 SDK VERSION 1.5.0 78
6.18 SDK VERSION 1.5.1 79
6.19 SDK VERSION 1.6.0 79
6.20 SDK VERSION 1.6.1 80
6.21 SDK VERSION 1.6.1P 80
6.22 SDK VERSION 1.6.1P2 80

7 APPENDIX : FREQUENTLY ASKED QUESTIONS 81

TABLE OF FIGURES

Figure 1: Launch certificate request link. 18
Figure 2: Certificate request application. 18
Figure 3: Certificate choice screen. 19
Figure 4: Key store location. 20
Figure 5: Open key store password screen. 21
Figure 6: Recip-e Solution Overview 30
Figure 7: Integration Modules - Executor 38
Figure 8: Connector module data artifacts 52
Figure 9: Pharmacy Software Integration 55
Figure 10: Single Message hierarchy overview 64
Figure 11: Single Message Specification 65
Figure 12: Delivered Medication Human Entity. 66
Figure 13: Delivered Medication Human History Entity 67
Figure 14: Registration message to TIP 70
Figure 15: Status request & response 71
Figure 16: Data Type Location Entity 72
Figure 17: Get data, type medication history, request to PCDH 72
Figure 18: Error response structure 73

LIST OF TABLES

Table 1: External references 8
Table 2: Authorization combinations 22
Table 3: Kmehr validation checks 32
Table 4: Connector module single message status 52
Table 5: PCDH Module Construction 56
Table 6: PCDH Module Construction 56
Table 7: PCDH Module Construction 59
Table 8: Get Data Types 59
Table 9: Get Data 60
Table 10: Get Pharmacy Details 60
Table 11: PCDH Module Construction 61
Table 12: Register Data 63
Table 13: Register Data 63
Table 14: Revoke Data 63
Table 15: Get Data Types 64
Table 16: Possible attributes for a Motivation Entity 67
Table 17: Quality indicator for PCDH 73

1 EXTERNAL REFERENCES

Ref ID	Name	URL
1.	Barcode Symbology	Specified in ISO/IEC 15417:2007
2.	Non addressed Encryption	https://www.ehealth.fgov.be/nl/support/basisdiensten/systeem-voor-end-end-vercijfering
3.	Addressed Encryption	https://www.ehealth.fgov.be/nl/support/basisdiensten/systeem-voor-end-end-vercijfering
4.	STS Cookbook	https://www.ehealth.fgov.be/nl/support/sts-secure-token-service https://www.ehealth.fgov.be/fr/support/sts-secure-token-service
5.	eHealth certificates	https://www.ehealth.fgov.be/nl/support/basisdiensten/ehealth-certificaten https://www.ehealth.fgov.be/fr/support/services-de-base/certificats-ehealth
6.	eMed-ecare Use cases	https://www.ehealth.fgov.be/nl/support/basisdiensten/systeem-voor-end-end-vercijfering
7.	Barcode print guideline	Specified in ISO/IEC 15416

Table 1: External references

1 INTRODUCTION

1.1 OBJECTIVES

This document establishes a set of requirements and constraints for the interface between Pharmacy Software and the GFD-DPP and Recip-e systems. It also explains how to use the SDK for the connector module. Apart from this document, if possible, we recommend the reader to also take notion of the documents that are referenced in this document for more understanding of the GFD-DPP and Recip-e project, such as:

- **Application Analysis**: e.g. use cases.
- **General Application Design document**: e.g. eHealth processes.
- **Application Design – Connector Module document**: technical design of the connector module

1.2 INTENDED AUDIENCE

This document is intended for use by architects, developers and testers of the applications identified.

- Pharmacy Software Vendors
- Client TSC
- Client support teams

1.3 SCOPE

This document outlines the interface requirements to support the business events in the GFD DPP and Recip-e solution. It also describes the procedures with regard to certificate requests and details the requirements to support encryption & authentication.

Changes are required in Pharmacy software to integrate with GFD-DPP. This document will detail integration procedure expected to be implemented by the Software Providers of said software.

1.4 ASSUMPTIONS

Ref	Description	Validation
1	The pharmacy vendors are capable of either supporting a jar or dll for interfacing with the GFD-DPP and Recip-e solution.	

2 COMMON

2.1 BUSINESS CONNECTOR STRUCTURE

Integration modules: Recip-e provides a reference implementation of a full client able to achieve the different functionalities in scope (including authentication and encryption). These modules are delivered as JAR/DLL libraries. The code of the modules is Open Source (Java code delivered with Apache V2 License)

These files are necessary to ensure that all features of the connector work correctly. We changed the structure of the connector to make it easier to provide a patch if there is a bug. There is no need to release a full version of the connector. We can just provide a .jar and .dll file for the module where the bug is located. This method will only work if it is possible to be backwards compatible. Otherwise, a new version of the connector still needs to be released.

A standard implementation of the connector module as both a **JAR** application and Windows library (.NET DLL + COM wrapper) is provided.

- The .NET dll can be used from code source written in a large number of languages, the most popular being C#, J# or VB.net code. A COM wrapper is also provided, which allows easier integration with languages/environments such as Delphi.
- The JAR library can be used from any application run on the JVM.

The DLL is generated from the Java module. The conversion is performed using the tool IKVM. IKVM.NET is an implementation of Java for Mono and the Microsoft .NET Framework. It includes the following components:

- A Java Virtual Machine implemented in .NET
- A .NET implementation of the Java class libraries
- Tools that enable Java and .NET interoperability

2.1.1 CONFIGURATION

2.1.1.1 MODULE CONFIGURATION

In next section, following notation is used:

- ${INSTALL_DIR}: represents the folder containing the recipe client binary files (Jars or Dlls).
- ..

Configuration parameters for Integration Modules are defined in the file "${INSTALL_DIR}/conf/connector-client.properties"

The Path to the configuration file should be defined in argument of the module constructor.

2.1.1.2 LOGS

The log configuration parameters are defined in the file "${INSTALL_DIR}/conf/log4j.xml".

Refer to log4j documentation for the customization of logging parameters.

The default parameters are:

- The Log level is set to INFO
- The default Log file is set to "connector.log"
- A new file is created each new day; the previous file is then renamed to "connector.log.YYYYMDD".
- A history of 21 days (default) is stored on the file system, older files are removed. This can be set with the "MaxBackupIndex" parameter in log4j.xml.

The possible log levels are:

Level	Description
DEBUG	The DEBUG Level designates fine-grained informational events that are most useful to debug an application.
INFO	The INFO level designates informational messages that highlight the progress of the application at coarse-grained level.
WARN	The WARN level designates potentially harmful situations.
ERROR	The ERROR level designates error events that might still allow the application to continue running.
FATAL	The FATAL level designates very severe error events that will presumably lead the application to abort.

To change the log level you need to change the following value <priority value="**DEBUG**"></priority> to one of the above log levels.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">
        …
<root>
        <priority value="DEBUG"></priority>
        <appender-ref ref="FILE"/>
</root>
</log4j:configuration>
```

2.1.1.3 PROPERTIES

Since we use the eHealth technical connector, a lot of properties have been changed. If it is possible, we use the same properties that the eHealth technical connector needs to manage the calls to their services. We will list all the properties that can be changed. If the properties are not listed here, no change is needed.

```
# Path to the log file.
LOG4J=%CONF%/log4j.xml

# Endpoint locations (needs to be according the environment changes)
endpoint.codage=https://services-acpt.ehealth.fgov.be/codage_1_0/codage
endpoint.seals.v1=https://services-acpt.ehealth.fgov.be/seals/v2/EncodingService
endpoint.ts.consult=https://wwwacc.ehealth.fgov.be/timestampconsult_1_3/timestampconsult
endpoint.ts.authority=https://wwwacc.ehealth.fgov.be/timestampauthority_1_5/timestampauthority
endpoint.sts=https://services-acpt.ehealth.fgov.be/IAM/Saml11TokenService/Legacy/v1
endpoint.etk=https://services-acpt.ehealth.fgov.be/EtkDepot/v1
endpoint.kgss=https://services-acpt.ehealth.fgov.be/Kgss/v1
endpoint.executor=https://services-acpt.ehealth.fgov.be/pilot/Recip-e/v2/Executor_v2
endpoint.executor.v3=https://services-acpt.ehealth.fgov.be/pilot/Recip-e/v3/Executor_v3
endpoint.technical=https://services-acpt.ehealth.fgov.be/pilot/Recip-e/v1/Technical_v1
endpoint.insurability=https://services-acpt.ehealth.fgov.be/pilot/Insurability/v1
endpoint.insurability.v2=https://services-acpt.ehealth.fgov.be/Insurability/v2
endpoint.ch4.consultation.v1=https://services-
acpt.ehealth.fgov.be/ChapterIV/ChapterIVAgreementConsultation/v1
endpoint.ch4.admission.v1=https://services-
acpt.ehealth.fgov.be/ChapterIV/ChapterIVAgreementAdmission/v1
endpoint.wsconsent=https://services-acpt.ehealth.fgov.be/Consent/v1

# Location of the p12 folder
KEYSTORE_DIR=%CONF%/p12/

# The proxy settings
http.proxyHost=
http.proxyPort=
http.proxyUser=
http.proxyPassword=
https.proxyHost=
https.proxyPort=
https.proxyUser=
```

```
https.proxyPassword=

# Information about user, pharmacy-holder and pharmacy (SSIN and NIHII-numbers) used in
the SAML attributes
user.nihii=21463130002
pharmacy.nihii=27456344
pharmacy-holder.nihii=21463130002
pharmacy-holder.ssin=76020727360
pharmacy-holder.firstname=Tom
pharmacy-holder.lastname=Henkens

# Session validation (in hours)
sessionmanager.validity.token=12

# The keystores file names
sessionmanager.holderofkey.keystore=NIHII-PHARMACY=66666219 20130502-144031.p12
sessionmanager.encryption.keystore=NIHII-PHARMACY=66666219 20130502-144031.p12
sessionmanager.identification.keystore=SSIN=86041036109 20131030-155419.p12

# ID's of TIP, TIPSYSTEM and PCDH
default.tip.id=0406753266
pcdhsystem.id=0406753266
tipsystem.id=0406753266

# The message queue of the GFD DPP project
MESSAGE_QUEUE_FOLDER=%CONF%/msg_queue
UNENCRYPTED_FOLDER=%CONF%/unencrypted_folder
INVALID_FOLDER=%CONF%/invalid_folder

# The time, in seconds, between each check if there is something on the queue
MESSAGE_QUEUE_TIMER=5

# The status message queue of the GFD DPP project
STATUS_MESSAGE_QUEUE_FOLDER=%CONF%/status_msg_queue

# Folder of all the returned TIP System configuration files.
TIP_CONFIGURATION_PATH=%CONF%/update

# Folder of the productfilter files.
PRODUCT_FILTER_PATH=%CONF%/update/productfilter

# Folder of the system services files.
SYSTEM_SERVICES_PATH=%CONF%/update/systemservices

# Number of threads started to decrypt the getData products (important: if you change this
number it will have an impact on the performance)
thread.number.limit=10

#Activation/deactivation of the insurability when retrieving a prescription.
patient.insurability.disable=false
```

Following properties are added for the Chapter IV, WSconsent and RAOTD-implementation.

```
kmehr.default.hcpartylist=software,pharmacy,pharmacistholder,identifier

kmehr.default.software.id.local.sl.1=application_ID
kmehr.default.software.id.local.sv.1=1.0
kmehr.default.software.id.local.value.1=${package.id}
kmehr.default.software.cd.cdhcparty.value=application
kmehr.default.software.cd.cdhcparty.sv=1.1
kmehr.default.software.name=${package.name}

kmehr.default.pharmacy.id.idhcparty.value=${pharmacy.nihii}
```

```
kmehr.default.pharmacy.id.idhcparty.sv=1.0
kmehr.default.pharmacy.cd.cdhcparty.value=orgpharmacy
kmehr.default.pharmacy.cd.cdhcparty.sv=1.1
kmehr.default.pharmacy.id.name=${pharmacy.name}

kmehr.default.pharmacistholder.id.inss.value=${pharmacy-holder.ssin}
kmehr.default.pharmacistholder.id.inss.sv=1.0
kmehr.default.pharmacistholder.id.idhcparty.value=${pharmacy-holder.nihii}
kmehr.default.pharmacistholder.id.idhcparty.sv=1.0
kmehr.default.pharmacistholder.cd.cdhcparty.value=${main.kmehr.quality}
kmehr.default.pharmacistholder.cd.cdhcparty.sv=1.1
kmehr.default.pharmacistholder.firstname=${pharmacy-holder.firstname}
kmehr.default.pharmacistholder.lastname=${pharmacy-holder.lastname}

kmehr.default.identifier.id.inss.value=${user.inss}
kmehr.default.identifier.id.inss.sv=1.0
kmehr.default.identifier.id.idhcparty.value=${user.nihii}
kmehr.default.identifier.id.idhcparty.sv=1.0
kmehr.default.identifier.cd.cdhcparty.value=${main.kmehr.quality}
kmehr.default.identifier.cd.cdhcparty.sv=1.1
kmehr.default.identifier.firstname=${user.firstname}
kmehr.default.identifier.lastname=${user.lastname}

mycarenet.commonbuilder.class=be.ehealth.business.mycarenetcommons.builders.impl.Gene
ric CommonBuilderImpl
mycarenet.default.careprovider.nihii.quality=pharmacy-holder
mycarenet.default.careprovider.nihii.value=${pharmacy-holder.nihii}
mycarenet.default.careprovider.physicalperson.name=${pharmacy-holder.firstname}
mycarenet.default.careprovider.physicalperson.ssin=${pharmacy-holder.ssin}
mycarenet.default.careprovider.organization.name=${pharmacy.name}
mycarenet.default.careprovider.organization.nihii.quality=pharmacy
mycarenet.default.careprovider.organization.nihii.value=${pharmacy.nihii}

main.kmehr.quality=perspharmacist
package.name=
mycarenet.licence.username=
mycarenet.licence.password=

chapterIV.keydepot.identifiertype=CBE
chapterIV.keydepot.identifiersubtype=
chapterIV.keydepot.identifiervalue=820563481
chapterIV.keydepot.application=MYCARENET
validation.incoming.chapterIV.consultation.message=false
validation.incoming.chapterIV.admission.message=false

chapterIV.package.name=${package.name}
chapterIV.package.licence.username=${mycarenet.licence.username}
chapterIV.package.licence.password=${mycarenet.licence.password}
chapterIV.kmehrbuilder.class=be.ehealth.businessconnector.chapterIV.builders.impl
.GenericKmehrBuilder
mycarenet.chapterIV.usedefaultproperties=true
kmehr.chapterIV.usedefaultproperties=false
kmehr.chapterIV.identifier.id.idhcparty.value=${pharmacy-holder.nihii}
kmehr.chapterIV.identifier.id.idhcparty.sv=1.0

kmehr.chapterIV.transactionAuthor.usedefaultproperties=false
kmehr.chapterIV.transactionAuthor.hcpartylist=pharmacistholder
kmehr.chapterIV.transactionAuthor.pharmacistholder.id.inss.value=${pharmacy-
holder.ssin}
kmehr.chapterIV.transactionAuthor.pharmacistholder.id.inss.sv=1.0
kmehr.chapterIV.transactionAuthor.pharmacistholder.id.idhcparty.value=${pharmacy-
holder.nihii}
kmehr.chapterIV.transactionAuthor.pharmacistholder.id.idhcparty.sv=1.0
kmehr.chapterIV.transactionAuthor.pharmacistholder.cd.cdhcparty.value=${main.kmeh
```

```
r.quality}
kmehr.chapterIV.transactionAuthor.pharmacistholder.cd.cdhcparty.sv=1.1
kmehr.chapterIV.transactionAuthor.pharmacistholder.firstname=${pharmacy-
holder.firstname}
kmehr.chapterIV.transactionAuthor.pharmacistholder.lastname=${pharmacy-
holder.lastname}

kmehr.chapterIV.sender.usedefaultproperties=false
kmehr.chapterIV.sender.hcpartylist=pharmacy,pharmacistholder
kmehr.chapterIV.sender.pharmacy.id.idhcparty.value=${pharmacy.nihii}
kmehr.chapterIV.sender.pharmacy.id.idhcparty.sv=1.0
kmehr.chapterIV.sender.pharmacy.cd.cdhcparty.value=orgpharmacy
kmehr.chapterIV.sender.pharmacy.cd.cdhcparty.sv=1.0
kmehr.chapterIV.sender.pharmacy.name=${pharmacy.name}

kmehr.chapterIV.sender.pharmacistholder.id.inss.value=${pharmacy-holder.ssin}
kmehr.chapterIV.sender.pharmacistholder.id.inss.sv=1.0
kmehr.chapterIV.sender.pharmacistholder.id.idhcparty.value=${pharmacy-
holder.nihii}
kmehr.chapterIV.sender.pharmacistholder.id.idhcparty.sv=1.0
kmehr.chapterIV.sender.pharmacistholder.cd.cdhcparty.value=${main.kmehr.quality}
kmehr.chapterIV.sender.pharmacistholder.cd.cdhcparty.sv=1.0
kmehr.chapterIV.sender.pharmacistholder.firstname=${pharmacy-holder.firstname}
kmehr.chapterIV.sender.pharmacistholder.lastname=${pharmacy-holder.lastname}

kmehr.chapterIV.sender.identifier.id.inss=${user.inss}
kmehr.chapterIV.sender.identifier.id.idhcparty.value=${user.nihii}
kmehr.chapterIV.sender.identifier.id.idhcparty.sv=1.0
kmehr.chapterIV.sender.identifier.cd.cdhcparty.value=${main.kmehr.quality}
kmehr.chapterIV.sender.identifier.cd.cdhcparty.sv=1.0
kmehr.chapterIV.sender.identifier.firstname=${user.firstname}
kmehr.chapterIV.sender.identifier.lastname=${user.lastname}

timestamp.signature.keystore.path=tsacertificate.jks
timestamp.signature.keystore.pwd=ttsa
timestamp.signature.cert.alias.1=ttsa01
timestamp.signature.cert.alias.2=ttsa02
timestamp.signature.cert.alias.3=ttsa03
timestamp.signature.cert.alias.4=ttsa04
timestamp.signature.cert.alias.5=ttsa06
timestamp.signature.cert.alias.6=ttsa07
timestamp.signature.cert.alias.7=ttsa08

be.ehealth.businessconnector.wsconsent.builders.HcPartyInfoBuilder.class=be.eheal
th.businessconnector.wsconsent.builders.impl.DelegatingHcPartyInfoBuilderImpl
kmehr.wsconsent.usedefaultproperties=true
wsconsent.maxrows=1000
validation.incoming.wsconsent.message=true
package.id=

# RAOTD-specific properties
ArchivePrescription-xml=true
ArchivePrescriptionDirectory=%CONF%/Archive/Prescription/
validate.befor.archiving=true
sdk.version=1.6.0
supported.prescription.version=1.17.1
prescription.version=1.17.1
```

2.1.1.3.1 KEYSTORE PROPERTIES DEFINED IN CONFIGURATION FILE

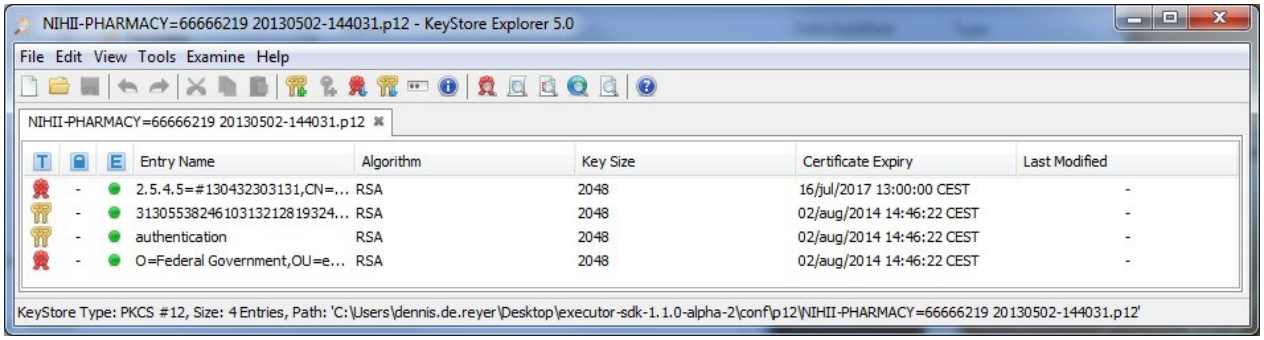
The properties of the key store can be defined in the configuration file as follows:

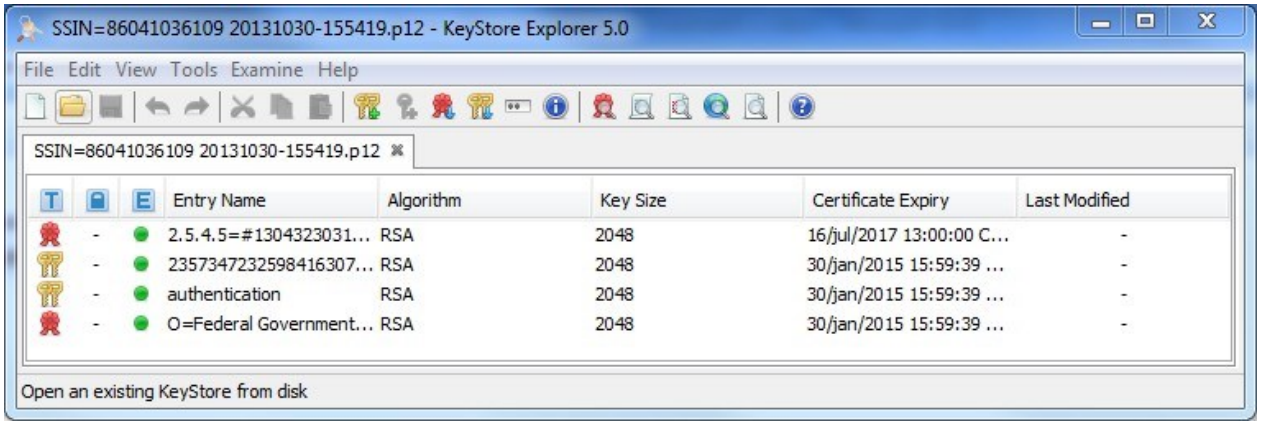
- KEYSTORE_DIR: PATH of the p12 folder where the keystores has been stored. The keystore must be a PKCS12 keystore ("*.p12"). The authentication certificate must have the alias "authentication".
- This folder must also contain the caCertificateKeystore.jks used for connection to eHealth services. The correct caCertificateKeystore file needs to be copied from the **acc or prod folder** (depending on the environment you want to test) to the p12 folder.

Executor:

- sessionmanager.holderofkey.keystore: name of the pharmacy certificate.
- sessionmanager.encryption.keystore: name of the pharmacy certificate.
- sessionmanager.identification.keystore: personal certificate in case of a fallback session.

Overview of a system key store opened with Keystore explorer:



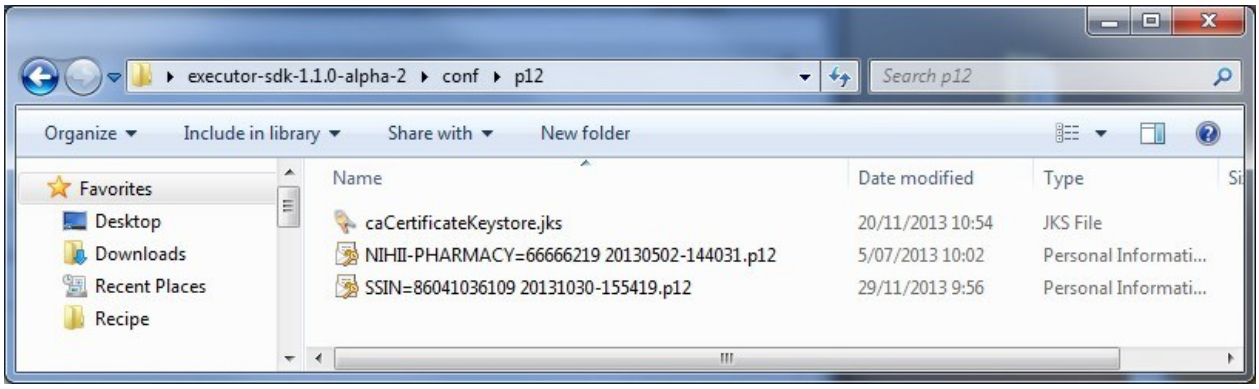


Sample key store section of the connector-client.properties configuration file:

```
KEYSTORE_DIR=%CONF%/p12/
sessionmanager.holderofkey.keystore=NIHII-PHARMACY=66666219 20130502-144031.p12
sessionmanager.encryption.keystore=NIHII-PHARMACY=66666219 20130502-144031.p12
sessionmanager.identification.keystore=SSIN=86041036109 20131030-155419.p12
```

For multi-user access or fall-back session creation (e.g.: several doctors sharing the same workstation), it is possible to copy the personal key pair in a specific folder identified by the configuration parameter KEYSTORE_DIR. The naming convention for the keypair is "SSIN=<niss of the care provider>_<creation time>.p12". (Same name then the one generated by e-health procedure). See Annex "Certificate Creation"

Example of content of the KEYSTORE_DIR:



2.1.1.3.2 KEYSTORE PROPERTIES PROVIDED WITH METHOD SETSYSTEMKEYSTOREPROPERTIES

The method setsystemkeystoreproperties makes it possible to set your system keystore properties in code and not with the configuration file.

```
setSystemKeystoreProperties(String systemKeystorePassword, String systemKeystorePath, String systemKeystoreDirectory, String systemKeystoreRizivKBO){
```

The minimal information that needs to be provided is

- The system keystore password.

2.1.1.3.3 PREVIOUS KEYSTORE PROPERTIES PROVIDED WITH METHOD SETOLDSYSTEMKEYSTOREPROPERTIES

The method setOldSystemKeystoreProperties makes it possible to provide a previous system keystore to unseal recip-e notifications or feedbacks that were sealed with an older certificate.

```
setOldSystemKeystoreProperties(String systemKeystorePassword, String systemKeystorePath, String systemKeystoreDirectory, String systemKeystoreRizivKBO){
```

The minimal information that needs to be provided is

- The system keystore password.

This method will need to be set when a user requests a new certificate. If the method is not set and a notification/feedback, sealed with the old certificate, is retrieved you will receive the following exception:

```
[ERR200004] An error occurred while unsealing the data. If the problem persists, please contact the helpdesk.
```

2.2 CERTIFICATE PROCEDURES

GFD-DPP and Recip-e are end-to-end secured systems using the eHealth platform. Before any interaction with eHealth can be established, the system and user must be able to identify themselves:

- **At least two eHealth certificates must be configured** (even if there's only one employee in a pharmacy):
  - The **system or organization certificate**, specific to the pharmacy.
  - The **personal certificate**, specific to the user/pharmacist.

  Both can be requested on the eHealth website as explained next.
- Next to the personal certificate, **the eID is used for user authentication and session negotiation with eHealth**, preferably at software startup. After 12 hours, the session is invalidated and must be established again.
  - Consequently a correctly installed eID reader is required.
  - If there is a problem with the eID or the reader, a fallback session can be created using only the personal certificate and corresponding password. However to discourage this less secure authentication method, the fallback sessions only have a validity of one hour before needing renewal.

Keep in mind that a certificate request is a two-step process. Any request will be evaluated and approved individually. Between the initial request and completion, a few business days may pass.

2.2.1 ENVIRONMENTS

Certificates are only valid for a specific runtime environment (integration, acceptance, production).

- **The link below requests certificates for the acceptance environment**. For production, only real pharmacists are able to request certificates. For integration, one must request the URL to eHealth.

- Which environment is used is configured by changing the end-point URLs for the web services.
  - In the connector module, the end-points for eHealth services are configurable in the connector properties.

2.2.2 SOFTWARE REQUIREMENTS

Java must be installed (http://www.java.com/en/download/manual.jsp).

- For Microsoft Windows, refer to http://support.microsoft.com/kb/827218 to find out what platform is running.
  - On a 32-bit platform, one can only install the 32-bit edition of Java, this is fine.
  - On a 64-bit platform, the **32-bit edition of Java must be used as well**. *The 64-bit edition of Java does not support reading eID cards (which uses PKCS#11 cryptography) as explained here:*
    ⇨ http://docs.oracle.com/javase/6/docs/technotes/guides/security/p11guide.html#Requirements

If both 32-bit and 64-bit versions of Java are installed on Windows, one must make sure the certificate request module and the connector module are run on the 32-bit version. *Remove the 64-bit version to remove this complexity – all java programs will continue to run*.

The eID middleware (http://eid.belgium.be) must be installed and an eID reader must be connected with the eID of the requestor inserted.

2.2.3 INITIATE CERTIFICATE CREATION

1. Visit the website https://www.ehealth.fgov.be/nl/support/basisdiensten/ehealth-certificaten and click on the link “Please use this application”, above the download table (see figure below).

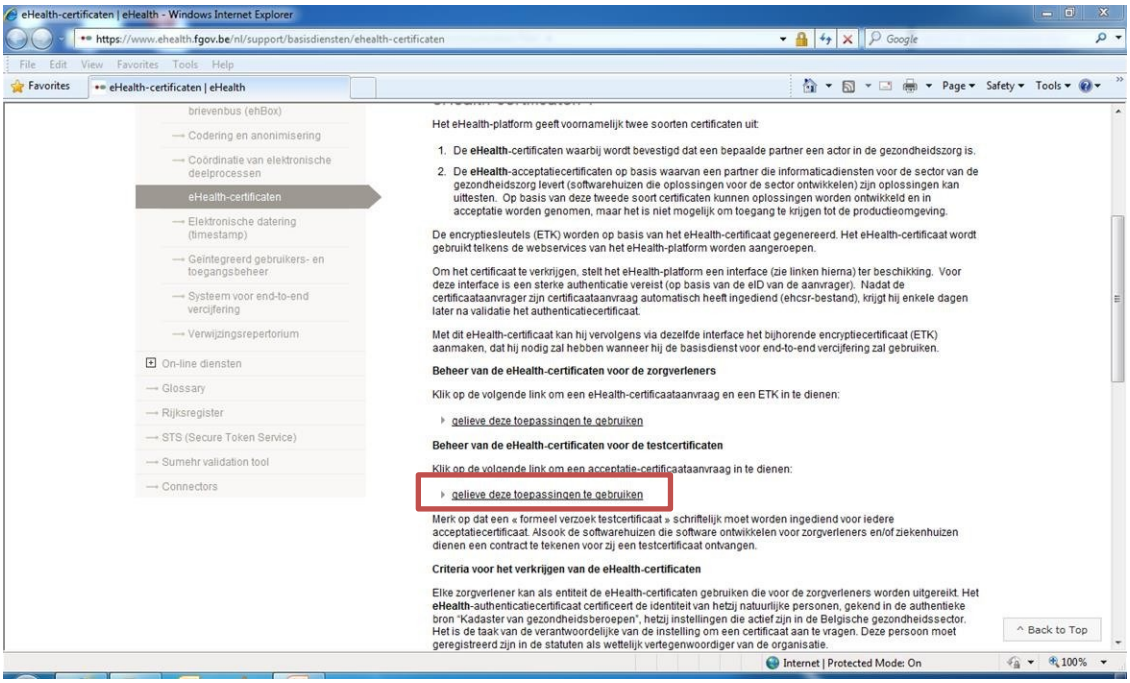


Figure 1: Launch certificate request link.

2. Click "Run” on the java dialog box that pops up.
3. Click the “Create new request” button (see figure).

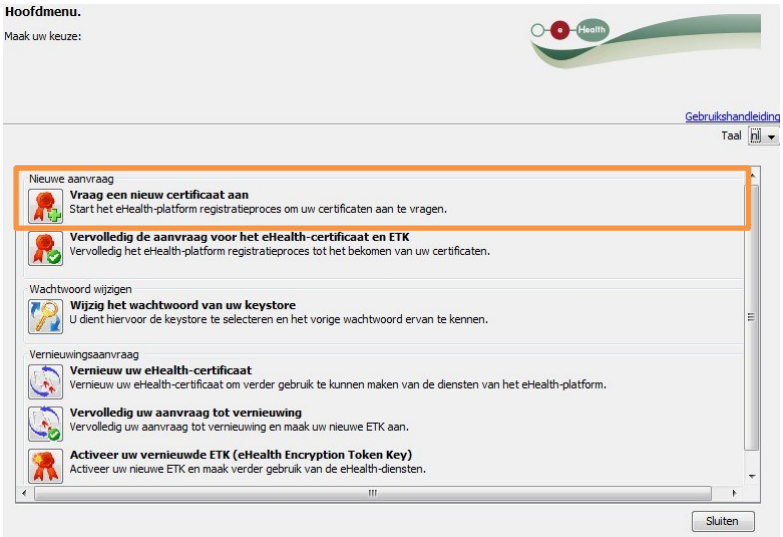


Figure 2: Certificate request application.

4. Read and accept the terms and conditions.
5. The eID card is read, click “next” (this button is only active when the eID can properly be read).
6. On the next screen (see figure), pick the option that displays your name for the personal certificate.

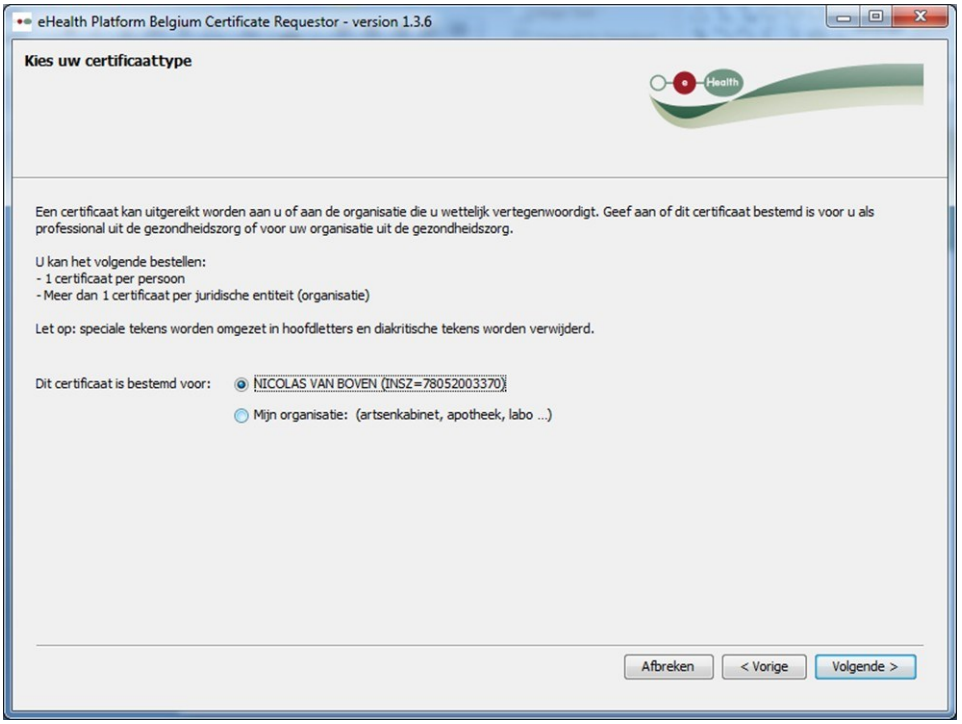


Figure 3: Certificate choice screen.

7. Fill your personal coordinates, only the email and telephone number are required.
8. A summary is displayed, click next.
9. Enter the pin for your eID when prompted.
10. Choose a password for your personal certificate. **This password is needed for the fallback session so remember it well**.
11. The next screen shows the location of the newly created key store. **This key store must be configured in the connector module in a later stage, so make sure to remember the location**.

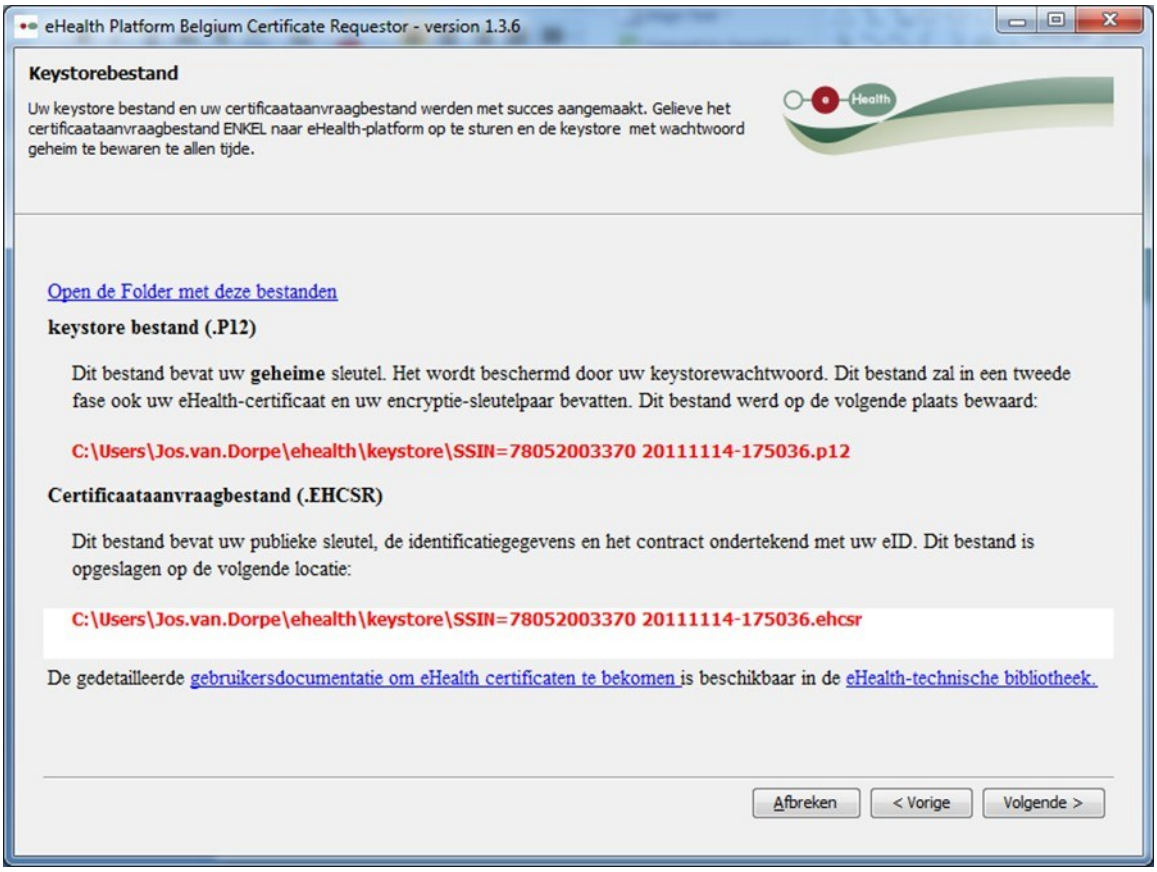


Figure 4: Key store location.

12. As a final step, the certificate request is sent to eHealth.

For the system certificate, the organization CBE number is required (lookup: http://kbopub.economie.fgov.be/kbopub/) and for pharmacies in particular, the NIHII number. Steps 1 – 6 and 9 – 13 are identical as for the personal certificate, on step 7 choose the second option instead.

8. Fill the information as requested of your organization. **Leave the application ID empty**.

If there is a problem with the certificate request, contact the eHealth helpdesk. A direct link is present within the certificate request application. Alternatively, the helpdesk can be contacted on the following phone number: 02 788 51 55 or via the website https://www.ehealth.fgov.be/nl/contact.

2.2.4 COMPLETE CERTIFICATE CREATION

When the requestor is contacted through one of the coordinates included in the certificate request, he/she must open the eHealth application again as explained in the previous section.

4. Click on the “Complete request” button.
5. Browse for and select the key store as created in the request.
6. Fill the password as chosen for the request.

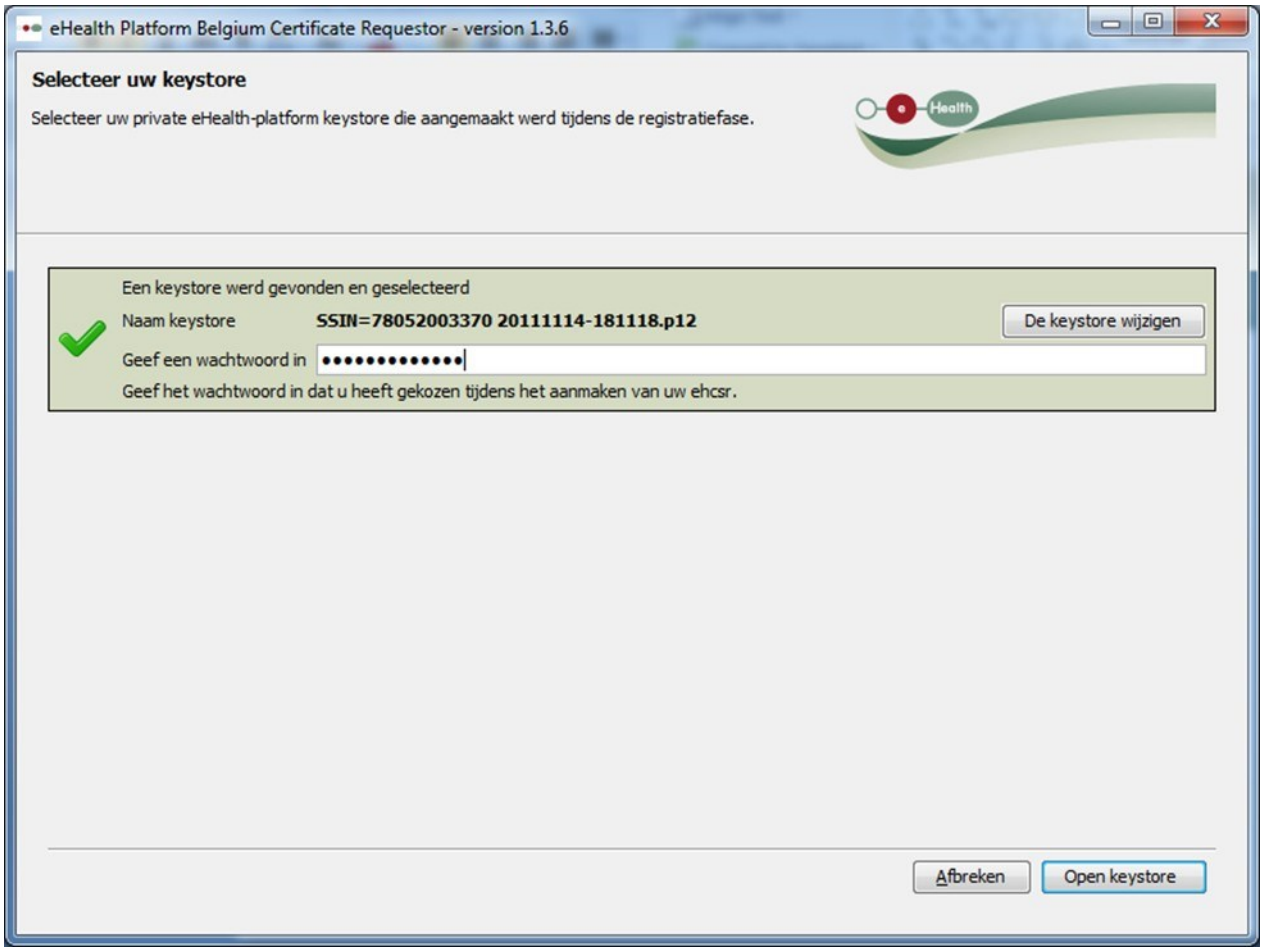


Figure 5: Open key store password screen.

7. Confirm certificate completion.

More documentation and other procedures such as revocation can be found on the eHealth website.

2.3 AUTHENTICATION & AUTHORIZATION

2.3.1 DEFINITION

In this document, what is named the "session" is in practice a SAML token generated by eHealth. Once obtained by the care provider, this token allows calling each one of the Recip-e services. The validity of this token is limited in time.

2.3.2 SESSION CREATION

The authentication must be done at session start-up and uses

- eID of a physical user (+ PIN code)
- organizational (Executor) or personal (Prescriber) eHealth Certificate, linked to a particular software installation and containing a responsible person

This authentication is validated by eHealth and a work session (by means of a SAML token) is generated. The validity of the session is sector dependant. This should be a variable defined with default and maximum values as 5H for Prescribers and 12H for Executors. This also means that the prescribers and executors will have not have to enter their eID and pin more than once or twice per day (i.e. after their session token expires)

Following combinations will be allowed to access the Recip-e system:

Care Provider	eID	Certificate
Prescriber (e.g. doctor)	e-ID of prescriber	Personal eHealth certificate (identified by the SSIN)
Pharmacist	e-ID of pharmacist	EHealth certificate linked to the pharmacy (identified by the NIHII-PHARMACY). This certificate also contains a responsible (does not have to be a pharmacist; e.g. owner of pharmacy).
Non-Pharmacist Executor (e.g. physiotherapist, nurse …)	e-ID of executor	eHealth certificate containing a responsible (does not have to be an executor; e.g. director in retirement home …)

Table 2: Authorization combinations

The software provider has the responsibility to make sure that the care provider:

- Has a valid card reader setup on his machine, this reader must be able to read Belgium EID card (card reader requirement are described on web site http://eid.belgium.be)
- Has a valid eHealth certificate stored in a secured Keystore (p12 file).

2.3.3 SESSION CREATION FALLBACK

A fallback exists to create the session in case of the care provider does not have his EID with him, his EID is broken, ... This fallback uses the personal encryption key of the executor (used for addressed encryption), instead of using the EID.

The session created with the fallback mechanism is limited in time.

Important: This fallback shall not be the default authentication solution. On top of that, it is expected from software vendors that they does not automate the user/password input (no password recording possible). Expected is that a warning message is shown in the software.

2.3.4 SESSION STOP

Once authenticated, modules keep in memory the SAML token. This SAML token is a key element is authentication mechanism and therefore must be destroyed if the care provider does not need to use services any more.

It is expected from software provider that this session is correctly destroyed (See API unloadSession) once the care provider shutdown his software/computer.

Software providers also have the possibility to secure even more this token by unloading it in case of care provider inactivity. Ex: if the prescriber do not use his software for a defined period (ex: 30 minutes), the session is automatically unloaded. If the prescriber wishes to use a service again, he will have to enter his PIN code.

2.3.5 ACCESS TO KEYSTORE

Since the keystore contains the different certificates and public/private keys of all people using the pc, special attention should be put on the protection and access to this keystore. The software vendors are expected to adhere to following points:

- The software should allow the end-user to easily delete his personal keys from the keystore. The taken approach should be user-friendly (e.g. without the user having to enter the different keystore commands)
- The software should allow an administrative person (e.g. the owner of the pharmacy, the main doctor …) to easily delete any of the personal keys from the keystore. The taken approach should be user-friendly (e.g. without the user having to enter the different keystore commands)
- The software should enforce that only the user to which the keys belong, has access to his personal keys. This to prevent for instance the case where a user would be able to start an emergency session in the name of another user whose keys reside in the same keystore. This can for instance be done by linking the concept of a user account to the password which is used to access the private keys of a particular user.

2.4 SERVICE INVENTORY

2.4.1 COMMON INTEGRATION MODULE

Once the certificates are available, the connector module must be pointed to the key stores containing them using one of the configuration options. A session can be started using the start session operation.

- Because of the decoupling of the technical connector and the business connector, session management is different from previous versions and is contained in its own component. This component is consulted by each business integration module, achieving single-sign-on for every service provided, not just recip-e/mycarenet.

Once authenticated, the integration module keeps the SAML token in memory. This SAML token is a key element in authentication mechanism and therefore must be destroyed if the care provider does not need to use any service any more.

- **Interface** be.business.connector.common.CommonIntegrationModule
- **Implementation**: be.business.connector.common.CommonIntegrationModuleImpl

The module is constructed using the above implementation class, it requires the following parameters (*there are extra helper constructors available, see source code*):

Signature	***CommonIntegrationModuleImpl(String propertyfile) throws IntegrationModuleException***
Parameters	***Propertyfile*** *(required): path to property file containing configuration for the connector module. This path can either be absolute (e.g. "C:\connector-client.properties") or relative to the class path of the application (e.g. "/connector-client.properties"). If using IKVM, some experimentation may be needed to find out this classpath.*

Signature	***CommonIntegrationModuleImpl(String propertyfile , String urlConfig) throws IntegrationModuleException***
Parameters	***Propertyfile*** *(required): path to property file containing configuration for the connector module. This path can either be absolute (e.g. "C:\connector-client.properties") or relative to the class path of the application (e.g. "/connector-client.properties"). If using IKVM, some experimentation may be needed to find out this classpath.* ***urlConfig*** *(required): URL that replace the %CONF% in the property file that contains the configuration for the connector module.* *The property "PERSONAL_KEYSTORE" is required in the property file to use the fallback session.*

2.4.1.1 CREATESESSION

Operation Name	***createSession***			
Class name	***CommonIntegrationModule***			
Description	*This function creates a user session based on the EID of the user (PIN code is then required) and his eHealth certificate.* *The session created is an XML SAML token; this token is not linked to a particular machine. Therefore, on the one hand, it must be protected to prevent malicious system to stole this token, on the other hand, the token can be shared in multi computer environment (such as pharmacy).See "loadSession"*			
When to be triggered	*When the user starts his application* *When the stored token has exceeded his validity period*			
Errors	*Technical Services Exception such as Connection failure, Encryption failure EID card reader issue (no card, unreadable card).*			
Inputs Integration Module	**Order**	**Type**	**Name**	**Description**
Output Integration Module		**Type**	**Name**	**Description**
		String	*xmlSamlToken*	*SAML response obtained from eHealth STS service. The SAML token has a validity of 5H for prescribers, 12H for executors. (This time is configurable, and will be further refined during the course of the pilot). This SAML token can be stored by the software provider (in a file or in a database). The function "loadSession" should be used to restore the session.*
Implementation Detail	**Step**	**Description**		
	1	*The module initializes the BEId framework. Read the EID card*		
	2	*PIN code of the end-user is requested*		
	3	*A SAML Request is created and signed with the certificate and sent to eHealth STS*		
	4	*The module returns the SAML token as an XML message. This XML message can be stored on the hard disk (Cookie like)*		

2.4.1.2 LOADSESSION"

Operation Name	***loadSession***			
Class name	***CommonIntegrationModule***			
Description	*This function should be used to reload a stored session*			
When to be triggered	*When the prescriber wants to reload an active session (after a computer shutdown)*			
Errors	*Technical Services Exception such as Connection failure, Encryption failure EID card reader issue (no card, unreadable card).*			
Inputs Integration Module	**Order**	**Type**	**Name**	**Description**
	1	*String*	*xmlSamlToken*	*The SAML token received by "createSession"*
Output Integration Module		**Type**	**Name**	**Description**
		#N/A	*#N/A*	*#N/A*
Implementation Detail	**Step**	**Description**		
	1	*The module is initialized with the given SAML token, all WS exchanges with Recip-e will use this token.*		

2.4.1.3 UNLOADSESSION

Operation Name	***unloadSession***			
Class name	***CommonIntegrationModule***			
Description	*Unload the current session.*			
When to be triggered	*When the user is shutting down his application* *After a long period of inactivity (definition of this period of inactivity under the responsibility of software providers).*			
Errors	*#N/A*			
Inputs Integration Module	**Order**	**Type**	**Name**	**Description**
Output Integration Module		**Type**	**Name**	**Description**
		#N/A	*#N/A*	*#N/A*
Implementation Detail	**Step**	**Description**		
	1	*The module destroys is current SAML token.*		

2.4.1.4 CREATEFALLBACKSESSION

Operation Name	***createFallbackSession***			
Class name	***CommonIntegrationModule***			
Description	*Create a session using eHealth certificates (EID is not needed)*			
When to be triggered	*When the user does not have his EID with him.*			
Errors	*Invalid password*			
Inputs Integration Module	**Order**	**Type**	**Name**	**Description**
	1	*String*	*userId*	*The NIHII Id of the care provider (INAMI/RIZIV for prescriber, NIHII-PHARMACY for pharmacists)*
	2	*String*	*passphrase*	*The passphrase of the Keystore containing eHealth certificates.*
Output Integration Module		**Type**	**Name**	**Description**
		String	*xmlSamlToken*	*SAML response obtained from eHealth STS service. The SAML token has a validity of 1H. This SAML token can be stored by the software provider (in a file or in a database). The function "loadSession" should be used to restore the session.*
Implementation Detail	**Step**	**Description**		
	1	*A SAML Request is created and signed with the certificate and sent to eHealth STS*		
	2	*The module returns the SAML token as an XML message. This XML message can be stored on the hard disk (Cookie like)*		

2.4.1.5 HASVALIDSESSION

Operation Name	***hasValidSession***			
Class name	***CommonIntegrationModule***			
Description	*Check if the current session (SAML token) is valid.*			
When to be triggered	*Before calling a Recip-e service, this method check that session has not expired. This method can be called at any time to check the validity of the SAML token.*			
Errors				
Inputs Integration Module	**Order**	**Type**	**Name**	**Description**
Output Integration Module		**Type**	**Name**	**Description**
		Boolean	*sessionValidity*	*A Boolean that states if the session is still valid.*
Implementation Detail	**Step**	**Description**		
	1	*The module checks if the current session is still valid.*		

2.4.1.6 SETPROPERTY

Operation Name	***setProperty***
Class name	***CommonIntegrationModule***
Description	*Sets a property in the PropertyHandler and the eHealth properties.*
When to be triggered	*When a user wants to set or override a property.*
Errors	

Inputs Integration Module	Order	Type	Name	Description
	1	*String*	*Key*	*The property key*
	2	*String*	*value*	*The property value*

Output Integration Module	Type	Name	Description

Implementation Detail	Step	Description
	1	*The property is added to the PropertyHandler*
	2	*The property is added to the eHealth properties*

2.4.1.7 SETSYSTEMPROPERTY

Operation Name	***setSystemProperty***
Class name	***CommonIntegrationModule***
Description	*Sets a property to the SystemProperties*
When to be triggered	*When a user wants to set a system property*
Errors	

Inputs Integration Module	Order	Type	Name	Description
	1	*String*	*Key*	*The property key*
	2	*String*	*value*	*The property value*

Output Integration Module	Type	Name	Description

Implementation Detail	Step	Description
	1	*The property is added to the SystemProperty*

3 RECIP-E

3.1 INTRODUCTION

3.1.1 RECIP-E SOLUTION

The Recip-e solution concerns the generic (i.e. valid for all types of prescription: pharmaceutical, physiotherapist, nursing ...) transfer of prescriptions from the prescriber to the care provider, for example from the general practitioner (GP) to the pharmacist, chosen freely by the patient or from the specialist to the physiotherapist.

With the Recip-e solution, data can be sent between various actors with a high level of security. This technological innovation also offers improvements for everyone involved in the project. Below is a list of the added value that the Recip-e solution offers:

- Ensure roles and responsibilities of everyone;
- Integration with medical platforms for the identification of the actors, the security of the data and the control of the insurability of patients to ensure (eg eHealth, MyCareNet)
- Improving the administrative process and reduce administrative burdens
- Reduce erroneous prescriptions (errors in prescriptions)
- Relationship between the electronic and the paper stream is guaranteed
- Traceability of the data between the different actors.
- Traceability of the data access (consult) for privacy logging

There are also immeasurable positive impacts foreseen thanks to the solution:

- Enhancing of the process (less fraud by avoiding patients to create fake prescriptions);
- …

Technically speaking Recip-e is not only a system that manages non-addressed messages in the health sector. Recip-e is also a system that provides advanced functionality such as prescription state, prescription validation, unique document numbering…

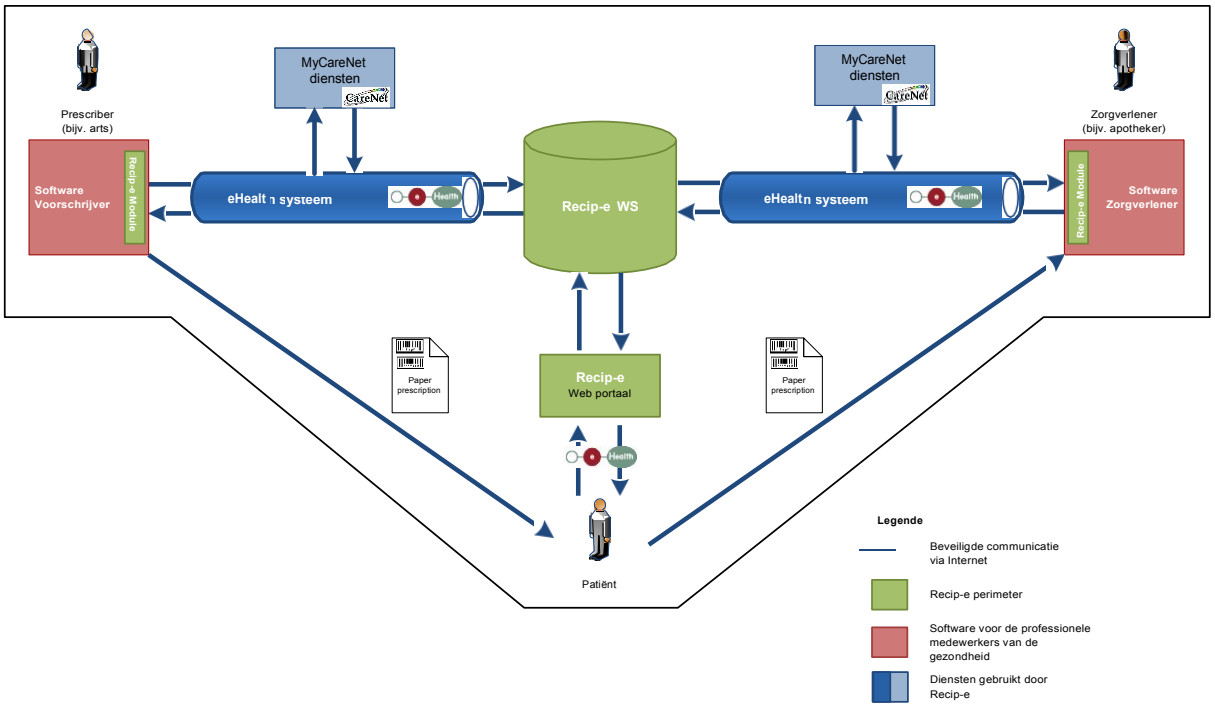


Figure 6: Recip-e Solution Overview

3.1.2 DESCRIPTION AND PURPOSE OF THE INTEGRATION SPECIFICATIONS

This document establishes a set of requirements for the interface between Executor Software and the RECIP-E system. It identifies agreed-upon design requirements and constraints that must be satisfied by the interfacing software. This document is intended for use by the developers of the applications identified, and by the test organizations responsible for the testing of these applications.

3.1.3 DOCUMENT SCOPE

This document outlines the interface requirements to support the following business events for Execution software (pharmacist, nurses …):

- (Un)Deliver a prescription
- Cancel a prescription
- Create a feedback for a prescription
- Read a prescription notification
- Read Prescription

The document also details the requirements to support the following technical events:

- Encryption
- Authentication

Changes are required in Executor software to integrate with RECIP-E. This document will detail integration procedure expected to be implemented by the Software Providers of said software.

Note: The scope of this document will be extended in the near future to include an “insurability” check of the patient coming from MyCarenet services.

3.2 RESPONSIBILITIES

The software provider will have to make sure that his application fulfils a set of requirements and thus whatever the integration option chosen.

3.2.1 ENCODING

All recip-e messages (prescriptions, feedback and notifications) should use the **UTF-8** encoding.

3.2.2 IDENTIFICATION OF THE PATIENT

The Patient must be identified by his INSS number (also named NISZ, NISS, and SSIN). Therefore, the health care software has the responsibility to provide a verified/validated INSS number.

3.2.3 PRESCRIPTION FORMAT

A prescription is defined as being a specific XML KMEHR message (Kind Message for Electronic Healthcare Record) of type pharmaceutical prescription. This format is further described on eHealth website: https://www.ehealth.fgov.be/standards/kmehr/.

On the one hand, the prescription software has the responsibility to generate a valid KMEHR prescription. On the other hand, executor software must be able to load such valid prescription.

The validation of the prescription consists in a two-step validation process:

- XSD validation
- Additional validation

The validation is automatically performed by Recip-e Integration Modules.

3.2.3.1 KMEHR XSD VALIDATION

The XML schema defining the KMEHR standard (XSD file) is provided in the Recip-e-client package and can be found on eHealth website at this URL: https://www.ehealth.fgov.be/standards/kmehr/.

At this moment, the prescription must be compliant with the version "20100601-kmehr" of the XSD definition (XSD packaged in a zip can be downloaded at this URL: https://www.ehealth.fgov.be/standards/kmehr/sites/default/files/assets/kmehr/20100601/20100601-kmehr.zip).

As from version 1.6.0 of the SDK, the prescription should be compliant to Kmehr version 1.17.1.

3.2.3.2 ADDITIONAL VALIDATION

The kmehr standard defines many different type of message regarding the healthcare sector. In the first stage of Recip-e (pilot), only pharmaceutical prescription is accepted. However, in the next phases, the system will accept other kind of prescription (new version of this document will be available).

For pharmaceutical prescription, additional verifications must be performed before uploading the prescription in Recip-e system.

The table below describes the different checks to be performed. For each check, a XML XPATH is defined. The result of the XPATH query must return the result defined in column "Expected Number of record"

Description	XPATH	Expected Number of record
Must contain 1 prescription	/kmehrmessage/folder/transaction/cd[@S='CD-TRANSACTION' and @SV='1.0' and text()='pharmaceuticalprescription']	1
Must contain between 1 and 10 items	/kmehrmessage/folder/transaction/heading/item/cd[@S='CD-ITEM' and @SV='1.0']	1 to 10 (included)
Valid Patient First name	/kmehrmessage/folder/patient/firstname	1
Valid Patient Family name	/kmehrmessage/folder/patient/familyname	1
Valid Patient ID	/kmehrmessage/folder/patient/id[@S='ID-PATIENT' and @SV='1.0']	1
Valid Prescriber First name	/kmehrmessage/header/sender/hcparty//firstname	1
Valid Prescriber Family name	/kmehrmessage/header/sender/hcparty//familyname	1
Valid Prescriber ID	/kmehrmessage/header/sender/hcparty/id[@S='ID-HCPARTY' and @SV='1.0']	1

Table 3: Kmehr validation checks

For more information about XPATH queries, please refer to http://www.w3.org/TR/xpath/.

3.2.3.3 SAMPLE PRESCRIPTION

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<kmehrmessage xmlns="http://www.ehealth.fgov.be/standards/kmehr/schema/v1">
      <header>
         <standard>
               <cd S="CD-STANDARD"SV="1.1">20100601</cd>
         </standard>
         <id S="ID-KMEHR"SV="1.0">14675011004.20090110090000000</id>
         <id S="LOCAL" SL="ID-RECIPE"SV="1.0">8e1c4ea4-3825-48e4-bcc2b8cadfa7a897</id>
          <date>2010-08-01</date>
          <time>09:00:00</time>
          <sender>
               <hcparty>
                     <id S="ID-HCPARTY" SV="1.0">14675011004</id>
                     <cdS="CD-HCPARTY" SV="1.0">persphysician</cd>
                     <name>Dr. Duck Donald</name>
               </hcparty>
          </sender>
          <recipient>
               <hcparty>
                     <id S="ID-HCPARTY" SV="1.0">RECIPE</id>
                     <cd S="CD-HCPARTY"SV="1.0">orgpublichealth</cd>
                     <name>Recip-e</name>
               </hcparty>
          </recipient>
      </header>
      <folder>
            <id S="ID-KMEHR" SV="1.0">1</id>
            <patient>
                  <id S="ID-PATIENT" SV="1.0">87990949113</id>
                  <firstname>Fred</firstname>
                  <familyname> Flintstone</familyname>
                  <birthdate>
                        <date>1933-10-23</date>
                  </birthdate>
                  <sex>
                        <cd S="CD-SEX" SV="1.0">male</cd>
                  </sex>
            </patient>
            <transaction>
                  <id S="ID-KMEHR" SV="1.0">1</id>
                  <cd S="CD-TRANSACTION"SV="1.1">pharmaceuticalprescription</cd>
                  <date>2010-08-01</date>
                  <time>09:00:00</time>
                  <author>
                        <hcparty>
                              <id S="ID-HCPARTY" SV="1.0">14675011004</id>
                              <cd S="CD-HCPARTY"SV="1.0">persphysician</cd>
                              <name>Dr. Duck Donald</name>
                        </hcparty>
                  </author>
                  <iscomplete>true</iscomplete>
                  <isvalidated>true</isvalidated>
                  <expirationdate>2013-08-01</expirationdate>
                  <heading>
                        <id S="ID-KMEHR" SV="1.0">1</id>
                        <cd S="CD-HEADING" SV="1.1">prescription</cd>
                        <item>
                              <id S="ID-KMEHR" SV="1.0">1</id>
                              <cd S="CD-ITEM" SV="1.1">medication</cd>
                              <content>
                                    <medicinalproduct>
                          <intendedcd S="CD-DRUG-CNK" SV="2.0">0318717</intendedcd>
                          <intendedname>Adalat Oros 30 (c) 30mg </intendedname>
                                    </medicinalproduct>
                              </content>
                              <lifecycle>
                                <cd S="CD-LIFECYCLE" SV="1.0">prescribed</cd>
                              </lifecycle>
                              <quantity>
                                    <decimal>1</decimal>
                              </quantity>
                              <frequency>
                                    <periodicity>
                                     <cd S="CD-PERIODICITY" SV="1.0">D</cd>
                                    </periodicity>
                              </frequency>
                              <dayperiod>
```

```
                                <cd S="CD-DAYPERIOD" SV="1.0">evening</cd>
                            </dayperiod>
                            <posology>
                                <text L="nl">1x</text>
                            </posology>
                        </item>
                        <item>
                            <id S="ID-KMEHR" SV="1.0">2</id>
                            <cd S="CD-ITEM" SV="1.1">medication</cd>
                            <content>
                             <medicinalproduct>
                          <intendedcd S="CD-DRUG-CNK" SV="2.0">1085885</intendedcd>
                                <intendedname>Actrapid HM Penfill (c) 100IU/ml
                                </intendedname>
                                    </medicinalproduct>
                            </content>
                            <lifecycle>
                                <cd S="CD-LIFECYCLE" SV="1.0">prescribed</cd>
                            </lifecycle>
                            <quantity>
                                    <decimal>1</decimal>
                            </quantity>
                            <posology>
                                <text L="nl">2x/d 12E voor de maaltijd SC</text>
                            </posology>
                        </item>
                        <item>
                            <id S="ID-KMEHR" SV="1.0">3</id>
                            <cd S="CD-ITEM" SV="1.1">medication</cd>
                            <content>
                                 <medicinalproduct>
                          <intendedcd S="CD-DRUG-CNK"S V="2.0">1077718</intendedcd>
                                <intendedname>Insulatard HM Penfill (c) 100IU/m
                                </intendedname>
                                    </medicinalproduct>
                            </content>
                            <lifecycle>
                                <cd S="CD-LIFECYCLE" SV="1.0">prescribed</cd>
                            </lifecycle>
                            <quantity>
                                    <decimal>1</decimal>
                            </quantity>
                            <posology>
                                    <text L="nl">1x/d 7E voor het slapen</text>
                            </posology>
                        </item>
                        <item>
                            <id S="ID-KMEHR" SV="1.0">4</id>
                            <cd S="CD-ITEM" SV="1.1">medication</cd>
                            <content>
                                <medicinalproduct>
                         <intendedcd S="CD-DRUG-CNK" SV="2.0">1057959</intendedcd>
                                    <intendedname>Spironolactone E.G. (c) 100mg
                                            </intendedname>
                                    </medicinalproduct>
                            </content>
                            <lifecycle>
                                 <cd S="CD-LIFECYCLE" SV="1.0">prescribed</cd>
                            </lifecycle>
                            <quantity>
                                    <decimal>1</decimal>
                            </quantity>
                            <frequency>
                                    <periodicity>
                                      <cd S="CD-PERIODICITY" SV="1.0">D</cd>
                                    </periodicity>
                            </frequency>
                            <dayperiod>
                                    <cd S="CD-DAYPERIOD" SV="1.0">morning</cd>
                            </dayperiod>
                            <posology>
                                    <text L="nl">1x/d</text>
                            </posology>
                        </item>
                    </heading>
                </transaction>
        </folder>
</kmehrmessage>
```

Example of a 'product' medication item:

```
<item>
    <id SV="1.0" S="ID-KMEHR">1</id>
    <cd SV="1.2" S="CD-ITEM">medication</cd>
    <content>
        <medicinalproduct>
            <intendedcd SV="2010-07" S="CD-DRUG-CNK">1484229</intendedcd>
            <intendedname>panadol tab 50x 1 g</intendedname>
        </medicinalproduct>
    </content>
    ...
</item>
```

Example of a 'substance' medication item:

```
<item>
    <id SV="1.0" S="ID-KMEHR">1</id>
    <cd SV="1.2" S="CD-ITEM">medication</cd>
    <content>
        <substanceproduct>
            <intendedcd SV="2010-07" S="CD-INNCLUSTER">8038747</intendedcd>
            <intendedname>paracetamol 1 g</intendedname>
        </substanceproduct>
    </content>
...
</item>
```

“Magistral prescription” : non-standardized (textual description)

```
<item>
    <id SV="1.0" S="ID-KMEHR">1</id>
    <cd SV="1.2" S="CD-ITEM">medication</cd>
    <content>
        <compoundprescription L="FR">Prescription magistrale</compoundprescription>
    </content>
...
</item>
```

3.2.3.4 CHECKING/SETTING THE AUTHOR OF THE PRESCRIPTION

Since the Recip-e system does not see the content of the prescription itself, it is up to the software of the executor to check that the name of the prescriber corresponds with the author as entered in the Kmehr message.

3.2.3.5 PRESCRIPTION TYPE

When the prescription software is creating a prescription, the attribute “prescription type” must be correctly defined. During the pilot phase, only pharmaceutical prescriptions are in scope. Currently we have defined following 4 types of pharmaceutical prescriptions

- P0 or PP: Pharmaceutical prescription
- P1 : Pharmaceutical prescription that necessitates information on the patient’s insurability
- P2 : Pharmaceutical prescription that necessitates attestation information (not used for now).

The PP pharmaceutical prescription is only a temporary one, and will be dropped in the future in favor of the P0, P1 and P2 types. The software providers are encouraged to already implement the distinction between P0, P1, and P2. This distinction can be made by looking at the BCFI database and the prescribed medication.

In general is expected from software provider to implement a solution open to any future updates of this prescription type.

Note: this prescription type is important because it is used by the Recip-e system to define access rights for executors. The executor will only be able to read (decrypt) messages of types to which he has access. (e.g. a pharmacist has access to pharmaceutical prescriptions)

3.2.4 FEEDBACK FORMAT

When requested by the prescriber, a Feedback may be sent back by the prescription executor (pharmacist) once prescription is delivered. The Feedback before encryption is a XML message that must be compliant with following description.

3.2.4.1 XSD VALIDATION

The File Feedback.xsd provided in the Recip-e client package describes the format of a notification.

Content of the file:

```
<xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema
        " xmlns="http://services.recipe.be"
        targetNamespace="http://services.recipe.be">
<xs:element name="feedback">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="text" type="xs:string" maxOccurs="1"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

3.2.4.2 SAMPLE FILE

```
<?xml version="1.0" encoding="UTF-8"?>
<p:feedback xmlns:p="http://services.recipe.be"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-
        instance" xsi:schemaLocation="http://services.recipe.be
        feedback.xsd">
 <text>this is a feedback</text>
</p:feedback>
```

3.2.5 BARCODE SPECIFICATION

On top of the prescription, a new barcode will be found that contains the Recip-e ID of the prescription. This barcode needs to be compatible with the current barcode readers found in pharmacies.

3.2.5.1 FORMAT RECIP-ID

The Recip-ID (or RID) has format BEPPXXXXXXXX where

- BE is a 2-character alphanumerical id that stands for the country (BE = Belgium)
- PP is a 2-character alphanumerical id that stands for the type of prescription (PP = 'general' pharmaceutical prescription). Refer to "Prescription Type" section for more information on available prescription types.
- XXXXXXXX an 8-character alphanumerical sequence ID

The alphanumerical characters can be numbers or uppercase letters:

- Possible characters are 0123456789ABCDEFGHKLMNPRSTVWXYZ
- Due to possible ambiguity letters O, Q, I, J, U and V are not used

The Recip-ID will be provided by eHealth during the creation of a prescription.

3.2.5.2 FORMAT BARCODE

The 128A format is used for the barcode. The barcode symbology is specified in ISO/IEC 15417:2007

Example:



3.2.6 ARCHIVING OF PRESCRIPTIONS

The executor has the responsibility to archive the prescription and its meta-data once delivered.

Recip-e recommends that the archive contain for each delivered Prescription:

- The prescription in format "SealedForUnknown" (this format is further described in the Encryption part of the document). It contains digital signature of the prescriber.
- The time stamp provided by Recip-e. It gives the proof that the prescription has existed at a specific point in time. The timestamp contains the Hash of the prescription and the creation date in an envelope signed by e-health.
- The Encryption Key for the prescription, provided by eHealth.

All required information is returned by the getprescription function:

- Prescription: variable 'sealedContent' in the return of the getPrescription function
- Time stamp: variable 'timestampingId' in the return of the getPrescription function
- e-Health encryption key: variable 'encryptionKey' in the return of the getPrescription function

3.2.7 PERFORMANCE METRICS UPLOAD

During the pilot, Recip-e needs to receive information about performances of services used in the solution (such as eHealth response time, create Prescription response time). These performance metrics do not contain information regarding care provider work (only response times of technical services). The care provider must be warned that such metrics are collected on his workstation. Recip-e expects that software providers warned their care provider before the activation of the collect.

- The collection of performance measures is included in the module. A file is generated under the execution directory folder. This file is uploaded on a regular basis on Recip-e server (Activation and frequency is defined in the log4j.xml configuration file, by default it is uploaded once a day in background).

3.3 INTEGRATION

3.3.1 OVERVIEW

3.3.1.1 EXECUTOR SOFTWARE

The figure below shows all the actions to be performed by the Software Provider's for executor (Executor/Pharmacist) and the way it is to be integrated with the Recip-e system and eHealth services.

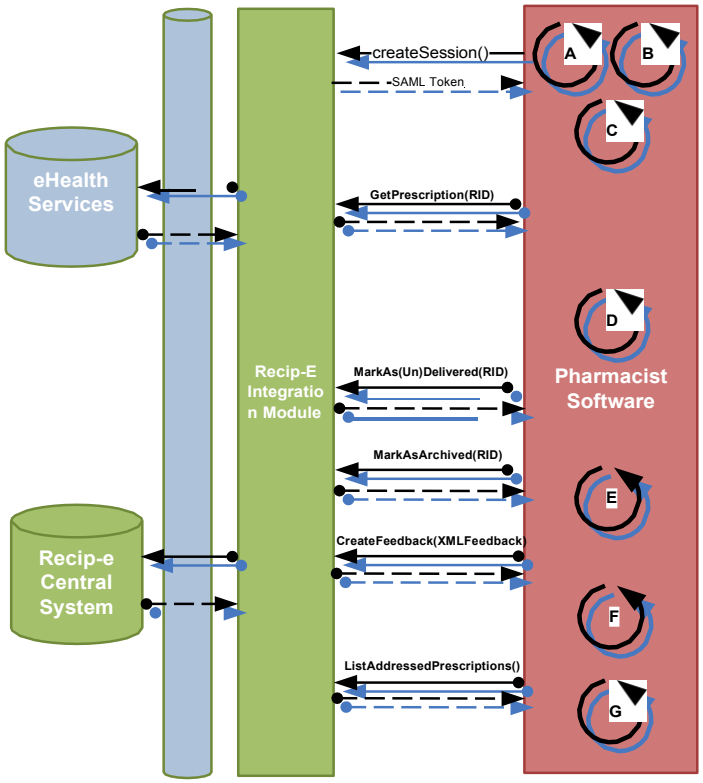


Figure 7: Integration Modules - Executor

Communication functionality

1. **Session Management :**
   a. **Create Session** : Send a session creation request to eHealth
   b. **Load Session**: Load a previously loaded session
   c. **Unload Session** : Unload the current session
2. **Get Prescription:** Get the prescription using the Recip-e ID found in barcode format on the prescription provided by the patient. For P1 prescriptions there is also the possibility to request the insurability of the patient.
3. **Mark Prescription As (un)delivered:** Software Provider software signals the Recip-e system that the prescription (or at least one item on the prescription) has been delivered. The Integration module returns a **Confirmation**
4. **Mark Prescription As archived:** Software Provider software signals the Recip-e system that the prescription has been archived. The Integration module returns a **Confirmation**
5. **Send feedback:** Send feedback on the prescription to the prescriber system (the feedback must contain the name of prescriber, patient name and prescription to which it relates). The Integration module returns a **Confirmation.**
6. **List Notifications:** Get all notifications from prescribers directed to this executor (The prescriber can send a text message to a specific executor. (e.g. could contain a recipe to be prepared)). The Integration module returns a Confirmation.
7. **Get Insurability:** retrieve the patients insurability information. The integration module returns a string representation of the xml.

Integration Specifications Executor

Internal functionality

A. Identify the patient through
   - the NISS/NISZ number found on the patients ID card
   - information on the patients SIS card
   - information on the patients mutuality sticker
   - directly into the system the executor (if the patient is already in the system)

B. Identify the prescription by reading the barcode containing the prescription number on the paper prescription

C. Check the insurability of the patient on MyCareNet

D. Translate the prescription returned from the Recip-e system into the internal format of the Software Provider software, save the prescription in the Software Provider software, and show the prescription to the pharmacist

E. Create feedback for the prescription - providing an additional screen that allows the introduction of feedback for a prescription. Multiple feedbacks on the same prescription are possible.

F. Get all notifications – The prescriber can send a text message to a specific executor. (e.g. could contain a prescription to be prepared). The pharmacist software needs to be able to show this.

G. Archiving of the prescription: once archived (out of scope of the specification), the executor software notifies recip-e system.

3.3.2 INTEGRATION DETAIL

3.3.2.1 APPROACH

The approach suggested to use Recip-e integration modules is:

1. Integrate the "mock" API in your software (no Recip-e backend needed).
2. Configure the modules to use Recip-e Acceptance web services (basic authentication) via the Belgacom VPN (ask Accenture for url/user & passwords)
3. Configure the modules to use eHealth platform (ESB, SAML Services and EID authentication).

3.3.2.1.1 MOCK

Each integration module has a "mock" version of the API. This mock version has the same API footprint than the real API. This mock version allows software vendors to update their application without needing a real Recip-e back-end.
For executors, the mock module is

"be.business.connector.recipe.executor.mock.ExecutorIntegrationModuleMock". Sample instantiation of the mock

module:

```
ExecutorIntegrationModule module = New ExecutorIntegrationModuleMock();
```

3.3.2.1.2 EHEALTH SERVICES

Once you have successfully integrated Recip-e API, you can replace the integrationModuleMock by the e-health connected module. This module are available :
For Executor : in the class ExecutorIntegrationModule

Both modules take in argument the path to the Recip-e configuration file.

3.3.2.2 JAVA

The JAR files contained in the folder "jar/." of the Executor zip package must be added to your class path.

Then, add following import in your code:

```
Import be.business.connector.common.CommonIntegrationModule
Import be.business.connector.recipe.executor.ExecutorIntegrationModuleImpl
```

Sample code for Executor:

```
// Module initialization
ExecutorIntegrationModule module = new ExecutorIntegrationModule();

// Login with the EID
String session = module.createSession();
System.out.println("User successfully logged-in with session : " + session );

// The RID is entered by the pharmacist
String rid = "BEPPa1b2c3d4";

// The prescription is retrieved from Recip-e
GetPrescriptionForPrescriberResult prescriptionResult = module.getPrescription(rid);

byte[] prescription = prescriptionResult.getPrescription();

System.out.println("The prescription " + rid + " has been successfully loaded : " +
        new String(prescription));
```

Refer to “Service Inventory” for the full list of all available functions and their associated input/output parameters.

For performance reasons, the initialization of the class “xxxxIntegrationModule() should be done once at application start-up.

3.3.2.3 DLL

The DLL files contained in the folder “dotNet/.” of the Executor zip package must be added to your project dependencies.

Then, the Recip-e client can be used with the code (VB.NET)

3.3.2.4 J2PCSC.DLL

The j2pcsc.dll needs to be placed in the working directory (the folder where you start the application .exe, .bat or …).

3.3.2.5 MYCARENET

As pharmacist there is the possibility to request information about the insurability of the patient. To request this information it will be necessary to add the following property in the configuration file.

```
pharmacy-holder.ssin=86041036109
```

This parameter represents the pharmacy holder, it is mandatory to create your session with eHealth and MyCarenet.

The executor module provides two ways to request insurability:

- When requesting a prescription (orchestration)
- Directly with MyCareNet thru eHealth ESB

3.3.2.5.1 ORCHESTRATION

When the getPrescription functionality is called, from the Executor module, there is the possibility to request more information about the insurability of a patient. . The insurability can only be requested for P1 prescriptions.

With the following parameters in the configuration file you can enable and disable the insurability retrieval.
The "pharmacy-holder.ssin=86041036109" is the NISS number of the pharmacy holder.

```
pharmacy-holder.ssin=86041036109
patient.insurability.disable=false
```

3.3.2.5.2 DIRECT CONNECTION

There is an API integrated into the Executor module to call the MyCareNet directly. The following steps need to be done to use the "direct connection":

- Request User, password and package name to MyCareNet support
- Create a e-Health STS session with the module (normal or fallback or mandate)
- Use the getInsurability() method of the Executor module to request the insurability.

More information on the input parameters can be found on MyCarenet sharepoint (http://share.intermut.be/home/MyCareNet/pharmacist/default.aspx).

Sample Insurability response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:GetInsurabilityForPharmacistResponse xmlns:ns2="urn:be:fgov:ehealth:insurability:core:v1"
xmlns:ns4="urn:be:fgov:ehealth:recipe:protocol:v2" xmlns:ns3="urn:be:fgov:ehealth:insurability:protocol:v1">
        <Status>
                <Code>200</Code>
                <Message>Success</Message>
        </Status>
        <ns3:CommonOutput>
                <ns2:Reference>12345123451234</ns2:Reference>
        </ns3:CommonOutput>
        <ns3:RecordCommonOutput>
                <ns2:Reference>278</ns2:Reference>
                <ns2:IoReference>0</ns2:IoReference>
                <ns2:UserReference>86041036109</ns2:UserReference>
        </ns3:RecordCommonOutput>
        <ns3:ReturnInfo>
                <ns2:ReturnCode>
                        <ns2:Major>01</ns2:Major>
                        <ns2:Minor>00</ns2:Minor>
                        <ns2:Detail>00000</ns2:Detail>
                </ns2:ReturnCode>
        </ns3:ReturnInfo>
        <ns3:InsurabilityResponse>
                <ns2:CareReceiver>
                        <ns2:Ssin>86041036109</ns2:Ssin>
                        <ns2:RegNrWithMut> </ns2:RegNrWithMut>
```

```
                        <ns2:Mutuality>111</ns2:Mutuality>
                        <ns2:FirstName>YANNICK ERIK</ns2:FirstName>
                        <ns2:LastName>LANDUYT</ns2:LastName>
                        <ns2:Birthday>1986-04-10</ns2:Birthday>
                        <ns2:Sex>male</ns2:Sex>
                </ns2:CareReceiver>
                <ns2:Coverage>
                        <ns2:Communicated>2011-05-23</ns2:Communicated>
                        <ns2:Period>
                                <ns2:BeginDate>2011-01-01</ns2:BeginDate>
                                <ns2:EndDate>2011-01-31</ns2:EndDate>
                        </ns2:Period>
                        <ns2:Entitlement>
                                <ns2:Code1>110</ns2:Code1>
                                <ns2:Code2>110</ns2:Code2>
                                <ns2:ThirdPartyPayerRegime>standard</ns2:ThirdPartyPayerRegime>
                        </ns2:Entitlement>
                </ns2:Coverage>
                <ns2:Verification>
                        <ns2:PaymentApproval>F321C0FE9A63A5EC144D5C790DF33290</ns2:PaymentApproval>
                        <ns2:PaymentApprovalSeed>1461301420</ns2:PaymentApprovalSeed>
                        <ns2:InvoicingOfficeCheckDigit>Sh</ns2:InvoicingOfficeCheckDigit>
                </ns2:Verification>
        </ns3:InsurabilityResponse>
</ns3:GetInsurabilityForPharmacistResponse>
```

3.3.3 SERVICE INVENTORY

This section lists all available API in the integration modules.

3.3.3.1 EXECUTOR INTEGRATION MODULE

Once authenticated, the common integration module keeps the SAML token in memory and the executor functionalities
can be called.

- **Interface** be.business.connector.recipe.executor.ExecutorIntegrationModule
- **Implementation**: be.business.connector.recipe.executor.ExecutorIntegrationModuleImpl

The module is constructed using the above implementation class, it requires the following parameters (*there are extra helper constructors available, see source code*):

Signature	***public ExecutorIntegrationModuleImpl(String propertyfile, RequestorProvider requestorIntegrationModule, CommonIntegrationModule commonIntegrationModule) throws IntegrationModuleException***
Parameters	***Propertyfile****: path to property file containing configuration for the connector module. This path can either be absolute (e.g. "C:\connector-client.properties") or relative to the class path of the application (e.g. "/connector-client.properties"). If using IKVM, some experimentation may be needed to find out this class path.* ***CommonIntegrationModule***: module that contains the information related to the SAML and has to be initialized before using the prescriber functionalities. ***RequestorProvider***: module that contains specific information related to the requestor.

3.3.3.1.1 GETPRESCRIPTION

Operation Name	***getPrescription (for Executor)***			
Class Name	***ExecutorIntegrationModule***			
Description	*This action returns the content of the prescription (KMEHR Standard) When the executor receives the prescription, Recip-e locks it. The executor MUST notify Recip-e of taken actions (MarkAsDelivered, MarkAsUnDelivered())* *Also there is the possibility to received the insurability of the patient.*			
When to be triggered	*When the barcode of the prescription has been read*			
Errors	*Technical Services Exception such as Connection failure, Encryption failure Inconsistent Status : when the prescription has been revoked , archived, is in Process*			
Inputs Integration Module	**Order**	**Type**	**Name**	**Description**
	1	*String*	*rid*	*RID of the prescription (size = 12 chars)*
Output Integration Module		**Type**	**Name**	**Description**
		GetPrescription ForExecutorResult	*prescription*	*A structure containing :* - *creationDate : Calendar* - *encryptionKeyId : String* - *encryptionKey : byte[]* - *feedbackAllowed : boolean* - *patientId : Long* - *prescriberId : long* - *prescription : byte[] (The XML clear content)* - *sealedContent : byte[] (The sealed content to be archived)* - *prescriptionType : String* - *rid : String* - *timestampingId : String*
Implementation Detail	**Step**	**Description**		
	1	*The module checks the SAML token.*		
	2	*A XML request is generated*		
	3	*The message is ciphered using the Encryption Token of the recipient (seal()), the personal Encryption Token of the sender is attached to the request.*		
	4	*Adding parameter to message to request insurability or not*		
	5	*Message is sent to Recip-e, Recip-e returns the result as a SOAP message*		
	6	*The message is decrypted. (Method "unseal()" of the ETEE framework is used).*		

7 *The module requests the Encryption Token (Signed Public Key) of the KGSS service to ETK Depot (eHealth service)*

8 *The module generates a "Symmetric Key request"*

9 *The message is ciphered using the Encryption Token of the recipient (seal()), the personal Encryption Token of the sender is attached to the request. (Recipient is KGSS)*

10 *The Sealed Response is received from Kgss, the module decrypt it. It contains the secured encryption key*

11 *The message is decrypted using unsealForUnknown()*

The executor has the responsibility to save the received sealed prescription and its meta-data once received.

This information is saved in the location defined through the connector properties file with the prescription RID as an xml file name. An example is as follow:

Old Prescription without TimestampedPrescription:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:archiveStandard xmlns:ns2="http:/services.recipe.be/executor">
  <creationDate>2016-09-30T08:42:59.162+02:00</creationDate>
  <executorId Type="PHARMACY" IdType="NIHII">62599147</executorId>
  <rid>BEPP11KYVYZE</rid>
  <timestampedPrescriptionVersion>v1</timestampedPrescriptionVersion>
  <prescriptionSealed> </prescriptionSealed>
  <timestampeId></timestampeId>
  <prescriptionType>PP</prescriptionType>
  <patientId Type="CITIZEN" IdType="SSIN">84071230581</patientId>
  <prescriberId Type="DOCTOR" IdType="NIHII">10050881001</prescriberId>
  <encryptionKeyId>UF3LPX8gHHv0fx3C8pWMwg==</encryptionKeyId>
  <encryptionKey>nNh9tSFSBTSgPR8SYJsJ1w==</encryptionKey>
  <unsealedPrescription> </unsealedPrescription>
  <prescriptionPrescriberId Type="persphysician" IdType="NIHII">14675011004</prescriptionPrescriberId>
  <properties>
    <property key="sdkVersion" value="jdk1.7.0_79"/>
    <property key="SupportedPrescriptionVersion" value="1.17"/>
    <property key="prescriptionVersion" value="1.17"/>
  </properties>
  <validationWarnings>
    <validationWarning>
      <ruleId IdType="RECIPE">1010</ruleId>
      <ruleMessage>
        <messageText language="NL">text!! [81112623980]&lt;&gt;[10050881001]</messageText>
      </ruleMessage>
      <severity>ERROR</severity>
    </validationWarning>
  </validationWarnings>
</ns2:archiveStandard>
```

An example with TimeStampedPrescription:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:archiveStandard xmlns:ns2="http:/services.recipe.be/executor">
  <creationDate>2016-09-29T09:50:41.889+02:00</creationDate>
  <executorId Type="PHARMACY" IdType="NIHII">62599147</executorId>
  <rid>BEP0SDDMT18K</rid>
  <timestampedPrescription></timestampedPrescription>
  <timestampedPrescriptionVersion>v2</timestampedPrescriptionVersion>
  <prescriptionType>P0</prescriptionType>
  <patientId Type="CITIZEN" IdType="SSIN">84071230581</patientId>
  <prescriberId Type="DOCTOR" IdType="NIHHI">10050881001</prescriberId>
```

```
  <encryptionKeyId>dtsOxsiwWORLpACST0Ix8w==</encryptionKeyId>
  <encryptionKey>C3WNwnv3RfuG1PZv6Y8PVg==</encryptionKey>
  <unsealedPrescription></unsealedPrescription>
  <prescriptionPrescriberId Type="" IdType="NIHII">81112623980</prescriptionPrescriberId>
  <properties>
    <property key="sdkVersion" value="jdk1.7.0_79"/>
    <property key="SupportedPrescriptionVersion" value="1.17"/>
    <property key="prescriptionVersion" value="1.17"/>
  </properties>
  <validationWarnings>
    <validationWarning>
      <ruleId IdType="RECIPE">1005</ruleId>
      <ruleMessage>
        <messageText language="NL">Prescriber mismach between kmehr and SAMLToken!!
[81112623980]&lt;&gt;[10050881001]</messageText>
      </ruleMessage>
      <severity>CRITICAL</severity>
    </validationWarning>
  </validationWarnings>
</ns2:archiveStandard>
```

3.3.3.1.2 MARKASUNDELIVERED

Operation Name	***markAsUnDelivered***			
Class Name	***ExecutorIntegrationModule***			
Description	*Update the status of the prescription in the central system. Set it back to 'Not delivered'. This action unlocks the prescription.*			
When to be triggered	*When no items have been delivered*			
Errors	*Technical Services Exception such as Connection failure, Encryption failure Inconsistent Status : when the prescription has been revoked , archived, is in Process*			
Inputs Integration Module	**Order**	**Type**	**Name**	**Description**
	1	*String*	*rid*	*RID of the prescription (size = 12 chars)*
Output Integration Module		**Type**	**Name**	**Description**
		#N/A	*#N/A*	*#N/A*
Implementation Detail	**Step**	**Description**		
	1	*The module checks the SAML token.*		
	2	*The module generates an XML request.*		

3	*The message is ciphered using the Encryption Token of the recipient (seal()), the personal Encryption Token of the sender is attached to the request. (The recipient is Recip-e)*
4	*The message is sent to Recip-e, nothing is returned if the action is processed successfully (otherwise, an exception is thrown)*

3.3.3.1.3 MARKASDELIVERED

Operation Name	***markAsDelivered***			
Class Name	***ExecutorIntegrationModule***			
Description	*Update the status of the prescription in the central system. Set it to 'delivered'. Once delivered, the prescription won't be delivered by another pharmacist*			
When to be triggered	*When at least one item of the overall prescription is delivered.*			
Errors	*Technical Services Exception such as Connection failure, Encryption failure Inconsistent Status : when the prescription has been revoked , archived, is in Process*			
Inputs Integration Module	**Order**	**Type**	**Name**	**Description**
	1	*String*	*rid*	*RID of the prescription (size = 12 chars)*
Output Integration Module		**Type**	**Name**	**Description**
		#N/A	*#N/A*	*#N/A*

Implementation Detail	**Step**	**Description**
	1	*The module checks the SAML token.*
	2	*The module generates an XML request.*
	3	*The message is ciphered using the Encryption Token of the recipient (seal()), the personal Encryption Token of the sender is attached to the request. (The recipient is Recip-e)*
	4	*The message is sent to Recip-e, nothing is returned if the action is processed successfully (otherwise, an exception is thrown)*

3.3.3.1.4 MARKASARCHIVED

Operation Name	***markAsArchived***			
Class Name	***ExecutorIntegrationModule***			
Description	*This action allows Recip-e to remove the prescription from the central storage. Encryption key is also removed from eHealth.*			
When to be triggered	*When the pharmacist has successfully archived the prescription with external back-up system. The prescription must be archived in his encrypted version including signature + encryption keys*			
Errors	*Technical Services Exception such as Connection failure, Encryption failure Inconsitant Status : when the prescription has not been delivered Invalid Prescription : when the prescription does not exist Access Rights : when the executor does not have rights for an update (ex : pharmacist that want to deliver a nurse prescription)*			
Inputs Integration Module	**Order**	**Type**	**Name**	**Description**
	1	*String*	*rid*	*RID of the prescription (size = 12 chars)*
Output Integration Module		**Type**	**Name**	**Description**
		#N/A	*#N/A*	*#N/A*
Implementation Detail	**Step**	**Description**		
	1	*The module checks the SAML token.*		
	2	*The module generates an XML request.*		
	3	*The message is ciphered using the Encryption Token of the recipient (seal()), the personal Encryption Token of the sender is attached to the request. (The recipient is Recip-e)*		
	4	*The message is sent to Recip-e, nothing is returned if the action is processed successfully (otherwise, an exception is thrown)*		

3.3.3.1.5 CREATEFEEDBACK

Operation Name	***createFeedback***			
Class Name	***ExecutorIntegrationModule***			
Description	*This action creates a feedback for the prescriber about a specific prescription.*			
When to be triggered	*When something special about the delivery should be notified to the Prescriber (ex : generic medicine delivery, dosage change)*			
Errors	*Technical Services Exception such as Connection failure, Encryption failure*			
Inputs Integration Module	**Order**	**Type**	**Name**	**Description**
	1	*Long*	*prescriberId*	*NIHII id of the recipient (the same one as the one received by GetPrescription.prescriberId).*
	2	*String*	*rid*	*RID of the prescription (size = 12 chars)*
	3	*byte[]*	*feedbackText*	*Xml message containing the feedback*
Output Integration Module		**Type**	**Name**	**Description**
		#N/A	*#N/A*	*#N/A*
Implementation Detail	**Step**	**Description**		
	1	*The module checks the SAML token.*		
	2	*The module requests the Encryption Token (Signed Public Key) of recipient to ETK Depot (eHealth service)*		
	3	*The module generates an XML request.*		
	4	*The message is ciphered using the Encryption Token of the recipient (seal()), the personal Encryption Token of the sender is attached to the request. (The recipient is the prescriber)*		
	5	*The message is enriched with meta data (recipient id…)*		
	6	*The message is ciphered using the Encryption Token of the recipient (seal()), the personal Encryption Token of the sender is attached to the request. (The recipient is Recip-e)*		
	7	*The message is sent to Recip-e, nothing is returned if the action is processed successfully (otherwise, an exception is thrown)*		

3.3.3.1.6 LISTNOTIFICATIONS

Operation Name	***listNotifications***			
Class Name	***ExecutorIntegrationModule***			
Description	*This function returns a list of notifications Prescriptions coming from Prescribers. When notifications are retrieved, they are automatically marked in the system as read, they stay in the system for a defined period of time (ex : 7 days).*			
When to be triggered	*There are different options:* *- can be invoked on a regular basis (2x per day) and display the messages as popup to the executor* *- can be invoked on demand by the prescriber (when he opens a specific screen)*			
Errors	*Technical Services Exception such as Connection failure, Encryption failure*			
Inputs Integration Module	**Order**	**Type**	**Name**	**Description**
Output Integration Module		**Type**	**Name**	**Description**
		List<ListNotificationsItem>	*List of notifications*	*ListNotificationsItem is structure containing :* - *content : byte[]* - *sentBy : Long* - *sentDate : Calendar*
Implementation Detail	**Step**	**Description**		
	1	*The module checks the SAML token.*		
	2	*The module generates an XML request.*		
	3	*The message is ciphered using the Encryption Token of the recipient (seal()), the personal Encryption Token of the sender is attached to the request. (The recipient is Recip-e)*		
	4	*Message is sent to Recip-e, Recip-e returns the result as a SOAP message*		
	5	*The message is decrypted. (Method "unseal()" of the ETEE framework is used).*		

3.3.3.1.7 GETINSURABILITY

Operation Name	***getInsurability***			
Class Name	***ExecutorIntegrationModule***			
Description	*This function retrieves the insurability of the patient at a specific date*			
When to be triggered	*After a prescription delivery.*			
Errors	*Refer to Mycarenet documentation*			
Inputs Integration Module	**Order**	**Type**	**Name**	**Description**
		String	*the user name*	
		String	*the password*	
		String	*the pharmacy holder*	
		String	*the pharmacy ssin*	
		String	*the pharmacy nihii*	
		String	*the date*	
		String	*the type*	
		String	*the care receiver ssin*	
		String	*the package name*	
		String	*the reference*	
		String	*the user reference*	
Output Integration Module		**Type**	**Name**	**Description**
		String	*Insurability*	*The XML Insurability (including e-health ESB service status)*
Implementation Detail	**Step**	**Description**		
	1	*The module retrieves the insurability from Mycarenet.*		

4 GFDDPP

4.1 REQUIREMENTS FOR INTEGRATION

The end-users of the GFD-DPP solution do not directly use the Integration Module, TIP System or Pharmaceutical Care Data Hub. Instead they use their familiar software packages. **The pharmacy software is expected to integrate with the GFD-DPP functionality by means of web services, the connector module in Java/.Net or by using the COM wrapper**.

4.1.1 OPERATION SUPPORT

Regardless of integration approach, the following business operations are needed for the GFD-DPP solution. A description of these operations when using the connector module can be found in chapter 5.3:

1. **Session Management – needed before calling any other operation**
   a. **Create (Fall-Back) Session** : Send a session creation request to eHealth (fall-back: using the personal certificate instead of the eID) OR
   b. **Load Session**: Load a previously created session
   c. **Unload Session** : Unload the current session
2. **Register Data:** registers Single Message in GFD-DPP.
   - In case of a medication delivery, a session GUID (for the delivery session) and dispensation GUIDs for every dispensed medication will be in the message (generated in connector module if using this approach).
   - In case of other events (e.g. archiving, tarification), only a session GUID is needed.
3. **Update Data:** update the data (based on dispensation id) that was previously registered in GFD-DPP.
4. **Revoke Data:** remove the data (based on dispensation id) that was previously registered in GFD-DPP.
5. **Get Status Messages:** get feedback on the register/update/revoke actions for a pharmacy. Most of the status messaging is asynchronous; the reasons are explained in the next chapter. *The software will need to reiteratively check the status message queue (Get Status Messages) and interpret the status messages.*
6. **Get Data Types:** retrieve the data types available for a patient in the referral registry within the PCDH.
7. **Get Data:** retrieve the patient data from the pharmaceutical care data hub identified through data type. When data type equals 'medication history', then the consolidated patients medication history is retrieved.
8. **Get Pharmacy Details:** retrieve the pharmacy id of the pharmacy that dispensed a medication (dispensation GUID). The integration module returns a string representation of the response xml. This response only contains the APB number of the pharmacy. This operation can only be called with explicit patient consent.

The pharmacy software will call these operations during normal business processing, as discussed in the next chapter, and the user interface needs to be updated/created to show retrieved information.

4.1.2 DATA SUPPORT

- The local pharmaceutical dossier will need to store both session and dispensation GUIDs. The software must also be able to retrieve these ID's.

The single message format must be supported in order to offer and parse information to/from the care services, and also to do a complete consolidation of the local dossier and the information retrieved. This format is discussed in its own chapter.

The software should be able to extract from its own local pharmaceutical dossier the medication history of a particular patient and present it in an XML structure as defined in the connector module application design [Ref 4]

- The following XML entities specified in the Single Message definition must be explicitly supported initially (creation and interpretation):
  - Delivered Medication Human Entity (i.e. dispensation data to be stored in PCDH as medication history)
  - Delivered Medication Human History Entity (i.e. dispensation data retrieved from PCDH when retrieving medication history).

4.2 CONNECTOR MODULE DESIGN

There's **no real persistency store in the connector module**. The data artifacts this module manages are:

- **Logs**
- **Configuration** (properties, XSD, WSDL, …)
- **Persisted XML messages** (*support for offline operation*, see subsequent sections)

The *data model* consists of an **object representation of the single message XSD,** filled with data for the current message being treated, and for which there is also an XML file persisted on the local file system. The same object model/file also contains meta-information on the status of processing.

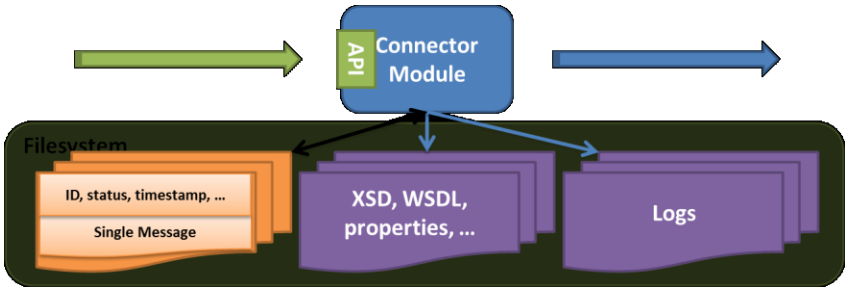


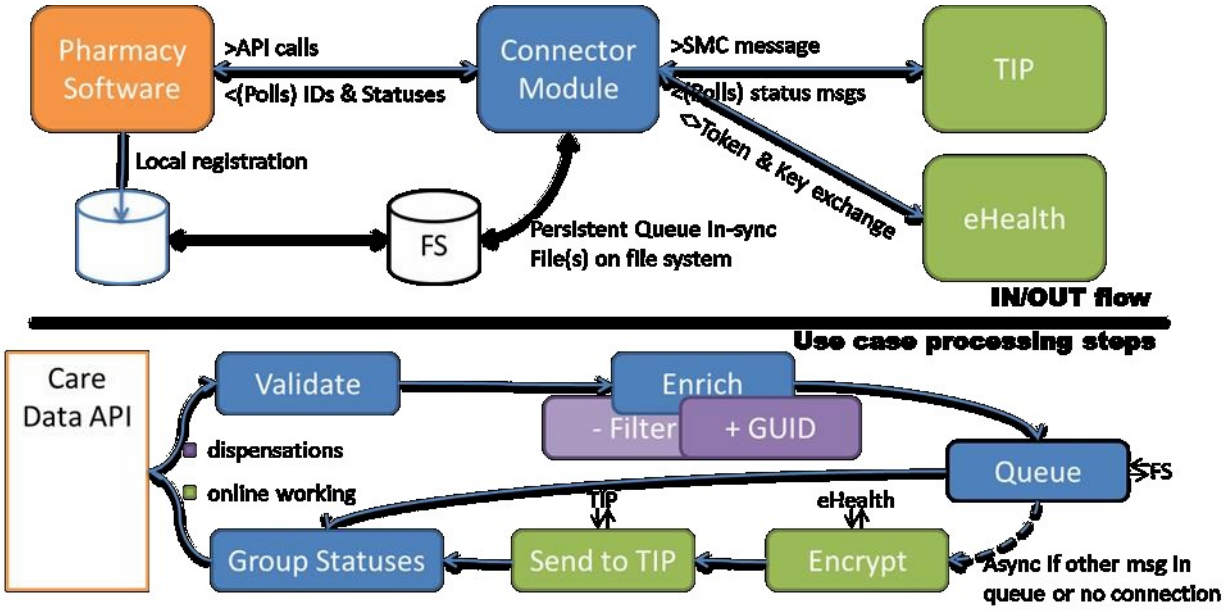
Figure 8: Connector module data artifacts

To register, a single message with session ID and multiple "delivered medication" entities goes through multiple processing states. The possible states are listed below and are inferred using status messages (see operations).

ID	Status	Final	Description
1	Queued	N	The message with dispensation is queued in the connector module.
2	TIP Processing	N	The message is being processed in the TIP.
3	TIP Error	Y	An error occurred during TIP processing.
3	TIP Processed	~	The message finished TIP processing (but not necessarily care services!).
4	Error/OK	Y	Return messages of care services stored in the module. For each care service there could either be an error or confirmation of processing.

Table 4: Connector module single message status

The connector module is responsible for assigning a dispensation globally unique ID to each new medication dispensation, and to the delivery session. It allows CRUD operations and governance to trace what happens with each medication dispensation.



4.2.1 APPROACH

The approach suggested to use GFD-DPP integration modules is:

- Test the "mock" API software (no GFD-DPP backend needed) to get a feel of in/output without being concerned with eHealth configuration.
- Configure the modules to use GFD-DP Acceptance web services (basic authentication) via the BELGACOM VPN (ask Accenture for URL/user & passwords)
- Configure the modules to use eHealth platform (ESB, SAML Services and EID authentication).

4.2.1.1 MOCK

Each integration module has a "mock" version of the API. This mock version has the same API footprint as the real API. This mock version allows software vendors to update their application without needing a real GFD-DPP back-end. The only functions performed by the mock:

- Validation of input (error messages when wrong)
- Return a canned response (always identical) of what a normal response through the online systems would look like. For test purposes, this response can be configured as well.

For the pharmacy software, the mock module is:

- be.connector.executor.pcdh.mock.PCDHIntegrationModuleMock
- be.connector.executor.tip.mock.TIPSystemIntegrationModuleMock

Sample instantiation of the mock module:

- `PCDHIntegrationModule mock = new PCDHIntegrationModuleMock()`

4.2.1.2 EHEALTH SERVICES

Once you have successfully integrated GFD-DPP services, you will need to update your configuration to use eHeath platform:

- **Authentication**: eHealth SAML services should be used to create a user session (SAML token). To enable SAML authentication: remove the "password" property of the configuration file. Update the property "wsdl.sts" with the eHealth STS URLs (line in comment to be uncommented), cf. application design connector module Ref[4].
- **ESB**: Send a request to GFD-DPP team to get the URL of GFD-DPP services via the eHealth ESB (an update of the integration module may be required), this request should include SSIN of your test users (those SSIN will be declared as Pharmacist in the STS configuration).

From version 1.1.0 of the Recipe connector, the eHealth services are implemented with the technical connector of eHealth. Integration specification is described in the eHealth documentation: https://www.ehealth.fgov.be/nl/support/connectors. More details about the Recip-e connector can be found in the Recip-e integration specifications.

4.2.2 JAVA

The JAR files contained in the folder "jar/." of the GFD-DPP-client.zip package must be added to your class path. The next chapter explains how to construct the modules and use them.

For performance reasons, initialization should be done once at application start-up.

4.2.3 DLL

The DLL files contained in the folder "dotNet/." of the GFD-DPP-client.zip package must be added to your project dependencies. Add an import statement in your code. Sample code in VB.NET:

```
Imports be.business.connector.recipe.executor
```

The GFD-DPP client can be used with following code (VB.NET):

```
Dim executorIntegrationModule = New ExecutorIntegrationModule()

' Create the session with EID, user enters his PIN code
executorIntegrationModule.createSession()

Dim patientId As String
Dim pharmacyId As String
Dim dataTypesResult
Dim dataTypes As String

patientId = "" 'INSZ
pharmacyId = "" 'NIHII

dataTypesResult = executorIntegrationModule.getDataTypes(patientId, pharmacyId)
dataTypes = dataTypesResult.getDataTypes()

Debug.Print("The returned data types are: "& dataTypes)
```

4.3 OPERATIONS AND BUSINESS FLOW

This chapter describes how the GFD-DPP solution extends the normal day-to-day activities of the pharmacist; when the call to register data must be made, with what information, etc.

- The data present in the local pharmaceutical record should be amended with the information present in the shared pharmaceutical record (get request), whenever the pharmacist requires this information.
- The aim**, as stated in the project overview, is to send all information gathered during a pharmaceutical care provisioning event with a single patient. Ideally data is registered or updated with the GFD-DPP at each event closure as to enable a high-quality one-shot registration for the pharmacist**. Not all specified information will (or can) be present in each message, but the more information is provided, the higher the quality can be guaranteed. If necessary, the pharmacist should be consulted to provide missing information

The figure below shows a schematic overview of the business flow:

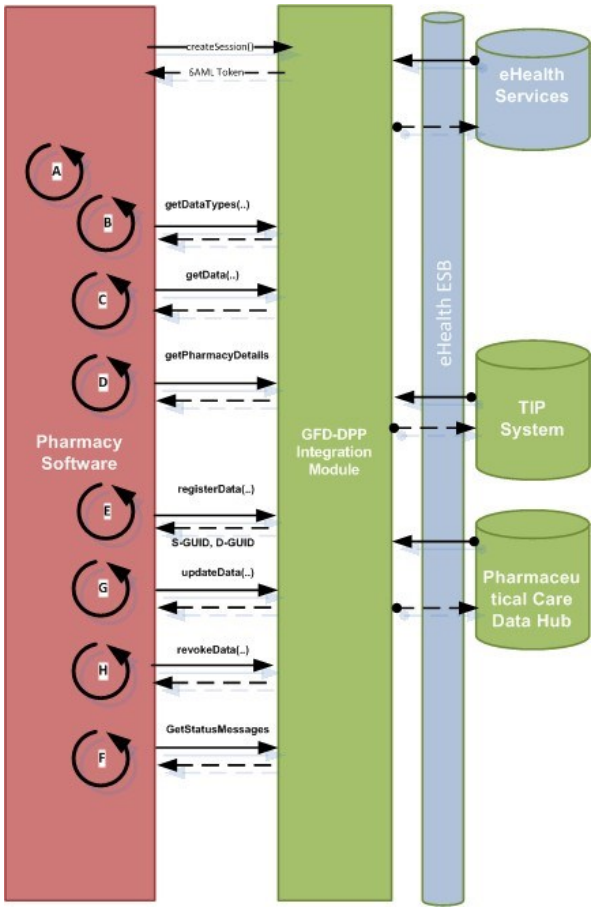


Figure 9: Pharmacy Software Integration

4.3.1 SESSION MANAGEMENT

Once the certificates are available, the connector module must be pointed to the key stores containing them using one of the configuration options (see chapter 6). A session can be started using the start session operation.

- Because of the decoupling of the technical connector and the business connector, session management is different from previous versions and is contained in its own component. This component is consulted by each business integration module, achieving single-sign-on for every service provided, not just recipe/mycarenet.

Once authenticated, the integration module keeps the SAML token in memory. This SAML token is a key element in authentication mechanism and therefore must be destroyed if the care provider does not need to use GFD-DPP services any more.

- **Interface** be.connector.common.CommonIntegrationModule
- **Implementation**: be.connector.common.CommonIntegrationModuleImpl

The module is constructed using the above implementation class, it requires the following parameters (*there are extra*

helper constructors available, see source code):

Signature	`CommonIntegrationModuleImpl(`
	`  String propertyfile`
	`) throws IntegrationModuleException`

Parameters	• `Propertyfile` (optional): path to property file containing configuration for the connector module. This path can either be absolute (e.g. "C:\connector-client.properties") or relative to the class path of the application (e.g. "/connector-client.properties"). If using IKVM, some experimentation may be needed to find out this class path. ○ **The bundled properties are used when not provided**. However, *use another constructor without this parameter instead of a null reference in this case*!

Table 5: PCDH Module Construction

Signature	`CommonIntegrationModuleImpl(`
	`  String propertyfile , String urlConfig`
	`) throws IntegrationModuleException`

Parameters	• `Propertyfile` (required): path to property file containing configuration for the connector module. This path can either be absolute (e.g. "C:\connector-client.properties") or relative to the class path of the application (e.g. "/connector-client.properties"). If using IKVM, some experimentation may be needed to find out this class path. • `urlConfig` (required): URL that replace the %CONF% in the property file that contains the configuration for the connector module. ○ **The property "PERSONAL_KEYSTORE" is required in the property file to use the fallback session.**

Table 6: PCDH Module Construction

Available operations are:

- `String createSession() throws IntegrationModuleException;`
- `void loadSession(String token) throws IntegrationModuleException;`
- `void unloadSession();`
- `String createFallbackSession(String niss, String passphrase) throws IntegrationModuleException;`
- `boolean hasValidSession();`
- `void assertValidSession() throws IntegrationModuleException;`
- `void setSystemKeystoreProperties(String systemKeystorePassword, String systemKeystorePath, String systemKeystoreDirectory, String systemKeystoreRizivKBO);`
- `void setOldSystemKeystoreProperties(String systemKeystorePassword, String systemKeystorePath, String systemKeystoreDirectory, String systemKeystoreRizivKBO);`
- `Element getSamlToken() throws IntegrationModuleException`
- `void setProperty() throws IntegrationModuleException`

It is expected from the software provider that this session is correctly destroyed (using API unloadSession) once the care provider shuts down his software/computer.

Software providers also have the possibility to secure even more this token by unloading it in case of care provider inactivity. Ex: if the pharmacist does not use his software for a defined period (ex: 30 minutes), the session is automatically unloaded. If the prescriber wishes to use a GFD-DPP service again, he will have to enter his PIN code.

4.3.1.1 ACTIONS PERFORMED ON SESSION AND MODULE STARTUP

Some online services (and related processing) are already called during session start-up, which should be implemented in the pharmacy software if the connector module is **not** used.

- The **eHealth STS service is contacted and a SAML token negotiated**. Reading from an eID card is required for this service.
- The common TIP is checked for an **updated system configuration** (versioned). If there is one available, this new configuration is downloaded, saved locally as a backup and parsed into memory for use
  - The configuration lists all known CBEs for each service type (TIP, PCDH, …) and the endpoint URL associated with that CBE.
  - The configuration is versioned to prevent unneeded load on the server and the client.
- Using the updated configuration, the common TIP is checked for an **updated product filter** (versioned). If there is one available, this new configuration is downloaded, saved locally as a backup and parsed into memory for use.
  - The filter lists a black and a white list for medications and preparations. The precedence of the white and black list can also change.
  - The configuration is versioned to prevent unneeded load on the server and the client.
- For the modules that need it, **a persistent queue and associated worker thread is started in support of offline operation**.
  - As a consequence, the modules that need a queue introduce a lifecycle and should be properly shut-down, to prevent data loss when closing the application while the worker thread is performing actions.
  - The location of the persistent queue is configured in the properties.

Important: software vendors that use the API don't need to use the system configuration or the product filter file that is updated after every creation of a session. These files are used and managed in the background by the API. When you make unnecessary changes to these files there is possibility the application will stop working correctly.

4.3.2 REFERENCE FLOW

The following is a possible implementation of the operations as provided in a business flow. The functionality provides registration, retrieval and visualization of medication history and pharmacy data. The purpose towards pharmacist is to have all functionality for a GET in one screen. A GUI proposal will be delivered by APB separately.

Except for the technical restrictions and requirements, the flow can be adapted to comply with existing business flows within the application.

A. Identify the patient through one of many available means
   a. the NISS/NISZ number found on the patients ID card (if eID inserted, then strong patient authentication can be enabled for data retrieval (see steps C, D))
   b. information on the patients SIS card
   c. information on the patients mutuality sticker
   d. directly into the pharmacist system (if the patient is already in the system)

B. Get the data types available for patient in Pharmaceutical Care Data Hub. Show the data types available on the screen of the pharmacist (or pick the data types as supported, or in context, automatically).

C. Let the pharmacist **select any information data type** he/she wants to retrieve in the screen from the retrieved data types (connector module operation) (or select by default depending on screen type)
   - Allow the pharmacist the option to retrieve the medication history for the patient by default.
   - Allow the pharmacist to exclude his/her own registrations
   - Allow the pharmacist to request the default history or the full history (strong patient authentication required)
   - Allow the pharmacist to enter additional information (patient consent, therapeutical relationship) (or prefilled)

   Show the retrieved information to the pharmacist in the screen. This information should be consolidated with the local pharmaceutical care information. *The connector module supports consolidation with local data for the get data operation*.

D. In case of additional information required on a specific dispensation, when retrieved using get data operation, allow the pharmacist to retrieve additional information on a dispensation – **get pharmacy details operation - meaning the pharmacy id of the pharmacy who dispensed the medication**.

- Strong Patient authentication is required for this function
- A motivation must be entered or selected by pharmacist for this function.

This ID can be used to lookup contact information of the pharmacist that registered the dispensation. The GFDDPP does not have an authentic source on pharmacy information so an external system should be contacted with the ID.

E. When pharmaceutical care (e.g. delivery of a medication) is provided by the pharmacist to a patient, allow him/her to register this as part of his/her normal process of medication or pharmaceutical care registrations or through an additional screen where details on the care provision can be entered. Details that are available are prefilled, e.g. by product registration. The preference goes to make the registration into the GFD-DPP part of the normal (as now) registration process of the pharmacist (and add the GFD-DPP registration in the background), and thus to limit the manual entry in the pharmacy software by the pharmacist during a registration.
When registration process is locally performed or/and all required information is calculated or entered, store the registration in the local pharmaceutical record (database) and submit registration to connector module.

The connector module with return the generated session-GUID and dispensation-GUID(s).

F. **Frequently, e.g. every 5 minutes, request a status update on all registrations performed** (if not all registrations are finished) in the GFD-DPP for the pharmacy. These can be registrations, updates or revokes.

Store the status (associated to a session id and dispensation id) in the local pharmaceutical record. *In case of multiple terminals at one pharmacy, we assume a central database is used, if not a synchronization of data should occur between the different terminals.*

Allow the pharmacist to go to an overview screen to see his/her dispensations performed and associated statuses.
A feedback message of message type 'Routing Info' will be provided by connector module to pharmacy software for each service that is called with a registration (session and dispensation GUID) in TIP. Based on this feedback information, the pharmacy software can determine if all services are completed for a dispensation.

G. When the pharmacist is reviewing the previously made medication registrations, **allow the pharmacist to update a previous registered dispensed medication**

- A motivation should be entered or selected by pharmacist for this function.

H. When the pharmacist is reviewing the previous made medication registrations, **allow the pharmacist to revoke a previous registered dispensed medication**

- A motivation should be entered or selected by pharmacist for this function.

-

4.3.3 INTERFACES & OPERATIONS

In the GFD-DPP project there are two business interfaces, one for the PCDH servicing requests (Get …) and one for the TIP servicing commands (Register / Update / Revoke Data).

The original Java interfaces are used to discuss the operations; evidently the semantics are identical for the other technologies in which the connector module is available.

Remember, these operations can only be executed when there is a valid eHealth session.

4.3.3.1 PCDH INTEGRATION MODULE

- **Interface**: be.connector.executor.pcdh.PCDHIntegrationModule
- **Implementation**: be.connector.executor.pcdh.PCDHIntegrationModuleImpl

4.3.3.1.1 CONSTRUCTION

The module is constructed using the above implementation class, it requires the following parameters (*there are extra helper constructors available, see source code*):

Signature	`PCDHIntegrationModuleImpl(RequestorProvider requestorProvider, CommonIntegrationModule commonIntegrationModule) throws IntegrationModuleException`
Parameters	• `RequestorProvider` (required): Object implementing the RequestorProvidor interface, used to fill this information in each request. ○ TIP: the CommonIntegrationModuleImpl object implements this interface and is the right object (default) to give in most circumstances. It extracts the information out of the session. Take care to always provide the same instance as both parameters in the constructor in this case! • `CommonIntegrationModule` (required): Object managing the session with eHealth, necessary to create valid requests. Creation is discussed in the session management section.

Table 7: PCDH Module Construction

4.3.3.1.2 GET DATA TYPES

Signature	`String getDataTypes(String patientIdType, String patientId, boolean requestPatientSignature, Boolean therapeuticRelationShip) throws IntegrationModuleException`
Parameters	• `PatientIdType`: The type of ID the patient has, initially only “SSN” is supported. Later iterations may support foreign IDs. • `PatientId`: The actual ID in the format as indicated by the type. • `RequestPatientSignature` (required): true or false. Indicate whether the connector module will try to sign the request with the eID that’s currently inserted in the eID reader. It’s up to the pharmacy software to check (or warn the pharmacist) that this is the correct eID. Whether a signature is required or not for a request will change over time. • `TherapeuticRelationShip` (optional): Indicate true/false whether a therapeutical relationship, as legally established, for this patient is present. Leave null or equivalent when that information is not available.
Output	A string in the single message XML format containing a list of data types (see next chapter for a specification and example).

Table 8: Get Data Types

4.3.3.1.3 GET DATA

Signature	`String getData(`
	`String patientIdType,`
	`String patientId,`
	`String dataType, String`
	`dateRange, boolean`
	`excludeOwnData, byte[]`
	`localData,`
	`boolean requestPatientSignature,`
	`Boolean therapeuticRelationShip`
	`) throws IntegrationModuleException`

Parameters	• `PatientIdType` (required): *see Get Data Types*.
	• `PatientId` (required): *see Get Data Types*.
	• `DataType` (required): The pharmacist or software needs to select one of the data types that was returned by the "Get Data Types" call
	• `DateRange` (required): The keyword "DEFAULT" or "FULL", or a date range in the format "YYYYMMDD-YYYYMMDD" – e.g. "20120501-20120601".
	  ○ The keywords are case sensitive.
	  ○ The date range expects numerical values and the first date *must* be before the second.
	• `ExcludeOwnData` (required): whether to return data in this call that was originally registered by this pharmacy. *Indicate true if no consolidation is required*.
	• `LocalData` (optional): If given, is only used when *excludeOwnData* is false. Local database record (available for patient) transformed in proper SMC format, like in REGISTER DATA (delivered medication human).
	• `RequestPatientSignature` (required): *see Get Data Types.*
	• `TherapeuticRelationShip` (optional): *see Get Data Types.*

Output	A string in the single message XML format containing a list of data entities of the same type as given (see next chapter for a specification and example). If consolidating, this single message will include the given local data and the data is ordered chronologically.

Table 9: Get Data

4.3.3.1.4 GET PHARMACY DETAILS

Signature	`String getPharmacyDetails(`
	`String patientIdType,`
	`String patientId,`
	`String dGuid,`
	`String motivationType,`
	`String motivationText,`
	`boolean requestPatientSignature,`
	`Boolean therapeuticRelationShip`
	`) throws IntegrationModuleException`

Parameters	• `PatientIdType` (required): *see Get Data Types*. • `PatientId` (required): *see Get Data Types*. • `DGUID` (required): The ID of the dispensation (not session) for which the pharmacy ID is desired. • `motivationType` (required): A string indicating why the pharmacy ID is desired. This string can be used to implement a tagging principle locally. Motivations are registered in the PCDH. • `motivationText` (optional): If the pharmacist provides additional details to the request. • `RequestPatientSignature` (required): *see Get Data Types.* • `TherapeuticRelationShip` (optional): *see Get Data Types.*
Output	A string containing a pharmacy NIHII (*not* in XML format)

Table 10: Get Pharmacy Details

4.3.3.2 TIP SYSTEM INTEGRATION MODULE

- **Interface**: be.connector.executor.tipsystem.TIPSystemIntegrationModule
- **Implementation**: be.connector.executor.tipsystem.TIPSystemIntegrationModuleImpl

4.3.3.2.1 CONSTRUCTION

The module is constructed using the above implementation class, it requires the following parameters (*there are extra helper constructors available, see source code*):

Signature	`TIPSystemIntegrationModuleImpl(RequestorProvider requestorProvider, CommonIntegrationModule commonIntegrationModule) throws IntegrationModuleException`
Parameters	• `RequestorProvider` (required): *see PCDH Integration Module construction.* • `CommonIntegrationModule` (required): *see PCDH Integration Module construction*.

Table 11: PCDH Module Construction

4.3.3.2.2 SHUTDOWN GRACEFULLY

Remark that the TIP System Integration Module has a lifecycle method that should be called on shutdown. This method stops the worker thread responsible for emptying the queue and uploading to the TIP system.

- The same effect will be achieved by forcefully quitting the VM in which the thread runs, however this is not recommended.
- If this method is not called, the VM may in some cases continue to run while the original calling process has already terminated.

```
void shutdownGracefully()
```

4.3.3.2.3 REGISTER DATA

Signature	`String registerData(byte[] singleMessage, String tipId, boolean requestPatientSignature, Boolean patientConsent) throws IntegrationModuleException`

Parameters	• `SingleMessage` (required): The single message containing as much pharmaceutical care information from the delivery session as possible. Take note that:
	  ○ SGUID and DGUID are generated in the connector and thus not required in the message.
	  ○ Meta data specified in the single message format to indicate quality information.
	  ○ An example message can be found in the next chapter.
	• `TipId` (optional): The ID of the TIP (CBE number) to use instead of the default as configured in the connector module properties. *This is effectively an override mechanism*.
	• `RequestPatientSignature` (required): true or false. Indicate whether the connector module will try to sign the request with the eID that's currently inserted in the eID reader. It's up to the pharmacy software to check (or warn the pharmacist) that this is the correct eID. Whether a signature is required or not for a request will change over time.
	• `PatientConsent` (optional): Indicate whether there is patient consent, as legally established, to store any sent information in the PCDH. Leave null or equivalent when that information is not available. *When the GFD-DPP cannot determine consent either, defaults to false*.

Output

A string in a custom XML format containing the generated GUIDs:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://services.gfddpp.be"
targetNamespace="http://services.gfddpp.be">
<xs:element name="events">
     <xs:complexType>
          <xs:sequence>
            <xs:element name="pharmaceuticalCareEventType"
type="pharmaceuticalCareEventType" maxOccurs="1" />
          </xs:sequence>
          </xs:complexType>
</xs:element>
<xs:complexType name="pharmaceuticalCareEventType">
    <xs:sequence>
          <xs:element name="sguid" type="xs:string" maxOccurs="1"/>
          <xs:element name="dispensation" type="dispensation"
maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="dispensation">
    <xs:sequence>
     <xs:element name="localReference" type="xs:string" maxOccurs="1" />
     <xs:element name="dguid" type="xs:string" maxOccurs="1" />
        <xs:choice>
          <xs:element name="product" type="product" maxOccurs="1" />
          <xs:element name="magistral" type="magistral" maxOccurs="1" />
         </xs:choice>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="product">
    <xs:sequence>
     <xs:element name="productId" type="xs:string" maxOccurs="1" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="magistral">
    <xs:sequence>
     <xs:element name="magistralIds" type="xs:string" maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
</xs:schema>
```

An example return is:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:events xmlns:ns2="http://services.gfddpp.be">
     <pharmaceuticalCareEventType>
          <sguid>d237a926-e504-550f-d0cb-89e13c5c408f</sguid>
          <dispensation>
               <localReference>1234567</localReference>
               <dguid>c40a5497-440f-735e-b62e-6a8ee370baac</dguid>
               <product>
                    <productId>0592261</productId>
               </product>
          </dispensation>
          <dispensation>
               <localReference>1234567</localReference>
               <dguid>88c3adc3-14ef-5b93-2aec-df4baeb427bf</dguid>
               <product>
                    <productId>0592261</productId>
               </product>
          </dispensation>
     </pharmaceuticalCareEventType>
</ns2:events>
```

Remark that not all medication deliveries will have a DGUID as some will have been filtered out. When two medication deliveries have the same CNK, the order in the response is the same as in the original single message.

Output with Magisterial	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <ns2:events xmlns:ns2="http://services.gfddpp.be"> <pharmaceuticalCareEventType> <sguid>12345678-1234-1234-1234-0123456789AB</sguid> <dispensation> <localReference>1000709951</localReference> <dguid>12345678-1234-1234-1234-0123456000AB</dguid> <magistral> <magistralIds>0538710</magistralIds> <magistralIds>0509869</magistralIds> </magistral> </dispensation> </pharmaceuticalCareEventType> </ns2:events>

Table 12: Register Data

4.3.3.2.4 UPDATE DATA

Signature	`String updateData(`
	`byte[] singleMessage,`
	`String tipId,`
	`boolean requestPatientSignature,`
	`Boolean patientConsent`
	`) throws IntegrationModuleException`

Parameters	• `SingleMessage` (required): The single message containing updated pharmaceutical care information. Take note that: ○ SGUID and DGUID must be included and identical to the ones used for registration. ○ A motivation entity is expected in the message, this motivation entity contains the same fields as described for the Get Pharmacy Details request. ○ An example message can be found in the next chapter. • `TipId` (optional): *see Register Data*. • `RequestPatientSignature` (required): *see Register Data*. • `PatientConsent` (optional): *see Register Data*.
Output	*See Register Data*.

Table 13: Register Data

4.3.3.2.5 REVOKE DATA

Signature	`void revokeData(`
	`byte[] singleMessage,`
	`String tipId,`
	`boolean requestPatientSignature,`
	`Boolean patientConsent`
	`) throws IntegrationModuleException`

Parameters	• `SingleMessage` (required): The single message containing only the entity IDs of the care data that needs to be revoked – the actual care data need not be included. Take note that: ○ Session GUID is generated in the connector module and not required in the message. ○ A motivation entity is expected in the message, this motivation entity contains the same fields as described for the Get Pharmacy Details request. ○ An example can be found in the next chapter. • `TipId` (optional): *see Register Data*. • `RequestPatientSignature` (required): *see Register Data*. • `PatientConsent` (optional): *see Register Data*.
Output	Nothing – this operation only throws an exception when something goes wrong. To follow up on the revocation operation, consult the status messages.

Table 14: Revoke Data

4.3.3.2.6 GET STATUS MESSAGES

The get status message can be used in 3 different ways:

- When the S-GUID & D-GUID are not filled in, the 250 oldest messages for the pharmacy will be retrieved
- When the S-GUID is filled in, all the messages for the pharmacy with that S-GUID will be retrieved
- When the S-GUID & DGUID are filled in, all the messages for the pharmacy with that S-GUID & D-GUID will be retrieved.

Signature	`String getStatusMessages(`
	`String sGuid,`
	`String dGuid`
	`) throws IntegrationModuleException`

Parameters	• SGUID (optional): Limits the returned status messages for the requestor to those concerning the given SGUID. • DGUID (optional): May only be effective if the SGUID is effective. Further limits the returned status messages to those concerning the given DGUID. If both parameters are null or equivalent, all status messages for the requestor are returned.
Output	A string in the single message XML format containing a list of status message entities (see next chapter for a specification and example).

Table 15: Get Data Types

4.3.4 ERROR CODES

As evidenced in above descriptions, each method can throw an exception that includes an error code and a message in three different languages (nl, fr, en).

The error code list is maintained in its own deliverable as part of the SDK. The first chapter includes a reference to this list.

4.4 SINGLE MESSAGE FORMAT

The single message format is an XML specification used to communicate pharmaceutical information The message structure contains a *hierarchy of events and entities in its body*.

- **Events correspond to pharmaceutical care actions** performed by the pharmacist.
- **Entities are the data objects** manipulated by or associated with events.
  - *These are either part of, or referenced by, events*. The latter is preferred if multiple events reference the same entity.

The following simplified diagram contains *fictional* elements to illustrate the difference between events and entities. In this case, a revocation is an event that references a dispensation delivery.

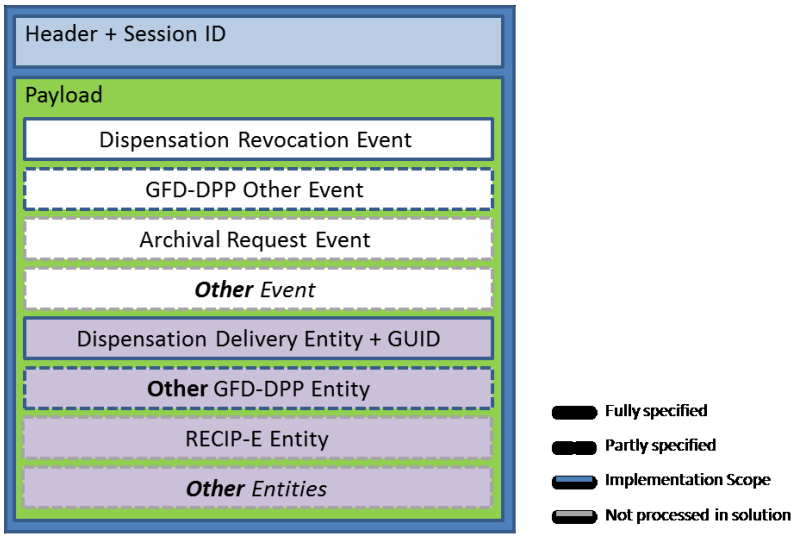


Figure 10: Single Message hierarchy overview

More specifically, the diagram below shows the structure as it exists within the actual schema.

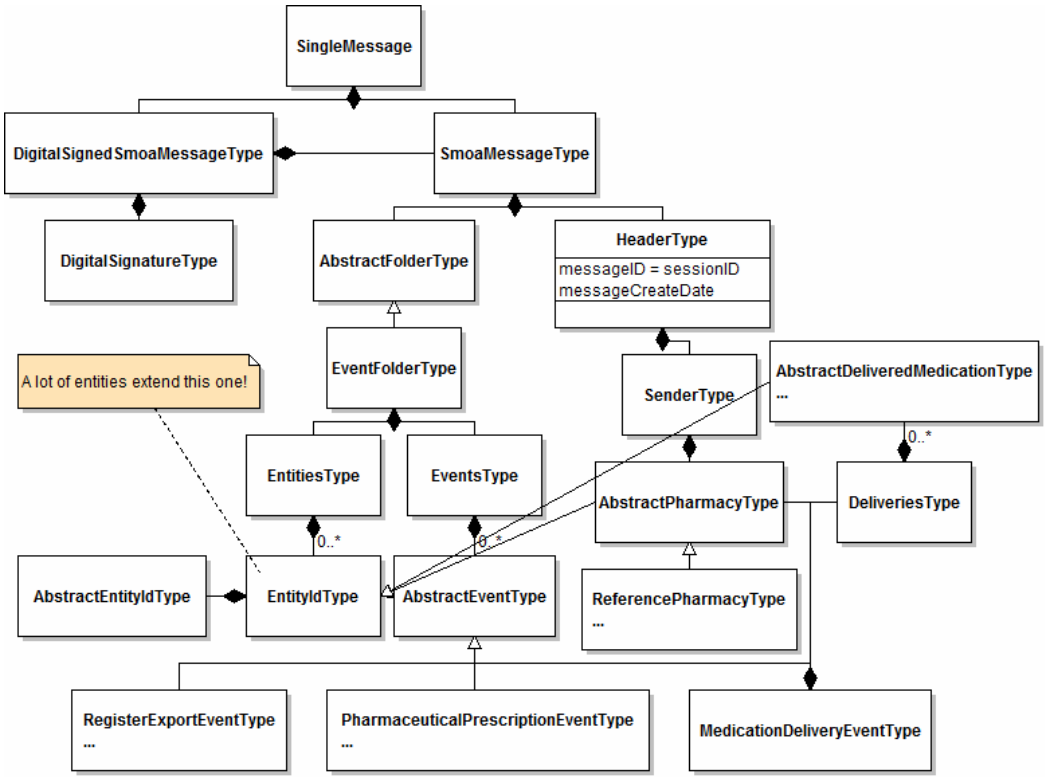


Figure 11: Single Message Specification

- Black diamond mean "is composed of" (contains)
- White arrow means "extends"
- If no cardinality is written, it means 0..1
- Ellipses (…) in an entity mean the entity has more attributes or extensions not shown in the diagram.

4.4.1 DELIVERED MEDICATION ENTITY

A subtype of *Abstract Entity* and contained in the *Event Folder*, this element stores medication delivery content. It is the main type of information processed and stored in the first release of the GFD-DPP project.

The representation *also* makes use of some of the KMEHR specification for medicines and related information.

- KMEHR stands for Kind Messages for Electronic Health care Record. It was introduced in 2002 and is an implementation of the fourth recommendation of the Belgian Healthcare Telematics Commission, enabling the exchange of structured clinical information.

A dispensation has a maximal and minimal dataset. The maximal dataset is what is sent from the pharmacy, the minimal dataset is what is actually stored in the Pharmaceutical Care Data Hub and will be returned in a "get".

4.4.1.1 DELIVERED MEDICATION HUMAN EVENT

The diagram on the right shows the representation of human delivered medication entity in the Single Message XSD. **It is expected in the SMC on the API of the connector module.**

- **D-GUID** is the dispensation ID either assigned by the connector module or given by the pharmacy software for an update or revoke.
- **Meta-data is for the pharmacy software to fill with the goal of verifying quality of the transmitted information.**
  - Only between the connector module and TIP, it's used during quality control for quality indicators and not to be forwarded to any back-end services.
  - **It uses a key-value pair approach**. Possible keys and values are not specified in the interface but in the systems.
    ⇨ Consequently, new pairs can be introduced without changing the interface.
  - **Example** of meta-data: *boolean – has Delphi care run*?

Integration Specifications Executor

⇨ Other meta-data will be introduced in design if needed.

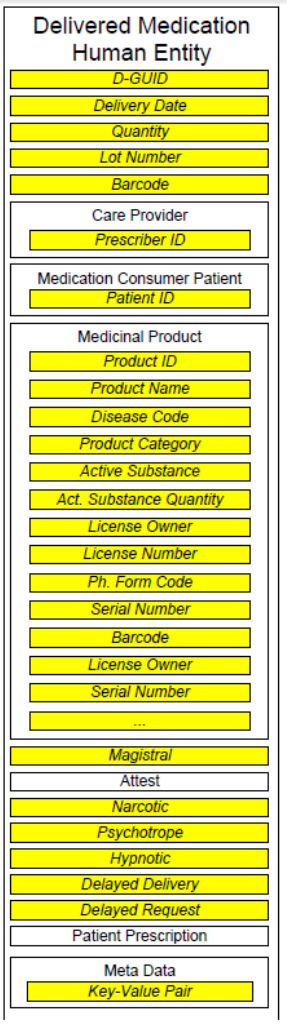


Figure 12: Delivered Medication Human Entity.

4.4.1.2 DELIVERED MEDICATION HUMAN HISTORY EVENT

This **entity** is generated at the TIP system from the maximal dataset. This entity is sent as a response from the pharmaceutical care data hub to a "get data"-request for the medication history.

Refer to Error! Reference source not found. to get a sense of the entity structure. It shows **some information is ymmetrically encrypted and will not be visible to the Pharmaceutical Care Data Hub itself. This information is decrypted in the connector module**. The separation encrypted/clear is determined by the needs of the Pharmaceutical Care Data Hub:

- The **patient ID** is the basis for medication history lookup by the pharmacist.
- The **dispensation ID** is the basis to allow update and revocation functionality, as well as pharmacy details lookup.
- The **delivery date** allows the PCDH to group data from various time periods (full, default, user defined).
- The **encryption key ID** is needed to ask eHealth for the symmetrical key by which part of the message is encrypted. *The* Pharmaceutical Care Data Hub *is not allowed to consult those decryption keys as specified in the Access Control List*.
- **Meta-data is included using a key-value pair approach, similar to meta-data between the connector module and the TIP system**. The difference being here that this meta-data is put by the TIP system and is used to describe the kind of dispensation this is. It could be used by the fine-grained authorization engine on the Pharmaceutical Care Data Hub to decide if a dispensation needs to be returned or not.

Some important information not present in the event will be available in the wrapper single message. The quality indicator structure of this History event is composed out of following:

- **TIP Quality Indicator**: The quality indicator is a score from 1 to 10 indicating the quality of the dispensation.
- **Quality Reasons:**
  - Type: a pre-defined reason why the quality indicator was assigned
  - Comment: a free text field where more details on the quality reason are entered

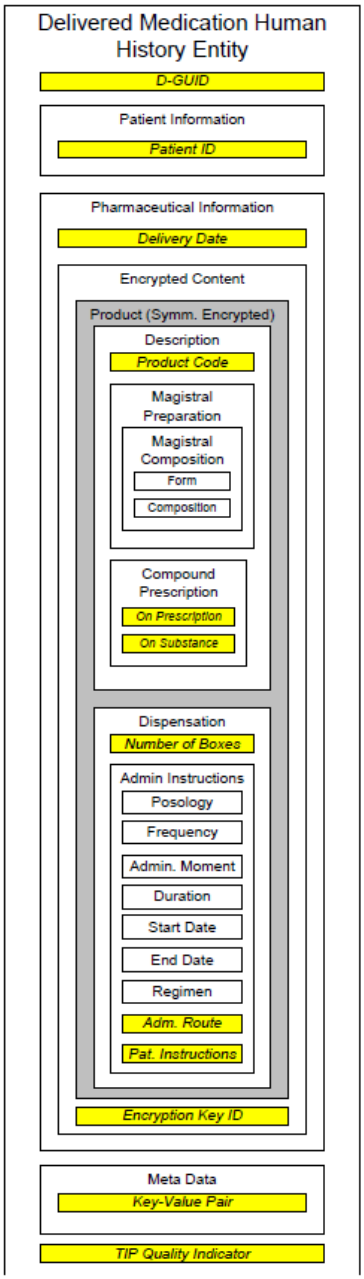


Figure 13: Delivered Medication Human History Entity.

4.4.2 MEDICATION DELIVERY EVENT

There is no notion of register, update or revoke in the event system; there only exists a medication delivery event.

This information is provided through the use of operations defined on the interfaces instead – refer to the use cases. The event contains the following :

- A **Pharmacy Entity** that identifies the *creator of the medication delivery event*, in this case always the dispenser.
- Multiple **Delivered Medication Entities** describe the actual dispensations and is shown in the previous section.

In addition, a motivation can be added to the event:

- **Both a revoke and update require a motivation of the pharmacist**. This is not a legal obligation but necessary to learn about any problems pharmacists might have and incorporate improvements in later iterations.

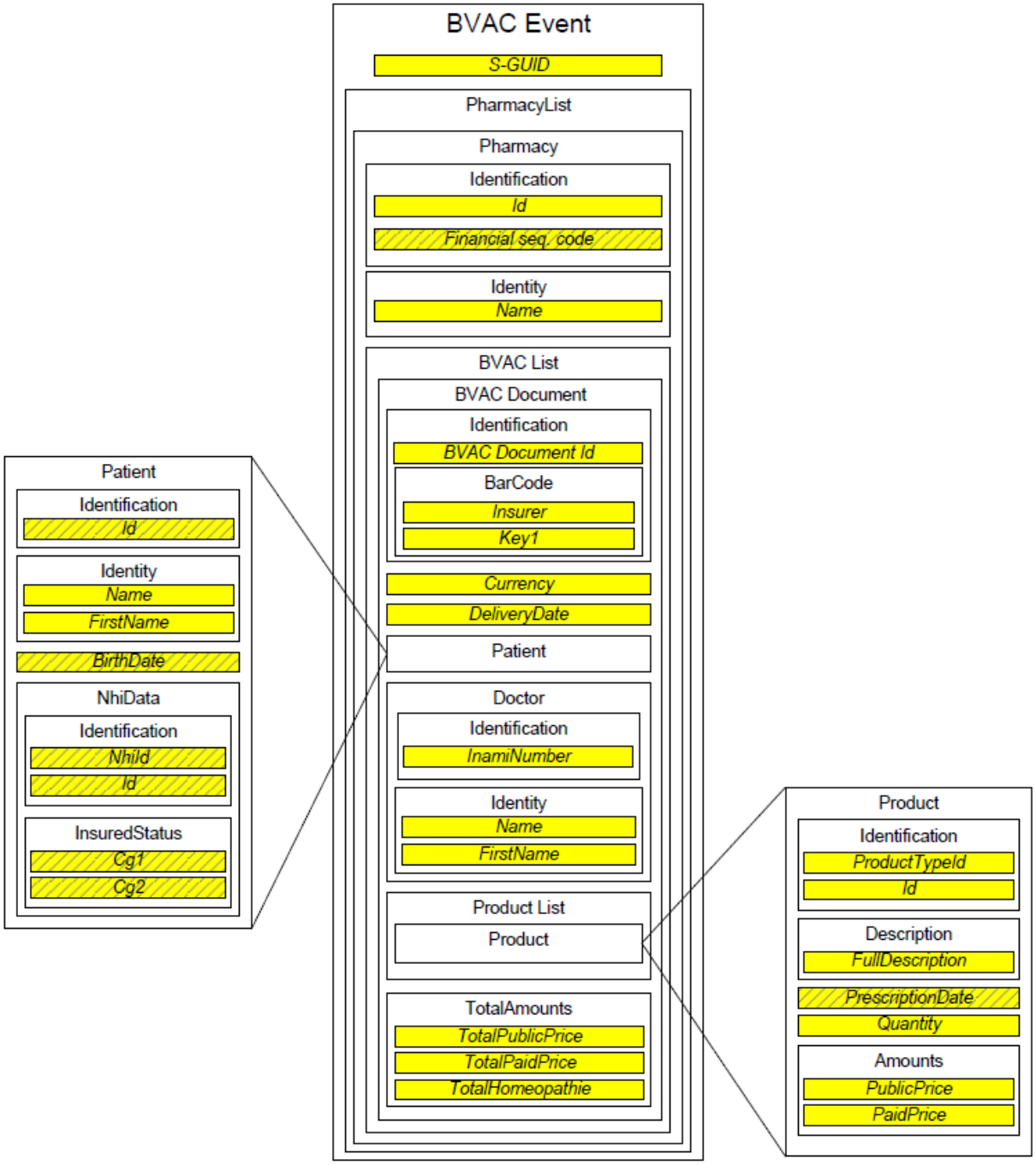
Attribute	Type	Required	Description
Motivation Type	Token	Y	Limited list as part of the tagging principle.
Motivation Text	String	N	Free text field for optional clarification.

Table 16: Possible attributes for a Motivation Entity

4.4.3 ASSURPHARMA – BVAC EVENT

As the personal share of medical expenses (portion of the cost not covered by INAMI) is increasing, patients are increasing their complementary coverage. At present an estimated 3 million BVAC attestations are issued yearly.
In the context of the Assurpharma project, an Assuralia flow has been implemented in the TIP System that will automate the transfer of the information contained within the BVAC attestation from pharmacist to insurer enabling reimbursement to the patient/insured person.

The BVAC event, in the single message, is used for doing this transfer between pharmacist and insurer.



4.4.4 STATUS ENTITY

This *new entity within the single message format* contains all information associated with a status update and is structured in three different sections:

- **Identification of the status update origin**: the IDs used are agreed and consistent within the solution.
  - The “Originator ID” is used for the supplier, e.g. APB Dossier
  - The “Service ID” is used for the service type, e.g. tarification
- **Identification of the data for which the status update is communicated**, currently supporting the session ID and dispensation ID. These are both optional parameters and will be extended if needed.
- **The status update content**, containing a timestamp of occurrence and a mandatory ID for tracking. This ID must be unique within the origin system so that, in combination with the originator information, it is globally unique.
- **The message type and subtype both contain entries from a pre-agreed list-of-values**. Care services are expected to deliver a list of possible message types and subtypes, and their meaning, so that the proper routing may be set up in the TIP system.
- **A free-text field is used for additional information on a status update**. The intended recipient is a human user, this field is not checked or parsed by the TIP and supports multiple locales.

4.4.5 OTHER TYPES OF DATA

It is important to distinguish what is meant with other types *in the single message* (the only type of data supported):

- Types **also stored in the pharmaceutical care data hub** but different from medication deliveries.
- Types **meant for other back-end services** as the pharmaceutical care data hub.

Because of their inclusion in the Single Message, processing of these other types will be easily enabled in a later release.

- The logic as present in the connector module and the inbound processing of the TIP is similar and will be reused, barring the introduction of new IDs, filtering, validations, … .
- Routing any Single Message entity/event *unmodified, within the TIP,* is present out-of-the-box.

4.4.5.1 ALREADY PRESENT TYPES FOR OTHER CARE SERVICES

The principle examples are tarification, archival and recipe-archive – also shown in the Single Message Concept section.

- The **tarification event smoa:APBFileEventType** contains the entire file as one element with date information and will be sent enciphered. There is only one tarification file and service for each pharmacist.
- The **archival event smoa:RegisterExportEventType** combines delivered medication entities and incoming medication entities, registering the stock and history of a pharmacy – on a fine granularity (ingredients, …).
- The **recipe-archival event smoa:PharmaceuticalPrescriptionEventType** is a consequence of delivering a RECIP- E prescription and meant to archive these prescriptions. The entity is defined by RECIP-E.

4.4.6 MESSAGE STRUCTURES FOR EACH OPERATION

4.4.6.1 REGISTER, UPDATE OR REVOKE DATA (CARE AND OTHER TYPE)

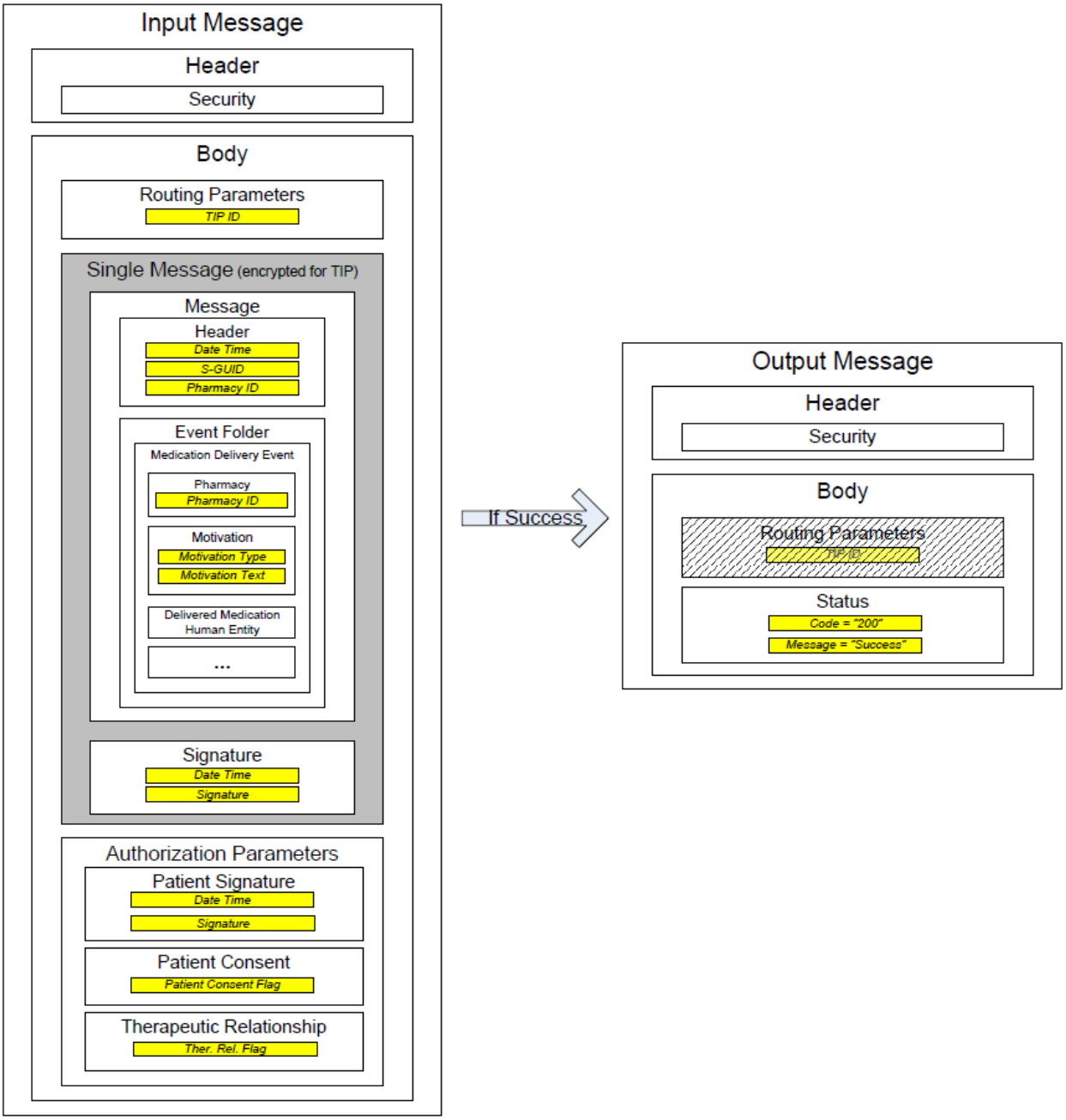


Figure 14: Registration message to TIP

The connector module generates a command message (the outer white box) based on input provided by the pharmacy software (the gray box = the SMC – minus the SGUID). This message is sent to the TIP using one of the register, update or revoke operations. *To be clear, the input message as depicted is not the Single Message as defined previously*. A wrapper envelope surrounds it with information for technical orchestration, this envelope is constructed in the connector module.

- The **top level header** groups eHealth required information to *orchestrate security* (SAML token, timestamp, …).
- The **TIP ID** allows initially the global TIP to *route the Single Message payload for further processing*. It is a parameter in the register/update/revoke operation.
- The **Single Message** in the body is *asymmetrically encrypted for a specific TIP*. If it is signed, a signature element in the encrypted payload provides the timestamp and actual signature for the message content
  - The header contains connector-module-generated-data such as the **Session GUID** described in the use cases.
  - The event folder contains multiple events and entities, see section 5.4.2.

- **The patient consent flag is included for registrations**. eHealth should orchestrate this flag in the future.

The response to this message in this case is a functional ACK/NACK ((non-)acknowledgement) message after syntactical validation and time-stamping.

4.4.6.2 STATUS MANAGEMENT

Each care service may asynchronously send status updates to the TIP system. This could be anything from "processing completed", to errors that may have occurred, or feedback as a result of processing. **Besides routing there is no tracking or processing foreseen on status updates**.

Status updates have the same format regardless of care service; each must implement the format defined. The only difference between care services are the possible types contained in the message and the message free-text.

- Note that the normal statuses for a dispensation are already defined for the Pharmaceutical Care Data Hub, however the possible errors will be defined and appended during design and build.

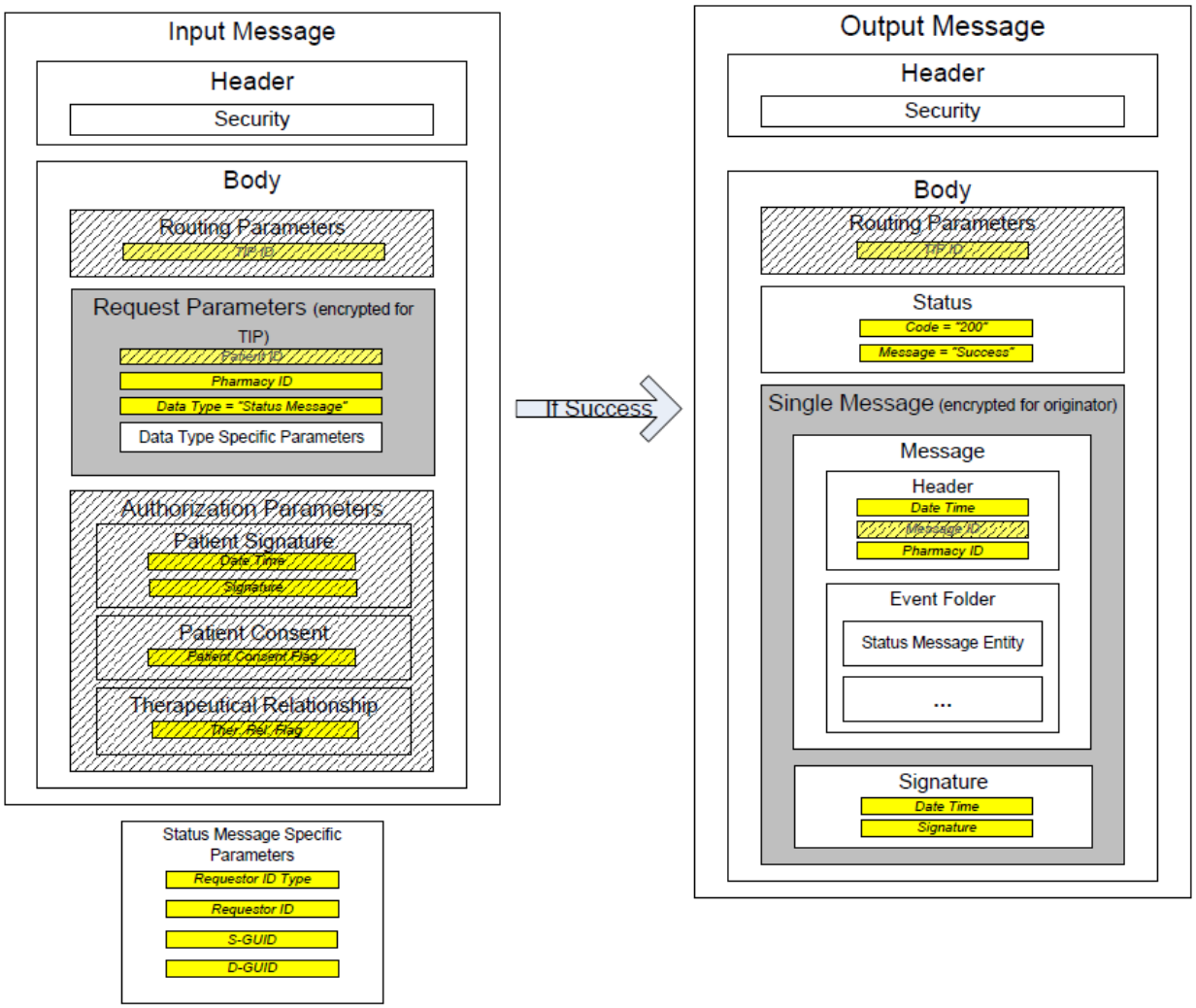


Figure 15: Status request & response

All request messages generated by the Connector Module have the same structure as pictured above, the pharmacy software is not required to build these. **However the above diagram is relevant to illustrate the responses**.

- Placeholder fields are present in each request for *Pharmacy ID*, *Patient ID*, *Patient Consent* and *Signature*.
  - The **data type field describes what kind of data is requested**. While also inferred from the service and/or operation called, this parameter presents another option to further restrict the data set returned.
  - The "data type specific parameters" element uses a **hierarchy to express possible parameters for each data type**. The business logic is responsible to check that the correct elements are used for the given data type.

The status response format does not differ structurally or functionally from the corresponding command message, it will only contain a bundling of status messages in a different envelope. *This does not mean there's a 1-to-1 relationship*

to what the Pharmacy receives and the status messages an external systems sends. The TIP may have filtered out outdated messages, or interpreted and generated the status message.

4.4.6.3 GET DATA TYPES

The same request message structure, the connector module uses this message to investigate which data types are available for a certain patient ID. The Pharmaceutical Care Data Hub consults the referral register and consolidates the available types in Data Type Location Entities per patient ID.

- A system data type list element groups all data types for a certain physical system. Multiple are possible in the Data Type Location Entity.
  - **One ID to designate the system**. This ID corresponds to an ID in the *system services list*.
  - **Multiple data type elements**. These are entries from a list of values of which some may be unknown to the connector module, e.g. *when not the latest version available*.

This interface and service has two purposes: *one is to inform and the other is to prevent the transfer of data types that the connector module does not support*.

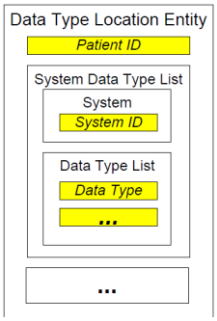


Figure 16: Data Type Location Entity

4.4.6.4 GET DATA

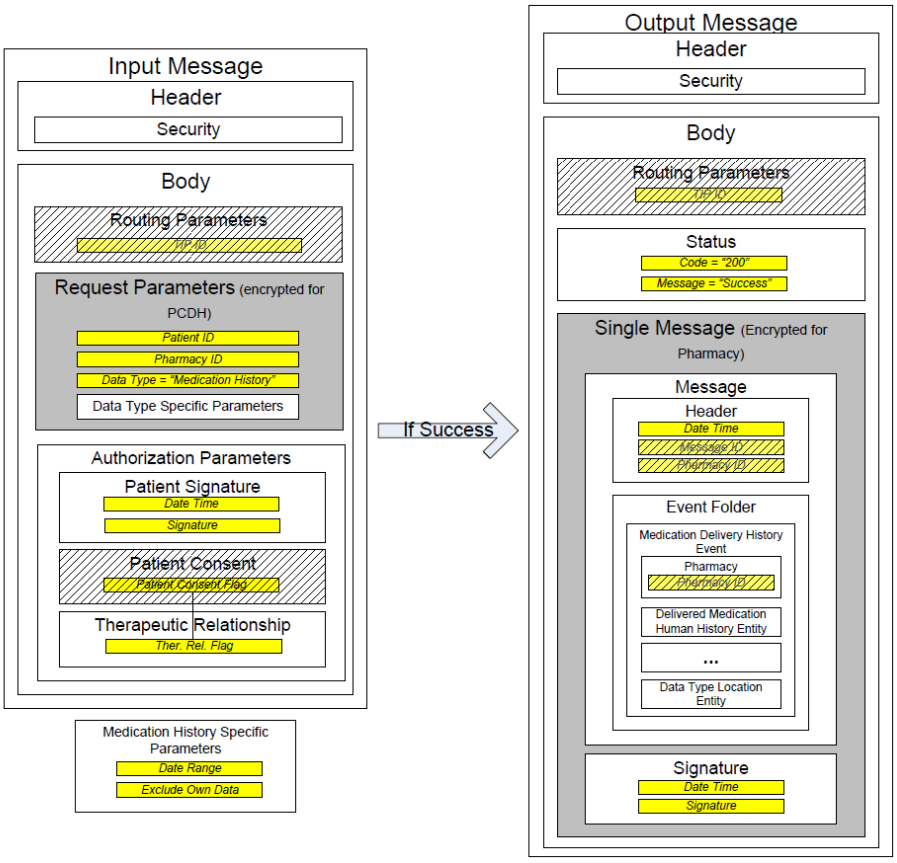


Figure 17: Get data, type medication history, request to PCDH

- The **patient ID** determines for which patient the medication history is requested.
- The **pharmacy ID** is the pharmacy invoking the request, checked with security information.
- The **data type** signifies medication history but could also contain "other type of data" when formally specified.
- **Patient consent and/or therapeutic relationship** is needed for some types of medication history and can be included, in addition to strong patient authentication using his/her signature.
- **The medication history specific parameters are**
  - The range of data to return
  - A flag to withheld or include data from the invoking pharmacist

Lookup responses contain entities stored in the Pharmaceutical Care Data Hub, e.g. Delivered Medication Human History Entity. In addition to this, some meta-information is added. The response again uses the same structure as the command messages.

- Note that due to the encrypted nature of the "medication delivery history events", it's possible there are some entities using an older version of the entity format. **The connector module and pharmacy software must support formats that have been put in production for the past archival period length**.
- The **Data Type Location Entity** is again used by the Pharmaceutical Care Data Hub to communicate the presence of the same type of data, for that particular patient, in other systems..

The Pharmaceutical Care Data Hub quality indicator, not shown in the diagram above, represents some information on what has been returned, as a consequence of authorization rules, TIP quality indicators, … .

Quality Indicator	Description
1	full dispensation details found
2	dispensation with potential incorrect (not validated) data elements
3	dispensation with no details attached due to the fact that no consent is available
4	dispensation record with a product that is not registered in Belgium

Table 17: Quality indicator for PCDH

4.4.6.5 GET PHARMACY DETAILS

The request message has the same structure as elaborated before. The **pharmacy entity is already standardized in the Single Message**, the structure is thus reused in the response. Unneeded information is left out.

Note that only an identifier is returned. *The pharmacy software must use this identifier to look up more details in the correct authentic source of pharmacies*.

4.4.6.6 ERROR RESPONSES

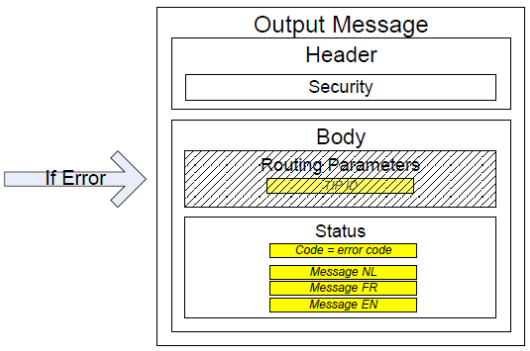


Figure 18: Error response structure

If any synchronous exchange defined in this chapter results in a *functional* error, the status message depicted in Figure is returned.

- The error code is first parsed to see if the error can be automatically handled and if so, the original request will be resubmitted with modified data, in response to the error.

If the error is fatal, the localized message is shown to the user and/or logged, depending on the system that receives the error status.

5 APPENDIX: EXECUTOR PACKAGE CONTENT

The package "Executor.zip" contains following files:

- conf: Recip-e sample configuration files.
- examples : sample application using Recip-e integration modules
- src : source files in Java of the integration modules
- wsdl : wsdl & xsd files for input/output of Recip-e web services
- lib/dotNet : The integration modules in dotNet (See examples/gui for sample use)
- lib/java : The integration modules in java (See examples/command-line for sample use)
- log : folder containing the logs of the examples.

6 APPENDIX: RELEASE NOTES

6.1 SDK VERSION 0.9.5T (GFD-DPP)

In this release the following items or issues were addressed:

- Unused Recip-e code has been removed (Bug#223, Feature#225)
- Error handling was improved (Bug#221, Bug#226)
- Properties file can be referenced by URL (Feature#222)
- Incorrect GetDataType in Mock implementation (Bug#219)
- Modules configured to use the Integration environment.

6.2 SDK VERSION 0.9.6T (GFD-DPP)

In this release the following items or issues were addressed:

- Create session problems fixed (Bug#229 , Bug#234 , Bug#238)
- Com interface problems fixed (Bug#227 , Bug#232, Bug#236)
- Environment properties are updated (Bug#231, Bug#235)
- Log4j appender warning removed (Bug#230)
- Multiple pom version (Bug#228)

6.3 SDK VERSION 0.9.7T (GFD-DPP)

In this release the following items or issues were addressed:

- Magistral preparation issue solved. (Bug#283)
- Logging improvement. (Bug#293)
- Performance improvement. (Bug#295)
    - The API is using multi-threading and to manage the threading per pc a property has been introduced thread.number.limit.
      Important: Changing this property can also have a negative effect on the API.
- SH Scripts updated (Bug#292)
- Update of System-services.xml (Bug#307)
    - The system service xsd has been changed on the TIP System and the API.
    - To be backwards compatible with the older version of the TIP System, that is deployed in Acceptance, the property system.services.enable.old has been added.
    - GFD DPP endpoint have been removed from the property files. The endpoints provided in the system services xml file are used instead. Default the acceptance environment is configured.
- Error message improved (Bug#221)

Performance metric log file (e.g. –metric.log) are split up in several files for the projects that use the performance metric logic. These are used by the API and should not be changed by other parties.
A next step of the GFD DPP project is to have multi TIP instances on the TIP System. The API contains changes that support this functionality. At this point the property has to be multitenancy.disabled=true till the multi tenancy is fully developed.
Important: In the release 0.9.7T some properties have been adapted and other are added for these reasons it's necessary to use the property file provided in 0.9.7T release.

6.4 SDK VERSION 0.9.8T (MANDATORY) (GFD-DPP)

In this release there have been some features added and some items have been changed. The following changes have been made:

- Multi Tenancy is supported; the property multitenancy.disabled has been removed in the config.

- The response of a register data has been changed. (The XML; the products tag changed to dispensation, see 5.3.3.2.3)

The following topics based on red mine tickets have been made:

- Product filter has been improved in various ways, e.g. magisterial preparations are now supported (see Table 12: Register Data) (Bug#316 , Bug#287 , Bug#322, Bug#310, Bug#315, Bug#283)
- Offline queue has completely been refactored (Bug#300)
- Single Message have been updated (Bug#323, Bug#270)
- Motivation is checked for, required, and put on the queue (Bug#312)
    - As specified in the SMC specification (xds), for an update and revoke a motivation is required under the SGUID tag, with the following format:

```
<model:motivation>
        <model:type>wrong medicine</model:type>
        <model:freeText>any text you feel like</model:freeText>
</model:motivation>
```

- Tip and PCDH call fixes (operation specific) (Bug#245, Bug#321 , Bug#267, Bug#319, , Bug#287 , Bug#290)
- Therapeutic relationship & Patient Consent (Bug#328, Bug#324)
- Single Message have been updated (Bug#271, Bug#252)

From now on, usage of version 0.9.8T is mandatory!

6.5 SDK VERSION 0.9.9T (GFD-DPP)

In this release there has been a feature added:

- The possibility to register BVAC events.

The following red mine tickets have been solved in this release:

- Use of exclusions on general informed consent (Bug#447)
- Use of specific eHealth extension for certificates .Acc-p12 (Bug#432)
- Error messages with eHealth connection issues have been made more explicit (Bug#336)
- Typos in status messages resolved (Bug#327)
- Performance issues on first message have been resolved (Bug#274)
- Error message duplication issue (Bug#317)
- Register data with no connection – off line queuing (Bug#318)
- French message error due to regional language setting (Bug#335)
- The date range of the get data (Bug#444)
- UnknownHostException on the performance metric (Bug#338)
- Message when a session is expired is wrong (Bug#419)
- hasValidSession method incomplete (Bug#449)

6.6 SDK VERSION 1.0.0

In this release the following items were addressed:

- The possibility to use only the personal certificate as prescriber.
- Code packing structure has been changed to a module structure.
    - be.connector.common.CommonIntegrationModule
        - Session related functionalities
    - be.connector.executor.recipe.ExecutorIntegrationModule
        - Recip-e Executor related functionalities
    - be.connector.prescriber.recipe.PrescriberIntegrationModule
        - Recip-e Prescriber related functionalities
- A new property “care.provider.type=PHARMACIST” is introduced for the executor module. This is a **mandatory** property and is part of the restructuring of the code.

Important: the old executor integration module class is deprecated and will be removed in future releases.

6.7 SDK VERSION 1.1.0

In this release the following items were addressed:

- Implementation of the eHealth technical connector version 3.3.0 beta-1. This implementation manages the calls to the session service, ETK and KGSS. For more info about this technical connector visit: https://www.ehealth.fgov.be/nl/support/connectors
- To read the eID we use the fedict eID library version 0.5.1
- IKVM update to version *ikvm-7.2.4630.5*
- A lot of properties has been changed, you can find more information in chapter 5.2.5 Properties also in the comments that are in the example .properties files delivered in the executor and prescriber *conf*folder.
- j2pcsc.dll needs to be added in the working directory (the folder where you start the application .exe, .bat or …)

Important:

- **We recommend to use version 1.1.0 of the business connector because we made some improvements in code.**
- **You cannot use the windows-com-dll in this version (this will be fixed in future release)**

6.8 SDK VERSION 1.1.1

In this release the following items were addressed:

- The problem with the eID in the windows-com-dll is fixed

Important:

- **We recommend to use version 1.1.1 of the business connector because we made some improvements in code.**

6.9 SDK VERSION 1.1.2

In this release the following items were addressed:

- Added 2 versions of caCertificateKeystore.jks (acceptance and production)
- Added setProperty method in CommonIntergrationModule
- Improvement exception handling

6.10 SDK VERSION 1.2.0

In this release the following items were addressed:

- Chapter IV added
- New structure of business connector (different modules, other package names)
- The same IKVM version of eHealth is used
- Changed the document history to version 1.2.0.

6.11 SDK VERSION 1.3.0

In this release the following items were addressed:

- MSCOR exception on .Net 4.0 is fixed
- Removed pharmcayNihii typo in commandline
- Correct Chapter 4 xml and properties are used in examples

6.12 SDK VERSION 1.3.1

In this release the following items were addressed:

- Using eHealth 3.3.1 version of technical connector
- Using new IKVM
- FileLock added on offline queue
- Wsconsent example added in commandline

6.13 SDK VERSION 1.4.0

In this release the following items were addressed:

- Added setSystemProperty method to CommonIntegrationModule
- Business change, unsealing changed.
- Added isTest to comwrapper interface (ChapterIV)

6.14 SDK VERSION 1.4.1

In this release the following items were addressed:

- Register/Update/Revoke is successfully added to offline queue when no session is available.
- Connector can start with empty productfilter folder/file.
- Changed chapter IV in properties file.
- truststore.jks added to p12 folder.
- Message queue will unlock locked files older than 2 minutes when starting a new session.

6.15 SDK VERSION 1.4.2

In this release the following items were addressed:

- truststore.jks properties added to the connector-client.properties file.
- Bugfix register Bvac
- Bugfix encoding

6.16 SDK VERSION 1.4.3

In this release the following items were addressed:

- Addition of consultation MyCareNet Insurability v2 (Extended insurability) next to v1
  Please note that “Orchestrated” requests (recip-e and MyCareNet insurability request in a single call) only return MyCareNet Insurability v1 responses.

6.17 SDK VERSION 1.5.0

In this release the following items were addressed:

- Migration to latest version of eHealth Technical Connector (3.4.0-beta-9).
- Migration to latest version of Chapter IV and WSconsent webservices (compatible with Technical Connector 3.4.0-beta-9).
- Integration Orchestrated MyCareNet Insurability v2 consultation.
- Security modifications.
- Minor modifications to COM-interface (based on input ISV).
- Addition of self-signed Government CA 2011 certificate to caCertificateKeystore.jks.
- Bugfix connector not correctly loading truststore.jks.
- Cosmetic changes to connector-client.properties file.

6.18 SDK VERSION 1.5.1

This version of the connector contains several important bug-fixes and enhancements, mainly regarding GFD-DPP and Assurpharma. Main changes can be found below:

- Migration to latest version of eHealth Technical Connector (3.4.0-beta-10)
- Migration to latest version of Chapter IV and WSconsent webservices (compatible with Technical Connector 3.4.0-beta-10).
- Logging adaptations:
  Logging levels have been revised: the “INFO” level (set by default) is again recommended for production.
  Log level can be set through conf/log4j.xml:

```
<root>
        <priority value="INFO"></priority>
        <appender-ref ref="FILE"/>
</root>
```

  A new parameter was added to conf/log4j.xml: “MaxBackupIndex”. This parameter allows setting the maximum number of days log files should be kept. Beware that this modification introduced a new naming convention for the connector log files.

```
<param name="MaxBackupIndex" value="21"/>
```

- Queue behavior:
  When the queue contains invalid messages (xsd non-compliant, incorrect UTF-8 encoding, wrong pharmacyID, …) or messages that cannot be deserialized, they will be moved to either conf/UNENCRYPTED_FOLDER or conf/INVALID_FOLDER (paths can be configured in the properties file). Those messages will thus no longer block the message queue.
  Software vendors are asked to regularly check for the presence of messages within these folders.
  In case of connectivity issues, sending of the messages on the queue is repeatedly retried. Initial retry is done after 5 seconds (default value, can be set via the property MESSAGE_QUEUE_TIMER) and an exponential back-off procedure starts. Waiting time is multiplied by 2 at every retry. 5, 10, 20, 40, 1:20, 2:40 …….
  Upon successful sending or at session creation, retry time is reset to default value.
- Additional validation of single messages (BVAC and GFD-DPP).
  Additional validation of “Single Messages” handed over to the connector will take place. These validation steps include, amongst others, the use of a correct PharmacyID, and xsd-compliance. In case of validation failure, an error will be returned by the connector. In that case no DGUID and SGUID will be returned!
-

6.19 SDK VERSION 1.6.0

This version of the connector contains important modifications mainly regarding legislative changes concerning recip-e.

- Validation of legal proof of a prescription
- Possibility to archive prescriptions and prescription comments in the RAOTD-archive

Other changes:

- Java to version 1.7
- Ehealth connectors to version 3.10.4 (including MyCareNet Insurability BCP functionality)
  a new property was therefore added in connector-client.properties :
  be.ehealth.technicalconnector.ws.genericsender.invokestrategy=be.ehealth.technicalconnector.ws.impl.strategy.RetryStrategy
- Single Message to version 2.0.3
- Addition of new SAML attributedesignator in connector-client.properties:
  sessionmanager.samlattributedesignator.8=urn:be:fgov:identification-namespace,urn:be:fgov:ehealth:1.0:holder-of-key:etk:serialnumber
- Addition of RAOTD-specific properties in connector-client.properties:
  ArchivePrescription-xml=true
  ArchivePrescriptionDirectory=%CONF%/Archive/Prescription/
  validate.befor.archiving=true
  sdk.version=1.6.0
  supported.prescription.version=1.17.1
  prescription.version=1.17.1

6.20 SDK VERSION 1.6.1

This version of the connector contains several bug fixes and optimizations.

Bug fixes:
all services: fixed problem loading truststore in .NET implementation
recip-e: critical bugfix when retrieving prescription of some dentist prescriptions
GFD-DPP: GetData call no longer gets stuck when KGSS returns SOA-error.
RAOTD: fixed problem with DGUID and SGUID being the same
RAOTD: fixed bug not correctly setting correct time in RID.xml file
RAOTD: fixed timestamp problem

Functional changes:

- Kmehr to version 1.19
- Statusmessages:
  In order to meet the need for returning statusmessages for RAOTD as fast as possible some modifications were made.
  - When the function “getstatusmessages” is called without specifying SGUID and/or DGUID the server will return all non-read statusmessages (with a maximum of 250) when the total number of incoming calls does not exceed a variable threshold. This variable threshold is based on the server’s load. In case the threshold is reached, an empty statusmessage will be returned. In that case, please retry later. We recommend an interval between 30 and 60 min for retrieval of statusmessages.
  - When the function “getstatusmessages” is called and SGUID or SGUID+DGUID are specified, the server will always return all available statusmessages for the SGUID (or SGUID+DGUID combination), regardless of the fact that the statusmessage(s) were retrieved before. The statusmessages will be available during 6 months on the server. After 6 months, they will be permanently removed.

6.21 SDK VERSION 1.6.1P

This version is a patched version of version 1.6.1:

- Allows connection to the new endpoints STS (endpoints STS Legacy will be no longer available as of 21/10/2018 - R2018.2)
- Deals with the SSL-configuration changes of services.ehealth.fgov.be (TLSv1.2) as of 21/10/2018 (R2018.2)

6.22 SDK VERSION 1.6.1P2

This version is a patched version of version 1.6.1P:

- Bugfix create session with eID
- Configuration of new timestamp certificate (production env)

7 APPENDIX : FREQUENTLY ASKED QUESTIONS

Following is a list of frequently asked questions concerning the implementation of the integration specifications and the use of the integration modules.

Question	Answer
How do we change the default prescription used in the mock version of the integration modules?	The prescription, notification and feedback used in the mock version are found in the \examples\resources folder. You can update these files.
Doesn't it make more sense to add a string in front of the Recipe ID to indicate it is a Recip-e ID (instead of immediately beginning with BEPP)	This is currently not foreseen. During the course of the pilot this will be evaluated.
Which doctor id should we use, the one in the Kmehr message, or the one in the header?	The INAMI/RIZIV number
Is it possible to store more than one prescription in the same Kmehr message?	No
Can we create our own prescriptions to use during testing?	Yes, using the sample prescriber application or, when using the mock version, by updating the example xmls.
Since there is no archiving currently foreseen, what has to happen with the MarkAsArchived function during the pilot? Should we always call it?	This should only be called when the prescription has been archived. For testing purposes, we will have test cases where this functionality needs to be called.
Can e-health provide a service to get public keys of doctors?	Use the ETK depot service.
How do we implement during the time when the e-Health ETK service is not yet up?	The service is already up.
Please explain more in detail the certificate request procedure. Are there any tools we can use? Could you provide us with an example of a certificate? Do we need to resign responsibility document for each certificate renewal?	e-Health will create a tool to automate this procedure. A sample certificate is provided in the conf folder of the module.
Please provide more information on the '10 items' limitation.	The printable area of the prescription is limited to 80 chars wide on 10 lines height. If the prescription requires more space, the content should be split on multiple prescription having each one its own RID
How use the mock functionality in Windev?	Declare the PrescriberIntegrationModuleMock class. This is valid for all programming languages. For Windev implementation specific questions please refer to your Windev documentation.
If a physical prescription contains more than 10 items, and we thus have to split it up over more than one electronic one, can we reuse eHealth keys for both prescriptions?	No, each prescription has its own encryption key.
In the notification, do we have to encapsulate the full Kmehr message in notification xml message?	The field is optional
Can dll modules be used to do end to end encryption? (For other purpose than Recip-e communication).	Yes

The feedback is in text (without code CNK?). In our application, we need to flag delivered drugs. How can make something with text when receiving the feedback ?	The feedback is linked to the overall prescription. No link is foreseen with delivery items.
Should the software provider maintain a list of executors (pharmacists) with correct IDs or are they available on e-Health?	Currently neither Recip-e nor e-Health provides a pharmacist inventory. So at this time, implement it in the software itself.
Where do we need to pass the SAML token in the different functions of the integration module?	The SAML token is stored in memory by the module itself. You do not need to pass it as an input.
When do we need to signal the prescription as delivered? After one of the items has been delivered, or after all of the items have been delivered?	If 1 of the items is delivered, the complete prescription is considered as delivered, and the MarkAsDelivered functionality should be called.
Which encoding should be used?	All recip-e messages (prescriptions, feedback and notifications) should use the UTF-8 encoding.