



FOR FILES IN TRANSIT

CONFIGURATION GUIDE

Reproduction and rights

Copyright © Prim'X Technologies 2003 - 2025.

All reproduction, even partial, of this document is strictly prohibited without the prior written approval of Prim'X Technologies or one of its legal representatives. Any request for publication, in any form whatsoever, must be accompanied by an example of the planned publication. Prim'X Technologies hereby reserves the right to reject any proposal, without providing justification.

All rights reserved. Use of the **ZEDPRO** software is subject to the terms and conditions of the license agreement undertaken with the user or their legal representative.

PRIMX

Head Office: 18 Rue du Général Mouton-Duvernet 69003 LYON – support@primx.eu

Sales Department: OPEN21 21 Rue Camille Desmoulins 92130 ISSY-LES-MOULINEAUX – Tel. : +33 (0)1 40 95 24 80 – business@primx.eu

www.primx.eu

Contents

Reproduction and rights.....	2
Contents.....	3
Foreword.....	4
1. Principles	5
1.1. Windows Architecture	5
1.2. Linux / Mac Architecture.....	6
1.3. Policies file format.....	7
2. Configuration tool.....	9
3. Policies	10
3.1. General policies.....	10
3.2. Password control policies.....	11
3.3. Policies pertaining to certificates	13
3.4. User interface.....	15
3.5. Advanced policies.....	17

Foreword

The purpose of this configuration guide of the **ZEDPRO** product is to help the user-administrator to configure the product:

Presentation of the policy system

Tool allowing to configure the policies

Presentation of the main policies

This guide is not directly dedicated to users of an enterprise deployment, but to people who will set the security policies of the solution.

1. PRINCIPLES

The policies represent the settings of the **ZEDPRO** product that can be configured by the Security Officer. They are globally applicable to a workstation: all users logging on to a workstation will therefore have the same settings.

Most of the policies are integrated **dynamically** (i.e.: immediately) by the product. Those that are not (and which require a system reboot in order to be considered) are indicated.

When a policy is "**non-configured**", this means that the product uses an internal (manufacturer's) default value, or else, in the case of management per domain, that the choice of the value is left at the lowest administrative level (culminating, ultimately, in the manufacturer's default value if the policy is not configured at a lower level).

All the policies are configured with default values recommended by Prim'X to enable the execution of the product in complete security.

Caution: The modification of the default configuration parameters may lead to a deterioration of the security level.

1.1. Windows Architecture

It is possible to define certain modes of behavior of the ZEDPRO product by modifying its configuration using policies configured in the group strategies (GPOs).

The policies comply with the standard schema of Windows architecture:

They are configurable using the standard **GPEDIT.MSC** tool (via the Management Console [MMC]);

They are only accessible to users with administration rights to the workstation;

They can be managed via the standard schemas of domain controllers, including with respect to remote administration and remote distribution (see Microsoft server documentation on this subject);

These are "machine/computer" type policies; you can assign different values to different computers via an "Organization-Units" classification on the domain controller.

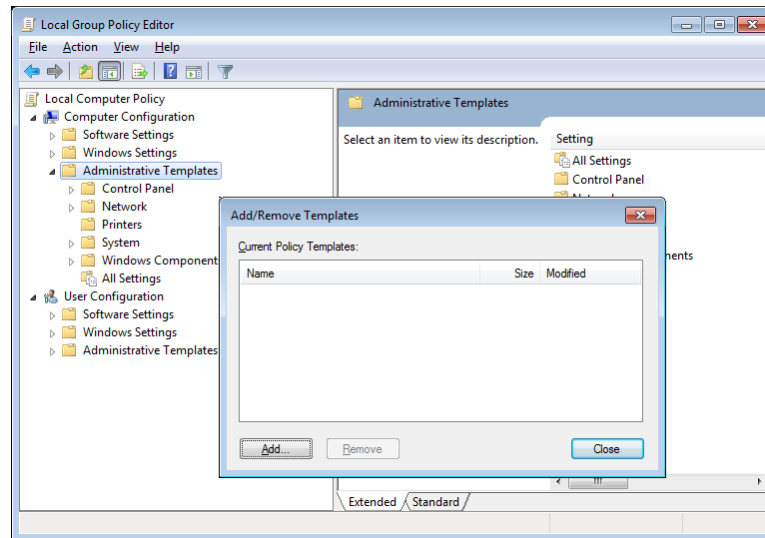
Configuring policies with the **GPEDIT.MSC** console is carried out using the administrative template file with the name "**zed_pro_policies.adm**". This file complies with the format required for Windows Administrative Templates.

Configuring policies on a workstation:

Copy the **zed_pro_policies.adm** file to the C:\Windows\inf folder

Run the local group policy editor **GPEDIT.MSC**

Under the category *Computer configuration*, right-click on *Administrative templates* then select *Add/Delete templates...*



You then need to add the ZEDPRO policies model by clicking on the *Add...* button in the *Add/Delete templates* window

The template can then be found in the `C:\Windows\inf` folder

Click on *Close* in the *Add/Delete templates* window, and you will find that the **ZEDPRO** policies have been imported in the local group policy editor of your computer.

To administer the policies from an Active Directory server, it is possible to copy this file to this server and declare it as the administrative template in the group strategy (or "Group Policy Object") of your choice (domain, domain controller, etc.). It is possible therefore to benefit from the management logistics of Windows server domains (a machine within a configured domain will receive the policies, generally updated every 90 minutes).

Classic pitfall: The Windows **GPEDIT.MSC** tool does not unfortunately display the policies **imposed** by the domain or the domain controller. It allows values to be modified even if they are already configured (and fixed) by the imposed policies. This frequently means that it is difficult to understand why a configuration has not been considered. You are recommended to use, at the same time, the "Resultant Set of Policy" view (RSOP). To do this, run an **MMC.EXE** console, add the RSOP snap-in, execute the "Generate RSOP data..." action, then consult the policies. Only those that are configured will appear, along with the origin of the configuration (domain, etc.). It is also possible to use the command line tools for consulting all the active policies (see product user guides).

With the **ZedCmd** tool it is possible to export the configured policies in an XML-format file (described in chapter 1.3), in order to distribute the implemented configuration on other workstations.

1.2. Linux / Mac Architecture

The policies are defined in XML-format files, described in chapter 1.3.

With Linux architecture, the policy file is:

```
/etc/primx/zed-pro-policies.xml
```

With Mac architecture:

```
/library/Preferences/Primx/zed-pro-policies.xml
```

These files cannot be directly modified by the user; administrator rights are required in order to edit them, either directly or via the dedicated commands.

These files can also be managed and updated from the server by means of a configuration files management tool (e.g., the *Puppet* tool).

1.3. Policies file format

The ZEDPRO policies file is presented as follows:

```
<root>
  <policies>
    <policy id="policy internal ID">
      <number>policy number</number>
      <label>policy name</label>
      policy value (format depends on the policy type)
    </policy>
  </policies>
</root>
```

There can be as many `<policy>...</policy>` blocks as required. In a policy block, the `id` field corresponds to the policy's internal ID. This field is required for **ZEDPRO** to recognize the policy. The number field indicates the policy number. This field is optional. It does not have to be stated in the file. The label field is also optional.

The policy value itself depends on its type: boolean, string, numerical or list.

Boolean type policy (or Yes/No).

```
<bool>True</bool>
```

Or

```
<bool>False</bool>
```

String type policy:

```
<sz>Free string</sz>
```

Numerical type policy:

```
<int>numerical value </int>
```

List type policy:

```
<list>
  <name>value name</name>
  <val>value</val>
</list>
```

```
<list>
  <name>value name</name>
  <val>value</val>
</list>
```

etc.

The configured values must not contain characters that are unauthorized by the XML format, in particular:

- < which must be replaced by **<**;
- > which must be replaced by **>**;
- & which must be replaced by **&**;

For the other characters, refer to the XML official standard.

To configure the policies, it is recommended not to edit the policies file directly but rather to use the dedicated commands in the **ZedCmd** tool.

File example:

```
<root>
  <policies>
    <policy id="MandatoryMembers">
      <number>139</number>
      <label>Mandatory access in encryptions</label>
      <list>
        <name>MY-MANDATORY.zaf</name>
        <val>sha256= A9 FB 90 9E 51 9C 74 D8 87 E9 B0 63 35 44 A0 9B 7B A9 0A
D3 AE EB 9A 48 F6 2A 93</val>
      </list>
    </policy>
    <policy id="ZedPwdMinimalQuality">
      <number>730</number>
      <label>ZED! passwords: minimal quality </label>
      <int>90</int>
    </policy>
  </policies>
</root>
```


2. CONFIGURATION TOOL

The **ZedCmd** command line can be used in particular for manipulating policies (export, import, edit, display). It can also be used to display the digital thumbprint of a file, which is useful in particular for configuring mandatory accesses.

Under Windows, this tool requires no installation and can be used directly on the workstation.

Under Linux, the installation of **ZEDPRO** is required to execute **ZedCmd**. It could be then used directly on the workstation.

Under Mac, the installation of **ZEDPRO** and the copy of **ZedCmd** into the folder `/Applications/ZEDPRO.app/Contents/MacOS` are required (Note: the application name can be « Zed ! Pro.app » depending on the version used).

In Windows architecture, the policies are read/written in the local workstation policies, whereas in Linux/Mac architecture they are read/written in the XML policies file.

Details of how to use a command may be obtained by means of the `/?` Windows option (e.g., `zedcmd ep /?`) or `-h` Linux option (`zedcmd ep -h`). In general, each command may be run by supplying it directly with all necessary options, or without any option, in which case the various items of information will be requested of the administrator interactively.

List of available sub-commands:

COMMAND NAME	ABBREVIATED NAME	DESCRIPTION
about	ab	Displaying the version of ZEDPRO with which the ZedCmd tool is associated. It is recommended to use a tool version identical to the ZEDPRO version installed on the workstation.
showpolicies	sp	Displaying the active policies on the workstation. Under Windows, this concerns the resultant set of policies deriving from the local policies and the domain policies. Under Linux / Mac, this concerns the policies configured in the XML file on the workstation. Policies that are not displayed are not configured. The <code>/a</code> option can be used to display the value of all the policies, even those that are not configured (it is then the default value that is displayed).
exportpolicies	ep	Exporting policies in an XML file. The file can then be used to be imported on to another workstation. Attention: under Linux / Mac this file cannot be used directly to replace the active policies file.
importpolicies	ip	Importing a policies file into the local policies.
modifypolicies	mp	Editing the local policies on a workstation. This command can be used quite simply without providing any settings: the policy to be modified and each new value are then requested interactively.
showhash	sh	Displaying the digital thumbprint of a file. This command is useful for obtaining the thumbprint for a mandatory access (see policy P139).
createmandatory	cm	Creating an access list file that can be used as a mandatory access (see policy P139). An access list may contain one or more accesses, of password or certificate type.

3. POLICIES

3.1. General policies

P139 – Mandatory accesses

→LIST value. The 'value name' must contain a file name (with or without its path), and the 'value' must contain the expression "*algo*=XX YY ZZ...", where *algo* indicates the algorithm used (sha1, sha256 or sha512) and XX YY ZZ... indicates the designated file's numeric footprint.

You can define as many files in the list as required. These files can be access files created by the command **createmandatory** (.zaf) or certificate files (.cer or .p7b, Base64 or binary coding).

This policy is **very important**. It is used to define a means for recovery, and to ensure that it is **applied automatically for all new encrypted container**.

The value name designates the file. It can be:

- A complete path: in this case, the file will first be searched for in the specified location. The file will then be searched in the search locations (see below);
- A filename: the file will be searched in the search locations.

Use the keyword "*None*" as the value name to indicate that no mandatory access will be added.

The search locations are (in order of search):

On Windows :

- user profile\Application Data\Zed-Pro
- ProgramData\Zed-Pro\AccessFiles
- ProgramData\Zed-Pro\Cache

On Linux :

- \$HOME\$/.primx/personalaccess
- /etc/primx/accessfiles
- /var/tmp/primx/cache
- \$HOME\$/.primx/cache

On Mac :

- \$HOME\$/Library/Application Support/Primx/personalaccess
- /Library/Application Support/Primx/accessfiles
- /Library/Caches/Primx
- \$HOME\$/Library/Caches/Primx

The value is used to specify the numeric footprint, the algorithm used and – optionally – the associated role. The syntax is as follows:

Algo=footprint;role=1

"*Algo*" can be "sha1", "sha256" or "sha512". The name "hash" can still be used, but should be avoided if possible. The algorithm indicates how the file's footprint is to be calculated, before being compared with the "*footprint*" part.

"*Role*" is used to specify the role of the mandatory access that will be added in the containers. The following values are available:

- "Admin=1": the administrative role (e.g. Local Security Officer). This role can only be specified for a certificate file.
- "normal=1": the normal user role (e.g. Director, wanting access to everything). This role can only be specified for a certificate file. An administrative access can modify the accesses of an existing encrypted container.

If the mandatory access designates a certificate file, its default role will be "recovery" (reminder: this role also automatically assigns it the "admin" role).

If this policy refers to a file that cannot be found, the user's action is refused.

To obtain the file's hash: the **zedcmd showhash** command must be used. You are advised against displaying the certificate's content and using the hash calculated by a certificate viewer; this is not to say that this content is invalid, but that its calculation method does not depend on the certificate file and format, rather on the certificate's value. Here, the product wants the hash of the file. The difference concerns the file's coding (Base64, etc.).

Important: when a new mandatory access of the access list type must be created, it is important to ensure the list is not an existing list that has simply been renamed, but rather a genuinely 'new list. If this were not the case, conflicts could occur when updating workstations.

Use cases: virtually systematic, and one of the first things to be done. The only case where this policy can be ignored is when the PKI 'impounds' the values of the users' keys.

3.2. Password control policies

Policies in the "Password control" category define the minimum strength and complexity required by a password in order for it to be accepted.

P730 – Password acceptance threshold

→ Internal ID: **ZedPwdMinimalQuality**

→ Numeric value (from 0 to 100%). Default value: 80%.

A password's strength is evaluated in relation to the various password control policies. A password's compliance with the policies is expressed as a percentage: a password that fully respects the constraints imposed by the policies will obtain a score of 100%.

This policy defines a password's acceptance threshold. A threshold set to 100% means that the password must respect all the constraints imposed by the policies (number of letters, digits, etc.).

Setting a threshold lower than 100% serves to accept passwords that only comply with a subset of rules. As a result, you can accept strong passwords that incorporate lowercase and uppercase letters, or lowercase letters and digits, depending on the user's preferences (a password that incorporates all 3 will be even stronger).

Use cases: this depends on the internal policy. The value of this policy must be decided on in harmony with the other password control policies.

P732 – Password length

→ Internal ID: **ZedPwdLengthTarget**

→ Numeric value (from 0 to 100%). Default value: 10.

P733 – Malus per missing character

→ Internal ID: **ZedPwdLengthMalus**

→ Numeric value (from 0 to 100%). Default value: 5%.

These policies determine the constraint on password length.

P732 determines the ideal length for a password. For each character below this length, the malus given by P733 is subtracted from the password's total quality.

Warning: the password's length is calculated without counting duplicates (identical characters). Thus, for example, the length of the password "abAab" is 3, because there are 3 unique characters in this password: a, b and A.

For example, if P732 indicates an ideal length of 10, and the user enters an 8-character password, a 10% malus will be applied. Moreover, if the password respects all the other constraints, its estimated quality will be 90% (which exceeds the default threshold – P730 – of 80%).

Use cases: this depends on the internal policy. The value of this policy must be decided on in harmony with the other password control policies.

P734 – Number of lowercase letters

→ Internal ID: **ZedPwdLowerCaseTarget**

→ Numeric value (from 0 to 100%). Default value: 2.

P735 – Malus per missing lowercase character

→ Internal ID: **ZedPwdLowerCaseMalus**

→ Numeric value (from 0 to 100%). Default value: 5%.

P734 indicates the ideal number of lowercase letters a password must contain. For each missing lowercase character, the malus given by P735 is subtracted from the password's total quality.

Warning: the number of lowercase letters is calculated without counting duplicates (identical characters). Thus, the password "ababa" contains 2 lowercase letters.

For example, P734 specifies 5 for the number of lowercase letters. The user keys in the password "123ABCabcabc": the malus subtracted from the total quality for lowercase letters will thus be 10%, because there are only 3 lowercase letters.

Use cases: this depends on the internal policy. The value of this policy must be decided on in harmony with the other password control policies.

P736 – Number of uppercase letters

→ Internal ID: **ZedPwdUpperCaseTarget**

→ Numeric value (from 0 to 100%). Default value: 2.

P737 – Malus per missing uppercase character

→ Internal ID: **ZedPwdUpperCaseMalus**

→ Numeric value (from 0 to 100%). Default value: 5%.

P736 indicates the ideal number of uppercase letters a password must contain. For each missing uppercase character, the malus given by P737 is subtracted from the password's total quality.

Warning: the number of uppercase letters is calculated without counting duplicates (identical characters). Thus, the password "ABACCBA" contains 3 uppercase letters.

For example, P736 specifies 4 for the number of uppercase letters. The user keys in the password "123ABCabcabc": the malus subtracted from the total quality for uppercase letters will thus be 5%, because there are only 3 uppercase letters.

Use cases: this depends on the internal policy. The value of this policy must be decided on in harmony with the other password control policies.

P738 – Number of symbols

→ Internal ID: **ZedPwdSymbolTarget**

→ Numeric value (from 0 to 100%). Default value: 1.

P739 – Malus per missing symbol

→ Internal ID: **ZedPwdSymbolMalus**

→ Numeric value (from 0 to 100%). Default value: 5%.

P738 indicates the ideal number of symbols a password must contain. For each missing symbol, the malus given by P739 is subtracted from the password's total quality.

Warning: the number of symbols is calculated without counting duplicates (identical characters). Thus, the password "&- ??&-" contains 3 symbols.

For example, P738 specifies 2 for the number of symbols. The user keys in the password "123ABCabcabc": the malus subtracted from the total quality for symbols will thus be 10%, because there are no symbols.

Use cases: this depends on the internal policy. The value of this policy must be decided on in harmony with the other password control policies.

P740 – Number of digits

→ Internal ID: **ZedPwdDigitTarget**

→ Numeric value (from 0 to 100%). Default value: 1.

P741 – Malus per missing digit

→ Internal ID: **ZedPwdDigitMalus**

→ Numeric value (from 0 to 100%). Default value: 5%.

P740 indicates the ideal number of digits a password must contain. For each missing digit, the malus given by P741 is subtracted from the password's total quality.

Warning: the number of digits is calculated without counting duplicates (identical characters). Thus, the password "12123412" contains 4 digits.

For example, P740 specifies 4 for the number of digits. The user keys in the password "123ABCabcabc": the malus subtracted from the total quality for digits will thus be 5%, because there are 3 digits.

Use cases: this depends on the internal policy. The value of this policy must be decided on in harmony with the other password control policies.

P743 – Help text for the user

→ Internal ID: **ZedPwdHelpTextLong**

→ "Entry field" value. Default value: empty (the default text is used).

In the various password entry interfaces, a help text is displayed to guide users in their choice of password, and which summarizes the constraints defined by the different policies.

By configuring this policy, this help text can be replaced with your own text. Your text can be fairly detailed, but no more than approximately 40-50 words so as not to be truncated.

Use cases: useful for adapting the text to selected constraints if the default text is deemed too vague.

3.3. Policies pertaining to certificates

P141 – Authorized roots

→ Internal ID: **RestrictCertRoots**

→ LIST value. The 'value name' contains a free name (e.g., 'My root'), and the 'value' must contain the expression "algo=XX YY ZZ...", where XX YY ZZ... contains the designated certificate's hash ("Algo" can be "sha1", "sha256" or "sha512").

This policy is used to limit the RSA keys users can use. It applies to the key's initial selection, to key renewals, and to added accesses (except for encrypted containers).

The purpose of this policy is to avoid users from using RSA keys they have generated themselves by whatever means (Internet, downloadable tool, personal key, etc.), and to ensure they only use keys issued by the official PKI (the underlying risk here is users encrypting with "whatever they like").

This policy is extremely important in one specific case: the policy is to not apply a recovery key (systematic addition of an 'in-house' key to all encrypted targets) because the PKI impounds users' RSA encryption keys. In this case, restricting usable keys to impounded keys only is clearly very important, because otherwise the recovery function is not ensured.

A number of certificates can be defined here. They do not have to be roots, but can be intermediate authorities (or even final certificates!). Note, however, that this policy does not provide the actual certificates, which must therefore be available in the PKI environment (local stores, LDAPs, attached to key holders).

Specifying an authorized CA certificate here does not mean that the checks are stopped for it. For example, if a certificate from an intermediate authority is specified, this means that you are permitted to use the certificates this authority has issued, provided both they and the authority are valid.

Use cases: essential if the PKI impounds users' keys; in all other cases, specifying this policy is worthwhile.

P142/143 – Check validity dates (public key)

This policy is composed of two parameters:

P142 – Authorizing use outside of validity dates

P143 – Tolerance period for the certificate associated with the private key

P142 – Authorizing the use of certificates beyond validity dates

→ Internal ID: **CanUseOutOfDateCertificates**

→ Yes/No value. Default value: No (complete check).

This policy allows a certificate to continue being used for some time after its expiration, when creating or adding an access. The following policy (P143) is used to define this amount of time.

This policy applies when a certificate is used for its public key. This is the case when you add an access via certificate to an encrypted container, or when you initialize your personal access.

When this period expires, or when this policy is not configured, the use of the certificate is refused with an explicit message.

Use cases: free

P143 – Period during which the use of expired certificates is tolerated

→ Internal ID: **OutOfDateCertificatesDelay**

→ Numeric value (in days). Minimum: 0, Maximum: 365

Default value: 90 days, but only applicable if the policy P142 is enabled.

See P142.

P225 – Trusted certificates

This policy only applies to Linux and Mac architectures.

→ Internal ID: **TrustedCertificates**

→ LIST type value. The "value name" designates the certificate, and the value designates the digital thumbprint of the certificate or the content encoded in base-64 of the certificate itself.

This policy can be used to define the trusted certificates (roots or intermediate certificates) in the user's Prim'X certificate store. These certificates are used when checking the validity of a certificate that is used as a new access in a zone, an access list or an encrypted container, or as a new personal access of the user.

Each element of this policy corresponds to a trusted certificate. A certificate may be indicated in the form of a file path, or directly in base-64 encoded form.

In file form:

The value name designates the file. This may involve:

- A complete file path: the file will first of all be searched for in the stated location. Next, the file will be searched for in the access list locations defined in the other policies (P120, P121 and P122).
- A simple file name: the file will be searched for in the access list locations defined in the other policies (P120, P121 and P122).

Use the keyword "none" as the value name to indicate that no trusted certificate is configured.

The value makes it possible to indicate the digital thumbprint and the algorithm used. The syntax is as follows:

Algo=thumbprint

"Algo" can be "sha1", "sha256" or "sha512". The "hash" name continues to remain usable, but should be avoided if possible. The algorithm indicates how the file thumbprint must be calculated, before being compared to the "thumbprint" part.

To obtain the file thumbprint, you need to use the "zedcmd showhash" command. We recommend against displaying the certificate content and using the thumbprint calculated by the Windows certificates viewer. Not that it is wrong to do so, but its calculation mode depends not on the certificate file and its format, but rather on the certificate value. What the product wants is the file thumbprint. There is a difference in the file encryption function (Base64, etc.).

In direct form:

A practical solution also consists in directly configuring the content of the trusted certificates in this policy. This avoids having to make the other certificates available.

The value name can be chosen freely, and may be used to indicate the designation of a certificate.

The value designates the certificate content, encoded in base-64:

content={base-64 encoded content}

The content is in the classic form: "-----BEGIN CERTIFICATE-----". The content must be framed by { and }.

Cases of use: This policy must be used to designate all trusted certificates, whether internal within the company or external. It must include, in particular, the root of the enterprise PKI, if there is one.

P226 – Rejected certificates

This policy only applies to Linux and Mac architectures.

➔ Internal ID: **DistrustedCertificates**

➔ LIST type value. The "value name" designates the certificate, and the value designates the digital thumbprint of the certificate or the content encoded in base-64 of the certificate itself.

This policy can be used to designate explicitly all certificates which are not to be trusted. When checking a certificate, carried out when creating a personal access or when adding an access to zone, an access list or an encrypted container. When a rejected certificate is found in the inspected certificate trust chain, it will then be refused.

This policy is configured in precisely the same way as policy P225.

Cases of use: this policy makes it possible, in particular, to indicate old roots that are no longer used or roots that are explicitly identified.

3.4. User interface

P289 – Options for encrypted containers

➔ Internal identifier: **ZedOptions**

➔ Text value of options list. Options list has the form Option={Value=X;Forced=Y} separated by semi-colons.

This policy allows selecting default options for encrypted containers. It is possible to force some, so they are not user-editable.

The configuration string is written as follows:

OpenMode={Value=xxxx;Forced=0 or 1};Location={Value=yyyy;Forced=0 or 1};WatermarkFile={Value=zzzz;Forced=0 or 1};WatermarkSha256={Value=hhhh}; WatermarkSha512={Value=hhhh};DisableWatermark={Value=0 or 1;Forced=0 or 1};NotifyUserOfReintegration={Value=0 or 1;Forced=0 or 1};DisableWorkFile={Value=0 or 1};NotifyUserOfReintegration={Value=0 ou 1;Forced=0 ou 1};DisableInactivity={Value=0 ou 1;Forced=0 ou 1};InactivityTimer={Value=kkkk;Forced=0 ou 1};

+ **OpenMode** option indicates how files are opened. Available values are:

- **Open**: opening for read only;
- **OpenOnly**: opening for read only, masking choice '**Modify**';
- **Modify**: opening for modification;
- **ModifyOnly**: opening for modification, masking choice '**Open**'.
- + **Location** value indicates where the extracted files are saved:
 - **Auto** value indicate a location in the temporary folder of the user
 - **A custom path value** that can contain environment variables and that will be created if it does not exist at the first use of containers.
- + **WatermarkFile**: path to a watermark file to use. Full path of file must be indicated.
Forcing this option prevents the user to change the watermark.
Supported file types are **.bmp, .jpg, .jpeg, .gif, .png**.
- + **WatermarkSha256**: optional, SHA-256 footprint of the defined watermark to verify the file integrity of the file defined in the WatermarkFile option.
- + **WatermarkSha512**: optional, SHA-512 footprint of the defined watermark to verify the file integrity of the file defined in the WatermarkFile option.
- + **DisableWatermark**: disables watermark display and modification.

Notes:

- Forcing a watermark breaks the compatibility with versions prior to 6.0 (an error message will be displayed at the opening of the container).
- If you force a watermark and the corresponding file is not found, the creation of the container is denied.

The (optional) hhhh value indicates the footprint (SHA256 or SHA512) of the file defined by the 'WatermarkFile' option in order to verify the file integrity.

Warning: since version 2021, "SHA-1" algorithm is not supported anymore.

The option *DisableWatermark* dictates the enabling (value 0) or disabling (value 1) of the watermark display and modification into encrypted containers.

- + **NotifyUserOfReintegration**: enables or disables the window displayed when saving a file in edition mode.
The option *NotifyUserOfReintegration* allows the activation or deactivation of the window displayed when saving a file in edition mode. If disabled, every time the file is saved, it will trigger a reintegration into the container.
- + **DisableWorkFile**: disables the creation of a temporary work file when opening an encrypted container.
- + **DisableInactivity**: allows you to disable the automatic saving of changes on inactivity. The backup is always performed when the container is closed.
- + **InactivityTimer**: allows you to define the time (in minutes) of inactivity before triggering the saving of changes in the container. By default, it is triggered after 5 minutes of inactivity.

For all options, keyword '**Forced**' indicates if the user can (value 0) or cannot (value 1) edit the option.

Tip : to get the configuration string in the options window, use the context menu of the title bar by holding 'ctrl' key down. Choice 'Copy options to clipboard' is then available.

Use case: used to configure options for encrypted containers with ability to force options for users.

P310 – Encrypted mail reply enable/disable

➔ Internal identifier: **ZED_DisableMailReply**

➔ Yes/No value. Default value: No (Reply allowed).

This policy allows choosing whether the user can reply to all recipients of an encrypted email or not. If set to Yes, the "Reply" button will be disabled when reading an opened email, restricting the feature.

Use cases: this depends on the internal policy, limits ZED to only be able to read emails.

3.5. Advanced policies

P296 – PKCS#11 smart cards and tokens supported/authorized

→ Internal identifier: **Tokens**

→ LIST value. The "value name" corresponds to the driver's published name; the "value" is not used (a comment can be inserted here).

This policy is used to specify one or more PKCS#11 extensions not supported by default, in other words, to use a different smart card or USB token model.

By default, the product searches for certain versions of PKCS#11 extensions by the following manufacturers: Bull Trustway, Gemalto, ActivCard, Aladdin, Rainbow, Oberthur Card Systems and (on Linux and Mac) the generic middleware OpenSC and Charismatics.

If this policy is configured, the product no longer searches for these recognized extensions, only that/those configured.

On Windows:

- You must specify the name of the supplier's PKCS#11 module (DLL name), without the path if this module is installed in a standard system directory, or with the path if not.

On Linux:

- You must specify the name of the supplier's PKCS#11 module, for example « /usr/lib/MyP11.so ».

On Mac:

- You must specify the name of the supplier's PKCS#11 module, for example « /usr/lib/MyP11.dylib ».

On each start of ZEDPRO, the configured modules are loaded. When a container is opened, if one of these modules contains a key holder with a valid access key for the zone to open, it will be used.

The fact that a PKCS#11 extension is declared and not present on the workstation is not considered to be an error; the extension will simply not be operational. This can serve, if necessary, to predefine several external "suppliers", whether installed or not.

Caution: on Windows, this policy only concerns the PKCS#11 interface; it has no effect on the CSP interface. Moreover, if other cards or tokens are available via the CSP interface, they are likely to be used.

Use case: when specific PKCS#11 extensions are used, or when the list is reduced to authorized cards only.

P329 – Allow pasting in password fields

→ Internal identifier: **EnablePasswordPasting**

→ Yes/No value. Default value: No (pasting in password fields is not allowed)

This rule allows pasting in password fields such as the open key window or create/add access windows.

Use case: this depends on the internal policy.

P399 – Version of encrypted containers and encrypted messages

→ Internal identifier: **ZedFormatVersion**

→ Value of the Choice type (Version 1, Version 2 "December 2014"). Default value: Version 2.

This policy defines the version number of encrypted containers created by users (there is no impact on existing encrypted containers).

The possible values are:

- + Version 1.
- + Version 2 (December 2014).

It is recommended to always choose the latest available version. However this requires that the contact uses a recent enough version of ZED! or ZEDMAIL to read this version.

Version 1 can be read by all ZED! or ZEDMAIL compatible software.

Version 2 can only be read by ZED! or ZEDMAIL compatible software published after December of 2014. This version brings enhanced security level for very small files.

Since version 6.1.2236, container watermark is not displayed if its integrity cannot be checked (container version 1 or version 2 created with version older than 6.1.2236 version).

P234 – Validity thresholds for encrypted data elements

➔ Internal identifier: **SecurityControl**

➔ LIST type value. The “value name” is the keyword of a feature to be validated; the “value” is used to indicate the action to be applied when validating the functionality.

This policy allows you to control features by defining the action to be applied when validating them.

The value name indicates the functionality to be controlled.

The value indicates the action to be applied when the feature is validated;

The value name syntax is: FEATURE

The syntax value is: ACTION={group=<users>};...;ACTION={group=<users>}

The keywords of possible functionalities to validate (to be used as value name, instead of 'FEATURE') are the following:

- + ZedIntegrityFailure: checks for corrupted encrypted containers.
By default, their use is prohibited.

Here are the possible actions to be applied (to be used as a value, instead of 'ACTION'):

- + Allow : the functionality is allowed;
- + AllowAndWarn : the functionality is allowed with warning display;
- + AllowReadOnly: the feature is allowed in read-only mode with warning display;
- + Deny: Feature is prohibited.

It is possible to filter the users for which an action is applied by using the keyword 'group=' followed by the user or user group name.

To indicate the members of a domain, it is preferable to use the Netbios Domain\Username syntax.

Examples of configuration :

- + Value name: ZedIntegrityFailure
- + Value: AllowAndWarn

Allows the use of corrupted encrypted containers by displaying a warning.

P499 – Reserved rules

➔ Internal identifier: **Reserved**

➔ LIST value.

This policy is used to configure unpublished features, reserved for highly specific cases of technical compatibility.

These features interfere with certain technical parameters, but on no account can they be used to reduce the workstation's security.

Use case: specific.