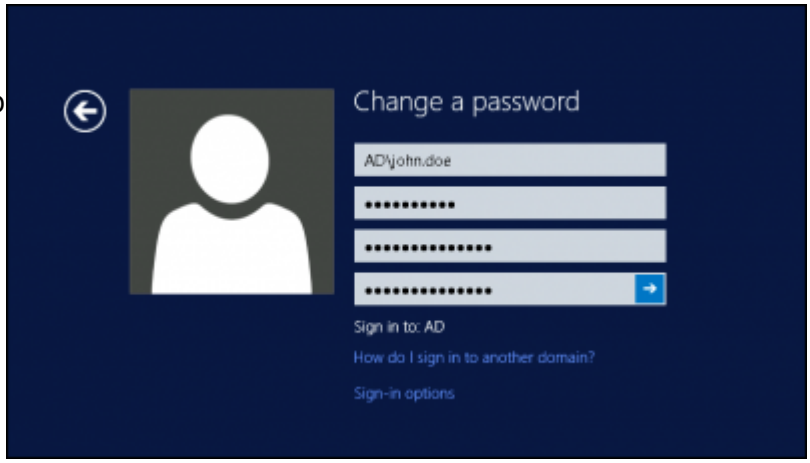
synchronization, password, filter, passwordfilter, active, directory, AD, dll

Password filter - dll library

What purpose does it serve to?



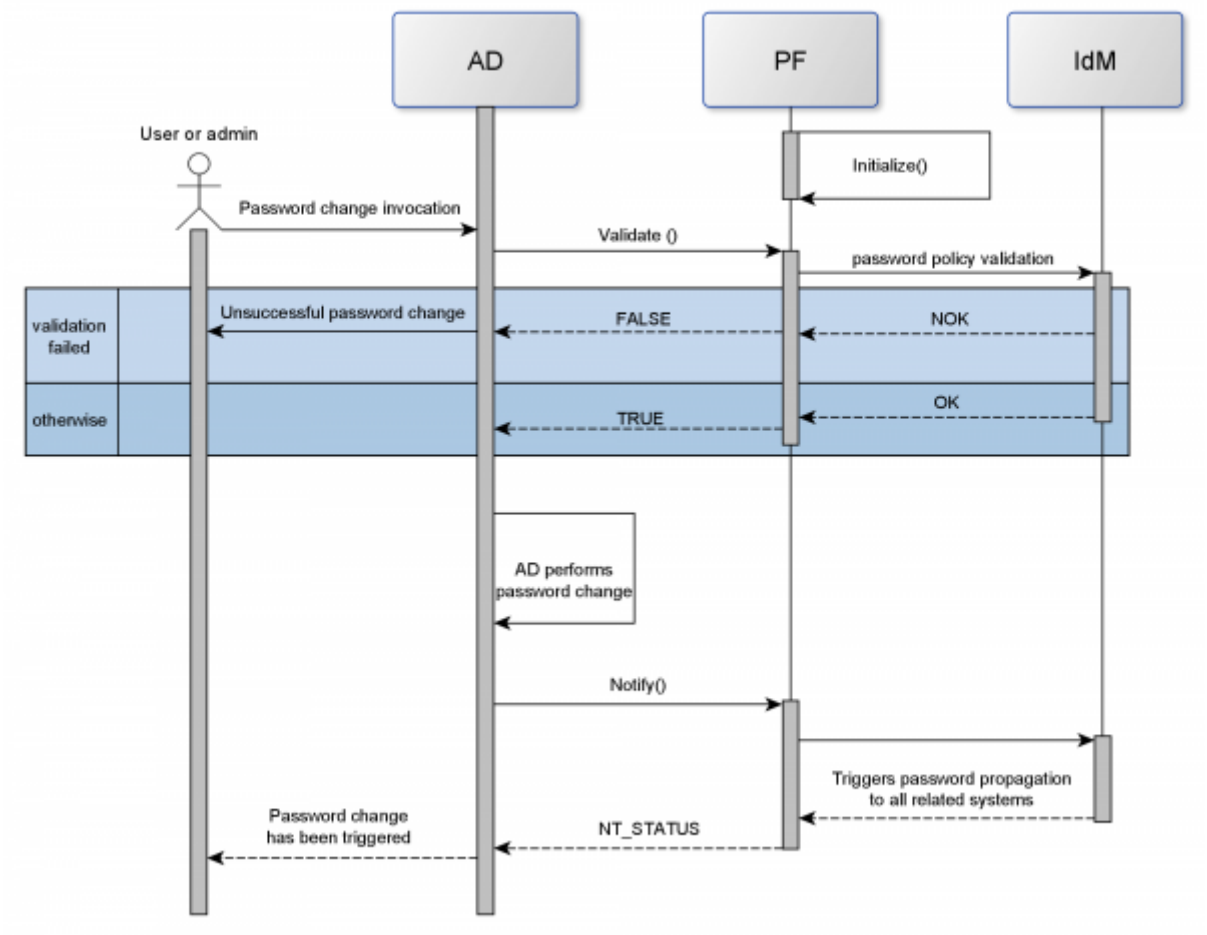
Password filter is a useful Active Directory extension which provides a manner for common MS Windows users to change their password by a standard way such as (ctrl + alt + del) and have this new password propagated to IdM. Depending on the IdM configuration, the new password may be propagated into other related systems and thus maintain unified password on them. An integral part is also validation, if the new password meets all password policies.

The current version isPasswordFilterDll-1.1.0_x64.zip Please check the changelog for important changes.

Password filter usage

Basic function

Password filter is, in general, a dynamic library (dll) which is loaded by Local Security Authentication Server (LSASS) process responsible for user authentication in MS Windows. It allows to intercept the process of password change and insert there some logic. Typical usage scenario is with MS Active Directory (AD) to check the new password, whether it meets requirements of user defined password policies before the password is passed to AD. It is also able to provide a notification after successful password change. This password filter implements password policy validation by querying IdM and following IdM notification if password has been successfully changed in AD. Password filter dll (PF) implements three methods serving as the interface with Local Security Authentication Server (LSASS) process. These methods implement all the logic providing communication with IdM and deduction of responses to LSASS. PF communicates with IdM via REST api. It supports secured communication and performs server certificate validation. This validation can be turned off which is recommended for debug or testing purpose only. User is also allowed to set more than one destination URL which serve as a backup in case of primary target unavailability. The number of connection attempts is also configurable. It is applied on every target URL independently. Password filter authenticates to IdM by provided token. The body of the REST request sent to IdM contains `AccountName` and new `password` provided by AD, `systemId` (see Configuration section), `SessionId` (see Logging section) and PF `version`. The following picture shows the whole process of PF calling.



Password Filter methods

InitializeChangeNotify (aka **Initialize**) method This method is called only once at the very beginning of PF start and it can inform by its return value whether it has been properly initialized. It returns always true - means correctly initialized - in our implementation, because PF is able to be reinitialized during runtime (see Configuration paragraph). In case of initialization failure this fact is logged and PF switches into inactive state e.i. it makes no policy validation nor notification and allows all password changes.

PasswordFilter (aka **Validate**) method Validate method is one of PF pillars. Any time a password change happens, it is invoked and queries IdM whether the new password for given AD account meets password policies of all related systems. The response from IdM is transformed into **PF result** which may be: **TRUE** ~ change **allowed** or **FALSE** ~ change **rejected**. Those values are immediately returned from Validate method. The third option, **TRY_AGAIN**, is to make another query attempt. **PF result** is based on returned HTTP status code and IdM specific response if present. The table below shows possible results.

Transformation of an IdM response to the PF result

HTTP status	has IdM result	PF result	Note
200	n/a	TRUE	Password passed policy validation

HTTP status	has IdM result	PF result	Note
400	PASSWORD_DOES_NOT_MEET_POLICY	FALSE	Password did not pass policy validation
	n/a	FALSE	None or unknown IdM result
404	PASSWORD_FILTER_SYSTEM_NOT_FOUND, PASSWORD_FILTER_IDENTITY_NOT_FOUND, PASSWORD_FILTER_IDENTITY_NOT_FOUND	TRUE	Identity is not managed by IdM or PF is not set to be used
	n/a	FALSE	None or unknown IdM result
423	n/a	TRUE	PF is disabled in IdM and must not block password change
408	n/a	TRY_AGAIN	Client time out. A new attempt will be done
504	n/a	TRY_AGAIN	Server/proxy time out. A new attempt will be done
others	n/a	FALSE	All other results

The validation request from PF to IdM may also end without any particular HTTP response. This typically happens when the PF gives up waiting for response, when IdM server certificate validation fails, the destination URL is unavailable and so on. Common reaction is to try it again up to the number of configured attempts to every configured base URLs. If the **last result** is the **certificate validation failure**, the **PF result** is set to **FALSE** ~ change **rejected**. In all other cases **PF result** depends on configuration item `allowChangeByDefault`.

PasswordChangeNotify (aka **Notify**) method Is the second cornerstone of PF. It is called after successful password change in AD and it notifies IdM about this. Notify method does not provide any specific result. HTTP results received from IdM is logged only. If there is no HTTP response notification is repeated until all configured attempts are used up.

Configuration

Password filter (PF) is controlled by configuration file. By default the configuration file is searched at the location `c:\CzechIdM\PasswordFilter\etc\PasswordFilterConfig.cfg`. The file location can be specified by setting of the environmental variable `BCV_PWF_CONFIG_FILE_PATH` with the full path to the file. When the configuration file is read it is first searched at the location specified by `BCV_PWF_CONFIG_FILE_PATH`. If variable not set or the path to the file does not exist then the default location is used. Configuration file uses JSON structure and has to be in UTF-8 character encoding format without BOM. It is a mandatory part of password filter library and if not found or an error occurs during its parsing, the password filter is not able to work properly and switches to the **inactive state**. It means that it acts as no password filter was installed i.e. it allows all password changes without policy validation and performs no IdM notification. All configuration properties are mandatory and incomplete configuration file is invalid. Configuration file is read during PF initialization phase and then checked every 3 seconds whether it has changed. If so, it is reloaded. If the configuration file is missing or has wrong name etc., the monitoring system stops working. The only current solution is to provide config file and restart AD. Configuration file has to contain following items:

- `restBaseUrl` - is an array of base addresses with common parts of REST API; if connection is not successful next address is tried
- `restCheckUrl` - specifies rest API for invocation of policies validation in IdM; it usually needs not to be changed
- `restNotifyUrl` - specifies rest API for IdM notification that password has been changed; it usually needs not to be changed
- `connectionAttempts` - determines number of attempts of connection to every item in `restBaseUrl` array
- `connectionTimeoutMs` - setting of timeout how long PF waits for response from IdM before giving up; the value is in milliseconds
- `token` - contains token string used for authentication in IdM
- `ignoreCertificate` - a flag allowing to disable certificate validation; it should be used for test/debug purpose only
- `systemId` - is an identifier specifying what system PF resides on; it has to correspond to setting in IdM
- `allowChangeByDefault` - determines whether PF allows or rejects password change in cases of falling to default setting
- `logLevel` - controls log verbosity; allowed settings are (case insensitive): debug, info, warning, error; debug provides the most logs, error the least
- ~~`forbiddenInitChars`~~ (until v1.1.0) - AD accounts starting with any of listed characters will be allowed password change without policy validation or IdM notification; **do not use any separator**
- `skippedAccPrefix` (since v1.1.0) - an array of reserved strings; replaces the item `forbiddenInitChars`; if user account starts with any of these prefixes the password change is allowed without policy validation and IdM notification
- `passwordFilterEnabled` - enables or disables password filter; if disabled it allows all password changes and performs no policy validation or notification in IdM

Configuration file may look like this:

```
{
  "restBaseUrl" :
["https://CHANGE_IT_TO_CZECHIDM_SERVER_URL/idm/api/v1/systems/password-filte
r"],
  "restCheckUrl" : "validate",
  "restNotifyUrl" : "change",
  "connectionAttempts" : 2,
  "connectionTimeoutMs" : 30000,
  "token" : "CHANGE_IT_TO_VALID_AUTHENTICATION_TOKEN",
  "ignoreCertificate" :false,
  "systemId" : "CHANGE_IT_TO_CORRECT_SYSTEM_NAME",
  "allowChangeByDefault" : false,
  "logLevel" : "info",
  "skippedAccPrefix": ["testPrefix","testPREFIX", "testPrefix2"],
  "passwordFilterEnabled": true
}
```

Logging

Password filter (PF) is currently able to write logs into a logging file only. The default logging file is located at `c:\CzechIdM\PasswordFilter\log\PasswordFilterLog.log`. The log folder can be user specified by environmental variable `BCV_PWF_LOG_FILE_FOLDER`. The name of the file remains the same `PasswordFilterLog.log`. PF logging file is rotated by its size at the moment it reaches 10MB. Logging library which takes care of logging maintains always one file where PF currently logs to and one previous file with the name `PasswordFilterLog.log.1`. Logging verbosity level can be set via `logLevel` in the configuration file. Setting of this item is case insensitive and accepts following values:

* **debug** - the most verbose mode * **info** - provides errors, warnings and common service logs * **warning** - errors and warnings only * **error** - messages marked as error only

If set log level is wrong or misspelled the default one is used. The default log level is **debug** Each line of the logging file contains one log. The first part is a timestamp followed by log level and `SessionId`. `SessionId` is a unique number generated at the beginning of PasswordFilter and PasswordChangeNotify methods and is maintained until the end of such method. `SessionId` helps to pair log records of events in PF dll and IdM. The last portion is the message. The example below shows separated logs of usual phases the PF walks through.

```
##### the phase of password filter initialization
17-06-2021 14:21:07,665 INFO    SessionId: 0000000000 Logging initialized
17-06-2021 14:21:07,675 INFO    SessionId: 0000000000 Configuration file
will be loaded from the path
c:/CzechIdM/PasswordFilter/etc/PasswordFilterConfig.cfg
17-06-2021 14:21:07,675 INFO    SessionId: 0000000000 Configuration has been
successfully initialized from the file:
"c:/CzechIdM/PasswordFilter/etc/PasswordFilterConfig.cfg"
17-06-2021 14:21:07,676 DEBUG   SessionId: 0000000000 restBaseUrl:
https://czechidm-test.bcv/idm/api/v1/systems/password-filter
17-06-2021 14:21:07,676 DEBUG   SessionId: 0000000000 restCheckUrl: validate
17-06-2021 14:21:07,676 DEBUG   SessionId: 0000000000 restNotifyUrl: change
17-06-2021 14:21:07,676 DEBUG   SessionId: 0000000000 connectionAttempts: 2
17-06-2021 14:21:07,676 DEBUG   SessionId: 0000000000 connectionTimeoutMs:
30000
17-06-2021 14:21:07,676 DEBUG   SessionId: 0000000000 token: EXCLUDED FROM
LOG
17-06-2021 14:21:07,676 DEBUG   SessionId: 0000000000 ignoreCertificate:
true
17-06-2021 14:21:07,676 DEBUG   SessionId: 0000000000 systemId: AD 190
17-06-2021 14:21:07,676 DEBUG   SessionId: 0000000000 allowChangeByDefault:
false
17-06-2021 14:21:07,676 DEBUG   SessionId: 0000000000 logLevel: debug
17-06-2021 14:21:07,676 DEBUG   SessionId: 0000000000 skippedAccPrefix:
abcdefg
17-06-2021 14:21:07,676 DEBUG   SessionId: 0000000000 skippedAccPrefix:
skippedPrefix
17-06-2021 14:21:07,676 DEBUG   SessionId: 0000000000 passwordFilterEnabled:
true
```

```
17-06-2021 14:21:07,676 INFO    SessionId: 0000000000 Configuration
monitoring thread successfully started
17-06-2021 14:21:07,676 INFO    SessionId: 0000000000 Inside Dll main -
DLL_PROCESS_ATTACH: PID 580
17-06-2021 14:21:07,676 DEBUG   SessionId: 0000000000 Calling
InitializeChangeNotify

###### successful calling of IdM to perform password policy validation
17-06-2021 15:12:02,340 DEBUG   SessionId: 3145420471 Calling PasswordFilter
- password policy validation
17-06-2021 15:12:02,340 INFO    SessionId: 3145420471 Account: jtester -
Starting password policy validation
17-06-2021 15:12:02,635 INFO    SessionId: 3145420471 Password policy
validation passed
17-06-2021 15:12:02,636 INFO    SessionId: 3145420471 Account: jtester -
Password policy validation completed with the result: APPROVED

###### successful IdM notification
17-06-2021 15:12:02,694 DEBUG   SessionId: 0386167646 Calling
PasswordChangeNotify
17-06-2021 15:12:02,694 INFO    SessionId: 0386167646 Account: jtester -
Notifying IdM about password change
17-06-2021 15:12:03,029 INFO    SessionId: 0386167646 Account: jtester - IdM
notification is successful
```

—

Password filter deployment

Password filter distribution and installation

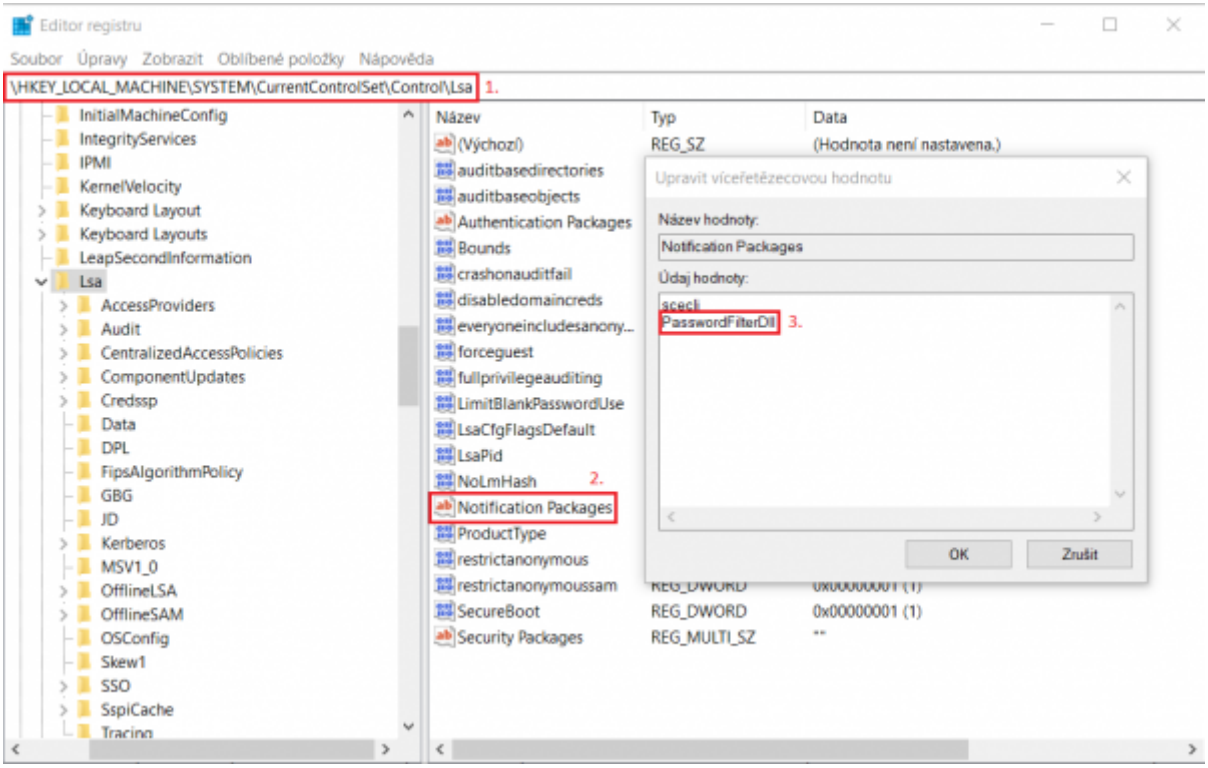
Before starting installation make sure there is a dedicated user in IdM system with sufficient permissions for password filter to use!

Do not use a user token with **"Application administration"** permission for password filter. Not even for testing or development purpose. It is both a security risk and it affects some IdM functions such as minimum password validity.

When you change permissions of the user later you will need to **generate a new toke**n and **update the config file** on all DCs.

Password filter is distributed as a zip package which contains Password filter dll and all its dependencies. In order to deploy password filter follow these steps. Password filter has been tested on `Windows Server 2012 R2` and `Windows Server 2016`.

1. Unpack the supplied archive
2. Install MS MSVC Redistributable package by running `VC_redist.exe` in `bin` directory
3. Create a directory structure `c:\CzechIdM\PasswordFilter\etc` in the local filesystem for PF configuration file
4. Copy a configuration file to the location created during the step 3). An example is situated in `share\CzechIdM\PasswordFilter\etc\PasswordFilterConfig.cfg`. The file needs to have UTF-8 encoding. When editing in `notepad.exe` the encoding is usually changed. Set all necessary parts, especially:
    1. restBaseUrl - Url to IdM server.
    2. token - Token represents credentials for authentication of PF when it connects to IdM. Ask IdM admin for providing the token.
    3. systemId - The name of the connected system in IdM, the password filter is used on.
5. Copy all `\*.dll` files located in `lib\` directory and from all its sub-directories directly into `c:\Windows\System32\` (do not follow original directory structure)
6. Add (in `regedit` or `regedit.exe`) the password filter name (by default **PasswordFilterDll**) to the register key `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Notification Packages` (see the picture below)
7. Restart server



The structure of the distribution package is as follows:

```
PasswordFilter_VERSION_ARCH
|__bin
|  |__VC_redist.exe
|__doc
|  |__manual.txt
|__lib
|  |__cpprestsdk
|  |  |__copyright.txt
```

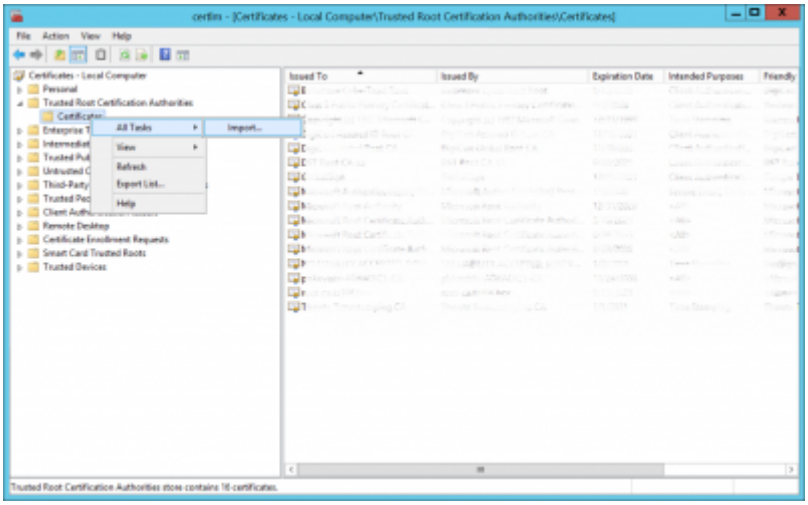
```
|   |   |__cpprest_2_10.dll
|   |   |__ref.txt
|   |__log4cpp
|   |   |__copyright.txt
|   |   |__orocos-log4cpp.dll
|   |   |__ref.txt
|   |__passwordFilter
|   |   |__LICENSE.txt
|   |   |__PasswordFilterDll.dll
|__share
|   |   |__CzechIdM
|   |   |   |__PasswordFilter
|   |   |   |   |__etc
|   |   |   |   |   |__PasswordFilterConfig.cfg
|__readme.txt
```

Password filter redeployment

If a password filter dll file already exists in the `c:\Windows\System32\` and needs to be rewritten, the record in the register key, made during the step 6 of the installation process, has to be removed and server restarted. Replace the old dll file with the new one and create the register record (according to the step 6) again. Server restart is also necessary.

Secured connection setting

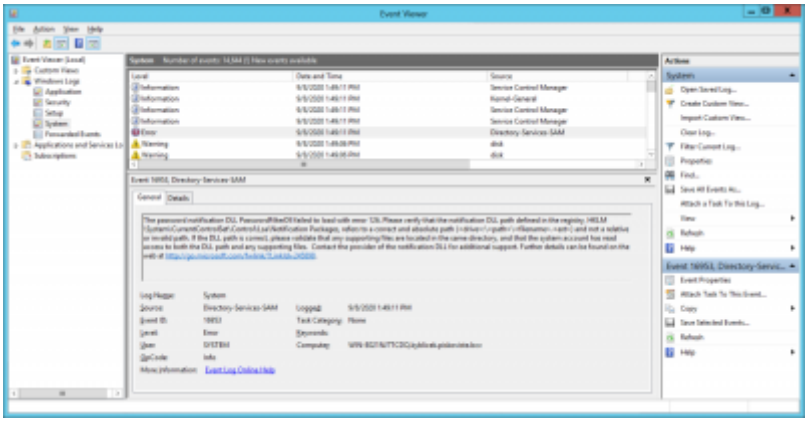
Password filter is able to use secured connection (https) to IdM server which is strongly recommended. Server with the PF needs to have imported certificate of the IdM server or an appropriate certificate authority (CA). Use Windows certificate manager to import such certificate. The certificate validation may be disabled by turning `ignoreCertificate` configuration property to **true**. But this is recommended for testing or debug purpose only.



Deployment troubleshooting

In case of troubles with deploying the password filter AD does not provide much information. The error log can be found in Windows event manager (see the picture below). This error log is common and may occur when some of following cases happens:

- `PasswordFilterDll` *.dll file is not present in expected directory i.e. `c:\Windows\System32\`.
- The record in the register `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Notification Packages` does not correspond to real password filter dll's name (by default PasswordFilterDll).
- Password filter dll is not able to find all its dependencies. Check the installation process manual for details.



-

Building password filter dll

To build PF dll it is necessary to install development tools and libraries PF depends on. MS Visual Studio (VS) was used for PF dll development. Along with VS installation there are installed common runtime libraries. Beside them we need to install cpprestsdk and log4cpp libraries. Both are available on Gitlab, but we will utilize package system called VCPKG provided by Microsoft. It will save us from manual download and installation of libraries. Please follow the installation and configuration steps on the original vcpkg site including integration of installed libraries to Visual Studio. It will ease your further development. The process of integration may have to be repeated after installation of libraries listed in the following paragraph.

Libraries to install

- `cpprestsdk` - a library providing REST communication; currently tested version: 2.10
- `log4cpp` - a logging library which enables writing logs to various destinations; currently tested version: 2.9

PF dll depends on above listed libraries. It is good idea to invoke the library installation command with specification of particular platform. Otherwise the default platform is used and it may not meet our

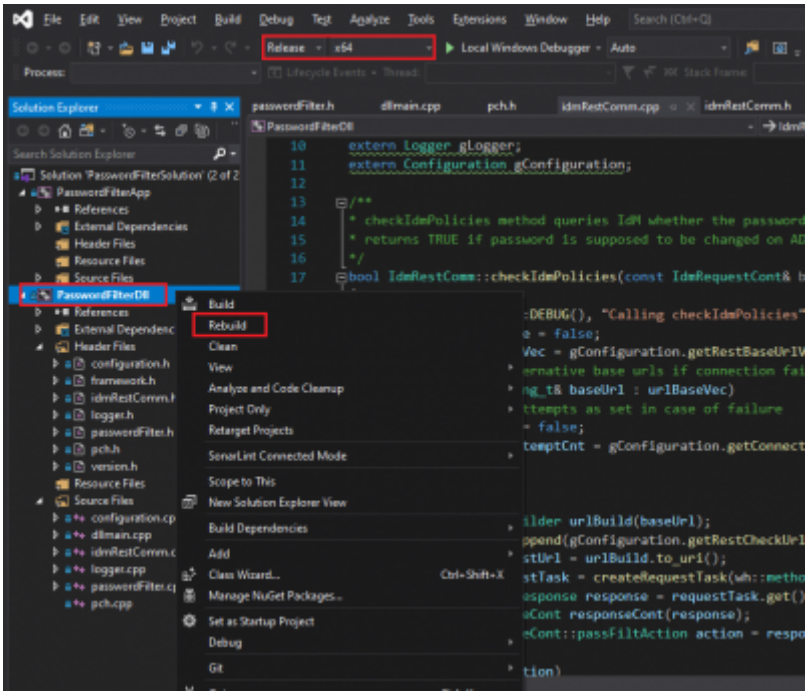
needs. The command for cpprestsdk lib for 64bit Windows platform will look like that one below. Package manager takes care of all dependencies the installed library needs.

```
.\vcpkg.exe install cpprestsdk:x64-windows
```

At the time of writing this doc, vcpkg system doesn't allow to control versions of installed libraries easily. It is worth using the latest available version of libraries. This is also the default behavior of vcpkg manager. In case of troubles with compatibility it may be necessary to return back to the original version of libraries the PF dll was developed with.

Building project in Visual Studio

Project build in Visual Studio is quite straightforward. Download PF project from Github and open its Solution in VS. There is no need of project adjusting if you performed the integration of vcpkg libraries as mentioned in the previous paragraph. The only thing which needs to be done is the selection of destination platform and compilation profile. Choose *Debug* for development and *Release* for deployment. Run *Rebuild* command applied on *PasswordFilterDll* project in the Solution. Compiled libraries can be found in <SOLUTION_DIR>/<PLATFORM_DIR>/<BUILD_PROFILE>. We will need PasswordFilterDll.dll, orocos-log4cpp.dll and cpprest_X_XX.dll (where X is a version number). Create or reuse the structure of installation package and place there these libraries. Overwrite the existing ones.



From:
https://wiki.czechidm.com/ - **IdStory Identity Manager**

Permanent link:
https://wiki.czechidm.com/devel/documentation/uniform_password/password_filter_dll

Last update: **2021/08/09 10:53**