SIEMENS EDA

Calibre® SVRFencrypt User's Manual

Software Version 2024.1
Document Revision 25

SIEMENS

About Siemens Digital Industries Software

Siemens Digital Industries Software is a global leader in the growing field of product lifecycle management (PLM), manufacturing operations management (MOM), and electronic design automation (EDA) software, hardware, and services. Siemens works with more than 100,000 customers, leading the digitalization of their planning and manufacturing processes. At Siemens Digital Industries Software, we blur the boundaries between industry domains by integrating the virtual and physical, hardware and software, design and manufacturing worlds. With the rapid pace of innovation, digitalization is no longer tomorrow’s idea. We take what the future promises tomorrow and make it real for our customers today. Where today meets tomorrow. Our culture encourages creativity, welcomes fresh thinking and focuses on growth, so our people, our business, and our customers can achieve their full potential.

Support Center: support.sw.siemens.com
Send Feedback on Documentation: support.sw.siemens.com/doc_feedback_form

Revision History

Revision	Changes	Status/ Date
25	Modifications to improve the readability and comprehension of the content. Approved by Lucille Woo. All technical enhancements, changes, and fixes listed in the *Calibre Release Notes* for this products are reflected in this document. Approved by Michael Buehler.	Released January 2024
24	Modifications to improve the readability and comprehension of the content. Approved by Lucille Woo. All technical enhancements, changes, and fixes listed in the *Calibre Release Notes* for this products are reflected in this document. Approved by Michael Buehler.	Released October 2023
23	Modifications to improve the readability and comprehension of the content. Approved by Lucille Woo. All technical enhancements, changes, and fixes listed in the *Calibre Release Notes* for this products are reflected in this document. Approved by Michael Buehler.	Released July 2023
22	Modifications to improve the readability and comprehension of the content. Approved by Lucille Woo. All technical enhancements, changes, and fixes listed in the *Calibre Release Notes* for this products are reflected in this document. Approved by Michael Buehler.	Released April 2023

Author: In-house procedures and working practices require multiple authors for documents. All associated authors for each topic within this document are tracked within the Siemens EDA documentation source. For specific topic authors, contact the Siemens Digital Industries Software documentation department.

Revision History: Released documents maintain a revision history of up to four revisions. For earlier revision history, refer to earlier releases of documentation which are available on https://support.sw.siemens.com/.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction to Calibre SVRFencrypt

You can use Calibre SVRFencrypt to encrypt sensitive data in a Standard Verification Rule Format (SVRF) rule file.

Overview

The Calibre SVRFencrypt program provides the ability to encrypt an SVRF rule file for the purpose of distributing the encrypted rule file to a third party Calibre customer without disclosing any proprietary information. If you distribute an encrypted rule file to your customer, it is your responsibility to support the customer. Siemens Digital Industries Software can only support a rule file that does not contain encrypted information.

Note

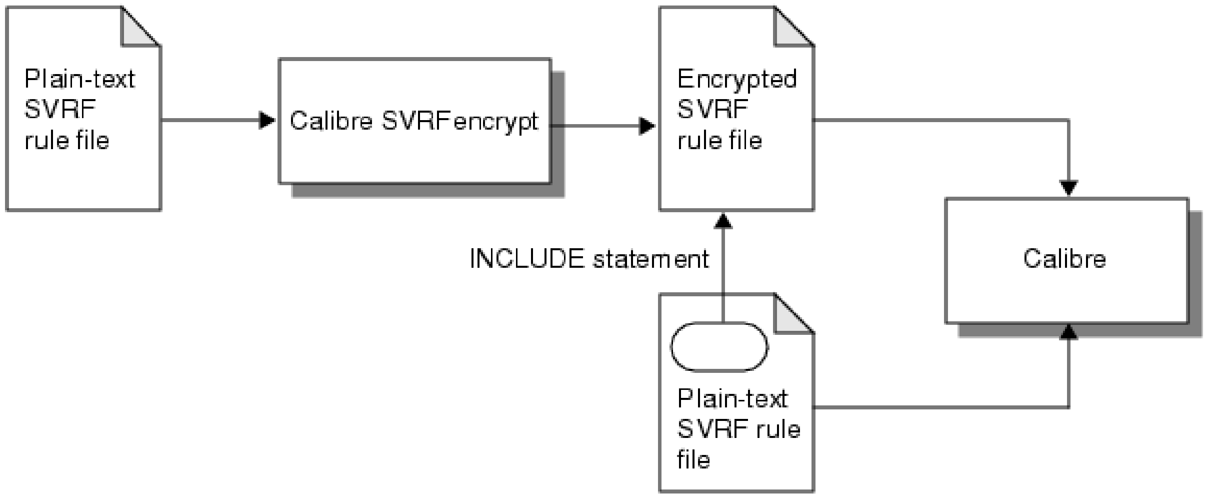
In order to protect the proprietary and sensitive information of our customers, Siemens Digital Industries Software cannot decrypt an encrypted file. Any customer who encrypts a file is responsible for maintaining its corresponding source and decryption information.

Workflow

You can either input an encrypted rule file directly into Calibre, or reference it from an unencrypted rule file using the INCLUDE statement.

Figure 1-1 illustrates the basic workflow for using Calibre SVRFencrypt.

Figure 1-1. Calibre SVRFencrypt Workflow



Supported Tools

Calibre SVRFencrypt is available on all platforms supported by the Calibre product line. It is backwards-compatible (newer versions of Calibre can decrypt files encrypted with older versions of Calibre).

Encrypted rule files are supported by any Calibre tool that accepts an SVRF rule file as input.

Licensing Requirements

You must have a Calibre SVRFencrypt license to use this utility.

For specific licensing information on Calibre SVRFencrypt, refer to the *Calibre Administrator's Guide*.

Syntax Conventions

The command descriptions use font properties and several metacharacters to document the command syntax.

Table 1-1. Syntax Conventions

Convention	Description
Bold	Bold fonts indicate a required item.
Italic	Italic fonts indicate a user-supplied argument.
`Monospace`	Monospace fonts indicate a shell command, line of code, or URL. A bold monospace font identifies text you enter.

Table 1-1. Syntax Conventions (cont.)

Convention	Description
<u>Underline</u>	Underlining indicates either the default argument or the default value of an argument.
UPPercase	For certain case-insensitive commands, uppercase indicates the minimum keyword characters. In most cases, you may omit the lowercase letters and abbreviate the keyword.
[ ]	Brackets enclose optional arguments. Do not include the brackets when entering the command unless they are quoted.
{ }	Braces enclose arguments to show grouping. Do not include the braces when entering the command unless they are quoted.
‘ ’	Quotes enclose metacharacters that are to be entered literally. Do not include single quotes when entering braces or brackets in a command.
| or ||	Vertical bars indicate a choice between items. Do not include the bars when entering the command.
…	Three dots (an ellipsis) follows an argument or group of arguments that may appear more than once. Do not include the ellipsis when entering the command.
Example: **DEVice** {***element_name*** [‘(’*model_name*‘)’]} ***device_layer*** {***pin_layer*** [‘(’*pin_name*‘)’] …} [‘<’*auxiliary_layer*‘>’ …] [‘(’*swap_list*‘)’ …] [<u>BY NET</u> | BY SHAPE]	

Chapter 2
Preparing an Encrypted Rule File

You prepare a rule file for encryption by including encryption directives and enabling or disabling sections of the file for encryption. You can fully or partially encrypt a rule file.

Rule File Encryption Directives

You can add SVRF encryption directives to your rule file to explicitly identify the sections you want to encrypt.

The SVRF encryption directives include:

- #ENCRYPT — Identifies the beginning of a section to be encrypted.
- #ENDCRYPT — Identifies the end of a section to be encrypted.

These directives must be the first keywords on their respective lines; Calibre SVRFencrypt ignores any other information on these lines. Calibre SVRFencrypt issues an error if you attempt to nest #ENCRYPT and #ENDCRYPT pairs or do not specify matching #ENCRYPT and #ENDCRYPT pairs.

The Calibre SVRFencrypt program encrypts all information between the #ENCRYPT and #ENDCRYPT directives.

When encrypting a rule file, the #ENCRYPT and #ENDCRYPT directives must be completely outside of any SVRF statements or commands. Specification statements that affect general runtime environments, such as Layout Path and DRC Results Database, which can be set through the use of an environment variable, will be retained when decrypted. For this reason, when providing an encrypted rule file that contains environment variables to a third-party, you should also provide the name of the variables and the valid values.

Calibre SVRFencrypt resolves macro language variables. Therefore, a Calibre run using the encrypted rule file will work as expected.

To encrypt setup files or model files used with Litho operations you must use an inline setup or model file. You can insert the #ENCRYPT and #ENDCRYPT directives anywhere within the setup file section.

SVRFencrypt will not encrypt external setup and model files. Because of this, you should provide any external setup or model files directly to the third-party user when providing the encrypted rule file.

Selectively Disabling Encryption Protection

In certain cases, it is necessary to disable encryption-dependent protection for various outputs generated from an encrypted rule file. The #PRAGMA TRANSPARENT statement can be used to selectively display plain-text output for a number of rule file elements. The syntax is:

```
#PRAGMA TRANSPARENT option1 [option2 ...]
```

where each *option* may be one of the following:

- ERROR — The output of SVRF Error statements is not suppressed even if the SVRF Error statements are encrypted.
- MESSAGE — The output of SVRF Message statements is not suppressed even if the SVRF Message statements are encrypted.
- @COMMENT — Rule check comments are not suppressed, even if the rule checks are encrypted.
- LAYERS — Layer names are not replaced by the <ENCRYPTED_LAYER_X> string, even if the operations producing the layers are encrypted.
- NONE — All options are canceled.

For each line of the rule file, there can be only one #PRAGMA TRANSPARENT statement in effect. Each #PRAGMA TRANSPARENT statement cancels any previously-specified statements.

#PRAGMA TRANSPARENT statements have effect only inside the encryption block in which they appear. A #PRAGMA TRANSPARENT statement added before an encrypted block has no effect on the encrypted statements inside the block. Similarly, the effect of a #PRAGMA TRANSPARENT statement ends when the encrypted block containing the statement ends. This ensures that the encryption cannot be weakened by anyone who does not have access to the plain-text version of the encrypted rules. The only way to relax the encryption requirements using a #PRAGMA TRANSPARENT statement is to insert the statement into an encrypted section of a rule file prior to encryption.

The #PRAGMA TRANSPARENT statement can be used inside both the #ENCRYPT and #ENDCRYPT and the #PRAGMA ENCRYPT and #PRAGMA ENDCRYPT statements.

In the following example, the #PRAGMA TRANSPARENT statements do the following:

- #PRAGMA TRANSPARENT @COMMENT causes “COMMENT 2” to be output to the transcript.

- #PRAGMA TRANSPARENT MESSAGE causes “MESSAGE 3” to be output to the transcript.
- #PRAGMA TRANSPARENT @COMMENT MESSAGE causes “MESSAGE 4” and “COMMENT 4” to be output to the transcript.

All other comments are hidden and the remaining SVRF Message statements are suppressed.

```
#ENCRYPT
RES1 { @ COMMENT 1
COPY METAL1
}
SVRF MESSAGE "MESSAGE 1"
#PRAGMA TRANSPARENT @COMMENT
RES2 { @ COMMENT 2
COPY METAL2
}
SVRF MESSAGE "MESSAGE 2"
#PRAGMA TRANSPARENT MESSAGE
RES3 { @ COMMENT 3
COPY METAL3
}
SVRF MESSAGE "MESSAGE 3"
#PRAGMA TRANSPARENT @COMMENT MESSAGE
RES4 { @ COMMENT 4
COPY METAL4
}
SVRF MESSAGE "MESSAGE 4"
#PRAGMA TRANSPARENT ERROR
RES5 { @ COMMENT 5
COPY METAL5
}
SVRF MESSAGE "MESSAGE 5"
#ENDCRYPT
```

Next, consider the following example:

```
#PRAGMA ENCRYPTED
D = A OR B
E = A NOT B
#PRAGMA TRANSPARENT LAYERS
F = B AND C
G = D OR E
H = F OR G
#PRAGMA ENDCRYPT
RES { DFM COPY D E F G H }
```

In this example, the layers D and E are encrypted, and their names are not revealed in the transcript nor in the results database. The layers F, G, and H are transcripted under their original names, while the operations that produce these layers are hidden:

```
<ENCRYPTED OPERATION(S)>
------------------------
<ENCRYPTED_LAYER_4> (TYP=1 CFG=1 ECT=6 OCT=3 SRT=1 CMP=F MPN=1)

<ENCRYPTED OPERATION(S)>
------------------------
f (TYP=1 CFG=1 ECT=4 OCT=2 SRT=1 CMP=F MPN=2)

RES::<1> = DFM COPY <ENCRYPTED_LAYER_4> <ENCRYPTED_LAYER_5> f g h
```

Encryption of Include Files

During encryption, Calibre SVRFencrypt performs syntax checking on the rule file and any included rule files.

If you wish to encrypt only your include file, you can follow either of the following methods:

- Specify the #ENCRYPT and #ENDCRYPT directives within the include files.
- Specify the #ENCRYPT and #ENDCRYPT directives around the include statement of the parent rule file, as shown:

```
#ENCRYPT
Include name.file
#ENDCRYPT
```

When you encrypt a rule file that contains include statements, the tool flattens and echoes to the transcript the contents of the include file (unless you use the -e option). This behavior is different than a Calibre run that does not include encryption.

Calibre SVRFencrypt fully encrypts the contents of the rule file and any associated Calibre nmDRC/nmDRC-H include files between the #ENCRYPT and #ENDCRYPT directives. However, only certain SVRF commands and their transcripted results are supported for encryption. When the rule file is executed, those commands not supported will be visible in the transcript.

Chapter 3
Creating an Encrypted Rule File

Once you have prepared your rule files for encryption, you use the **calsvrfencrypt** program to encrypt your files.

Rule File Processing

Prior to encrypting a rule file, it is recommended that you use the unencrypted version of the rule file in a Calibre run to ensure that the rule file compiles without any errors.

Basic Encryption

The basic syntax for encrypting a rule file is:

```
calsvrfencrypt <input_file> <output_file>
```

To perform basic encryption, the rule file must contain the encryption directives (#ENCRYPT and #ENDCRYPT) as described in "Rule File Encryption Directives" on page 15.

When you use this command to encrypt a rule file, the **calsvrfencrypt** program puts the complete rule file and any included rule files through the standard Calibre preprocessor prior to encrypting the file. If the preprocessor encounters any errors, Calibre generates an error and stops the encryption run. Refer to "Error Messages" on page 39 for information on errors generated by the **calsvrfencrypt** program and errors encountered when Calibre attempts to read an encrypted rule files.

Encryption Options

The **calsvrfencrypt** program provides options to control the processing of the rule file including the ability to:

- Bypass preprocessing

  You can optionally specify the -e or -s argument to bypass preprocessing. The -e argument encrypts preprocessor and include directives, while the -s argument does not encrypt these directives. When preprocessing is bypassed using -e or -s, no compilation is done of the input rule file. The header in the output file is slightly different and includes the following line:

```
// This rule file has not been verified using any Mentor Graphics
software.
```

- Specify a runtime pass phrase

  You can optionally specify a pass phrase using the -r *runpassphrase* argument when you encrypt your rule file. The file can only then be decrypted by providing the pass phrase when invoking Calibre.

  When you assign a pass phrase to an encrypted file, you must supply the enduser of the encrypted rule file with the pass phrase for subsequent decryption during the Calibre run. The enduser must then set the CALIBRE_RUNPASSPHRASE environment variable to the pass phrase prior to invoking Calibre and specifying the encrypted rule file. If the pass phrase is incorrect, then decryption fails, and Calibre issues the following message in the transcript:

  ```
  Decryption impossible
  ```

- Generate output without special characters

  You can optionally specify the -a argument to output an encrypted file using only alphanumeric characters, colons, and periods. Special characters such as &, #, %, and $ are not output to the encrypted file.

  Encrypted files with special characters can be misinterpreted during compilation of TVF in a tvf::VERBATIM block. Using the -a argument, the **calsvrfencrypt** program produces encrypted output that is compatible with the tvf::VERBATIM command or your TVF rule file.

Refer to the "calsvrfencrypt" on page 24 for details on using these options.

Encryption of Rules Files with INCLUDE Statements

The **calsvrfencrypt** program handles preprocessing statements containing INCLUDE statements based on where such statements are specified in the preprocessor blocks.

Consider the following example, where two files are included based on the condition specified by *myVar*:

```
#DEFINE myVar
#IFDEF myVar
   INCLUDE myIncludeFile_1

...

#IFNDEF myVar
   INCLUDE myIncludeFile_2
```

The utility generates this output:

```
#DEFINE myVar
#IFDEF myVar
   //INCLUDE myIncludeFile_1
<contents of myIncludeFile_1>

...

#IFNDEF myVar
//INCLUDE myIncludeFile_2
<no content from myIncludeFile_2 is included>
```

After encryption, because *myVar* is defined (#IFDEF *myVar* evaluates as true), the contents of *myIncludeFile_1* is included in the encrypted output file.

The content of *myIncludeFile_2* is not included in the output file because the preprocessor directive (#IFNDEF *myVar*) evaluates as false.

Encrypted File Structure

The encryption process converts #ENCRYPT directives to #DECRYPT directives and adds a header statement.

#DECRYPT Directive — Calibre SVRFencrypt and Calibre TVFencrypt convert an #ENCRYPT directive into a #DECRYPT directive followed by an encrypted tag (highlighted below).

Before encryption:

```
#ENCRYPT
LAYOUT SYSTEM OASIS
LAYOUT PATH "../Shared/Design/fullchip.oas"
...
#ENDCRYPT
```

After encryption:

```
#DECRYPT &).~1E)JR5A!,'"W;K'U]S>7U5DD#KW*;>S1#@04,HOAR\U!#67...
G:.L>6!&7!\:G;M`5^-W6Q!!"+HH,YF<^-DR7.OK?)[Q!)@+#VR2A#Y(R;A]PPJB=C1A!#
IAP1$/6E;?B!SI5#P8/QI1!!"Z&Q<)[E\S7U(R;/]L;IC1JS6*V`XW-I)=PF"8IL3B6...
#ENDCRYPT
```

Header Statement — Calibre SVRFencrypt adds a header to the top of the encrypted rule file that includes a legal disclaimer and reference to the version of Calibre used to verify the rule file.

```
// Calibre SVRFencrypt v2020.2_0.61
// This rule file or portions thereof includes data that is
// encrypted. Lines between the #DECRYPT and the #ENDCRYPT
// statements must not be altered in any manner.
// Mentor Graphics Corporation does not provide support for the
// encrypted portion of the rules file. This is the sole responsibility of
// the providing organization and problems encountered are to be reported
// to the provider. Support by Mentor Graphics for an encrypted
// rules file requires the availability of the original rules file.
//

// This deck was developed using Mentor Graphics proprietary SVRF and TVF
// formats.  The deck is to be used only in Calibre tools.  For further
// information about SVRF/TVF File Use Restrictions, please go to
// http://www.mentor.com/terms_conditions/svrf-tvf.html
// LIMITATION OF LIABILITY: Mentor Graphics is not liable for any property
// damage, personal injury, loss of profits, interruption of business, or
// any other special, consequential or incidental damages, however caused,
// whether for breach of warranty, contract, tort (including negligence),
// strict liability or otherwise. In no event shall Mentor Graphics'
// liability exceed the amount paid for the product giving rise to the

//
// This rule file is verified using Calibre version v2020.2_0.61
//
```

Command Reference

In addition to the **calsvrfencrypt** command, the **caltxtencrypt** command allows you to encrypt plain text files.

calsvrfencrypt

Encrypts an SVRF rule file.

Usage

calsvrfencrypt {{*input_file output_file*} | {{-s | -e} '<' *input_file* '>' *output_file* [-a] [-t]}}
[-r *runpassphrase*]

Arguments

- *input_file*

  Specifies the path and filename of an SVRF rule file. When performing encryption without using the -e or -s argument, the rule file must contain encryption directives as described in "Rule File Encryption Directives" on page 15.

- *output_file*

  Specifies the path and filename of the encrypted output file.

- -s

  An optional argument that specifies to wrap the contents of the *input_file* with #ENCRYPT and #ENDCRYPT directives, bypass the Calibre preprocessor, and leave preprocessor directives and INCLUDE statements unencrypted in the *output_file*. Preprocessor directives that are preserved are #DEFINE, #IFDEF, #IFNDEF, #ELSE, #ENDIF, and #UNDEFINE. When preprocessing is bypassed no compilation is done on the input rule file and the generated output file header includes the following line:

  ```
  //This rule file has not been verified using any Mentor Graphics
  software.
  ```

  When using this argument, you must specify the redirection characters '<' before *input_file* and '>' before *output_file*. For example:

  ```
  calsvrfencrypt -s < drc.rules > drc_encrypt.rules
  ```

  This argument cannot be used with the -e argument.

- -e

  An optional argument that specifies to wrap the contents of the *input_file* with #ENCRYPT and #ENDCRYPT directives and bypass the Calibre preprocessor. When preprocessing is bypassed no compilation is done on the input rule file and the output file header includes the following line:

  ```
  //This rule file has not been verified using any Mentor Graphics
  software.
  ```

  This argument differs from the -s argument in that it does encrypt preprocessor and include directives.

When using this argument, you must specify the redirection characters '<' before *input_file* and '>' before *output_file*. For example:

```
calsvrfencrypt -e < drc.rules > drc_encrypt.rules
```

This argument cannot be used with the -s argument.

- -a

  An optional argument that specifies to output the encrypted file using alphanumeric characters, colons, and periods. No other special characters are output to the encrypted file. This argument can only be used with -s or -e.

- -t

  An optional argument that makes the #PRAGMA TRANSPARENT flags "true" by default. This argument can only be specified with the -s or -e argument. Refer to "Selectively Disabling Encryption Protection" on page 16 for more information on the #PRAGMA TRANSPARENT statement.

- -r *runpassphrase*

  An optional argument set used to specify a pass phrase for the encrypted rule file (*output_file*). The *runpassphrase* is an alphanumeric string. You must put quotes around *runpassphrase* if it contains spaces.

Note

If you specify a pass phrase for an encrypted rule file, then you must supply the Calibre application with the identical *runpassphrase* at runtime. To do this, set the CALIBRE_RUNPASSPHRASE environment variable to the value of *runpassphrase*.

Description

Reads a rule file and outputs an encrypted rule file.

Examples

Example 1

This example creates an encrypted rule file named *drc_encrypt.rules* from an input rule file named *drc.rules*. The #ENCRYPT and #ENDCRYPT directives identify the sections in the input rule file to be encrypted.

```
calsvrfencrypt drc.rules drc_encrypt.rules
```

Example 2

This example encrypts the rule file named *lvs.control* and outputs the results to *lvs_encrypt.control*. The contents of the rule file are automatically wrapped with #ENCRYPT and #ENDCRYPT directives and preprocessing is bypassed. The -s argument causes any preprocessor directives and INCLUDE statements in the rule file to remain unencrypted.

```
calsvrfencrypt -s < lvs.control > lvs_encrypt.control
```

Example 3

This example encrypts the rule file named *lvs.control* and outputs the results to *lvs_encrypt.control*. The contents of the rule file are automatically wrapped with #ENCRYPT and #ENDCRYPT directives and preprocessing is bypassed. The -e argument causes any preprocessor directives and INCLUDE statements in the rule file to be encrypted.

```
calsvrfencrypt -e < lvs.control > lvs_encrypt.control
```

Example 4

This example encrypts the rule file named *lvs.control* and outputs the results to *lvs_encrypt.control*. The contents of the rule file are automatically wrapped with #ENCRYPT and #ENDCRYPT directives and preprocessing is bypassed. The -e argument causes any preprocessor directives and INCLUDE statements in the rule file to be encrypted. The -a argument prevents special characters being output in the encrypted results.

```
calsvrfencrypt -e < lvs.control > lvs_encrypt.control -a
```

Example 5

This example uses the -r argument to specify the pass phrase “secret” to be associated with the generated encrypted file named *drc_encrypt.rules*.

```
calsvrfencrypt -e < drc.rules > drc_encrypt.rules -r secret
```

To use the encrypted rule file for a Calibre run, you must first set the CALIBRE_RUNPASSPHRASE environment variable to “secret”. For example:

```
setenv CALIBRE_RUNPASSPHRASE secret
calibre -drc -hier drc_encrypt.rules
```

Decryption fails if this variable is not set or the pass phrase is incorrect and Calibre issues the following message in the transcript:

```
Decryption impossible
```

caltxtencrypt

Encrypts plain-text files using the same license as Calibre SVRFencrypt.

Usage

caltxtencrypt {-stdin | *input_file*} *output_file* [-r *runpassphrase*]

Arguments

- -stdin

  Specifies to read input from standard input.

- *input_file*

  Specifies a path and file name for reading input.

- *output_file*

  Specifies a path and file name for writing output.

- -r *runpassphrase*

  An optional argument pair used to specify a string, or pass phrase, to associate with the encrypted rule file. The *runpassphrase* is an alphanumeric string. You must put quotes around *runpassphrase* if it contains spaces.

  Note

  If you specify a pass phrase for a rule file, then you must supply the Calibre application with the identical *runpassphrase* at runtime. You do this by setting the CALIBRE_RUNPASSPHRASE environment variable to the value of *runpassphrase*.

Description

Reads from standard input or a specified text file and outputs an encrypted text file.

Lines to be encrypted may be specified between #ENCRYPT and #ENDCRYPT directives. If the *input_file* does not have #ENCRYPT and #ENDCRYPT directives, the **caltxtencrypt** program automatically adds corresponding directives at the beginning and end of the *input_file* and encrypts the entire file.

Examples

Example 1

This example creates an encrypted file named *layer.inc.enc* from the input file named *layer.inc*.

```
caltxtencrypt layer.inc layer.enc.inc
```

Example 2

This example uses the -r argument to specify the pass phrase “secret” to be associated with the encrypted file named *layer.enc.inc*.

```
caltxtencrypt layer.inc layer.enc.inc -r secret
```

To use the encrypted file for a Calibre run, you must set the CALIBRE_RUNPASSPHRASE environment variable to “secret”. For example:

```
setenv CALIBRE_RUNPASSPHRASE secret
```

Decryption fails if this variable is not set or the pass phrase is incorrect and Calibre issues the following message in the transcript:

```
Decryption impossible
```

Chapter 4
Using an Encrypted Rule File

The transcript output from an encrypted rule file is protected. Calibre does not write derived layer names to the transcript if they are defined within an encrypted area of the rule file.

General Example

In the general example, the derived layer name *1_8_sc_vias* in the Example Rule File does not appear in the Encrypted Rule File and is replaced with a generic label *ENCRYPTED_LAYER_1* in the Transcript Output.

Table 4-1. Calibre nmDRC/nmDRC-H Example (General)

Example Rule File	Encrypted Rule File [1]	Transcript Output
LAYOUT PATH 'layout.gds' LAYOUT SYSTEM GDSII LAYOUT PRIMARY '*' DRC RESULTS DATABASE '/dev/null' LAYER V 0 #ENCRYPT **1_8_sc_vias** { WITH NEIGHBOR V == 1 SPACE == .008 SQUARE CENTERS } #ENDCRYPT… #ENCRYPT pgate = poly and pplus ngate = poly and nplus nsd = nplus not poly psd = pplus not poly connect metal1 pwell connect metal1 nsub #ENDCRYPT … #ENCRYPT device mn ngate ngate(g) nsd(s) nsd(d) pwell(b) [debug 1-5 property W, L W = 0.5 * (perim_co(S, ngate) + perim_in(D, ngate)) const = 0.1 L = area(ngate) / W if (W > L) W = W - const * bends(ngate) * L Else L = L - const * bends(ngate) * W] #ENDCRYPT …	LAYOUT PATH 'layout.gds' LAYOUT SYSTEM GDSII LAYOUT PRIMARY '*' DRC RESULTS DATABASE '/dev/null' LAYER V 0 #DECRYPT '"?~;UP\ YPN5^LX@@" … Y'S-P(5"C2FA33U;/&]-1!!"CD?&TV … -[+,);6&&)E_:^#"',>RCA!!"*)+"T/Y,& … J3@V+V*[Y73<X(K\Y='!A!!!"R:+\JB … Z9QE-"8=*VDQV[FC`QNO-Q!!">7S] … #ENDCRYPT… #DECRYPT'"?~;UP\ YPN5^LX@@"T … 0FO.[Q(X\>I&TU/ I4+*0$!!!"0&3%\= … I*J57B8>063%<MD/ K`VRZA!!":)Y!] … E$0AYQ8S2]$$S!U-?%D)MQ!!"! BS*<;7\ QG.$*DQ43.$K_Q!!":E?3^ … >%%!P(N^Q_N>H5V[YD@^D!!!"Q … E$0AYQ8S2]$$S!U-?%D)MQ!!"! #ENDCRYPT … #DECRYPT '"?~;UP\ YPN5^LX@@" … 7]C`Z$O>O5U'T@AJZ+YD*1!!"_8!C … 9<NT6`W4)`GL3/.;6-$GD1!!"0_3AE1 … &HKCT``KNU*=!',6PP`3%Q!!"?AOS … 4M1<3;LST8@"Q'2_Q]1Y2Q!!"AN0 … H[*\ L;PD'#==,Y6]%?Q.$1!!"3P_C+ … [G;$9X4>&FT=\=%)PN,*,1!!"3GV9;L … *E;>>MLZ_J[,DV@!4`A!!!!"?/`RL"R* … ><]7B8'G"%U)-4DN'*-V;A!!">J7QC- … IHGOB<U1P`$8R(U"V+=5.!!!")KE8'(… 6PT/F[+DD+)"VS!(L!^,>!!!"0UFT,$U … :IPH`;)ME.N@_WL"-N0#/1!!"I`?I>E … 4)`GL3/.;6-$GD1!!"0``KNU*=!'=%)P … U*=!',6PP`3%Q!!"Z$O>O5U'T@AJZ … #ENDCRYPT …	… V = OR V -------- V (HIER TYP=1 CFG=1 HGC=12 FGC=12 VHC=F VPC=F) CPU TIME = 0 REAL TIME = 0 LVHEAP = 0/3/3 Original Layer V DELETED -- LVHEAP = 0/3/3 <ENCRYPTED OPERATION(S)> ------------------------ <**ENCRYPTED_LAYER_1**>::<1> (HIER TYP=1 CFG=1 HGC=2 FGC=2 VHC=F VPC=F) CPU TIME = 0 REAL TIME = 0 LVHEAP = 0/3/3 … … <ENCRYPTED OPERATION(S)> ------------------------ nsd (HIER TYP=1 CFG=3 HGC=5 FGC=5 VHC=F VPC=F) CPU TIME = 0 REAL TIME = 0 LVHEAP = 5/7/7 … <ENCRYPTED OPERATION(S)> ------------------------ psd (HIER TYP=1 CFG=3 HGC=5 FGC=5 VHC=F VPC=F) CPU TIME = 0 REAL TIME = 0 LVHEAP = 5/7/7 … <ENCRYPTED OPERATION(S)> ------------------------ pgate (HIER TYP=1 CFG=3 HGC=3 FGC=3 VHC=T VPC=F) CPU TIME = 0 REAL TIME = 0 LVHEAP = 5/7/7 Layer Pplus DELETED -- LVHEAP = 5/7/7 <ENCRYPTED OPERATION(S)> ------------------------ ngate (HIER TYP=1 CFG=3 HGC=3 FGC=3 VHC=T VPC=F) CPU TIME = 0 REAL TIME = 0 LVHEAP = 5/7/7 Layer Poly DELETED -- LVHEAP = 5/7/7 Layer Nplus DELETED -- LVHEAP = 5/7/7 DEVICE RECOGNITION TOTAL LAYER PREPARATION TIME: CPU TIME = 0 REAL TIME = 0 LVHEAP = 5/7/7 DEVICE RECOGNITION TOTAL PRECISE INTERACTION TIME: CPU TIME = 0 REAL TIME = 0 LVHEAP = 5/7/7 SEED PROMOTIONS: seed layer seeds cell : layers touched SEED PROMOTIONS: ------------ ----- ------------------------- EXTRACTED DEVICE PROPERTY COMPUTATION DEBUG OUTPUT is not displayed with encrypted rules DEVICE RECOGNITION completed. CPU TIME = 0 REAL TIME = 0 LVHEAP = 5/7/7 MALLOC = 8/8/8 …

1. The lines that show encryption are shortened so it is easier to display in this format.

Partial Encryption

When rule files containing partially encrypted operations are run, the transcript output is *not* protected.

Table 4-2. Litho Application Example (Partial Encryption)

Example Rule File	Partially Encrypted Operation[1]	Transcript Output
…DEF_OPC { LITHO OPC POLY FILE# ------ Simulation models ----- modelpath inlineopticalmodel inlineresistpolyfile inline	…DEF_OPC { LITHO OPC POLY FILE# --------- Simulation models --------modelpath inlineopticalmodel inlineresistpolyfile inline	…DEF_OPC::<1> = LITHO OPC POLY FILE [# ---------------- Simulation models --------------- modelpath inline opticalmodel inline resistpolyfile inline
# ------------- OPC algorithm ------- iterations 4 tilemicrons 160.000000 stepsize 0.001 gridsize 0.001 aspect 0	# ------------- OPC algorithm ----------iterations 4 tilemicrons 160.000000 stepsize 0.001 gridsize 0.001 aspect 0	# ------------------ OPC algorithm ----------------- iterations 4 tilemicrons 160.000000 stepsize 0.001 gridsize 0.001 aspect 0

Table 4-2. Litho Application Example (Partial Encryption) (cont.)

Example Rule File	Partially Encrypted Operation[1]	Transcript Output
#ENCRYPTcornerSiteStyle SITES_ON_ARC lineEndAdjDist 0.190000 convexAdjDist 0.140000 concaveAdjDist 0.140000 # ---------- Fragmentation ----------minfeature 0.180000 minedgelength 0.06 maxedgelength 1000.000000 cornedge 0.130000 …inline_optical1 {version 5engine TCCcalc#ENDCRYPT	#DECRYPT '"?~;UP\YPN5^LX@@"T" …P_<S7(53%3#(TK&;O=&:>!!!":#<'4>.B …$Q.$NF+W'&P14RM0+!8EAA!!"!FY+\ ……DS^MP=<),)U!/8LB.'+W!Q!!">62/^%@ …`0^K*[HNQB"%I7!MD\&A^1!!"H6KI9 …DER**/`'.?IQ)*2DH$*:5Q!!"T;-(`,"4, …#ENDCRYPT	
opticalsystem 0.248 0.600 0.750defocuslevels 8 0.000 0.100approxorder 10hoodpix 1.280kerngrid 0.010illumtype STANDARD}]}	opticalsystem 0.248 0.600 0.750defocuslevels 8 0.000 0.100approxorder 10hoodpix 1.280kerngrid 0.010illumtype STANDARD}]}	opticalsystem 0.248 0.600 0.750 defocuslevels 8 0.000 0.100 approxorder 10 hoodpix 1.280 kerngrid 0.010 illumtype STANDARD }]------------ ---// Litho Libraries XXXXXXXXXXXXXXScalar diffraction model used for image calculationImage calculated in airTCCcalc: Radial symmetry detected…The calculation of TCCs will be completed when the index below reaches ** 5 **0…1…2…3…4…5…TCC count: 6133SOCS accuracy vs order:1 0.6269102 0.7106483 0.8319024 0.854065…Starting cell 'lab1a'. Finished 6 of 7.Completed cell 'lab1a'. CPU TIME = 0, REAL TIME = 0. LVHEAP = 1/3/3.TIME FOR ALL CELLS: CPU TIME = 1 REAL TIME = 1NOTICE: USING NEW_MERGEDEF_OPC::<1> (HIER TYP=1 CFG=1 HGC=69 FGC=80 VHC=F VPC=F)CPU TIME = 9 REAL TIME = 10 LVHEAP = 1/3/3Layer DEF_OPC::<1> DELETED -- LVHEAP = 1/3/3DRC RuleCheck DEF_OPC COMPLETED. Number of Results = 69 (80)

1. The lines that show encryption are shortened so it is easier to display in this format.

Table 4-3. Calibre nmDRC/nmDRC-H Example (Partial Encryption)

Example Rule File	Partially Encrypted Operation[1]	Transcript Output
…1_8_sc_vias { WITH NEIGHBOR V == 1 #ENCRYPTSPACE == .008 SQUARE CENTERS #ENDCRYPT}	…1_8_sc_vias { WITH NEIGHBOR V == 1 #DECRYPT '"?~;UP\YPN5^LX@@"TB …J3@V+V*[Y73<X(K\Y='!A!!!"R:+\JBA …#ENDCRYPT}	…1_8_sc_vias::<1> = V WITH NEIGHBOR == 1 SPACE == 0.008 SQUARE CENTERS--1_8_sc_vias::<1> (HIER TYP=1 CFG=1 HGC=2 FGC=2 VHC=F VPC=F)CPU TIME = 0 REAL TIME = 0 LVHEAP = 0/3/3…

1. The lines that show encryption are shortened so it is easier to display in this format.

Inline Setup and Model Files

When the setup or model files are used inline within rules and the rules are encrypted, the inline setup or model file transcript output is protected.

Examples of this are shown in Table 4-4 and Table 4-6. When encrypted rules that use external setup or model files are run, the setup or model file transcript output is *not* protected (see Table 4-5 and Table 4-7).

Table 4-4. Calibre OPC, ORC, PRINTimage example (1)

Example Rule File[1] - Inline setup file - External model file	Encrypted Rule File[2]	Transcript Output
DRC SUMMARY REPORT "opc_poly.rep"DRC MAXIMUM VERTEX 199#ENCRYPTDEF_OPC{ LITHO OPC POLY FILE[# --------- Simulation models ------modelpath ./models opticalmodel n6s75d8 resistpolyfile n6s75d8.mod# ----------- OPC algorithm -------iterations 4tilemicrons 160.000000…b4_close_le -last .06#fragmentTag aft_close_le -first.06]}#ENDCRYPT	DRC SUMMARYREPORT "opc_poly.rep"DRC MAXIMUM VERTEX 199#DECRYPT ""?~;UP\YPN5^@"T" …`7ML)<IH&L_,);)WABY9HA!AK ……J>,,<5`)_WM/TAN0%Q8FTE3J= …:IPH`;)ME.N@_WL"-N0#/1!!>EX …Z9QE-"8=*VDQV[FC`QN!">S]B …E$0AYQ8S2]$$S!U-?%D)MQ!"!#ENDCRYPTPOLY {COPY POLY}DIFF {COPY DIFF}	<ENCRYPTED OPERATION(S)>-----------------------// Litho Libraries XXXXXXXX.............................NOTICE: Resist model support = -1 dbu = -0.001 micronsNOTICE: Post-processed site numx = 12, cenx = 3, numy = 0Starting cell 'CELL1'. Finished 0 of 7.Completed cell 'CELL1'. CPU TIME = 0, REAL TIME = 0. LVHEAP = 1/3/3.…Starting cell 'CELL4'. Finished 5 of 7.Completed cell 'CELL4'. CPU TIME = 0, REAL TIME = 0. LVHEAP = 1/3/3.Starting cell 'lab1a'. Finished 6 of 7.Completed cell 'lab1a'. CPU TIME = 0, REAL TIME = 0. LVHEAP = 1/3/3.TIME FOR ALL CELLS: CPU TIME = 1 REAL TIME = 1NOTICE: USING NEW_MERGEDEF_OPC::<1> (HIER TYP=1 CFG=1 HGC=69 FGC=80 VHC=F VPC=F)CPU TIME = 2 REAL TIME = 2 LVHEAP = 1/3/3Layer DEF_OPC::<1> DELETED -- LVHEAP = 1/3/3

1. Note that the customer must have the external model files (not encrypted) even though they are encrypted in the transcript.
2. The lines that show encryption are shortened so it is easier to display in this format.

Table 4-5. Calibre OPC, ORC, PRINTimage example (2)

Example Rule File -External setup/model file	Encrypted Rule File[1]	Transcript Output
…DRC RESULTS DATABASE "opc_poly.gds" GDSII PSEUDODRC SUMMARY REPORT "opc_poly.rep"DRC MAXIMUM VERTEX 199#ENCRYPTDEF_OPC { LITHO OPC FILE "opc_poly.in" POLY }#ENDCRYPTPOLY {COPY POLY}DIFF {COPY DIFF}DRC CHECK MAP POLY…	…DRC RESULTS DATABASE "opc_poly.gds" GDSII PSEUDODRC SUMMARY REPORT "opc_poly.rep"DRC MAXIMUM VERTEX 199#DECRYPT ""?~;UP\YPN5^LX@@" …`7ML)<IH&L_,);)WABY9HA!!"W[)5D …J)I2'L@2ZD99O@P=M4I1J!!!"3QK5`_ …Z9QE-"8=*VDQV[FC`QNO-Q!!">7S] …#ENDCRYPTPOLY {COPY POLY}DIFF {COPY DIFF}…	<ENCRYPTED OPERATION(S)>-----------------------// Litho Libraries XXXXXXXXXXXXX............................-- LITHO SETUP FILE INTERPRETER MODULE --- SETUP FILE = opc_poly.inmodelpath ./models…epeBias 0.000000limitThreshold 0--- LITHO SETUP FILE INTERPRETER MODULE COMPLETED -…TIME FOR ALL CELLS: CPU TIME = 1 REAL TIME = 1NOTICE: USING NEW_MERGEDEF_OPC::<1> (HIER TYP=1 CFG=1 HGC=69 FGC=80 VHC=F VPC=F)CPU TIME = 2 REAL TIME = 2 LVHEAP = 1/3/3Layer DEF_OPC::<1> DELETED -- LVHEAP = 1/3/3DRC RuleCheck DEF_OPC COMPLETED. Number of Results = 69 (80)

1. The lines that show encryption are shortened so it is easier to display in this format.

Table 4-6. Calibre OPC, ORC, PRINTimage example (3)

Example Rule File -Inline setup/model file (vtre)	Encrypted Rule File[1]	Transcript Output[2]
DRC SUMMARY REPORT "opc_poly.rep"DRC MAXIMUM VERTEX 199#ENCRYPTDEF_OPC { LITHO OPC POLY FILE[# --------- Simulation models --------modelpath inlineopticalmodel inlineresistpolyfile inline# ----------- OPC algorithm ----------iterations 4 tilemicrons 160.000000 …inline_resist {3 2 -0.22750 1.95625 -1.14754 0.09951 -0.41644 0.26119 0.391 0.980 2.098 4.349 0.295 0.411 origDcComponent -0.22750epeBias 0.000000}inline_optical1 {version 5engine TCCcalcopticalsystem 0.248 0.600 0.750defocuslevels 8 0.000 0.100approxorder 10hoodpix 1.280kerngrid 0.010illumtype STANDARD}]}#ENDCRYPTPOLY {COPY POLY}DIFF {COPY DIFF}	DRC SUMMARY REPORT "opc_poly.rep"DRC MAXIMUM VERTEX 199#DECRYPT "'?~;UP\YPN5^L" …`7ML)<IH&L_,);)WABY9HA!! ……-Q!!">7S]B&L[]7RT_>:1);7$I1! …E$0AYQ8S2]$$S!U-?%D)MQ! …#ENDCRYPTPOLY {COPY POLY}DIFF {COPY DIFF}	<ENCRYPTED OPERATION(S)>------------------------// Litho Libraries vXXXXXXXXX Scalar diffraction model used for image calculationImage calculated in airTCCcalc: Radial symmetry detected…The calculation of TCCs will be completed when the index below reaches ** 5 **0…1…2…3…4…5…TCC count: 6133SOCS accuracy vs order:1 0.6269102 0.7106483 0.8319024 0.8540655 0.8799856 0.9068037 0.9150168 0.924112… Generating lookup tables… Generating lookup tables… Generating lookup tables… Generating lookup tables… Generating lookup tables… NOTICE: Resist model support = -1 dbu = -0.001 micronsNOTICE: Post-processed site numx = 12, cenx = 3, numy = 0Starting cell 'CELL1'. Finished 0 of 7.Completed cell 'CELL1'. CPU TIME = 0, REAL TIME = 0. LVHEAP = 1/3/3.… Starting cell 'lab1a'. Finished 6 of 7.Completed cell 'lab1a'. CPU TIME = 0, REAL TIME = 0. LVHEAP = 1/3/3.TIME FOR ALL CELLS: CPU TIME = 1 REAL TIME = 1NOTICE: USING NEW_MERGEDEF_OPC::<1> (HIER TYP=1 CFG=1 HGC=69 FGC=80 VHC=F VPC=F)CPU TIME = 9 REAL TIME = 9 LVHEAP = 1/3/3Layer DEF_OPC::<1> DELETED -- LVHEAP = 1/3/3DRC RuleCheck DEF_OPC COMPLETED. Number of Results = 69 (80)

1. The lines that show encryption are shortened so it is easier to display in this format.
2. Although the inline models are encrypted, the size and type of density kernels and the number of SOCS kernels are transcripted.

Table 4-7. Calibre OPC, ORC, PRINTimage example (4)

Example Rule File -External setup/model file	Encrypted Rule File[1]	Transcript Output[2]
LAYER assistO 6LAYER assistM 7#ENCRYPTopc_output=LITHO OPC feature assistO assistM FILE [# --------- Simulation models --------modelpath inlineopticalmodel inlineresistpolyfile inline# ----------- OPC algorithm ----------iterations 0 tilemicrons 100.000000 stepsize 0.0025 gridsize 0.00125 siteinfo SAMPLE -numx 40 -center 15 cornerSiteStyle SITES_ON_EDGE	LAYER assistO 6LAYER assistM 7#ENCRYPTopc_output=LITHO OPC feature assistO assistM FILE [# --------- Simulation models -------- modelpath inlineopticalmodel inlineresistpolyfile inline# ----------- OPC algorithm ----------iterations 0 tilemicrons 100.000000 stepsize 0.0025 gridsize 0.00125 siteinfo SAMPLE -numx 40 -center 15 cornerSiteStyle SITES_ON_EDGE	<ENCRYPTED OPERATION(S)>------------------------// Litho Libraries XXXXXXXXBuilt-in function 'tophat 0.060000', 32400 iterations …doneBuilt-in function 'tophat 0.120000 0.060000', 32400 iterations …doneBuilt-in function 'tophat 0.180000 0.120000', 32400 iterations …doneBuilt-in function 'tophat 0.240000 0.180000', 32400 iterations …done…Built-in function 'tophat 0.720000 0.660000', 32400 iterations …doneBuilt-in function 'tophat 0.780000 0.720000', 32400 iterations …doneBuilt-in function 'tophat 0.840000 0.780000', 32400 iterations …doneBuilt-in function 'tophat 0.900000 0.840000', 32400 iterations …doneVector diffraction model used for image calculationImage calculated within film number: 1TCCcalc: Radial symmetry detected…The calculation of TCCs will be completed when the index below reaches ** 6 **0…1…2…3…4…5…6…TCC count: 8037SOCS accuracy vs order:1 0.593064
# ------------ Fragmentation ----------- minfeature 0.130000 minedgelength 0.100000 …#----- Arbitrary Commands Can Follow This Line. Don't delete this line! -----sse OPC_ORTHOG_SITES 0sse LITHO_NEW_OPC 1sse INTENSITY_INTERP_CENTER 1inline_resist {type VT-5version 3sampleSpacing 0.02referenceThreshold 0.23bound IMAX 0.269636 0.808342…	LAYER assistO 6LAYER assistM 7#DECRYPT "'?~;UP\YPN5^L" …CRGRD&85-5)X2A*NF7`PV! …T?C,96=NBB/7YJX)$Z".N1!!"/ ……E$0AYQ8S2]$$S!U-?%D)MQ! …:IPH`;)ME.N@_WL"-N0#/1!!"I…#ENDCRYPT	2 0.6995423 0.8776294 0.9388095 0.9444716 0.9504867 0.9550288 0.9600709 0.964198… Generating lookup tables… NOTICE: Resist model support = 142 dbu = 0.1775 micronsNOTICE: Post-processed site numx = 40, cenx = 15, numy = 3Starting cell 'Topcell'. Finished 0 of 1.Processing tile # 1 of 2.Processing tile # 2 of 2.Completed cell 'Topcell'. CPU TIME = 2, REAL TIME = 2. LVHEAP = 0/9/9.

Table 4-7. Calibre OPC, ORC, PRINTimage example (4) (cont.)

Example Rule File -External setup/model file	Encrypted Rule File[1]	Transcript Output[2]
TTERM 0.0207271 FACTOR 1 D13 1TTERM 0.0201529 FACTOR 1 D14 1}inline_optical1 {version 5engine TCCcalcopticalsystem 0.248 0.700…film 0.18 2.1676 -3.7887 "Poly Silicon (0.955P/ 0.045C) at 248.5 wavelength"film 0.0062 1.5084 0 "Silicon Oxide at 248.5 wavelength"substrate 1.587 -3.57beamfocus -0.01vectormodelflag 1}]#ENDCRYPT		TIME FOR ALL CELLS: CPU TIME = 2 REAL TIME = 2NOTICE: USING NEW_MERGEopc_output (HIER TYP=1 CFG=1 HGC=786 FGC=786 VHC=F VPC=F)CPU TIME = 38 REAL TIME = 38 LVHEAP = 0/9/9Opc_Output::<1> = COPY opc_output-------------------------------- Opc_Output::<1> (HIER TYP=1 CFG=1 HGC=786 FGC=786 VHC=F VPC=F)CPU TIME = 0 REAL TIME = 0 LVHEAP = 0/9/9Layer opc_output DELETED -- LVHEAP = 0/9/9Layer Opc_Output::<1> DELETED -- LVHEAP = 0/9/9

1. The lines that show encryption are shortened so it is easier to display in this format.
2. Although the inline models are encrypted, the size and type of density kernels and the number of SOCS kernels are transcripted.

Calibre nmDRC Results Database (ASCII)

Encrypted rules generate protected output in ASCII nmDRC results databases. Partially encrypted rules will generate non-protected output in ASCII for Calibre nmDRC results databases.

Table 4-8. ASCII Calibre nmDRC Results Database (Complete Encryption)

Example Rule File	Encrypted Rule File[1]	Non-encrypted nmDRC Results Database	Encrypted nmDRC Results Database
LAYOUT PATH 'layout.gds' LAYOUT SYSTEM GDSII LAYOUT PRIMARY '*' DRC RESULTS DATABASE 'drc_results' DRC CHECK TEXT ALL LAYER V 0 #ENCRYPT 1_8_sc_vias { @ this is a comment WITH NEIGHBOR V == 1 SPACE == .008 SQUARE CENTERS } #ENDCRYPT	LAYOUT PATH 'layout.gds' LAYOUT SYSTEM GDSII LAYOUT PRIMARY '*' DRC RESULTS DATABASE 'drc_results' DRC CHECK TEXT ALL LAYER V 0 #DECRYPT '"?~;UP\ YPN5^LX@… #K4&U/ 0#.'X31TW,+KLNTA!!"8… -[+,);6&&)E_:^#"',>RCA!!"*)+"T/ … J3@V+V*[Y73<X(K\Y='!A!!!"R: … Z9QE-"8=*VDQV[FC`QNO-Q!!"… .'X31TW,+KLNT+V*[Y73<X(K\ … #ENDCRYPT	PRIMARYCELL 1000 1_8_sc_vias 2 2 4 Mar 2 14:06:09 2004 1_8_sc_vias { @ this is a comment WITH NEIGHBOR V == 1 SPACE == .008 SQUARE CENTERS } p 1 4 549 1692 553 1692 553 1696 549 1696 p 2 4 557 1696 561 1696 561 1700 557 1700	PRIMARYCELL 1000 1_8_sc_vias 2 2 5 Mar 2 14:06:32 2004 1_8_sc_vias { <ENCRYPTED LINE> <ENCRYPTED LINE> <ENCRYPTED LINE> <ENCRYPTED LINE> } p 1 4 549 1692 553 1692 553 1696 549 1696 p 2 4 557 1696 561 1696 561 1700 557 1700

1. The lines that show encryption are shortened so it is easier to display in this format.

Table 4-9. ASCII Calibre nmDRC Results Database (Partial Encryption)

Example Rule File	Encrypted Rule File[1]	Non-encrypted nmDRC Results Database	Encrypted nmDRC Results Database
LAYOUT PATH 'layout.gds' LAYOUT SYSTEM GDSII LAYOUT PRIMARY '*' DRC RESULTS DATABASE 'drc_results' DRC CHECK TEXT ALL LAYER V 0 1_8_sc_vias { @ a comment #ENCRYPT WITH NEIGHBOR V == 1 SPACE == .008 SQUARE CENTERS #ENDCRYPT }	LAYOUT PATH 'layout.gds' LAYOUT SYSTEM GDSII LAYOUT PRIMARY '*' DRC RESULTS DATABASE 'drc_results' DRC CHECK TEXT ALL LAYER V 0 1_8_sc_vias { @ a comment #DECRYPT "'"?~;UP\YPN5^LX@… -[+,);6&&)E_:^#"',>RCA!!"*)+"T/… J3@V+V*[Y73<X(K\Y='!A!!!"R: … JBA>:T9]=522!SCD[%&*I&GU<… #ENDCRYPT }	PRIMARYCELL 1000 1_8_sc_vias 2 2 4 Mar 2 13:55:11 2004 1_8_sc_vias { @ a comment WITH NEIGHBOR V == 1 SPACE == .008 SQUARE CENTERS } p 1 4 549 1692 553 1692 553 1696 549 1696 p 2 4 557 1696 561 1696 561 1700 557 1700	PRIMARYCELL 1000 1_8_sc_vias 2 2 4 Mar 2 13:55:01 2004 1_8_sc_vias { @ a comment <ENCRYPTED LINE> <ENCRYPTED LINE> } p 1 4 549 1692 553 1692 553 1696 549 1696 p 2 4 557 1696 561 1696 561 1700 557 1700

1. The lines that show encryption are shortened so it is easier to display in this format.

GUI Applications

An encrypted rule file is not decrypted or displayed in the GUI. Encrypted Calibre® nmDRC and Calibre® nmLVS operations are not revealed in the GUI. Calibre® RVE™ and Calibre Interactive™ can read the database generated by an encrypted rule file, but do not display the encrypted information.

Figure 4-1. Calibre Interactive Example

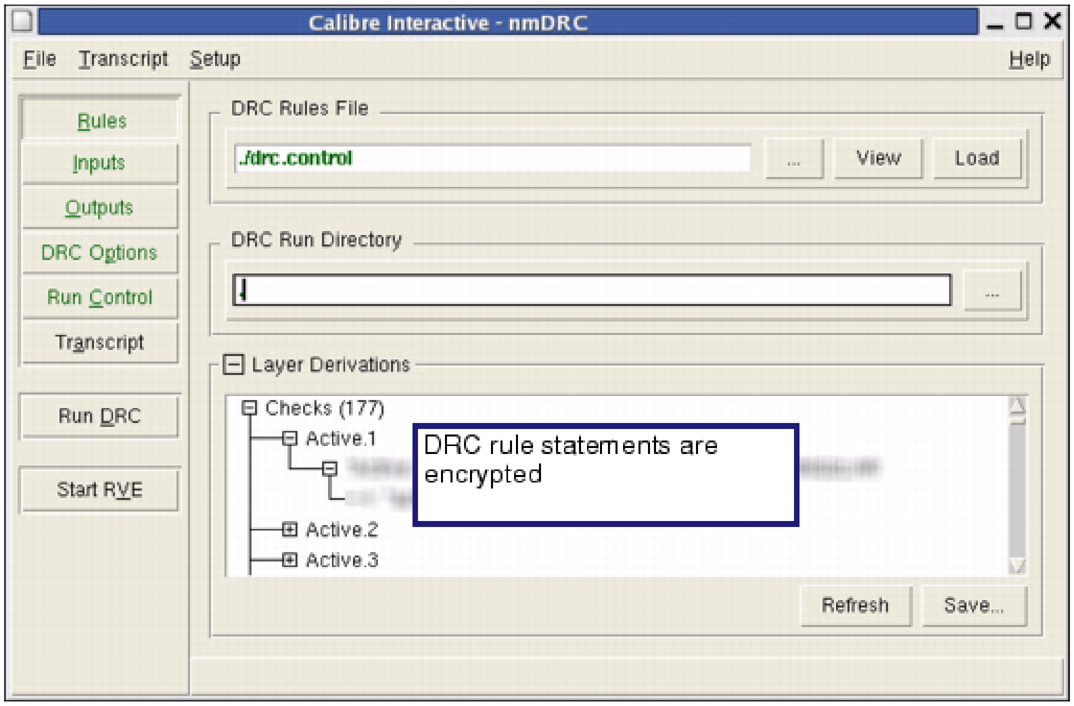
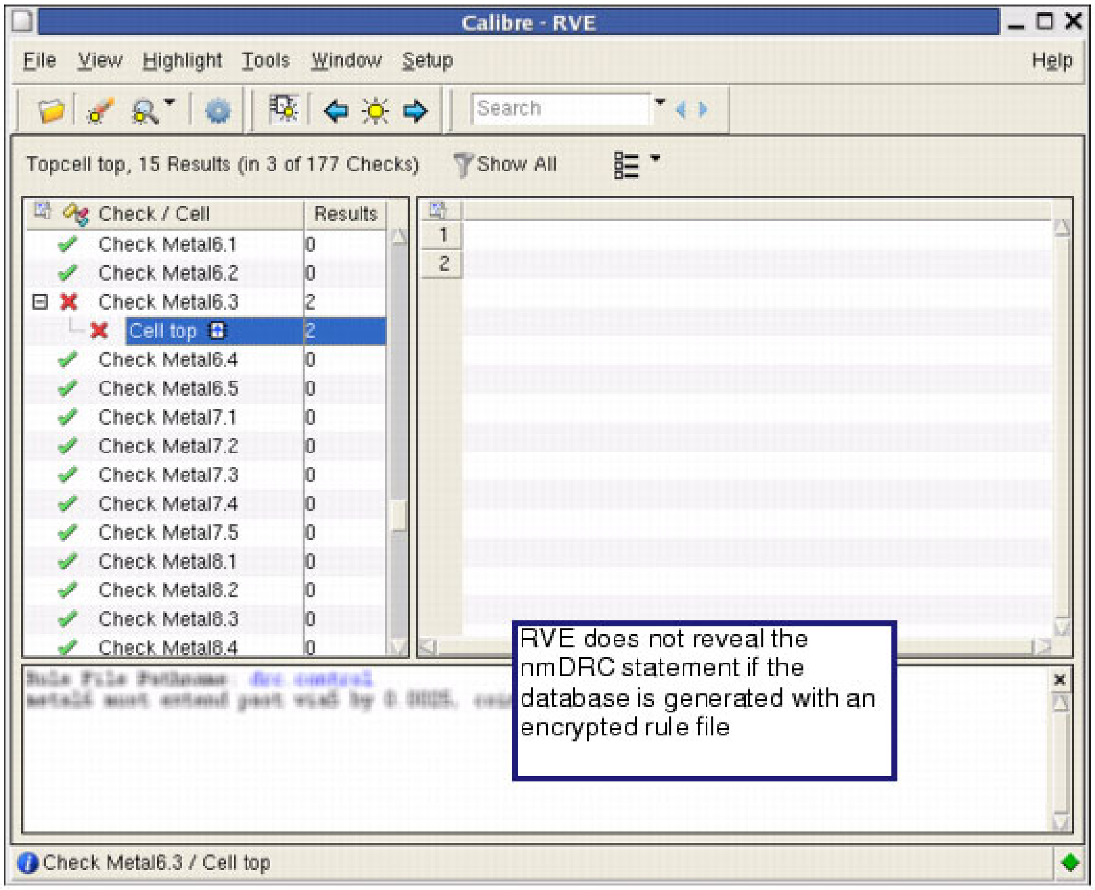


Figure 4-2. Calibre RVE Example



Chapter 5
Error Messages

Errors may occur when you encrypt a rule file or when running Calibre with an encrypted rule file.

Calibre SVRFencrypt Errors

You may see the following errors when encrypting an SVRF rule file.

Table 5-1. Calibre SVRFencrypt Error Descriptions

Error Text • Resolution
Undefined or empty environment variable • Ensure that the environment variable is set and resolvable.
ERROR: Error INCL1 on line <line_number> of <full_filename> - problem with access, file type, or file open of this include file: <included_filename>. • Ensure that the filename referenced in an INCLUDE statement is accessible.
ERROR: Error OPEN1 - problem with access, file type, or file open of file: <full_filename>. • Ensure that the rule file you are attempting to encrypt is accessible.
ERROR: Error INCL2 on line <line_number> of <full_filename> - this file name is included recursively: <included_filename>. • Do not specify INCLUDE statements that cause loops. For example: rule file A INCLUDEs rule file B, which INCLUDEs rule file A again.
ERROR: Error ENCRYPT2 on line <line_number> of <full_filename> - Nested #ENCRYPT (missing #ENDCRYPT?) • Ensure that you are not nesting #ENCRYPT and #ENDCRYPT directive pairs within each other.
ERROR: Error ENCRYPT3 on line <line_number> of <full_filename> - #ENDCRYPT without #ENCRYPT (missing #ENCRYPT?) • Ensure that you have an #ENDCRYPT directive for every #ENCRYPT directive.

Encrypted Rule File Errors

You may encounter errors when running Calibre with an encrypted SVRF rule file.

When you edit an encrypted rule file, you should be sure not to edit any of the encrypted lines between the #DECRYPT and #ENDCRYPT directives. The heading of an encrypted rule file specifically states:

```
Lines between the #DECRYPT and the #ENDCRYPT statements must not be
altered in any manner.
```

This includes copying the contents of an encrypted file to another encrypted file. Calibre issues an error if it determines that any of the encrypted lines are corrupted.

Refer to "Encrypted Rule File Errors" in the *Standard Verification Rule Format (SVRF) Manual* for explanations of possible encrypted rule file errors.

Index

Third-Party Information

Details on open source and third-party software that may be included with this product are available in the *<your_software_installation_location>/legal* directory.