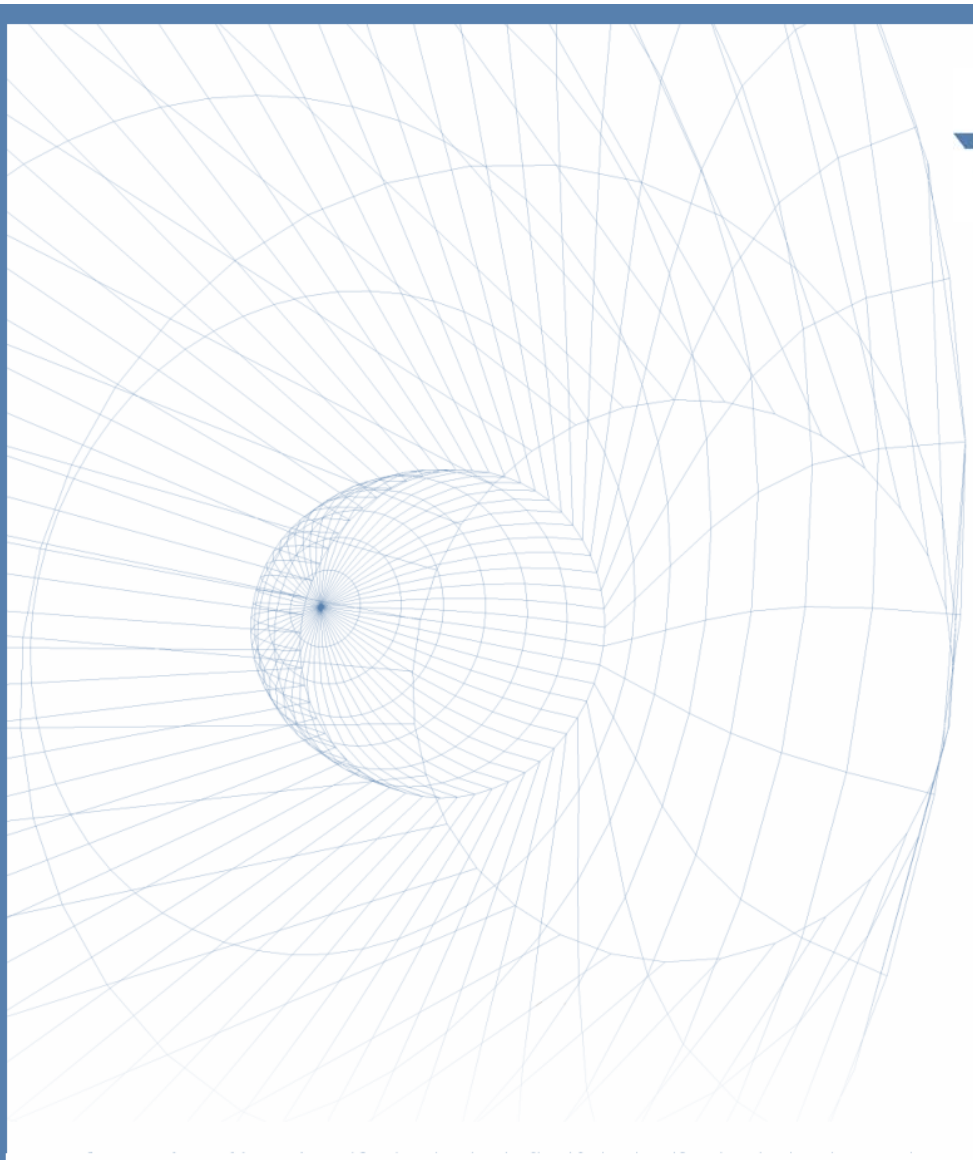

PML Publisher 1.0

User Guide

www.aveva.com

Disclaimer

Information of a technical nature, and particulars of the product and its use, is given by AVEVA Solutions Ltd and its subsidiaries without warranty. AVEVA Solutions Ltd and its subsidiaries disclaim any and all warranties and conditions, expressed or implied, to the fullest extent permitted by law.

Neither the author nor AVEVA Solutions Ltd, or any of its subsidiaries, shall be liable to any person or entity for any actions, claims, loss or damage arising from the use or possession of any information, particulars, or errors in this publication, or any incorrect use of the product, whatsoever.

Copyright

First published July 2006. This revision published January 2008

AVEVA Solutions Ltd, High Cross, Madingley Road, Cambridge, CB3 0HB, United Kingdom.

Trademarks

Revision History

Date	Version	Notes
July 2006	11.5.SP1	First Issue – limited distribution
September 2006	11.5.SP2	Revised – pmllib option added limited distribution
February 2007	11.6.SP4	Revised – limited distribution
January 2008	1.0	Revised for full distribution Version numbering restarted

Contents

1 Introduction

PML is the AVEVA Programmable Macro Language. You can find details of the language in the *Plant Design Software Customisation Guide* and the *Plant Design Software Customisation Reference Manual.*

PML functions, objects forms and macros may be encrypted using the tools described in this guide, and once encrypted may be used within PDMS but may not easily be read.

Encrypted PML files may be used in any compatible AVEVA program without an additional licence (see section 3 Using Encrypted Files in PDMS). The encryption utility described in section 2 Using the PML Encryption Utility Program is separately distributed and licensed.

Please note that the encryption used is of limited strength, and is not secure against all possible attacks — for details of the encryptions used, see section 2.5 .

This version includes a new encryption type. “Early Adopter” releases with PDMS 11.5.SP2 and 11.6.SP4 used a different encryption type which is no longer supported. AVEVA plans to release updated 11.5.SP2 and 11.6.SP4 versions supporting the new encryption type.

If you have existing encrypted files encrypted with the old encryption type 3, you must re-encrypt the original source to the new encryption type 4 if you wish to use them with version 12.0, or with updated 11.5.SP2 and 11.6.SP4 from January 2008.

1.1 Serious Warnings About Encryption

- Aveva may, from time to time, and at its sole discretion, change certain PML encryption algorithms. Customers must therefore acknowledge that existing encrypted PML applications may not work with the new encryption algorithms. On receipt of the new encryption algorithms Customers will need to re-encrypt the source code of the PML applications and therefore the customer must keep a record of the full and current PML source code.
- Please note that Aveva makes no guarantees or warranties as to the security of the encryption warranties and the customers use such encryption algorithms at their sole risk.
- The encryption used by PML publisher is shared by all users. If you encrypt a file for use in your company, it can be run by all users of a compatible version of PDMS, whether or not they are part of your company. (See section 2.7 for some example code to help address this issue)
- If you wish the use the same encrypted file with different program versions you must check each version for compatibility.
- Once a PML file has been encrypted, it can no longer be read or edited. When you publish a file make sure that you retain a safe copy of the original file, in case you want to make further modifications to it later, or in case a new encryption algorithm is required.
- The PML Publisher does **not** include a decryptor for encrypted files.

2 Using the PML Encryption Utility Program

2.1 Possible Workflow

pmlencrypt.exe, the encryption utility program supplied with this release, is a command-line program designed to be included in your PML software development process.

One possible workflow would be:

- Ensure that you have a current backup of the source PML
  No tool is supplied to decrypt an encrypted file, so it is very important that you keep good backups, in case you overwrite the source PML with an encrypted version.
- Copy the source folders to a new location
  Not all files within a PML folder hierarchy are always PML. Images, for example, should not be encrypted, but may need to be supplied with the encrypted versions of the PML.
- Encrypt from the source location to the new location.

Consider writing a batch file, a perl script, or a PML script to automate this procedure for your particular environment, to make it easy to create the encrypted PML environment correctly each time the source PML is updated.

2.2 Licensing

The pmlencrypt.exe utility program requires a PML Publisher licence in your license file (the feature name is VPD-PMLPUBLISHER). If this is not present then the program will not run.

2.3 Help

If pmlencrypt.exe is run without arguments, or with an invalid set of arguments, then a summary similar to this is output. The options are explained further in the following sections.

```
AVEVA PML Publisher Mk12.0 (Jan  4 2008)
(c) Copyright  2006 to 2008 AVEVA Solutions Limited

pmlencrypt [-rc4|-basic|-trivial|-none] [-buffer N] [-folder|-pmllib]
from_path to_path
```

-rc4	uses 40-bit RC4 encryption from the Microsoft Base Cryptographic Provider (default)
-basic	uses a simple low-security encryption algorithm
-trivial	uses a human-decipherable encryption scheme – for testing only
-none	no encryption, but can be used with -buffer N
-buffer N	causes the file to be retained in memory until a module switch once it has been read N times (the default is never)
-folder	is used to encrypt ALL files from the folder from_path to to_path
-pmllib	is used to encrypt ALL .pmlobj .pmlfnc .pmlfrm and .pmlmac files from the folders in a PMLLIB-type folder structure beneath from_path to to_path
from_path	is the file or folder to be encrypted
to_path	is the output file or folder

2.4 Choosing Files

PML files are not required to have particular file extensions. PML2 Functions, Objects, Forms and Macros are normally stored in files with the extensions .pmlfnc, .pmlobj, .pmlfrm and .pmlmac respectively, but other PML files, such as those in the pdmsui folder of a PDMS installation have no extension at all, and a PML file with any extension may be read with a $m command.

You must therefore be careful, when choosing files to encrypt, that you only encrypt PML files. Other files, such as icon images and configuration files cannot be used by PDMS when encrypted.

2.4.1 Single file

If neither of the –folder or –pmllib options are used the from_path and to_path arguments are taken to be single file-names or paths (which must not include embedded spaces). The to_path file is created or overwritten, as appropriate.

This option may be used whenever you have a single file to encrypt, and can also be useful within a script, where the file selection is handled by the script itself.

No assumptions are made about file extensions.

To encrypt a single file with one or more spaces in its name, move it into a folder without spaces in its name, and then encrypt the contents of that folder with the –folder option.

2.4.2 All files in a folder

If the –folder option is used the from_path and to_path arguments are taken to be names or paths of folders (which must not include embedded spaces, although files within the folders may). All files in the from_path folder are encrypted into the to_path folder. The to_path folder is created, if required, and the files inside it are overwritten.

No file extension is required, so this option is suitable for folders in the %PDMSUI% hierarchy, but you must be careful not to encrypt non-PML files.

2.4.3 Files in a pmllib-like folder tree

If the –pmllib option is used the from_path and to_path arguments are taken to be names or paths of folders (which must not include embedded spaces, although files and sub-folders within them may). All folders beneath the from_path folder are scanned, and files with extensions .pmlfnc, .pmlobj, .pmlfrm or .pmlmac are encrypted to a matching structure constructed or overwritten beneath the to_path folder.

As this option is file-extension sensitive, it will not encrypt image or other unrelated files in the hierarchy – but it will not copy them for you either.

2.4.4 File/Folder paths

Be careful when you give the from_path and to_path arguments that they are in that order –otherwise you may overwrite the wrong file.

The from_path and to_path arguments cannot be identical – this is to reduce the risk of accidental overwriting of the source-files.

Embedded spaces are not supported in the paths, but are allowed in files and sub-folders within from_path and to_path folders.

2.5 Encryption Algorithms

2.5.1 Encryption Type 4: RC4 Encryption

Encryption Type 4 (RC4 Encryption) is the recommended and default option. It can also be selected by the –rc4 option.

```
--<004>-- Published PML 1.0 >--
return error 99 'Unable to decrypt file in this software version'
$** d2b5c25a4eb20d0a540684e50a956e08
$** bs6mg5RrMcwxEsJcsWkPvI8wl0UzZbRe6k7aSK6MsfNn0Z1bI2+Qei7sparo
$** 07GrPZRJqvJvpyigRzDOR9OrbiaMj2O1nPllKJrLksfNKSMovO299idon3zg
$** SmwFahG-m1M4xgO4KfZ15tDf-k0n6wk45IsF9LMcX0lVc9hLEw+W64th
```

It is implemented using the Microsoft Base Cryptographic Provider, which is included in, among other operating systems, Windows 2000 and Windows XP. It is also included with Microsoft® Internet Explorer version 3.0 or later. 40-bit keys are used, to operate within limits imposed at one stage on exports of encryption technology.

It is therefore expected that all PDMS compatible computers will include the libraries required for this algorithm.

Please note that even this encryption is of limited strength, and is not secure against all possible attacks.

If you have existing encrypted files encrypted with encryption type 3, you must re-encrypt the original source to encryption type 4 if you wish to use them with version 12.0, or with the forthcoming updated 11.5.SP2 and 11.6.SP4.

2.5.2 Encryption Type 3: Obsolete

Encryption Type 3 is an obsolete format which is not supported at version 12.0.

```
--<003>-- Published PML 11.5.SP2 (Sep  6 2006) >--
return error 99 'This file is not readable by this version of PDMS'
$** 9ad7b51fc44384a8601979728b185f52
$** Ux1YR-LpiW-oRdjXdNJLy4-r8FE++c-LrEZsAzQebuwyRBKsrOv97U0h3dFR
$** M-5m1sMe41h2LlEXVpMadPyzRtVlUNMYdHhfBC8IYKtxe5BksX38RfF9mYUr
$** Vw3hBC9ZKUzMf80cvj0PIJJ
```

This format was the default for “Early Adopter” releases with PDMS 11.5.SP2 and 11.6.SP4. AVEVA plans to release updated 11.5.SP2 and 11.6.SP4 versions supporting encryption type 4 instead of encryption type 3 in January 2008.

If you have existing encrypted files encrypted with encryption type 3, you must re-encrypt the original source to encryption type 4 if you wish to use them with version 12.0, or with the updated 11.5.SP2 and 11.6.SP4.

2.5.3 Encryption Type 2: Basic Encryption

Encryption Type 2 (Basic Encryption) is an alternative simple encryption scheme which is implemented directly, and does not rely on external libraries. It can be selected by the –basic option.

```
--<002>-- Published PML 1.0 >--
return error 99 'Unable to decrypt file in this software version'
$** 4defaa8bf7dcf0d64dcd2aeda348703a
$** ppIlqUbi96dlUydmeuZkMLdkbWJ54Xp2Va4uR2M0RuZlkjPqQTLg5GoxqWrl
$** ZqN3Z65mRys0RuZlH7flJadi0+Zkampnc-4lKSJ2R64uRyYlOSZlabMzwLZh
$** 5+ZneHt2cmJgAi+sJqbmeWN8+jt0UzZnJKIz1uZlcOJ9
```

This algorithm is less secure than the RC4 algorithm, and is not recommended for general use.

2.5.4 Encryption Type 1: Trivial Encryption

Encryption Type 1 (Trivial Encryption) is designed for testing purposes only. It provides no security, as you can read the lines slowly (backwards), but you can use it to check that the decryption system is functioning correctly, and that, for example, an incompatible version of PDMS has not been installed.

```
--<001>-- Published PML 1.0 >--
orcam tset *$
)lasrever enil - laivirt( 1 mhtirogla htiw dedocne si elif sihT P$
cam.2ogla m$
cam.1ogla ni kcaB P$
```

It can be selected by the –trivial option.

For example, the line:

```
$p Decryption not available $*$
```

will be interpreted as a comment when read backwards as part of a trivially encrypted file, but will print a message if run on a version of PDMS that does not support any decryption.

2.5.5 Encryption Type 0: No Encryption

Encryption Type 0 (No Encryption) adds a standard Published PML header to the file, but does not otherwise encrypt the file.

It can be selected by the –none option.

```
--<000>-- Published PML 1.0 >--
$* test macro
$P This file (algo0.mac) is encoded with algorithm 0 (no encryption)
$m algo1.mac
$P Back in algo0.mac
```

You might choose to use this if you want to buffer the file for improved speed of access (particularly for widely used PML objects or functions accessed over a relatively slow network). For example, a file with the header

```
--<000-5>-- Published PML 1.0 >--
```

will be kept in memory after it has been read five times during a PDMS session.

2.6 Buffering

Decrypting a PML file takes longer than reading a plain-text version, and in some circumstances PML files may be re-read many times during a session. (A new command PML STATISTICS displays information on the numbers of times each file has been read and some extra information useful to AVEVA when testing the Published PML facilities).

In order to reduce the time taken to re-read the files, Published PML files may contain a buffering directive in the header-line (the first line in the file). If a dash and a number are included directly after the three-digit encryption algorithm ID, then PDMS will retain the file in memory indefinitely once it has been read that many times.

You may wish to edit heavily used files to add buffering to the header by hand, or may use the –buffer 5 option of pmlencrypt.exe to include a "buffer after five reads" tag in each file encrypted.

A value of five is a good number to start with. Many files are read precisely once during module start up — there is little benefit in buffering those files, and a value of five will avoid that, but apply to all heavily used files.

If a file you are actively developing has a header including buffering, it will not be re-read as often as you are used to. To force all buffered files to be cleared from memory if they are not in current use, you can issue the PML REHASH or PML INDEX commands, or switch modules.

2.6.1 Editing Published PML Files

Most changes made to an encrypted PML file will make it unusable (PDMS will report a corrupt file if you try) but there are a few exceptions:

You may add or change a buffering-value in the Published PML header-line, eg:

```
--<004>-- Published PML 1.0 >--
```

may be changed to

```
--<004-5>-- Published PML 1.0 >--
```

Adding a buffering-value of 5 (see later for details)

You may change the 2nd line of RC4 or Basic encrypted files to report a different error or message, eg

```
--<004>-- Published PML 1.0 >--
return error 99 'Unable to decrypt file in this software version'
$** 9ad7b51fc44384a8601979728b185f52
```

may be changed to

```
--<004>-- Published PML 1.0 >--
return error 66 'You need a PDMS patch – ring Ian on extension 6655'
$** 9ad7b51fc44384a8601979728b185f52
```

You may change lines within Trivial or unencrypted.

2.7 Examples

To encrypt a single file with the RC4 algorithm:

```
pmlencrypt  raw.txt  encrypted.txt
```

To encrypt a folder of files with the basic algorithm and buffering after three reads:

```
pmlencrypt -folder -basic –buffer 3 raw_folder  encrypted.folder
```

To encrypt a %PMLLIB% -structured hierarchy of files with no encryption algorithm but buffering after five reads:

```
pmlencrypt -pmllib -none –buffer 5 pmllib  pmllib_buffered
```

If you wish to not only encrypt a pml file, but also to restrict the sites at which it can be run, you can include extra tests within the pml before encrypting it. For example, the **q banner company** command returns a company dependent string from the license file, and you can test that within your encrypted pml file. In this case the test is that the string includes “AVEVA”

```
var !company banner company
if not !company.matchwild('*AVEVA*') then
   return error 99 'This file is not authorised for $!company'
endif
```

3 Using Encrypted Files in PDMS

Provided that you have a compatible version of PDMS then encrypted files can be read transparently in all modules that include PML.

PDMS 11.5.SP2 and 11.6.SP4 versions released before January 2008 support an “Early Adopter” encryption type 3, which is no longer supported from the first full release of PML Publisher 1.0. AVEVA is releasing updated 11.5.SP2 and 11.6.SP4 versions supporting the replacement encryption type 4, which is also supported in AVEVA Plant and Marine 12.0.

If you have existing encrypted files encrypted with encryption type 3, you must re-encrypt the original source to encryption type 4 if you wish to use them with version 12.0, or with the updated 11.5.SP2 and 11.6.SP4.

If you attempt to display or record encrypted PML using the $R commands, you will find that all lines are replaced by the text <hidden>. Error messages and trace-backs will include function names, but not the text of each line.

The only circumstance in which hidden lines can become visible is under certain circumstances during a macro which includes a module-switch. After a module switch, any remaining lines in that macro may or may not be traceable. This may change in a future release.

3.1 Error Messages

You may see the following error messages:

(46,103) PML: Encrypted file is corrupt or of unknown format	You are trying to read an encrypted file that has become corrupted (e.g. the encrypted text has been edited)
(46,104) PML: Encrypted file is in an obsolete and unsupported format	You are trying to read an encrypted file created with an algorithm that is no longer supported.
Unable to decrypt file in this software version	You are trying to read an encrypted file in an incompatible software version (e.g. the algorithms were created in a later software version) or: You are trying to read an RC4-encrypted file on a PC that doesn’t have the Microsoft Base Cryptographic Provider installed (this is not expected to occur)