

EVSc_1.x.y_Manual

- <DEPRECATED>
- Introduction
- Functionality
 - General
 - Input-folder
 - Which patient?
 - Which actor?
 - Which files?
 - What is a Kmehrmessage?
 - How is a Kmehrmessage identified?
 - Identification by Vitalink ID
 - Identification by EVS reference
 - Which actions?
 - Action "add"
 - Action "empty"
 - Action "export"
 - Action "removeID"
 - Action "removeREF"
 - Action "replace"
 - Action "updateID"
 - Action "updateREF"
 - Action "generateREF"
 - Action "updateschemeREF"
 - Processed-folder
 - Validation file
 - Global scheme PDF
 - Daily scheme PDF
 - Export file
 - Input file
 - Error file
- Configuration
 - How to add a patient?
 - How to add an actor?
- Parameterisation
- Appendix A: Folder structure EVS 1.x.y
- Appendix B: EVS-exporter
 - Launching
 - Output
 - Parameterisation

<DEPRECATED>

Introduction

EVS allows to manipulate vault contents using specific actors and specific patients, manually or based on previously exported vault contents.

After initial installation, some examples are available to get familiar with the functionalities of EVS, without the need for further configuration.

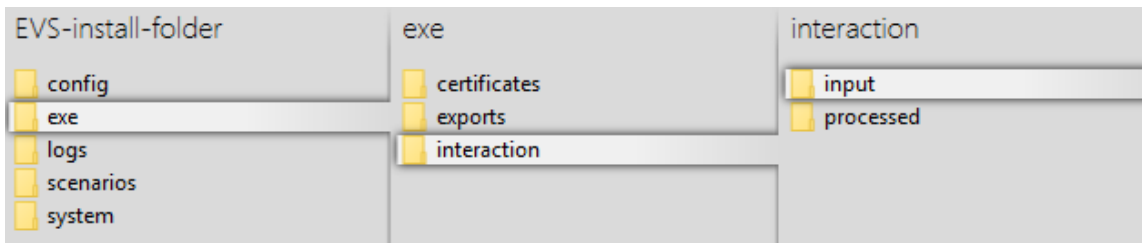
This manual describes EVS v1.2.2.

Functionality

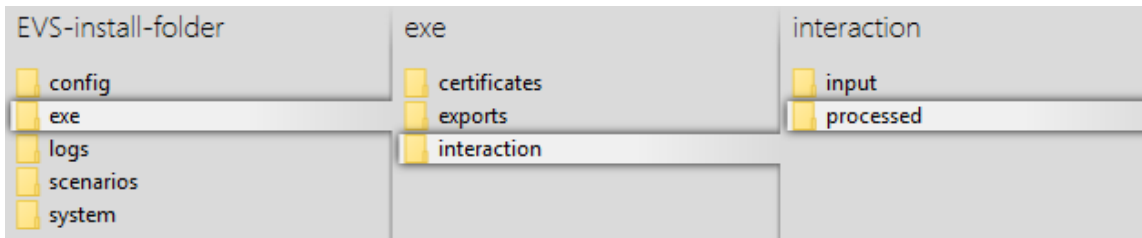
General

EVS allows a certain actor to perform, for a certain patient, a number of actions.

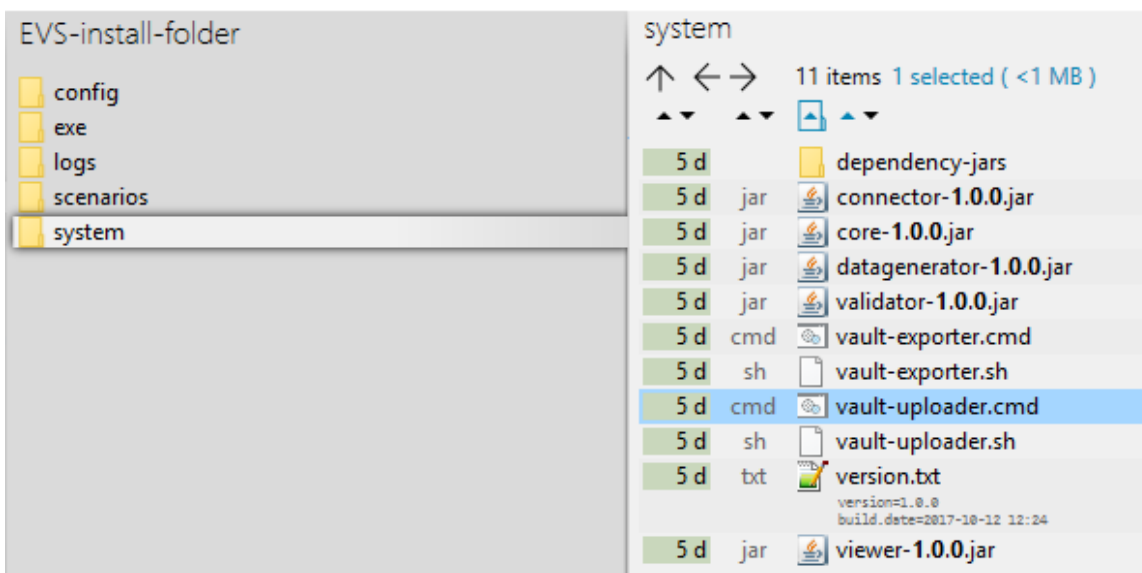
These actions are triggered by dropping input files in (subfolders of) the input-folder:



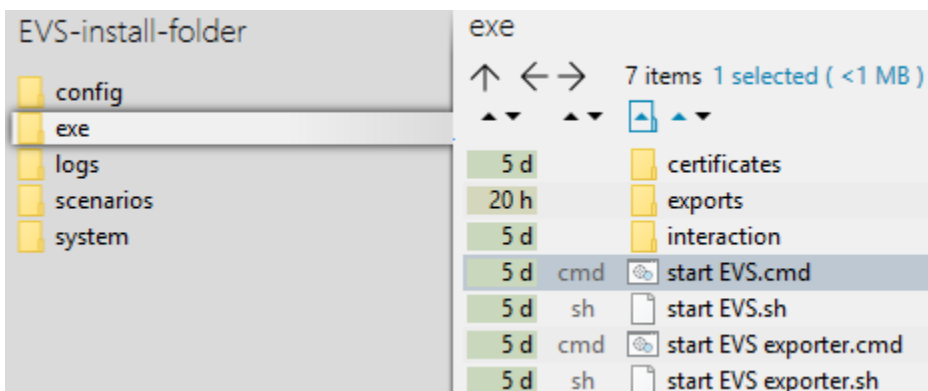
EVS watches these folder(s), executes the action(s) and generates output in the processed-folder:



EVS can be launched via the "vault-uploader.cmd" batch file:



The behaviour of EVS must be determined by passing some mandatory parameters. Instead of using the "vault-uploader.cmd" batch file, it is easier to use the example batch file "start EVS.cmd":



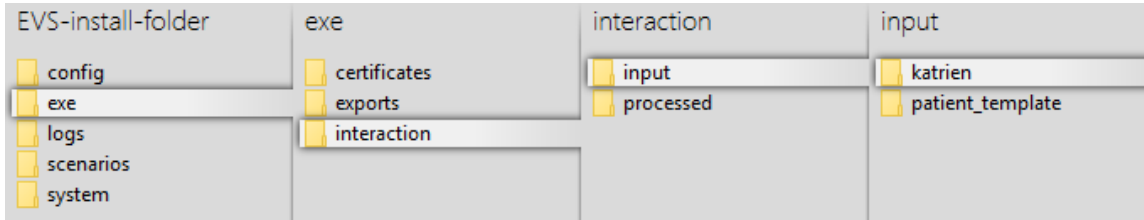
This batch file contains parameter values for a standard behaviour. How the parameters change the behaviour can be found in the paragraph [Parameterisation](#).

Input-folder

Which patient?

The patient is determined by the folder under "..\exe\interaction\input".

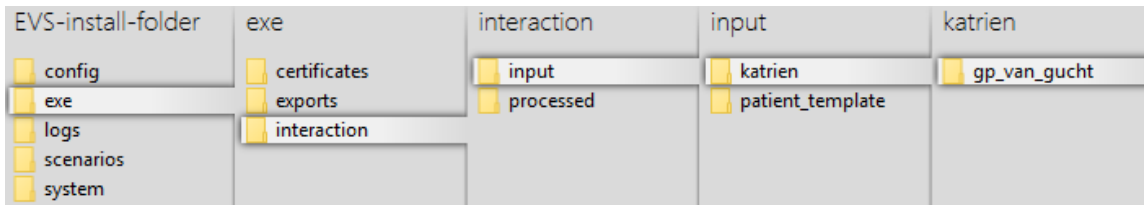
After initial installation, 1 patient named "katrien" is available:



Which actor?

The actor is determined by the folder under "..\exe\interaction\input\<patient folder>".

After initial installation, 1 actor named "gp_van_gucht" is available:



Which files?

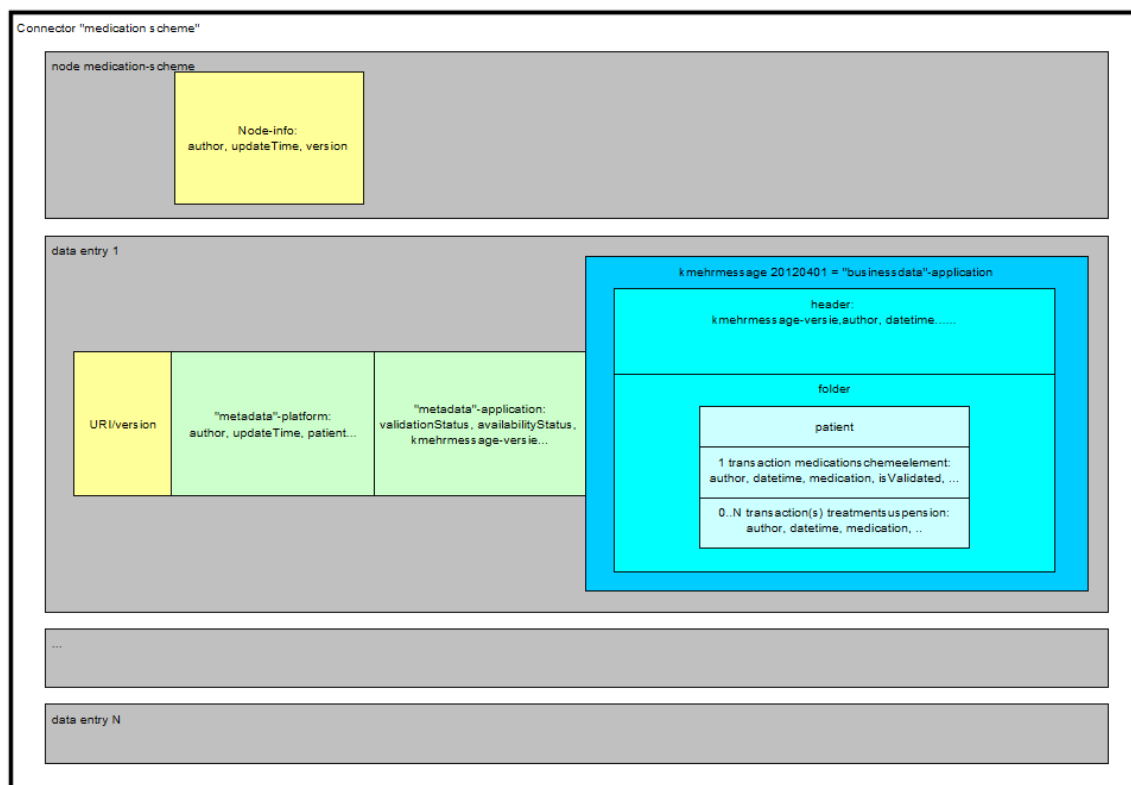
Any type of files, with any extension, can be dropped. They are considered as "input-files". EVS will, depending on the action folder, parse the files and extract the Kmehrmessage(s).

What is a Kmehrmessage?

A Kmehrmessage is that part of the file that starts with <kmehrmessage ...> and ends with </kmehrmessage>. One file can contain 0 or more Kmehrmessages.

EVS will only work with Kmehrmessages of Kmehr-standard 20120401. In this standard, 1 medicationschemeelement transaction and 0 or more treatmentsuspension transactions were considered as the business data of 1 "data entry".

All other data (among which the metadata) will be generated by the EVS and/or the Vitalink platform. As input the business data as depicted in blue here below will be used:



How is a Kmehrmessage identified?

For some actions, typically removing and updating data entries, the input Kmehrmessage needs to have an identification.

There are 2 ways to identify Kmehrmessages:

- by Vitalink ID,
- by EVS reference.

Identification by Vitalink ID

The Kmehrmessage should be preceded by a Vitalink URI.

The Vitalink ID can not be chosen by the sender when adding new data entries, but are dynamically generated by the Vitalink platform. By consequence, they are not deterministic and are hard to use when defining scenarios.

In the next example, this URI is `/subject/83051839468/medication-scheme/32055/1`. EVS detects the reserved format `"/subject/#####/medication-scheme/32055"` and finally uses "32055" as unique ID.

```

10 URI
11 -----
12 /subject/83051839468/medication-scheme/32055/1
13
14 MetaData
15 -----
16 encryptionFlag=encrypted
17 formatCode=KMEHR_20120401
18 authorRole=PHYSICIAN
19 creationTime=2017-10-12T16:07:04+02:00
20 patientID=83051839468
21 mimeType=text/xml
22 languageCode=nl-BE
23 authorPersonSSIN=83051839468
24 authorPersonNIHII=11425214001
25 authorPersonFamilyName=GUCHT
26 size=7312
27 authorPersonFirstName=KATRIEN VAN
28 availabilityStatus=active
29 validationStatus=validated
30 hash=24A2CAC5FF3EAFDEDF6F7CF610EFB11FA7D81FB4
31 authorPersonRole=PHYSICIAN
32
33 BusinessData
34 -----
35 <?xml version="1.0" encoding="UTF-8"?>
36 <kmehrmessage ...>
37   <header>
38     <standard>
39       <cd S="CD-STANDARD" SV="1.4">20120401</cd>
40     </standard>
41     ...

```

Identification by EVS reference

An EVS reference is put in 1 (and only 1) free text field in the concerned Kmehrmessage.

The EVS reference can be freely chosen, and facilitates the definition and execution of scenarios.

In the next example, this REF is "===EVSREF:901===". EVS detects the reserved format "===EVSREF:<any text>===" and finally uses "<any text>" as unique REF.

As from release 1.1.0 the REF must contain a minimum of 3 characters and can contain up to 20 characters, so both ===EVSREF:123=== and ===EVSREF:A123456789B123456789=== are considered valid REFs.

```

36 <kmehrmessage ...>
37 <header>
38 <standard>
39 | <cd S="CD-STANDARD" SV="1.4">20120401</cd>
40 </standard>
41 ...
42 </header>
43 <folder>
44 <id S="ID-KMEHR" SV="1.0">1</id>
45 <patient>
46 | <id S="ID-PATIENT" SV="1.0">83051839468</id>
47 | <firstname>Katrien</firstname>
48 | <familyname>Van Gucht</familyname>
49 | <sex>
50 | | <cd S="CD-SEX" SV="1.0">female</cd>
51 | </sex>
52 </patient>
53 <transaction>
54 ...
55 <item>
56 | <id S="ID-KMEHR" SV="1.0">3</id>
57 | <cd S="CD-ITEM" SV="1.4">medication</cd>
58 | <content>
59 | | <medicinalproduct>
60 | | | <intendedcd S="CD-DRUG-CNK" SV="1.0">1077718</intendedcd>
61 | | | <intendedname>Insulatard Penfill (5 x 3 ml)</intendedname>
62 | | </medicinalproduct>
63 | </content>
64 | ...
65 | <instructionforpatient L="nl">===EVSREF:901=== Actual instruction for patient.</instructionforpatient>
66 </item>
67 </transaction>
68 </folder>
69 </kmehrmessage>

```

Which actions?

Depending on the folder where the inout-file is dropped, EVS will execute an action.

Action "add"

This action will add a data entry to the vault for each Kmehrmessage found in all dropped files.

In the example below, 3 Kmehrmessages are dropped to be added to the vault:

The screenshot shows the EVS interface with a tree view on the left containing folders like 'config', 'exe', 'logs', 'scenarios', and 'system'. The main area displays a table of actions for the 'gp_van_gucht' folder. The 'add' action is highlighted. To the right, a list of 3 items is shown: 'file_without_kmehrmessages.any', 'file_with_1_kmehrmessage.any', and 'file_with_2_kmehrmessages.any'.

Action "empty"

This action will remove all data entries from the vault. EVS will do this action once for each dropped file, without any parsing.

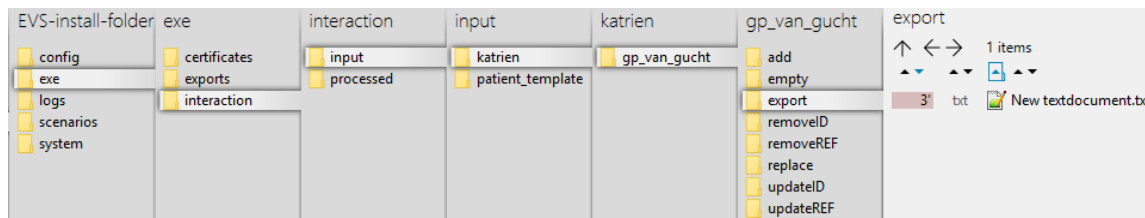
In the example below, a newly created file will trigger emptying of the vault by removing all existing data entries:

The screenshot shows the EVS interface with the 'empty' action selected for the 'gp_van_gucht' folder. The right panel shows 1 item: 'New textdocument.txt'.

Action "export"

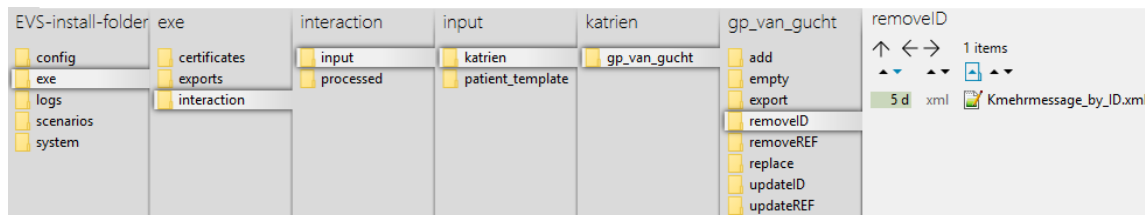
This action will export the contents of the vault, without any change to the vault itself. EVS will do this action once for each dropped file, without parsing this file.

In the example below, a newly created file will trigger an export of the contents of the vault:



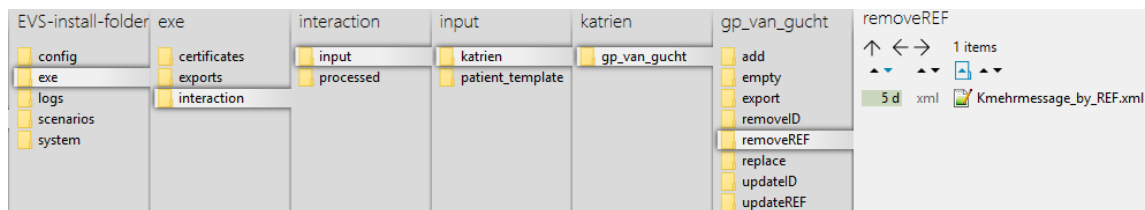
Action "removeID"

This action will remove the data entries identified by the Vitalink ID in the input file from the vault.



Action "removeREF"

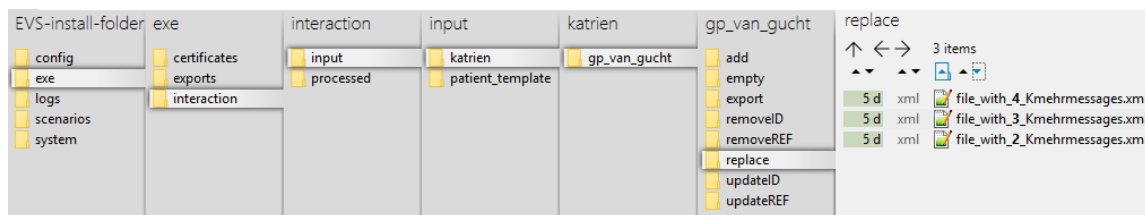
This action will remove the data entries identified by the EVS REF in the input file from the vault.



Action "replace"

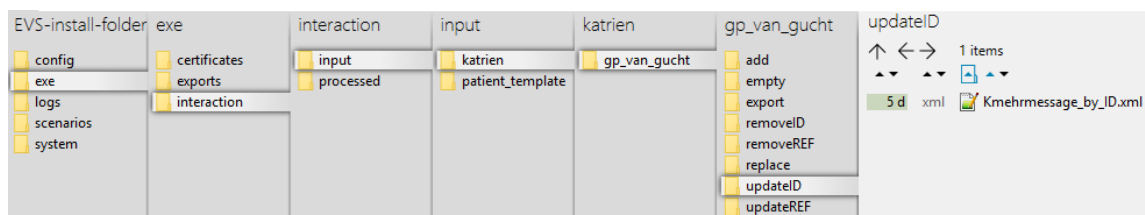
This action will replace the contents of the vault by all the Kmehrmessages found in the input file. Be aware of the fact that dropping multiple files in the replace-folder will result in a vault with as contents the Kmehrmessages of the last input file!

In the next example, after processing the next 3 input files, the vault contains 2 data entries.



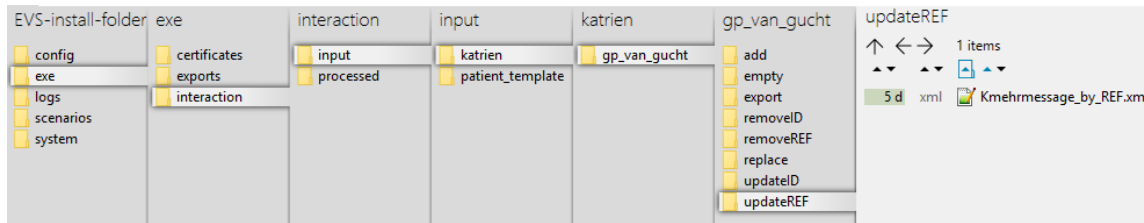
Action "updateID"

This action will update the data entries identified by the Vitalink ID in the input file.



Action "updateREF"

This action will update the data entries identified by the EVS REF in the input file.



Action "generateREF"

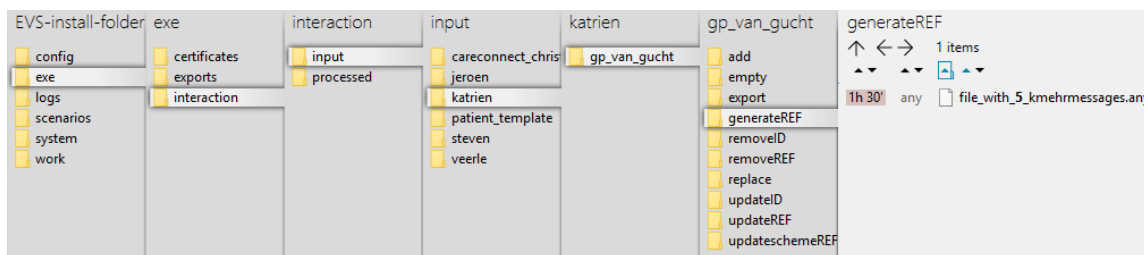
This action will generate an EVS REF for each Kmehrmessage in the input file, and will then replace the contents of the vault by all the Kmehrmessages found in the input file.

A Vitalink ID must be present for each Kmehrmessage in the input file as the EVS REF is derived from the Vitalink ID.

If an EVS REF exists already in the Kmehrmessage, the existing EVS REF is removed and the new EVS REF is put at the same location.

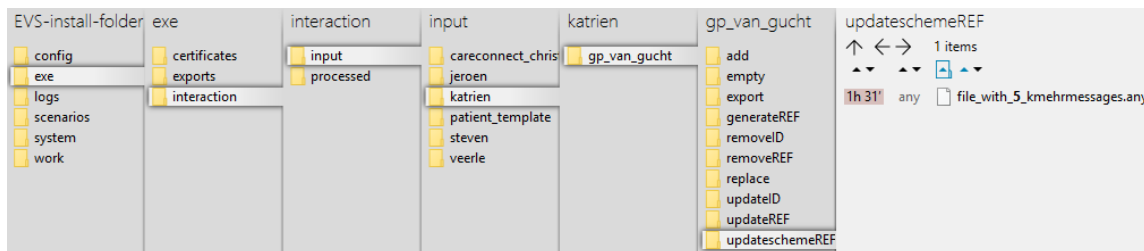
If multiple EVS REFs exist in the Kmehrmessage, all EVS REFs are removed and the new EVS REF is put in the instructionforpatient field.

Likewise, if no EVS REF exists, the new EVS REF is also put in the instructionforpatient field.



Action "updateschemeREF"

This action will update the complete contents of the vault, based on the input file compared with the current contents of the vault.



The next actions will take place:

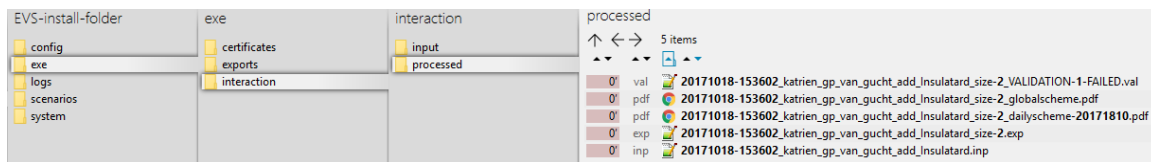
- an input-Kmehrmessage with EVS REF not yet existing in the vault will be added to the vault
- an input-Kmehrmessage with EVS REF already existing in the vault will cause an update but only if the input-Kmehrmessage differs from the vault-Kmehrmessage
- all vault-Kmehrmessages without corresponding EVS REF in the input file will be removed from the vault

If the input-file contains Kmehrmessages without EVS REF or if the EVS REFs used are not unique, the action will not be executed and an error will be thrown.

Processed-folder

After execution of an action a variable number of output files are generated in the processed folder.

The next screenshot shows the output of a successful add-action. Each output file will be explained below:

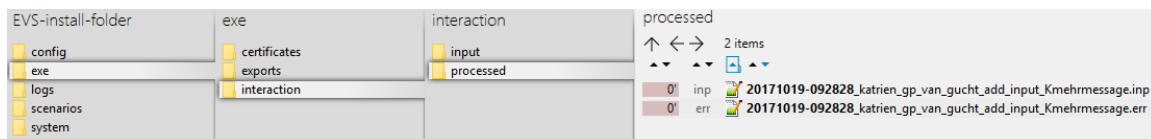


Each filename exists out of:

<date>-<time>_<patient>_<actor>_<action>_<input filename>_size-<nr of data entries>_<output suffix>.<output extension>

Name	Output suffix	Extension	Description	Remarks
Validation file	VALIDATION-OK VALIDATION-<###>-FAILED	.val	The report of the validation.	The filename contains the number of warnings and errors when the validation fails.
Global scheme PDF	globalscheme	.pdf	A visualisation of the global scheme in PDF format.	-
Daily scheme PDF	dailyscheme-<date>	.pdf	A visualisation of the daily scheme in PDF format.	This is the scheme of the medication that should be taken today. ⚠ For the moment, only "today" is generated. Future EVS releases will add free choice of the date.
Export file	-	.exp	An export of the contents of the vault.	-
Input file	-	.inp	The original input file.	The filename does not include the number of data entries in the vault.

If for some reason the action fails, an error output file is generated:



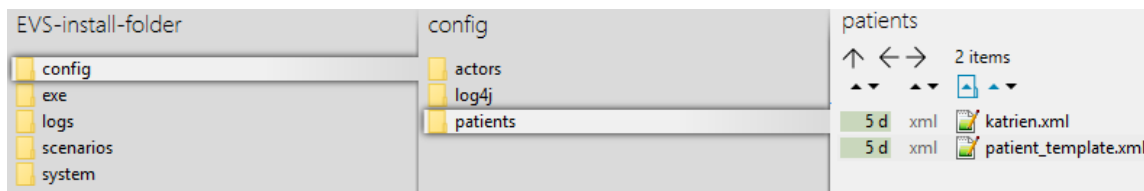
Name	Output suffix	Extension	Description	Remarks
Error file	-	.err	The report containing the error.	The content of the error file will identify the problem.

Configuration

This paragraph explains how to configure EVS.

How to add a patient?

Extra patients can be added by creating files in the next folder:



After initial installation, 2 patient config files are already present. Both can be used as example (copy-paste) to add extra patients. The name of the file, without the extension, needs to be used to identify for which patient the action needs to be performed.

After copy paste of the example files, insert the correct info for the new patient:

```

1 <patient>
2   <id>93051822361</id>
3   <firstName>Example Patient Firstname</firstName>
4   <lastName>Example Patient Lastname</lastName>
5   <gender>female</gender>
6   <birthDate>1993-05-18</birthDate>
7 </patient>

```

Since EVS follows all the rules for eHealth and Vitalink, it is up to the user to make sure the proper eHealth dependencies (informed consent, therapeutic relation, ...) are set in function of the wanted behaviour.

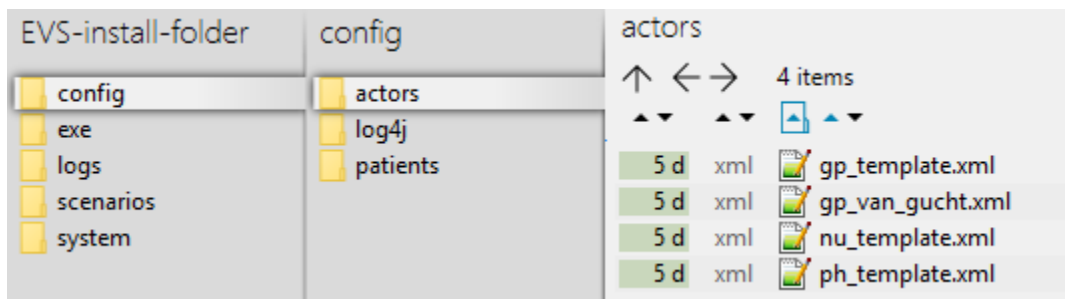


Restart EVS

EVS (and EVS-exporter) should be restarted when newly added patients will be used.

How to add an actor?

Extra actors can be added by creating files in the next folder:



After initial installation, some actor config files are already present. All can be used as example (copy-paste) to add extra actors. The name of the file, without the extension, needs to be used to identify by which actor the action needs to be performed.

The template examples are given for 3 different types of actors: doctor, nurse and pharmacy.

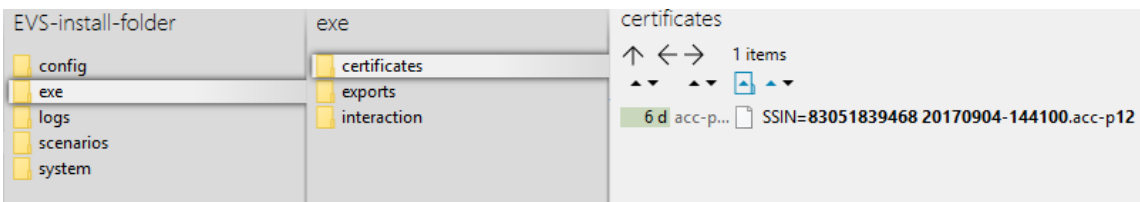
After copy paste of the appropriate example file, insert the correct info for the new actor:

```

1 <authenticationConfig>
2   <type>fallBack</type>
3   <mainCertificateLocation>..\exe\certificates\SSIN=83051839468 20170904-144100.acc-p12</mainCertificateLocation>
4   <ssin>83051839468</ssin>
5   <mainCertificatePassword>meds2017</mainCertificatePassword>
6   <samlAttribute>
7     <namespace>urn:be:fgov:identification-namespace</namespace>
8     <key>urn:be:fgov:ehealth:1.0:certificateholder:person:ssin</key>
9     <value>83051839468</value>
10  </samlAttribute>
11  <samlAttribute>
12    <namespace>urn:be:fgov:identification-namespace</namespace>
13    <key>urn:be:fgov:person:ssin</key>
14    <value>83051839468</value>
15  </samlAttribute>
16  <samlDesignator>
17    <namespace>urn:be:fgov:identification-namespace</namespace>
18    <key>urn:be:fgov:ehealth:1.0:certificateholder:person:ssin</key>
19  </samlDesignator>
20  <samlDesignator>
21    <namespace>urn:be:fgov:identification-namespace</namespace>
22    <key>urn:be:fgov:person:ssin</key>
23  </samlDesignator>
24  <samlDesignator>
25    <namespace>urn:be:fgov:certified-namespace:ehealth</namespace>
26    <key>urn:be:fgov:person:ssin:doctor:boolean</key>
27  </samlDesignator>
28  <samlDesignator>
29    <namespace>urn:be:fgov:certified-namespace:ehealth</namespace>
30    <key>urn:be:fgov:person:ssin:ehealth:1.0:doctor:nihiil1</key>
31  </samlDesignator>
32 </authenticationConfig>

```

The needed info in the above example is the SSIN, the location of the certificate and the password of the certificate. The location of the certificate can be freely chosen, but it is good practice to put it in the same folder as the example by installation:



Since EVS follows all the rules for eHealth and Vitalink, it is up to the user to make sure the proper eHealth dependencies (therapeutic relation, ...) are set in function of the wanted behaviour.



Restart EVS

EVS (and EVS-exporter) should be restarted when newly added actors will be used.

Parameterisation

The next parameters can be passed when launching EVS:

Name	Values	Meaning
rootdir	"..\exe\interaction"	Relative or absolute path to the folder that needs to be watched by EVS. This folder should contain the requested actions.
writeAsIs	true/false	<p>false: All patient data from the source Kmehrmessage will be replaced by the correspondign data of the used input patient. Since the Kmehr data model is used for this transformation, it is possible that other Kmehr structure elements are slightly changed too.</p> <p>true: The Kmehrmessage will be sent to the vault untouched. Use this when really no manipulation on the source Kmehrmessage is desired.</p>
exportAfterUpload	true/false	<p>true: Each action, excepted "export" itself, should be followed by an export.</p> <p>false: No export is needed after execution of the triggered action.</p>

validateExportAfterUpload	true/false	true: Each action should be followed by validation of the vault content. false: No validation is needed.
generateGlobalMedicationScheme	true/false	true: Each action should be followed by the generation of a global scheme visualisation PDF. false: No global scheme visualisation is needed.
generateDailyMedicationScheme	true/false	true: Each action should be followed by the generation of a daily scheme visualisation PDF. false: No daily scheme visualisation is needed. ⚠ This EVS functionality is still under development!

Example of a parameterisation:

```
start EVS.cmd
1 ..\system\vault-uploader.cmd -rootDir="..\exe\interaction" -writeAsIs=false -exportAfterUpload=true
  -validateExportAfterUpload=true -generateGlobalMedicationScheme=true -generateDailyMedicationScheme=false
```

Appendix A: Folder structure EVS 1.x.y

This paragraph gives a brief overview of the folder structure after initial installation. It can be used as reference while using and configuring the EVS.

Path							Reserved path	Reserved name	Explanation
E	VS						✗	✗	The root folder. The name and location can be freely chosen. Keep in mind that paths used in scenarios, patient and actor files are possibly impacted by changes to this!
		\co					✓	✓	Everything that defines the behaviour of EVS, configured as needed by the user.
			\actors				✓	✓	All the actors that can be used by EVS.
			\log4j				✓	✓	Settings of the log4j library. Please refer to https://logging.apache.org/log4j/2.x/manual/configuration.html for more explanation.
			\patien				✓	✓	All the patients that can be used by EVS.
		\exe					✓	✓	
			\certific				✗	✗	The certificates used in the actor configuration files.
			\exports				✗	✗	The folder where the EVS-exporter will put exported vault contents, see AppendixB:EVSexporter
			\intera				✗	✗	
				\inp			✓	✓	
					\katrien		✓	✗	
					\gp_v		✓	✗	
					an_g				
					ucht				
					\add		✓	✓	
					\empty		✓	✓	
					\export		✓	✓	
					\remove		✓	✓	
					ID				

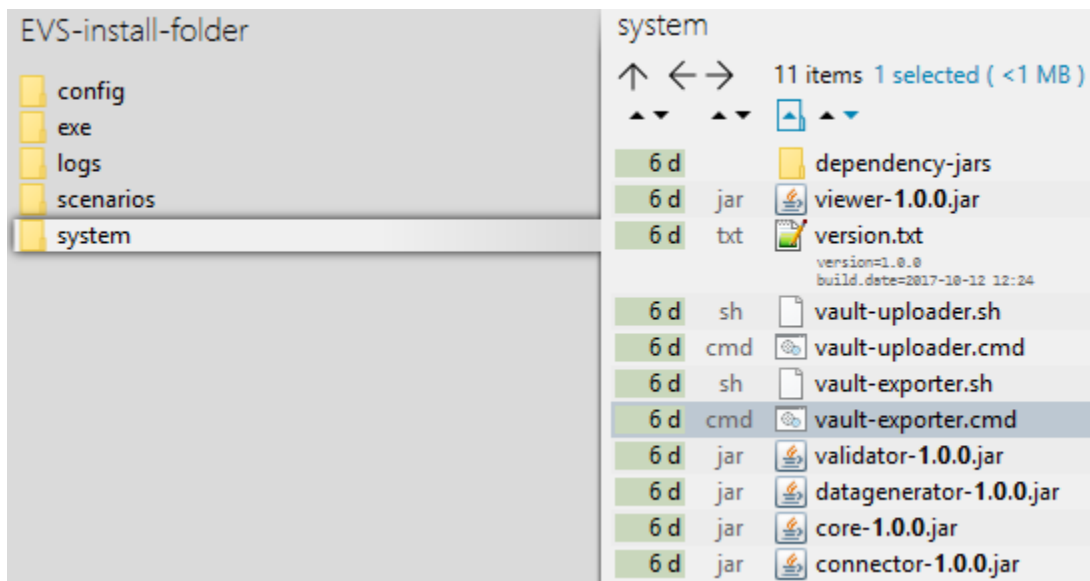
						\remove REF	✓	✓	
						\replace	✓	✓	
						\updateID	✓	✓	
						\update REF	✓	✓	
						\generat eREF	✓	✓	
						\update scheme REF	✓	✓	
				\patien t_temp late			✓	✓	
			\pro ces sed				✓	✓	
	\sc ena rios						✗	✗	
		\basic_ examp le					✗	✗	
	\sy stem						✓	✓	
		\depen dency- jars					✓	✓	

Appendix B: EVS-exporter

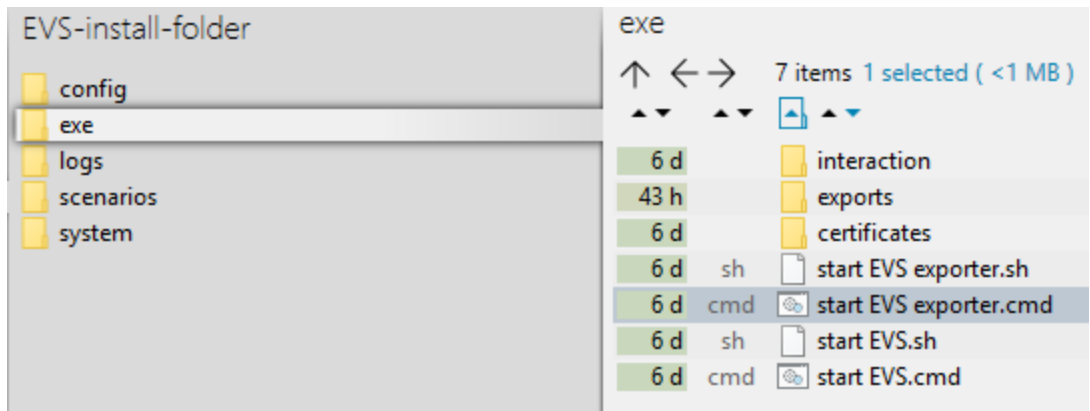
Besides the interaction provided by dropping files in the input folder, EVS offers as extended functionality the continuous monitoring of the vault contents. This functionality is provided by EVS-exporter.

Launching

EVS-exporter can be launched via the "vault-uploader.cmd" batch file:



The behaviour of EVS-exporter must be determined by passing some mandatory parameters. Instead of using the "vault-exporter.cmd" batch file, it is easier to use the example batch file "start EVS.cmd":



After initial installation, launching the EVS-exporter means that the vault contents of the patient "katrien" will be monitored.

Output

When EVS-exporter detects for the given patients a change in the vault contents, and export is executed. The export is also executed after initial launch.

The exported files are put in the next folder, with for each monitored patient a subfolder. The subfolder is automatically created when the monitoring for this patient is initially started.



The files contain the same as the files generated by EVS in the processed-folder, but the filenames differ.

For EVS-exporter, each filename exists out of:
 sv-<subject version>_nv-<node version>_<date>-<time>_<patient>_size-<nr of data entries>_<unique code>_updatedby-<actor>_<output suffix>.<output extension>

Name	Source
------	--------


Subject version	<p>"Version" from FetchDataEntriesResponse->Node as returned by the connector:</p> <p>4.4.4 FetchDataEntriesResponse</p> <p>See http://www.vitalink.be/sites/default/files/atoms/files/Safe_Cookbook_API_v4%202.pdf</p> <p>⚠ When the Kmehrmessage is returned by the Vitalink Gateway, this version can be found in folder/transaction[content/cd[text()='medicationscheme']/version.</p>
Node version	<p>"Version" from FetchDataEntriesResponse as returned by the connector.</p> <p>⚠ When the Kmehrmessage is returned by the Vitalink Gateway, this version can be found in folder/transaction[content/cd[text()='medicationscheme']/version.</p>
Date	Today.
Time	Now.
Patient	Name of the patient as defined in the EVS configuration.
Nr of data entries	Number of data entries in the vault as as returned by the connector.
Unique code	Code making the filename unique in case multiple exports where done in 1 second.
Actor	"Person" from FetchDataEntriesResponse->UpdatedBy as returned by the connector.
Output suffix	Hard coded, depending on file type. For the validation file, the number of warnings and errors and possible failure are shown.
Output extension	Hard coded, depending on file type.

When the export fails, an error file will be generated, which is the same behaviour as for the folder-triggered export action of EVS.

Parameterisation

The next parameters can be passed when launching EVS-exporter:

Name	Values	Meaning
transactionType	"medication-scheme"	This parameter is for future use. For the moment only 1 transaction type is supported.
patients	name(s) as defined in EVS configuration, comma separated	This(these) is(are) the patient(s) that will be monitored and whose vault content(s) will be exported.
actor	name as defined in EVS configuration	This is the actor that will be used for exporting.

exportDir	default "..\exe\exports"	This is the path where the output files will be generated. This location can be freely chosen.
validate	true false	true: Each export should be followed by validation of the vault content. false: No validation is needed.
generateGlobalMedicationScheme	true false	true: Each action should be followed by the generation of a global scheme visualisation PDF. false: No global scheme visualisation is needed.
generateDailyMedicationScheme	true false	true: Each action should be followed by the generation of a daily scheme visualisation PDF. false: No daily scheme visualisation is needed.  This EVS functionality is still under development!

Example of a parameterisation:

```
start EVS exporter.cmd
1 | ..\system\vault-exporter.cmd -transactionType=medication-scheme -patients=katrien -actor=gp_van_gucht -exportDir=
  | "..\exe\exports" -validate=true -generateGlobalMedicationScheme=true -generateDailyMedicationScheme=false
```