EVS_2.x.y_Manual

Introduction

EVS is an evolution of EVSc, using gateway-integration instead of Vitalink connector-integration. It allows the manipulation of 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.

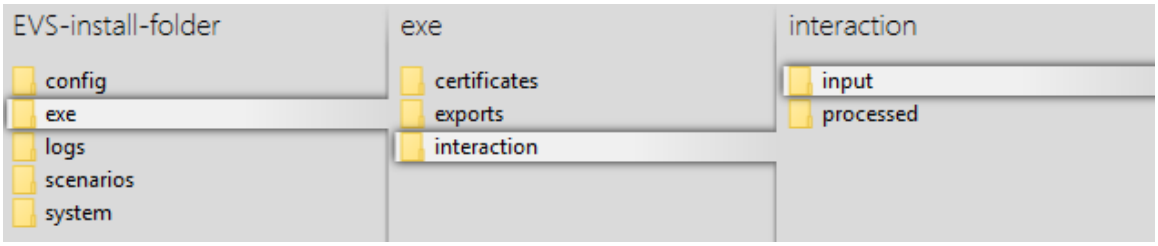
This manual describes EVS v2.1.0.

Functionality

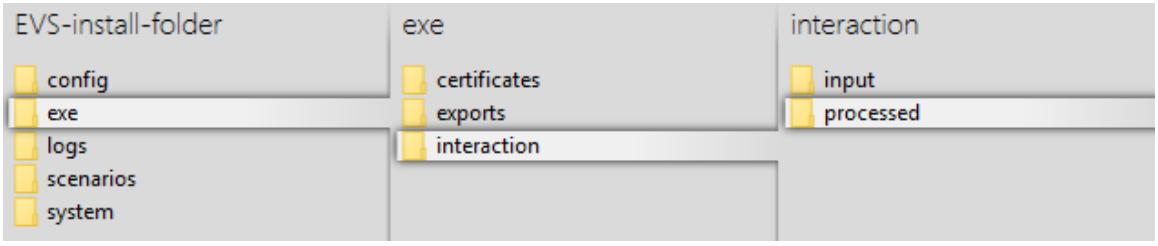
General

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

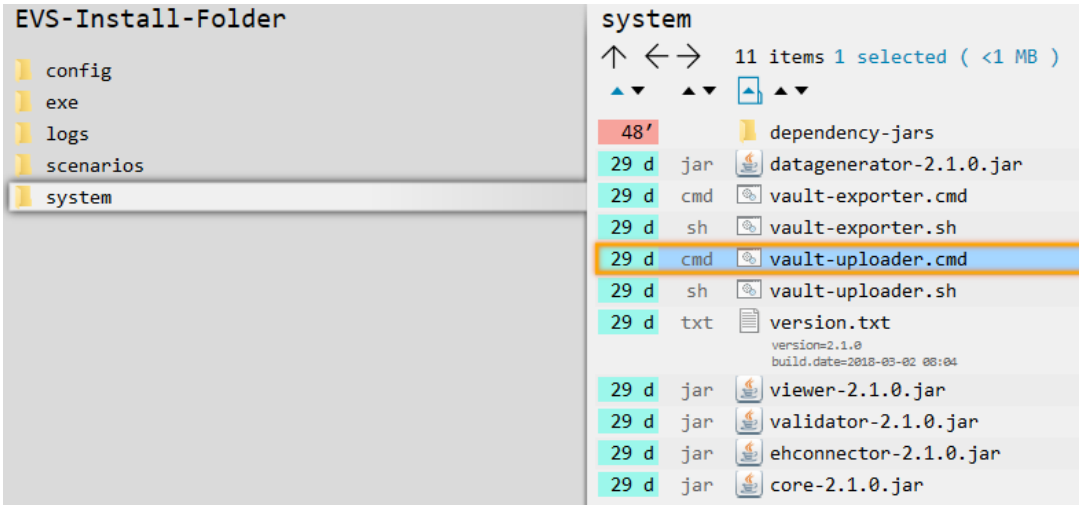
These actions are triggered by dropping input files in (subfolders of) the input-folder:



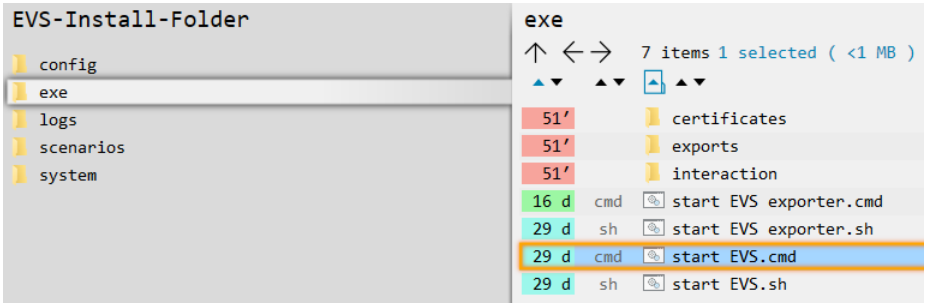
EVS watches these folder(s), executes the action(s) and generates output in the processed-folder:



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":



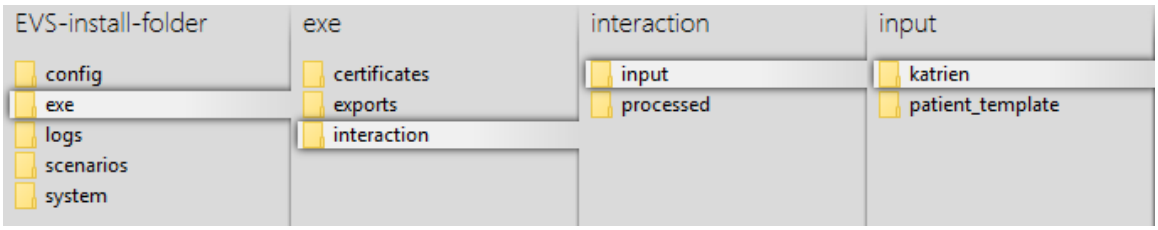
This batch file contains parameter values for a standard behaviour. How the parameters change the behaviour can be found in the paragraph Parameterisation.

Input-folder

Which patient?

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

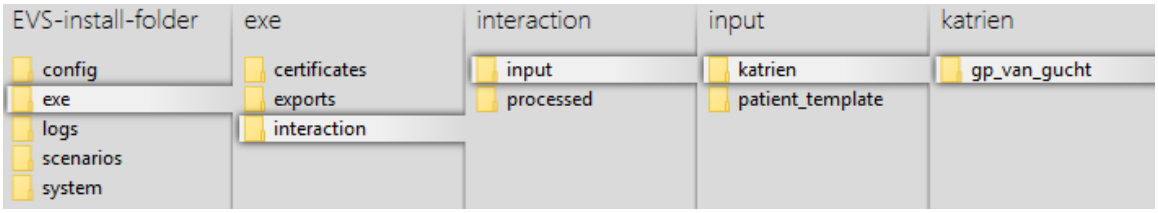
After initial installation, 1 patient named "katrien" is available:



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 MSE and TS transactions.

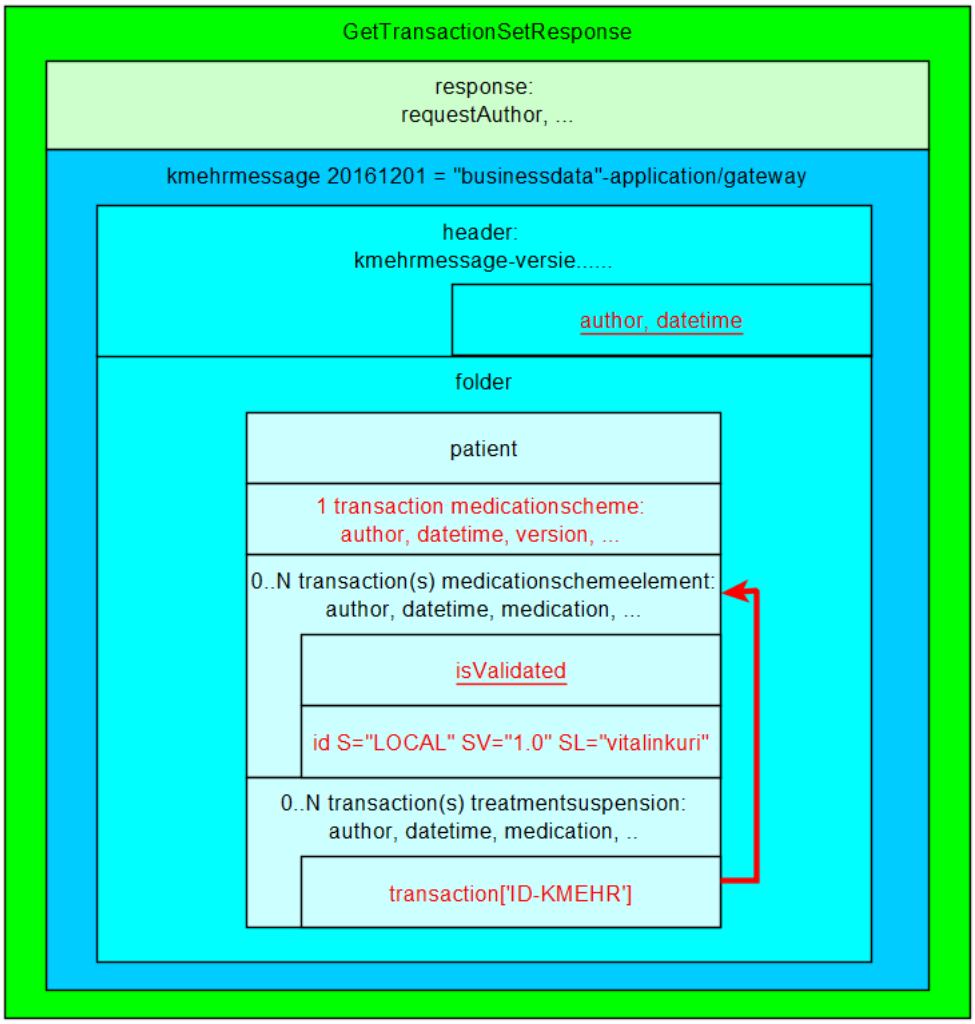
> ⚠ All MS transactions are ignored as input.

What is a Kmehrmessage?

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

EVS will work with Kmehrmessages of Kmehr-standard 20120401 and Kmehr-standard 20161201 as input.

All extra data needed for the communication with the gateway will be generated by the EVS. As input the data as depicted in the image below will be used:



How is a "medication" identified?

For some actions, typically removing and updating "medications", the medication that should be changed needs to have an identification. Those medications are expressed as MSE transactions. These MSE transactions are identified by using an EVS reference.

⚠ TS transactions can not be updated and should not contain EVS references! They can only be added or removed.

Identification by EVS reference

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

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.

The REF must contain a minimum of 3 characters and can contain up to 512 characters, so both ===EVSREF:123=== and ===EVSREF:A123456789B123456789=== are considered valid REFs.

If multiple EVS REFs have been given for 1 MSE transaction, EVS will not execute the action and will raise an error.

```
<kmehrmessage ...>
    <header>
        <standard>
            <cd S="CD-STANDARD" SV="1.4">20161201</cd>
        </standard>
        ...
    </header>
    <folder>
        <id S="ID-KMEHR" SV="1.0">1</id>
        <patient>
            <id S="ID-PATIENT" SV="1.0">83051839468</id>
            <firstname>Katrien</firstname>
            <familyname>Van Gucht</familyname>
            <sex>
                <cd S="CD-SEX" SV="1.0">female</cd>
            </sex>
        </patient>
        <transaction>
            ...
            <cd S="CD-TRANSACTION" SV="1.4">medicationschemeelement</cd>
            ...
            <item>
                <id S="ID-KMEHR" SV="1.0">2</id>
                <cd S="CD-ITEM" SV="1.4">medication</cd>
                <content>
                    <medicinalproduct>
                        <intendedcd S="CD-DRUG-CNK" SV="2010-07">2933901</intendedcd>
                        <intendedname>Dafalgan bruistab 20x 500mg</intendedname>
                    </medicinalproduct>
                </content>
                ...
                <instructionforpatient L="nl">===EVSREF:901=== Actual instruction for patient.</instructionforpatient>
            </item>
        </transaction>
    </folder>
</kmehrmessage>
```

Which actions?

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

Action "add"

This action will add a transaction to the vault for all transactions found in each Kmehrmessage found in all dropped files. If one of the MSE transactions within a kmehrmessage does not have an EVS REF yet, an EVS REF will be generated.

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

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 "generateREF"

This action will generate an EVS REF for each MSE transaction currently in the vault.

> The 'old' EVS, with Vitalink connector-integration, added the references to the input file, followed by putting this in the vault.

If an EVS REF exists already in the MSE transaction, no new EVS REF will be generated.

If no EVS REF exists, the new EVS REF is put in the instructionforpatient field.

Action "replace"

This action will replace the contents of the vault by all the transactions 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 transactions of the last input file! If one of the MSE transactions within a file does not have an EVS REF yet, this will be generated.

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

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 MSE transaction with EVS REF not yet existing in the vault will be added to the vault, together with all TS transactions which are linked with this MSE transaction
- an MSE transaction with EVS REF already existing in the vault will cause an update if any difference between the input-transaction and vault-transaction is found
- all MSE transactions without corresponding EVS REF in the input file will be removed from the vault

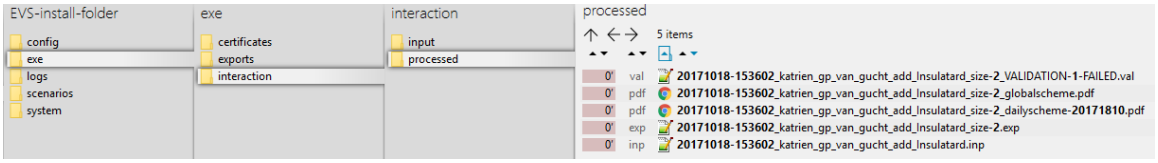
If the input-file contains MSE transactions with EVS REFs that are not unique, the action will not be executed and an error will be thrown.

> **Empty**
>
> The 'old' EVS, with Vitalink connector-integration, offered an action 'empty'. EVS doesn't offer this action anymore. Now, emptying the vault can be done by dropping an empty file for the actions "updateschemeREF" or "replace".

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:

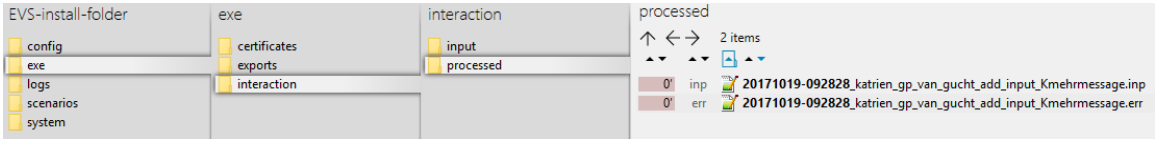


Each filename exists out of:

<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. blocked URL 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:



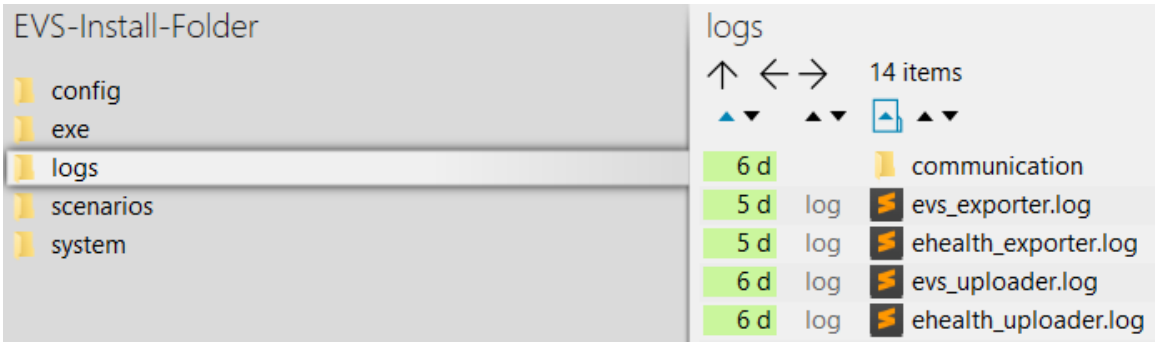
Name	Output suffix	Extension	Description	Remarks
Error file	-	.err	The report containing the error.	The content of the error file will identify the problem.

Logs-folder

Root

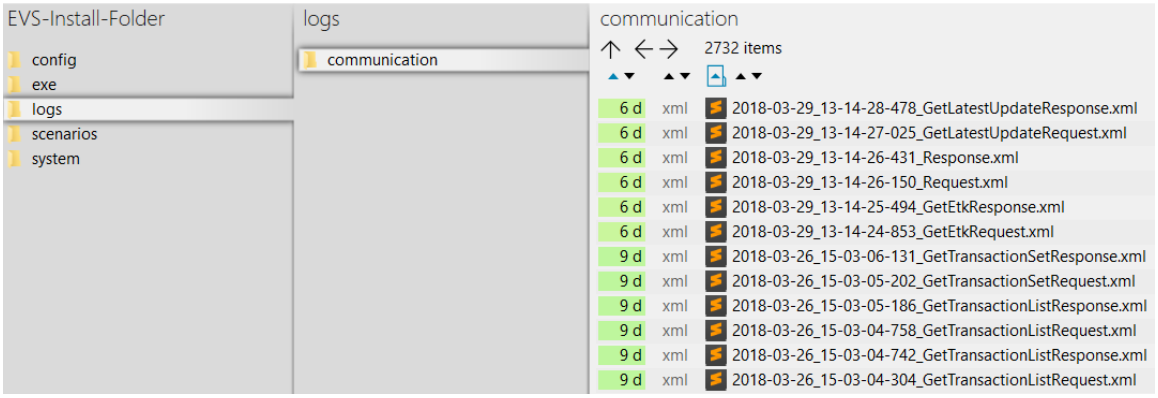
This folder contains:

- the eHealth technical connector logs for EVS: ehealth_uploader*.log
- the eHealth technical connector logs for EVS-exporter: ehealth_exporter*.log
- the proprietary EVS log for EVS: evs_uploader.log
- the proprietary EVS log for EVS-exporter: evs_exporter.log
- a folder 'communication'



Communicaton-folder

This folder contains all the requests and responses done by EVS when communicating with the gateway.

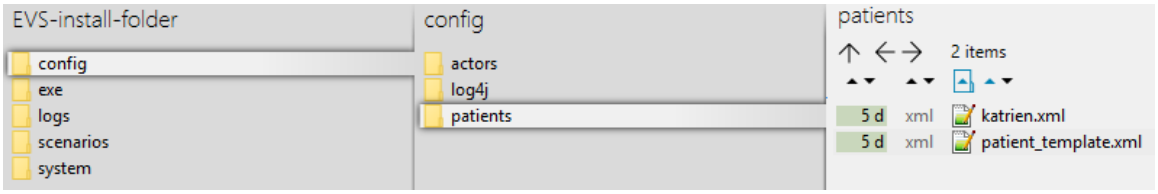


Configuration

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 performed.

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

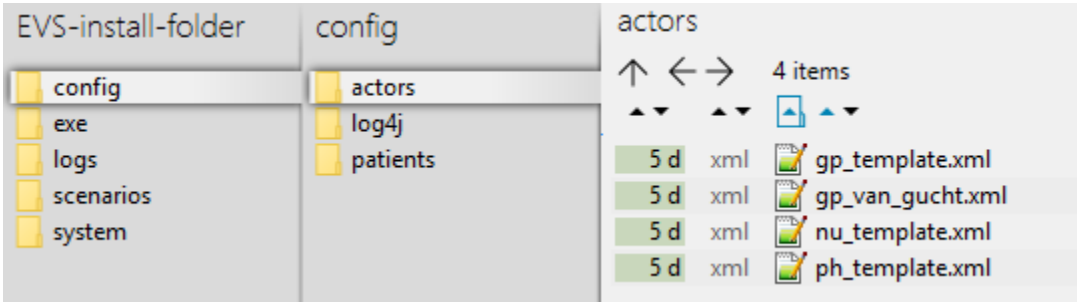
```xml
<patient>
    <id>93051822361</id>
    <firstName>Example Patient Firstname</firstName>
    <lastName>Example Patient Lastname</lastName>
    <gender>female</gender>
    <birthDate>1993-05-18</birthDate>
</patient>
```

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:



> The 'old' EVS, with Vitalink connector-integration, used another syntax in this files. EVS is not compatible with this old format!

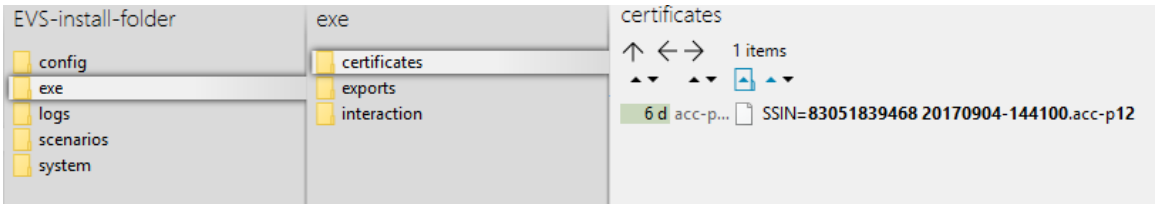
After initial installation, some actor config files are already present. All can be used as example (copy-paste) to add extra actors. The name of the file, without the extension, needs to be used to identify by which actor the action needs to be performed.

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:

```
<authenticationConfiguration>
    <evs>
        <type>fallback</type>
        <certificates>
            <certificate>
                <type>identification</type>
                <path>..\exe\certificates\[NAME_OF_ACC_P12_CERTIFICATE_FILE]</path>
                <password>[CERTIFICATE_PASSWORD]</password>
            </certificate>
        </certificates>
    </evs>
    <ehealth>
        <entry>user.inss=[SSIN]</entry>
        <entry>user.nihii=[NIHII11]</entry>
        <entry>user.firstname=[FIRSTNAME]</entry>
        <entry>user.lastname=[LASTNAME]</entry>
        <entry>careprovider.inss=${user.inss}</entry>
        <entry>careprovider.nihii=${user.nihii}</entry>
        <entry>careprovider.firstname=${user.firstname}</entry>
        <entry>careprovider.lastname=${user.lastname}</entry>
        <entry>main.kmehr.quality=persphysician</entry>
        <entry>kmehr.default.hcpartylist=identifier</entry>
        <entry>kmehr.hubservicev3.hcpartylist=identifier,software</entry>
        <entry>kmehr.hubservicev3.identifier.id.inss.value=${user.inss}</entry>
        <entry>kmehr.hubservicev3.identifier.id.inss.sv=1.0</entry>
        <entry>kmehr.hubservicev3.identifier.id.idhcparty.value=${user.nihii}</entry>
        <entry>kmehr.hubservicev3.identifier.id.idhcparty.sv=1.0</entry>
        <entry>kmehr.hubservicev3.identifier.cd.cdhcparty.value=persphysician</entry>
        <entry>kmehr.hubservicev3.identifier.cd.cdhcparty.sv=1.1</entry>
        <entry>kmehr.hubservicev3.identifier.firstname=${user.firstname}</entry>
        <entry>kmehr.hubservicev3.identifier.lastname=${user.lastname}</entry>
        <entry>kmehr.hubservicev3.identifier.address.cd.sv=1.0</entry>
        <entry>kmehr.hubservicev3.identifier.address.cd.value=</entry>
        <entry>kmehr.hubservicev3.identifier.address.cd.schemes=cdaddress</entry>
        <entry>kmehr.hubservicev3.identifier.address.nis=24062</entry>
        <entry>sessionmanager.samlattribute.1=urn:be:fgov:identification-namespace,urn:be:fgov:ehealth:1.0:certificateholder:person:ssin,${user.inss}</entry>
        <entry>sessionmanager.samlattribute.2=urn:be:fgov:identification-namespace,urn:be:fgov:person:ssin,${user.inss}</entry>
        <entry>sessionmanager.samlattributedesignator.1=urn:be:fgov:identification-namespace,urn:be:fgov:ehealth:1.0:certificateholder:person:ssin</entry>
        <entry>sessionmanager.samlattributedesignator.2=urn:be:fgov:identification-namespace,urn:be:fgov:person:ssin</entry>
        <entry>sessionmanager.samlattributedesignator.3=urn:be:fgov:certified-namespace:ehealth,urn:be:fgov:person:ssin:doctor:boolean</entry>
        <entry>sessionmanager.samlattributedesignator.4=urn:be:fgov:certified-namespace:ehealth,urn:be:fgov:person:ssin:ehealth:1.0:doctor:nihii11</entry>
    </ehealth>
</authenticationConfiguration>
```

The needed info in the above template is the certificate name, the password of the certificate, the SSIN, the NIHII number and the actor's name. 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	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. true: The Kmehrmessage will be sent to the vault untouched. Use this when really no manipulation on the source Kmehrmessage is desired.
exportAfterUpload	true|false	true: Each action, excepted "export" itself, should be followed by an export. false: No export is needed after execution of the triggered action.

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. blocked URL This EVS functionality is still under development!
dailyMedicationSchemeDate	date ("yyyy-MM-dd")	If no date has been given, it will generate a daily medication scheme of the current date. If a date has been given, it will generate a daily medication scheme of the given date.
startTransactionId	number	This number will be the number for the first transaction within the kmehrmessage of a putTransactionSetRequest in the context of a medicationscheme.

Example of a parameterisation:

start EVS.cmd

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

Appendix A: Folder structure EVS 2.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 EVS.

Path							Reserved path	Reserved name	Explanation
EVS							blocked URL	blocked URL	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!
	\config						blocked URL	blocked URL	Everything that defines the behaviour of EVS, configured as needed by the user.
		\actors					blocked URL	blocked URL	All the actors that can be used by EVS.
		\log4j					blocked URL	blocked URL	Settings of the log4j library. Please refer to https://logging.apache.org/log4j/2.x/manual/configuration.html for more explanation.
		\patients					blocked URL	blocked URL	All the patients that can be used by EVS.
	\exe						blocked URL	blocked URL	
		\certificates					blocked URL	blocked URL	The certificates used in the actor configuration files.
		\exports					blocked URL	blocked URL	The folder were the EVS-exporter will put exported vault contents, see AppendixB:EVSexporter
		\interaction					blocked URL	blocked URL	
			\input				blocked URL	blocked URL	
				\katrien			blocked URL	blocked URL	
					\gp_van_gucht		blocked URL	blocked URL	

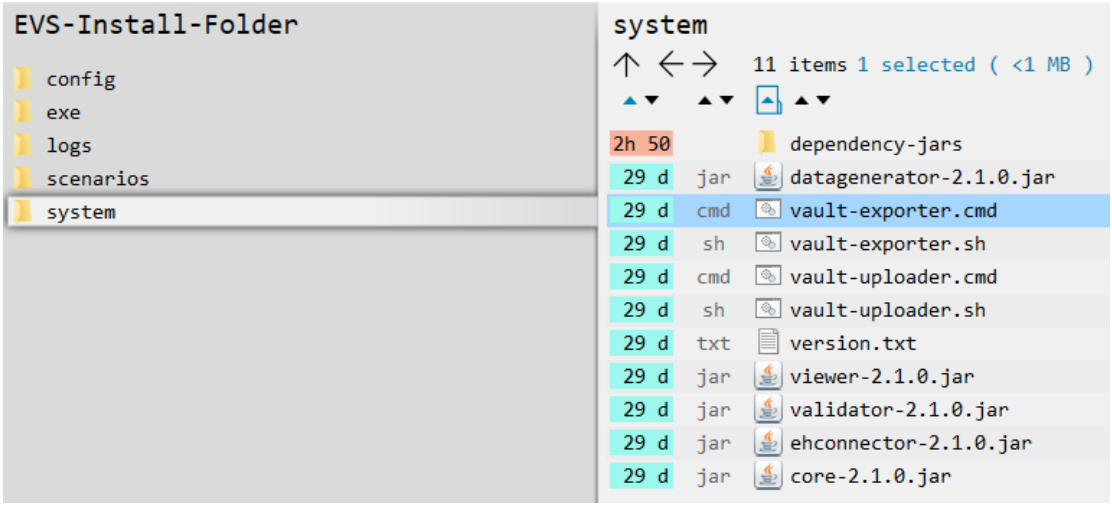
						\add	blocked URL	blocked URL	
						\export	blocked URL	blocked URL	
						\generateREF	blocked URL	blocked URL	
						\replace	blocked URL	blocked URL	
						\updateschemeREF	blocked URL	blocked URL	
				\patient_template			blocked URL	blocked URL	
			\processed				blocked URL	blocked URL	
	\logs						blocked URL	blocked URL	Can be configured through the log4j settings.
		\communication					blocked URL	blocked URL	
	\scenarios						blocked URL	blocked URL	
		\basic_example					blocked URL	blocked URL	
	\system						blocked URL	blocked URL	
		\dependency-jars					blocked URL	blocked URL	

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-exporter.cmd" batch file:



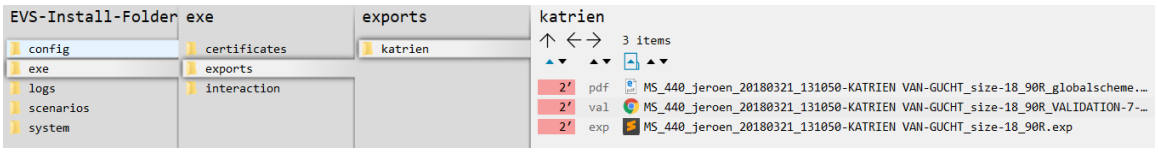
The behaviour of EVS-exporter must be determined by passing some mandatory parameters. Instead of using the "vault-exporter.cmd" batch file, it is easier to use the example batch file "start EVS.cmd":

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, an 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:
MS_<version>_<patient>_<date>_<time>-<author>_size-<nr of MSE transactions>_<unique code>_<output suffix>.<output extension>

Name	Source
Version	"Version" from the MS transaction. In case of an empty medicationscheme, the "version" is derived from the getLatestUpdate method.

Patient	Name of the patient as defined in the EVS configuration.
Date	Date of the latest update derived from the MS transaction.
Time	Time of the latest update derived from the MS transaction.
Author	"Author" of the latest update, derived from the MS transaction->UpdatedBy as returned by the gateway.
Nr of MSE transactions	The amount of MSE transactions in the vault.
Unique code	Code making the filename unique in case an export exists already.
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	"medicationscheme"	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. blocked URL This EVS functionality is still under development!
dailyMedicationSchemeDate	date("yyyy-MM-dd")	If no date has been given, it will generate a daily medication scheme of the current date. If a date has been given, it will generate a daily medication scheme of the given date.

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
```