

SIEMENS EDA

Calibre® Administrator's Guide

Software Version 2021.2
Document Revision 14

SIEMENS

Unpublished work. © 2021 Siemens

This material contains trade secrets or otherwise confidential information owned by Siemens Industry Software, Inc., its subsidiaries or its affiliates (collectively, "Siemens"), or its licensors. Access to and use of this information is strictly limited as set forth in Customer's applicable agreement with Siemens. This material may not be copied, distributed, or otherwise disclosed outside of Customer's facilities without the express written permission of Siemens, and may not be used in any way not expressly authorized by Siemens.

This document is for information and instruction purposes. Siemens reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Siemens to determine whether any changes have been made. Siemens disclaims all warranties with respect to this document including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement of intellectual property.

The terms and conditions governing the sale and licensing of Siemens products are set forth in written agreements between Siemens and its customers. Siemens' **End User License Agreement** may be viewed at: www.plm.automation.siemens.com/global/en/legal/online-terms/index.html.

No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Siemens whatsoever.

TRADEMARKS: The trademarks, logos, and service marks ("Marks") used herein are the property of Siemens or other parties. No one is permitted to use these Marks without the prior written consent of Siemens or the owner of the Marks, as applicable. The use herein of third party Marks is not an attempt to indicate Siemens as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A list of Siemens' trademarks may be viewed at: www.plm.automation.siemens.com/global/en/legal/trademarks.html. The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

Support Center: support.sw.siemens.com

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

Revision History

Revision	Changes	Status/ Date
14	Modifications to improve the readability and comprehension of the content. Approved by Lucille Woo. All technical enhancements, changes, and fixes listed in the <i>Calibre Release Notes</i> for this products are reflected in this document. Approved by Michael Buehler.	Released April 2021
13	Modifications to improve the readability and comprehension of the content. Approved by Lucille Woo. All technical enhancements, changes, and fixes listed in the <i>Calibre Release Notes</i> for this products are reflected in this document. Approved by Michael Buehler.	Released January 2021
12	Modifications to improve the readability and comprehension of the content. Approved by Lucille Woo. All technical enhancements, changes, and fixes listed in the <i>Calibre Release Notes</i> for this products are reflected in this document. Approved by Michael Buehler.	Released October 2020
11	Modifications to improve the readability and comprehension of the content. Approved by Lucille Woo. All technical enhancements, changes, and fixes listed in the <i>Calibre Release Notes</i> for this products are reflected in this document. Approved by Michael Buehler.	Released July 2020

<http://support.mentor.com>

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

Revision History

Chapter 1		
Introduction to Calibre Administration		17
About This Manual		17
Syntax Conventions		19
Chapter 2		
Configuration: Overview		21
Supported Operating Systems and Hardware		22
Supported Operating Systems		22
Minimum OS Patch Level		23
Mixed Operating Systems Environment		23
Supported x86-64 Hardware Configurations		23
Linux Distribution Compatibility Statement		25
Calibre Configuration Information		26
OPC Results Differences Between AOI and AOJ Calibre Trees		26
CPU Performance Settings for RHEL 7		26
RPM Package Manager (RPM) Requirements for Python and TensorFlow Support		27
RHEL 7 Requirements for Calibre_Xvfb Support		28
Older Calibre Versions and Newer x86-64 Hardware		28
Futex Issue Can Cause Calibre to Hang		29
Exceeding Opened File Limits		29
Sticky Bit Setting for Calibre		29
PTHREAD CREATE FAILURE for Calibre Runs on SLES 12		30
Variable CPU Speed Option for RHEL and SLES		30
Monitoring Multithreaded Calibre on Linux		30
Calibre Log File Configuration		30
Third-Party Software Information		32
Tcl Library in Calibre		32
Installation Information		33
Installing Calibre 2013.4 and Newer		33
Installing Calibre 2013.1 Through 2013.3		35
Installing Calibre 2012.2 Through 2012.4		36
Installing Calibre 2012.2 Through 2012.4 for Linux Platforms		36
Installing Calibre 2012.2 Through 2012.4 for Non-Linux Platforms		37
Installing Calibre 2009.3 Through 2012.1		38
Installing Calibre 2008.1 Through 2009.2		40
Installing Calibre 2007.2 Through 2007.4		42
Troubleshooting Installation		43

Chapter 3	
Configuration: Setting Up the Environment for Calibre Tools	45
CALIBRE_HOME Environment Variable	45
Setting the CALIBRE_HOME Environment Variable	46
Using the calibre_vco Utility	47
Running Calibre System Tests	47
Chapter 4	
Configuration: Managing Calibre Documentation	49
Documentation Directory Structure	49
Documentation Setup Options	51
Chapter 5	
Licensing: Overview	53
Calibre Licensing Requirements	53
Mentor Standard Licensing (MSL)	54
MSL License File and Environment Variables	54
Securing and Releasing Licenses	55
Lost Connection Between Calibre and the MSL License Server	55
Calibre Licensing Models	56
Calibre Classic Licensing	56
Calibre Loop Licensing	61
Calibre Licensing Configuration File	64
Calibre Licensing Configuration File Requirements	64
Retry Mode Classic	65
Retry Mode Loop	66
Calibre Licensing Command Line Options	68
Log Settings for Licensing	70
Licensing for Multithreaded Operations	71
Simultaneous Multithreading and Licensing Requirements	73
Calibre License Reference Description	74
Chapter 6	
Licensing: Queueing and Substitution	77
Setting Up Your Licensing Environment	77
Controlling Queueing for Calibre Classic Licensing	78
Disabling Queueing for Calibre Classic Licensing	78
Using License Substitution With Calibre Classic Licensing	79
Configuring Calibre Loop Licensing	80
Chapter 7	
Licensing: Calibre Platform and General Products	81
Calibre Cluster Manager (CalCM)	82
Calibre Cluster Manager Plus (CalCM+)	83
Calibre Dynamic Resource Allocator (DRA)	84
Calibre FullScale	85

Table of Contents

Chapter 8

Licensing: Physical Verification Products **87**

Calibre 3DSTACK	91
Calibre ADP (Advanced Device Properties)	92
Calibre AutoFix	94
Calibre Auto-Waivers	95
Calibre CB (Cell/Block)	96
Calibre CI (Connectivity Interface)	98
Calibre DESIGNrev	99
Calibre e2lvs	100
Calibre Interactive	101
Calibre Multi-Patterning	102
Calibre Multi-Patterning Advanced	103
Calibre nmDRC	105
Calibre nmDRCgold	106
Calibre nmDRC-H	107
Calibre nmLVS	109
Calibre nmLVS-H	110
Calibre nmLVS Advanced	111
Calibre nmLVS Reconnaissance	113
Calibre Pattern Matching	114
Calibre Pattern Matching Manufacturing	115
Calibre PERC	116
Calibre PERC Advanced	118
Calibre Physical Verification Suite (PVS)	119
Calibre Query Server	120
Calibre RealTime Custom	121
Calibre RealTime Custom Lite	122
Calibre RealTime Digital	123
Calibre Rule File Analyzer	124
Calibre RVE	125
Calibre SVRFencrypt	126
Calibre v2lvs	127
Tanner-Calibre One DRC/LVS	128

Chapter 9

Licensing: Design for Manufacturability (DFM) Products **129**

Calibre CMPAnalyzer	130
Calibre CMP Model Builder	131
Calibre LFD	132
Calibre YieldAnalyzer	133
Calibre YieldEnhancer	134
Calibre YieldServer	136

Chapter 10

Licensing: Parasitic Extraction Products **137**

Calibre License Broker Daemon (LBD) for Calibre xRC	138
lbd -xrc_flow	139

Using the LBD Server for Calibre xRC Jobs	140
Calibre xACT	142
Calibre xACT 3D	144
Calibre xACTView	145
Calibre xL	146
Calibre xRC	148
Calibre xRC CB	149
Calibre xRC to ADVance MS	150
Tanner-Calibre One xRC	151
xCalibrate Rule File Generator	152
Chapter 11	
Licensing: Resolution Enhancement Technology (RET) Products	153
Calibre Cell Array OPC	157
Calibre DSA Mask Synthesis	158
Calibre DSA Verification	159
Calibre EUV	160
Calibre LITHOview	161
Calibre LPE	162
Calibre mlOPC	163
Calibre MP Manufacturing	164
Calibre MP Manufacturing Advanced	165
Calibre nmCLBIAS	166
Calibre nmCLOPC	167
Calibre nmModelflow	168
Calibre nmOPC	170
Calibre nmSRAF	171
Calibre OPCpro	172
Calibre OPCsbar	173
Calibre OPCverify	174
Calibre OPCverify Classify Plus	175
Calibre ORC	176
Calibre PRINTimage	177
Calibre pxOPC	178
Calibre pxSMO	179
Calibre SONR	180
Calibre TDopc	181
Calibre WORKbench	182
Chapter 12	
Licensing: Mask Data Preparation (MDP) Products	183
Calibre D2DBDataPrep	185
Calibre DefectClassify	186
Calibre DefectReview	187
Calibre FRACTUREc	188
Calibre FRACTUREh	189
Calibre FRACTUREi	190
Calibre FRACTUREj	191

Table of Contents

Calibre FRACTUREm	192
Calibre FRACTUREn	193
Calibre FRACTUREp	194
Calibre FRACTUREt	195
Calibre FRACTUREv	196
Calibre Job Deck Editor	197
Calibre MASKOPT	198
Calibre MDP EMBED	199
Calibre MDP Embedded SVRF	200
Calibre MDPAutoClassify	201
Calibre MDPDefectAvoidance	202
Calibre MDPmerge	203
Calibre MDPstat	204
Calibre MDPverify	205
Calibre MDPview	206
Calibre Metrology API (MAPI)	207
Calibre MPCpro	208
Calibre MPCverify	209
Calibre nmCLMPC	210
Calibre nmMPC	211

Chapter 13

Distributed Calibre: Overview	213
Distributed Calibre Recommendations and Concepts	213
Multithreaded Processing Modes	216
Multithreaded (MT) Processing	218
Calibre MTflex Processing	219
Hyperscaling	220
Simultaneous Multithreading With Virtual CPUs	222
Remote Data Server	226
Hyper Remote	228
Calibre FullScale	233
Data Processing Modes	234
TURBOflex Data Processing — LAYOUT TURBO FLEX	234
ULTRAflex Data Processing — LAYOUT ULTRA FLEX	235
Configuration Requirements and Guidelines for Distributed Calibre	236
Hardware Configuration Guidelines	236
Network Configuration Guidelines	240
Calibre MTflex Network Setup	240
Calibre MTflex Network Management	243
Calibre MTflex Network Monitoring	244
Network Routing Examples	245
Linux Network Settings	245
Licensing Requirements for Distributed Calibre	249
Shared Memory Guidelines for Distributed Calibre	250
Calibre and Distributed Resource Management Tools	251
Job Checkpointing	251
Job Suspension	251

IBM Spectrum LSF Commands	253
Univa Grid Engine Commands	255
Known Issues	257
Output Differences	257
Exceeding a Shell’s File Descriptor Limit	257
NFSD Error “Too Many Open TCP Sockets”	258
Potential Network Errors	258
Chapter 14	
Distributed Calibre: Tool-Specific Support Information	259
Calibre Physical Verification Product Support for Multithreaded and Hyperscaling Modes	260
Calibre DFM Product Support for Multithreaded and Hyperscaling Modes	262
Calibre PEX Product Support for Multithreaded and Hyperscaling Modes	263
Calibre RET Product Support for Multithreaded and Hyperscaling Modes	264
Calibre MDP Product Support for Multithreaded and Hyperscaling Modes	266
Tool-Specific Information for Distributed Calibre	268
Calibre xRC Distributed Operations	268
Calibre OPCpro Distributed Operations	268
Calibre FRACTURE Distributed Operations	269
Calibre MDPmerge Distributed Operations	269
Calibre MDP Applications and Hyperscaling	270
Chapter 15	
Distributed Calibre: Creating a Configuration File	271
About the Configuration File	272
MGC_HOME Definition in the Remote Host Statement	272
xCalibrate Configuration File Requirements	273
Configuration File Statements	274
Using Environment Variables in a Configuration File	275
Chapter 16	
Distributed Calibre: Executing and Monitoring	277
Running MT Mode	277
Running Calibre MTflex in a Homogeneous Environment	279
Running Calibre MTflex in a Heterogeneous Environment	281
Using the LAUNCH CLUSTER and REMOTE COMMAND Statements	281
Using the LAUNCH MANUAL Statement and calibre -mtflex Command	283
Running MT and Calibre MTflex with Hyperscaling	285
Monitoring Remote Processes	288
Monitoring Distributed Calibre	292
Viewing PNG Files Using a Web Server	292
Viewing PNG Files Without a Web Server	293
Executing Calibre Using IBM Spectrum LSF	294
Executing Calibre Through IBM Spectrum LSF in MT Mode Using SMPs on the Same Host	
294	
Executing Calibre Through IBM Spectrum LSF in a Calibre MTflex Environment	295
Executing Calibre Through IBM Spectrum LSF for a Single CPU Run	297
Executing Calibre xRC Through IBM Spectrum LSF Using bsub Commands	298

Table of Contents

Executing xCalibrate With Calibre MTflex Using IBM Spectrum LSF	299
Comparing Calibre MTflex Runs	301
Running Calibre MTflex Without Hyperscaling	301
Running Calibre MTflex With Hyperscaling	302
Debugging Performance Issues Using a Calibre Log File.	305

Chapter 17

Distributed Calibre: Reference Dictionary **309**

Configuration File Reference Dictionary	310
HYPER	311
LAUNCH AUTOMATIC	313
LAUNCH CLUSTER	316
LAUNCH MANUAL	320
LOCAL HOST DIR	323
LOCAL HOST COMPUTE	324
MONITOR LOCAL	325
MONITOR REMOTE	329
MONITOR SYSTEM	332
MONITOR UTILITY	340
REMOTE COMMAND	341
REMOTE HOST	345
Transcript Reference	348
MONITOR LOCAL LOAD Transcript	349
Calibre MTflex Transcript	352
Distributed Calibre Environment Variables	354
CALIBRE_FLATTEN_LIKE_HYPER	355
CALIBRE_HDBFLEX_MIN	356
CALIBRE_HYPER_SMTFACTOR	357
CALIBRE_HYPER_RDSMULTIPLE	359
CALIBRE_HYPERFLEX_SAVE_LOGS	360
CALIBRE_LVS_REMOTE_CONNECTION	361
CALIBRE_MT_MONITOR_LOAD	362
CALIBRE_MT_MONITOR_MEMORY	363
CALIBRE_MT_SMT_FACTOR	364
CALIBRE_MTFLEX_HYPER	365
CALIBRE_MTFLEX_LAUNCH	366
CALIBRE_MTFLEX_LOCAL_HOST	367
CALIBRE_MTFLEX_LOCAL_HOST_DIR	368
CALIBRE_MTFLEX_MONITOR_LOCAL	369
CALIBRE_MTFLEX_MONITOR_REMOTE	370
CALIBRE_MTFLEX_REMOTE_DIR	371
CALIBRE_MTFLEX_REMOTE_MAXMEM	372
CALIBRE_MTFLEX_SAVE_LOGS	373
CALIBRE_REMOTE_CONNECTION	374
CALIBRE_REMOTE_COUNT	375
CALIBRE_REMOTE_MINCOUNT	376
Command Line Options Reference Dictionary	377
Calibre Command Line	377

-hyper	378
-mtflex	381
-remote	383
-remotecmd	385
-remotedata	387
-remotefile	391
-turbo	393
-turbo_all	394
-turbo_litho	395
Utilities	397
rcalibre	398
rxcalibrate	400
Errors and Warnings	402

**Appendix A
Calibre Environment Variables 405**

General and Administrative Environment Variables	406
MT and Calibre MTflex Environment Variables	407
Licensing Environment Variables	409
Documentation Environment Variables	410
Calibre DESIGNrev Application Environment Variables	411
Calibre nmDRC Application Environment Variables	412
Calibre nmLVS Application Environment Variables	413
Calibre OPCpro Environment Variables	414
Calibre PERC Environment Variables	415
Third Party Database Environment Variables	416

**Appendix B
Configuration: Operating Systems and Virtual Memory 417**

Memory Usage Best Practices	417
Virtual Memory Allocation Failures	417

Glossary**Index****Third-Party Information**

List of Figures

Figure 4-1. Calibre Documentation Directory Structure	50
Figure 5-1. MSL License Server Lost Connection Process.....	55
Figure 5-2. Calibre Classic License Queueing and Substitution Scheme	58
Figure 5-3. License Dropdown With Leveling	59
Figure 5-4. Queueing and Dropdown Example for 2-CPU OPC Run.....	60
Figure 5-5. Calibre Loop Licensing.....	62
Figure 5-6. Standard Multithreaded License Requirements Formula	71
Figure 10-1. Calibre xACT License	143
Figure 13-1. Distribution of a Design and SVRF Operations in Hyperscaling Mode.....	221
Figure 13-2. Hyper Remote Configuration of Pseudo HDBs and Remote Data Servers on a Medium Size (24-63 CPUs) Remote Cluster.....	229
Figure 13-3. Hyper Remote Configuration of Pseudo HDBs and Remote Data Servers on a Small (<24) Remote Cluster.....	230
Figure 13-4. Comparison Running Hyper Remote on a Small and Medium Remote Cluster	231
Figure 13-5. Executing Hyper Remote With Less Than Four Pseudo HDBs	232
Figure 13-6. Hyper Remote Transcript Running With Sufficient Resources	232
Figure 13-7. Hyper Remote Transcript Running With a 32-Bit Remote Host	233
Figure 13-8. Calibre nmDRC/nmDRC-H Customer Trends	238
Figure 13-9. Recommended Network Topology	241
Figure 13-10. Recommended Network Topology - Tiered Switches	242
Figure 17-1. Example RRDGRAPH Output of the Load Average on a Primary Host	338
Figure 17-2. Example RRDGRAPH Output of the Load Average on a Remote Machine ..	339

List of Tables

Table 1-1. Syntax Conventions	19
Table 2-1. Calibre 2021.2 Supported Operating Systems	22
Table 2-2. Supported Processors	23
Table 2-3. Platform Requirements for AOI	24
Table 2-4. Platform Requirements for AOJ	24
Table 2-5. Platform Requirements for AOK	25
Table 4-1. Naming Conventions for the Calibre Doc Tree Directories and Files	50
Table 5-1. Single Threaded and Multithreaded License Queueing Comparison	57
Table 5-2. CALIBRE_LM_LOG_LEVEL Environment Variable Settings	70
Table 5-3. License Requirements per # of CPUs	71
Table 5-4. Two-CPU Licenses	72
Table 5-5. License Consumption for SMT-Enabled Hardware With Eight CPUs	73
Table 7-1. Primary License Requirements for Calibre General Products	81
Table 8-1. Primary License Requirements for Physical Verification Products	87
Table 8-2. Calibre nmLVS Advanced License CPU Allocation	111
Table 9-1. Primary License Requirements for Design for Manufacturability (DFM) Products 129	
Table 10-1. Primary License Requirements for Parasitic Extraction Products	137
Table 10-2. CPUs Enabled by Calibre xACT Licenses	142
Table 10-3. CPUs Enabled by License Count	144
Table 10-4. CPUs Enabled When Calibre xL Is Launched From Calibre xACT	146
Table 10-5. CPUs Enabled When Calibre xL Is Launched From Calibre xRC and PEEC Inductance Extraction Mode Is Enabled	146
Table 10-6. CPUs Enabled by Calibre xRC Licenses	148
Table 10-7. CPUs Enabled by Calibre xRC to ADVance MS Licenses	150
Table 11-1. Primary License Requirements for Resolution Enhancement Technology (RET) Products	153
Table 12-1. Primary License Requirements for Mask Data Preparation (MDP) Products ..	183
Table 13-1. Overview of Multithreaded Processing Modes	216
Table 13-2. HDBs Created for Hyperscaling Runs in Multithreaded Mode	222
Table 13-3. Remote Compute Servers (RCS) per CPU for Calibre MTflex Clusters Using Hyperscaling	222
Table 13-4. Calibre Runtime Benefit From Simultaneous Multithreading	222
Table 13-5. SMT Environment Variable and Configuration File Setting Summary	224
Table 13-6. RDS and HDB Multipliers for Hyper Remote	228
Table 13-7. Overview of Calibre Data Processing Modes	234
Table 13-8. Recommended Hardware Specifications for and Remote Hosts	236
Table 13-9. Calibre MTflex Network Management Preventative Measures	244
Table 13-10. Linux Network Kernel and Driver Recommended Settings	247
Table 13-11. Commonly Used IBM Spectrum LSF Commands	253

Table 13-12. Commonly Used Univa Grid Engine Commands	255
Table 13-13. Guidelines for Calculating the maxproc Setting	258
Table 14-1. Calibre Physical Verification Product Support for Multithreaded and Hyperscaling Modes	260
Table 14-2. Calibre Design for Manufacturability (DFM) Product Support for Multithreaded and Hyperscaling Modes	262
Table 14-3. Calibre Parasitic Extraction (PEX) Product Support for Multithreaded and Hyperscaling Modes	263
Table 14-4. Calibre Resolution Enhancement Technology (RET) Product Support for Multithreaded and Hyperscaling Modes	264
Table 14-5. Calibre Mask Data Preparation (MDP) Product Support for Multithreaded and Hyperscaling Modes	266
Table 15-1. Configuration File Statements	274
Table 16-1. Options for Running Calibre MTflex in a Heterogeneous Environment	281
Table 16-2. bsub Command Line Arguments	295
Table 17-1. Calibre MTflex Error Messages	402
Table 17-2. Calibre MTflex Warning Messages	403
Table A-1. General and Administrative Environment Variables	406
Table A-2. MT and Calibre MTflex Environment Variables	407
Table A-3. Licensing Environment Variables	409
Table A-4. Documentation Environment Variables	410
Table A-5. General and Administrative Environment Variables	411
Table A-6. Calibre nmDRC Application Environment Variables	412
Table A-7. Calibre nmLVS Application Environment Variables	413
Table A-8. Calibre OPCpro Application Environment Variables	414
Table A-9. Calibre PERC Application Environment Variables	415
Table A-10. Calibre Third Party Database Environment Variables	416

Chapter 1

Introduction to Calibre Administration

Administration tasks include installing and configuring the Calibre software, installing and configuring licensing, and configuring Calibre jobs to run in a distributed environment.

About This Manual	17
Syntax Conventions	19

About This Manual

The audience for this manual includes individuals that are familiar with the systems and network at their site, and with the Calibre tools. Administrative tasks for Calibre include configuring the network and hardware, installing the Calibre software, installing and configuring the licensing environment, and configuring Calibre jobs to run in a distributed environment.

Manual Organization

- **Configuration**

This information is provided in Chapters 2 through 4, and covers the following topics:

- “[Supported Operating Systems and Hardware](#)” on page 22
- “[Calibre Configuration Information](#)” on page 26
- “[Installation Information](#)” on page 33
- “[Configuration: Setting Up the Environment for Calibre Tools](#)” on page 45
- “[Configuration: Managing Calibre Documentation](#)” on page 49

Related Appendices:

- “[Calibre Environment Variables](#)” on page 405
- “[Configuration: Operating Systems and Virtual Memory](#)” on page 417

- **Licensing**

This information is provided in Chapters 5 through 12, and covers the following topics:

- “[Mentor Standard Licensing \(MSL\)](#)” on page 54
- “[Calibre Licensing Models](#)” on page 56

- “[Log Settings for Licensing](#)” on page 70
- “[Calibre Licensing Configuration File](#)” on page 64
- Licensing information for the following products:
 - [Physical Verification products](#)
 - [Design for Manufacturability \(DFM\) products](#)
 - [Parasitic Extraction products](#)
 - [Resolution Enhancement Technology \(RET\) products](#)
 - [Mask Data Preparation \(MDP\) products](#)
- **Distributed Calibre**

This information is provided in Chapters 13 through 17, and covers the following topics:

 - “[Multithreaded Processing Modes](#)” on page 216
 - “[Configuration Requirements and Guidelines for Distributed Calibre](#)” on page 236
 - “[Tool-Specific Information for Distributed Calibre](#)” on page 268
 - “[Distributed Calibre: Creating a Configuration File](#)” on page 271
 - “[Distributed Calibre: Executing and Monitoring](#)” on page 277
 - “[Errors and Warnings](#)” on page 402

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.
<i>Italic</i>	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.
<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:	
<pre>DEvice {element_name [‘(‘model_name‘)’]}</pre> <pre>device_layer {pin_layer [‘(‘pin_name‘)’] ...}</pre> <pre>[‘<auxiliary_layer> ...]</pre> <pre>[‘(‘swap_list‘)’ ...]</pre> <pre><u>[BY NET BY SHAPE]</u></pre>	

Chapter 2

Configuration: Overview

Calibre runs on different hardware architectures and operating systems and, depending on your environment, may require you to perform some specific configuration tasks.

Supported Operating Systems and Hardware	22
Calibre Configuration Information	26
Third-Party Software Information	32
Installation Information	33

Supported Operating Systems and Hardware

The Calibre tools run on various combinations of operating systems and hardware architectures.

For the latest information on supported platforms and OS versions for Calibre tools, refer to the “Calibre Platform Support Overview and Roadmap” at:

http://calibre.mentorcloudservices.com/docs/Calibre_OS_Roadmap.htm

The roadmap provides information on the combinations of hardware architectures and operating systems that Calibre currently supports and intends to support in the upcoming year.

Supported Operating Systems	22
Minimum OS Patch Level.....	23
Mixed Operating Systems Environment	23
Supported x86-64 Hardware Configurations	23
Linux Distribution Compatibility Statement	25

Supported Operating Systems

The Calibre software is packaged into three CALIBRE_HOME executables: AOI, AOJ and AOK.

Table 2-1. Calibre 2021.2 Supported Operating Systems

CALIBRE_HOME Tree	Supported Operating System	Support Center Download Name	Executable Filename
AOI	RHEL/CentOS 6.8-6.10 RHEL/CentOS 7.3-7.9 SLES 11sp4 SLES 12sp5	Linux x86-64 for RHEL 6/7 & SLES 11sp4 with documentation	<i>aoi_cal_2021.2_<xx.xx>_mib.exe</i>
AOJ	RHEL/CentOS 6.8-6.10 RHEL/CentOS 7.3-7.9 SLES 11sp4 SLES 12sp5	Linux x86-64 for RHEL 6/7 & SLES 11sp4, SLES 12sp5 with documentation	<i>aoj_cal_2021.2_<xx.xx>_mib.exe</i>
AOK	RHEL/CentOS 8	Linux x86-64 for RHEL 8 with documentation	<i>aok_cal_2021.2_<xx.xx>_mib.exe</i>

Note

 Calibre support for Red Hat® Enterprise Linux® (RHEL) is determined by the operating system version, for example, RHEL 7.2. Red Hat uses different terminology for marketing RHEL, including “Server,” “Workstation,” and “Desktop.” These different options are supported as long as a supported operating system version is loaded. RHEL for IBM POWER systems is not supported. Also, Red Hat Fedora® Linux is maintained, distributed, and supported differently than RHEL, and is not supported for use with the Calibre product line. Any use of the Red Hat Fedora operating system is governed by the “[Linux Distribution Compatibility Statement](#)” on page 25.

Minimum OS Patch Level

For each OS version, the Calibre tools require a minimum OS patch level. Siemens EDA tests Calibre tools with the vendor-recommended OS patch sets. However, given the number of OS patches and the frequency of updates, Siemens EDA cannot test the Calibre tools with all possible versions and variations of a vendor’s OS patches. If you deviate from the OS patch sets recommended by Siemens EDA, there is no guarantee the Calibre tools will function correctly. Nevertheless, most OS vendors ensure, when possible, forward compatibility of patch versions.

Note

 Siemens EDA does not provide vendor-specific OS patches. If you require a specific patch, consult the OS vendor.

Mixed Operating Systems Environment

The different executables for the AOI, AOJ, and AOK trees are optimized for different operating system and hardware environments. A mixed operating system environment is not supported when using these trees.

If your network includes different distributions of Linux as listed in “[Supported Operating Systems](#)” on page 22, you can download and install the executables to a target directory. You then set the CALIBRE_HOME environment variable to the appropriate trees (see “[CALIBRE_HOME Environment Variable](#)” on page 45).

Supported x86-64 Hardware Configurations

The Calibre tools are qualified and supported on x86-64 processors supplied by different vendors.

Table 2-2. Supported Processors

Vendor	Processor
Advanced Micro Devices	AMD64

Table 2-2. Supported Processors (cont.)

Vendor	Processor
Intel®	Intel® Xeon®
Intel®	Intel® fourth generation core architecture processor (also known as the Haswell processor family)

The following tables identify the deprecated and supported operating systems for the AOI, AOJ, and AOK VCOs, respectively. A deprecated state indicates that Calibre is supported on the operating system, but that support will be discontinued for that OS within the next year.

Note

 It is recommended that you move to a newer operating system as soon as possible in order to maintain support by both the OS vendor and Calibre.

Table 2-3. Platform Requirements for AOI

OS	Support by Calibre Tools	Processor	Minimum OS Patch Level and Required Patches	Recommended OS Patch Level and Required Patches
RHEL/ CentOS 6.8- 7.6	Supported	AMD64 Intel Xeon	Use vendor's base load.	Same as Minimum.
RHEL/ CentOS 7.7	Supported			
SLES 11sp4	Supported			
SLES 12sp5	Supported			

Table 2-4. Platform Requirements for AOJ

OS	Support for Calibre Tools	Processor	Minimum OS Patch Level and Required Patches	Recommended OS Patch Level and Required Patches
RHEL/ CentOS 7.7- 7.9	Supported	Intel® fourth generation core architecture processor (also known as the Haswell processor family) or later.	Use vendor's base load.	Same as Minimum.
SLES 12sp5	Supported			

Table 2-5. Platform Requirements for AOK

OS	Support for Calibre Tools	Processor	Minimum OS Patch Level and Required Patches	Recommended OS Patch Level and Required Patches
RHEL/CentOS 8.2	Supported	Intel® fourth generation core architecture processor (also known as the Haswell processor family) or later.	Use vendor's base load.	Same as Minimum.

Linux Distribution Compatibility Statement

If you encounter problems using Siemens EDA tools on an unsupported Linux distribution, and the problem can be duplicated on a supported production Linux distribution, it will be addressed through normal Siemens EDA support policies. If a problem only exists on the unsupported Linux distribution, the fix lies in a change to the Linux environment, not the Siemens EDA product. In this case, you should contact your Linux supplier.

Any other version, update or service pack level which is not listed for a particular hardware platform is not supported, and any use is covered by the Linux Clone Compatibility Statement.

Calibre Configuration Information

Configuration tasks may include configuring virtual machines in order to run older versions of Calibre on newer hardware, monitoring multithreaded Calibre runs, and configuring the log file output.

OPC Results Differences Between AOI and AOJ Calibre Trees	26
CPU Performance Settings for RHEL 7	26
RPM Package Manager (RPM) Requirements for Python and TensorFlow Support ..	27
RHEL 7 Requirements for Calibre_Xvfb Support.....	28
Older Calibre Versions and Newer x86-64 Hardware	28
Futex Issue Can Cause Calibre to Hang	29
Exceeding Opened File Limits	29
Sticky Bit Setting for Calibre	29
PTHREAD CREATE FAILURE for Calibre Runs on SLES 12	30
Variable CPU Speed Option for RHEL and SLES	30
Monitoring Multithreaded Calibre on Linux	30
Calibre Log File Configuration	30

OPC Results Differences Between AOI and AOJ Calibre Trees

Differences may be observed in the results from OPC jobs that are run on an AOI and an AOJ Calibre tree. The differences in results are because of the different hardware processors and operating system libraries that are supported by the different Calibre trees.

CPU Performance Settings for RHEL 7

If you experience a performance degradation when running Calibre on an RHEL 7 operating system, there is a utility and profile available with RHEL 7 that you can use to improve performance.

In RHEL 7, there is a daemon process named “tuned” that monitors connected devices and tunes system settings according to a selected profile. You may experience a performance degradation with Calibre depending on the profile you are using. RHEL 7 includes a utility named “[tuned-adm](#)” that you can use to modify the active profile on a physical system. To use the tuned-adm utility, the tuned daemon process must be running and the utility must be executed as root.

A server profile named “throughput-performance” is included with RHEL 7 that focuses on improving throughput and is recommended for most systems. To view the currently activated profile, enter the following command:

```
# /sbin/tuned-adm active
```

If the throughput-performance profile is active, you will see the following message:

```
Current active profile: throughput-performance
```

If Calibre detects that tuned-adm exists on the host machine, and the performance profile is not set to throughput-performance (or virtual-guest on VMs, where this setting cannot be changed), then this warning is given in the transcript:

```
WARNING: Please enable throughput-performance profile for best performance
```

To activate the throughput-performance profile, enter the following command:

```
# /sbin/tuned-adm profile throughput-performance
```

For more information on tuned, refer to the [RHEL 7 Performance Tuning Guide](#). Also see the tuned-adm “man” page.

RPM Package Manager (RPM) Requirements for Python and TensorFlow Support

To support Python and TensorFlow usage, some RPM packages, which contain libraries needed for Python and TensorFlow, must be added to some of the standard SLES environments. You should also ensure your environment is properly configured for running the Calibre version of Python.

OS Version and Patch Level

To check the OS version and patch level, use the **cat** command to print the contents of the */etc/SuSE-release* file. For example:

```
% cat /etc/SuSE-release
SUSE Linux Enterprise Server 11 (x86_64)
VERSION = 11
PATCHLEVEL = 3
%
```

SLES 11 sp4 RPM Packages

- libgfortran3-32bit
- libquadmath0-32bit
- unixODBC_23

The unixODBC_23 package can be found on the distributed OS media, and the libgfortran3-32bit and libquadmath0-32bit packages can be found on the SLES 11 SDK (Software Development Kit) media.

Python

Calibre uses the version of Python located at *\$MGC_HOME/bin/calpython3*. If you encounter issues running Python in a Calibre environment, check your setting for the PYTHONSTARTUP environment variable:

```
% echo $PYTHONSTARTUP
```

You should unset this environment variable if it is defined.

RHEL 7 Requirements for Calibre_Xvfb Support

The xorg-x11-fonts-misc-* package is required in order for Calibre_Xvfb (virtual framebuffer X) to work. Releases prior to RHEL 7 loaded the xorg-x11-fonts-misc-* package by default. For Calibre_Xvfb to work on RHEL 7 systems, you must load the xorg-x11-fonts-misc-* package from the RHEL 7 distribution.

Older Calibre Versions and Newer x86-64 Hardware

Some customers (for example, foundries) are required to run older versions of Calibre to support production lines. Many of these same customers also want to upgrade to newer hardware to take advantage of performance improvements and power savings. However, newer x86-64 servers have processors that require more recent versions of Linux.

In addition, customers may not wish to spend the time nor incur the risk of re-qualifying an existing production environment on a new operating system version or a new Calibre version. Therefore, a legacy support strategy based on virtual machines (VMs) is recommended for customers who are required to run old versions of Calibre on new servers. While it is nearly always better to move forward with newer hardware and associated newer software, when that is not an option, a VM can be a suitable solution. In-house Calibre VM testing is done using the VMware ESX server.

There is a difference in licensing behavior between VMware and Red Hat KVM virtualization solutions when running a Calibre job using both physical and virtual CPUs. With VMware hypervisor, Calibre is prevented from distinguishing between physical and virtual CPUs and, as a result, the license requirements are based on the number of connected physical and virtual CPUs. Red Hat KVM hypervisor allows Calibre to distinguish between physical and virtual CPUs, so the licenses requirements are based only on the number of connected physical CPUs.

The Calibre product division will make a reasonable effort to support Linux VM usage (for example, VMware) on x86-64 hardware and will perform due diligence to assist our customers

when problems arise. If no solution or workaround is found and the problem does not occur when using physical hardware, you should follow up with your virtualization software vendor.

Using a VM of the older OS version on new, fast hardware provides the benefit of improved performance while still retaining the exact supported environment needed by the older software. Therefore, there is no risk of OS-introduced issues and no time needed for requalification. For more information, contact Calibre Customer Support.

Futex Issue Can Cause Calibre to Hang

There is a futex issue with the fourth generation Intel CPU. If you have this CPU, there are RHEL kernel versions available that include a fix.

If your system does not have such a kernel, then this warning is issued in the Calibre transcript:

```
WARNING: Detected an OS version on node <node> that may be incompatible
with identified hardware for multi-threaded applications, including
Calibre. Recommend upgrading to RHEL6 - 2.6.32-504.16.2.el6 or RHEL7 -
3.10.0-229.7.2.el7 or more recent versions.
```

If you continue to run with an OS that triggers this warning, a Calibre run can hang.

For more information, refer to:

- Calibre support

<https://support.sw.siemens.com/en-US/product/852852053/knowledge-base/MG590923?pid=sc%3Apc-typeahead?pid=sc%3Apc-typeahead>

- RedHat Customer Portal

<https://access.redhat.com/solutions/1386323>

Exceeding Opened File Limits

Each OS has limits on the number of files you can simultaneously have open. During normal operation, some Calibre tools can exceed this limit and produce undesirable results. Check your OS vendor's documentation if you experience issues with exceeding this OS limit.

Sticky Bit Setting for Calibre

For Calibre 2011.2 and later releases, Calibre tools will not invoke on Linux platforms with a sticky bit Calibre executable setting.

If the sticky bit is set, you may receive the following error:

```
error while loading shared libraries cannot open shared object file:
No such file or directory
```

The error occurs because Linux does not enable you to use a library path from a user environment when running an executable with the sticky bit set. To fix this issue, the Linux access control list (ACL) sticky bit must be removed by your system administrator.

PTHREAD CREATE FAILURE for Calibre Runs on SLES 12

In some cases, a Linux kernel setting for the SLES 12 operating system generates the message “PTHREAD CREATE FAILURE” in the transcript for a Calibre run.

The DefaultTasksMax option is an operating system security feature that is intended to keep one service from consuming all the server resources or spawning too many threads. The default value may be too low for Calibre to work effectively. To mitigate this behavior, set the DefaultTasksMax option as follows in the */etc/system/system.conf* file:

```
DefaultTasksMax=infinity
```

Variable CPU Speed Option for RHEL and SLES

The “cpuspeed” option for RHEL operating systems and the “CPUFREQ_ENABLED” option for SLES operating systems provide a variable speed capability that enables CPUs to throttle down below their full speed when they are not being utilized. You should disable these options to avoid the performance impact due to the ramp up of the CPUs when they are engaged. For RHEL operating systems, “cpuspeed” is the name of the process. For SLES-based operating systems, the CPUFREQ_ENABLED option is disabled in the powersave configuration file.

Because of the great variability of processor, BIOS, and operating system features, Siemens EDA does not provide support for problems with power save features. Contact your server vendor to discuss options.

Monitoring Multithreaded Calibre on Linux

When performing multithreaded Calibre runs on Linux platforms, you can monitor the thread status using either the ps or top command. The top command can give you misleading results on thread or process memory usage because it shows each thread the OS spawns from the primary thread; however, the memory size is identical to the primary thread.

Calibre Log File Configuration

You can use the CALIBRE_PRINT_TIME_STAMPS environment variable to configure Calibre to output an absolute time stamp to the log file. Set this variable to any value (including nil) to have Calibre output the date and time after the ELAPSED TIME.

The following example shows a comparison of the output with and without the use of CALIBRE_PRINT_TIME_STAMPS.

- With CALIBRE_PRINT_TIME_STAMPS:

```
CPU TIME = 0  REAL TIME = 0  LVHEAP = 7/34/35  OPS COMPLETE = 192 OF
193  ELAPSED TIME = 35  (2013/01/22 08:28:09)
```

- Without CALIBRE_PRINT_TIME_STAMPS:

```
CPU TIME = 0  REAL TIME = 0  LVHEAP = 7/34/35  OPS COMPLETE = 192 OF
193  ELAPSED TIME = 35
```

Third-Party Software Information

Some Calibre tools use third-party software to implement some of the interfaces in Calibre and provide customization capabilities. Occasionally, there are updates to third-party software packages that you should be aware of.

Tcl Library in Calibre..... 32

Tcl Library in Calibre

Calibre uses Tcl 8.6 starting with the 2019.2 release. Tcl is used to implement a number of interfaces in Calibre and several Calibre products use Tcl scripts to control execution and tool flow.

Tcl is an evolving open source language. Future versions of Tcl that are adopted by Calibre may not run some of the Tcl code that ran under a previous version of Tcl. For this reason, it is important to understand that some maintenance may be required for an older version of Tcl code to work with a new version of Calibre that uses a later version of Tcl.

Installation Information

Depending upon the Calibre release you plan to install, slightly different steps may be required. Refer to the appropriate installation instructions and supplementary information for the Calibre version you plan to install.

Installing Calibre 2013.4 and Newer	33
Installing Calibre 2013.1 Through 2013.3	35
Installing Calibre 2012.2 Through 2012.4	36
Installing Calibre 2009.3 Through 2012.1	38
Installing Calibre 2008.1 Through 2009.2	40
Installing Calibre 2007.2 Through 2007.4	42
Troubleshooting Installation.....	43

Installing Calibre 2013.4 and Newer

Effective with the 2013.4 and newer releases, the CALIBRE_HOME tree is packaged into the following trees: IXL (discontinued with 2021.1), AOI, AOJ and AOK. The different executables, which include both the software and documentation, are optimized for the different Linux distributions. The AOJ download requires AVX2-capable hardware in order to execute. If your network includes a mix of different Linux distributions, you can install and run the different CALIBRE_HOME trees.

Refer to “[Supported Operating Systems](#)” on page 22 for details on the latest supported operating systems.

For 2013.4:

- IXL supports RHEL 5 and SLES 11 distributions.
- AOI supports RHEL 6, RHEL 7, and SLES 11 distributions.

For 2014.1:

- IXL supports RHEL 5 and SLES 11sp1 distributions.
- AOI supports RHEL 6 and SLES 11sp2 distributions.

For 2017.4:

- IXL supports RHEL 5 distributions.
- AOI supports RHEL 6, RHEL7, and SLES 11sp2-11sp4 distributions.
- AOJ supports RHEL 6.7, RHEL 6.8, RHEL 6.9, RHEL 7, and SLES 11 sp4 distributions.

For 2020.4:

- AOI supports RHEL 6, RHEL 7, and SLES 11 distributions.
- AOJ supports RHEL 6, RHEL 7, SLES11, and SLES12 distributions.

For 2021.1:

- AOI supports RHEL/CentOS 6, RHEL/CentOS 7, SLES 11, and SLES12 distributions.
- AOJ supports RHEL/CentOS 7 and SLES12 distributions.
- AOK supports RHEL/CentOS 8 distributions.

Procedure

1. Download the Calibre executable from Support Center and then place the executable file in the directory in which you want to install the software.
2. Verify you have permissions to execute the file. If necessary, use the Change Mode (chmod) command to change the permissions. For example:

```
% chmod a+x aoi_cal_2020.4_34.17_mib.exe
```

3. At the shell prompt, type the name of the executable file to install it. For example:

```
% aoi_cal_2020.4_34.17_mib.exe
```

4. Review and approve the license agreement.

Type “D” to display the license agreement and “yes” to accept the license agreement.

The installation program starts and displays a message when the process is complete. For example:

```
Installation in progress ... Please Wait
0%--10%--20%--30%--40%--50%--60%--70%--80%--90%--100%
|||||||||||||||||||||||||||||||||||||||||||
Installation Completed. Installed to: /home/calibre_install
```

5. If you use a mix of Linux distributions on your network, download and install the executable for the other tree(s) to the same target directory.
6. Set the CALIBRE_HOME environment variable.

If you are installing trees for different Linux distributions, set the CALIBRE_HOME environment variable to either the AOI or AOJ tree. For example:

```
% cd /home/calibre_install
% ls -al
drwxr-xr-x 11 user group 4096 Nov 14 10:46 aoj_cal_2020.4_34.17
drwxr-xr-x 11 user group 4096 Nov 14 10:46 aoi_cal_2020.4_34.17
% setenv CALIBRE_HOME /home/calibre_install/aoi_cal_2020.4_34.17
```

When you invoke Calibre, it automatically detects the operating system and executes from the correct CALIBRE_HOME tree. Setting up your environment in this manner

enables the remotes to continue to be used transparently as an interchangeable resource pool.

With this change, if you attempt to execute Calibre and the correct CALIBRE_HOME tree is not installed, you will get an error. For example:

```
ERROR: Current execution environment is VCO=aoj. Software tree is  
for environment VCO=aoi.
```

```
ERROR: Current execution environment is VCO=aoi. Software tree is  
for environment VCO=aoj.
```

Installing Calibre 2013.1 Through 2013.3

For the 2013.1 through 2013.3 Calibre releases, the software and documentation are combined into one Mentor Graphics Install Bundle, with an *_mib.exe* suffix. When installing on a Linux box, the combined software and documentation installation file is run as a self-extracting executable.

Procedure

1. Download the self-extracting executable file from Support Center and then place the file in the directory in which you want to install the software.
2. Verify you have permissions to execute the file. If necessary, use the Change Mode command to change the permissions. For example:

```
% chmod a+x ixl_cal_2013.1_14.11_mib.exe
```

3. At the shell prompt, type the name of the executable file to start the installation process. For example:

```
% ixl_cal_2013.1_14.11_mib.exe
```

4. Review and approve the license agreement.

Type “D” to display the license agreement and “yes” to accept the license agreement.

The installation program starts and displays a message when the process is complete. For example:

```
Installation in progress ... Please Wait  
0%--10%--20%--30%--40%--50%--60%--70%--80%--90%--100%  
||||||||||||||||||||||||||||||||||||  
Installation Completed. Installed to: /home/calibre_install
```

Installing Calibre 2012.2 Through 2012.4

Effective with the 2012.2 release, both the software and the documentation installation files are packaged in separate Siemens EDA Install Bundles, with an “_mib.exe” suffix. The installation process is different for Linux and non-Linux platforms.

Installing Calibre 2012.2 Through 2012.4 for Linux Platforms	36
Installing Calibre 2012.2 Through 2012.4 for Non-Linux Platforms	37

Installing Calibre 2012.2 Through 2012.4 for Linux Platforms

When installing the 2012.2 through 2012.4 Calibre software and documentation on a Linux platform, both the software and the documentation installation files are run as self-extracting executables.

Procedure

1. Download the Calibre executables for the software and documentation from Support Center. Place the executable files in the directory in which you want to install the software and documentation.
2. Verify you have permissions to execute the files. If necessary, use the Change Mode command to change the permissions. For example:

```
% chmod a+x ixl_cal_2012.2_15.12_mib.exe  
% chmod a+x docs_cal_2012.2_17.11_mib.exe
```

3. At the shell prompt, type the name of the executable file to install the software. For example:

```
% ixl_cal_2012.2_17.11_mib.exe
```

4. Review and approve the license agreement.

Type “D” to display the license agreement and “yes” to accept the license agreement.

The installation program starts and displays a message when the process is complete. For example:

```
Installation in progress ... Please Wait  
0%--10%--20%--30%--40%--50%--60%--70%--80%--90%--100%  
|||||  
Installation Completed. Installed to: /home/calibre_install
```

5. At the shell prompt, type the name of the executable file to install the documentation. For example:

```
% docs_cal_2012.2_17.11_mib.exe
```

6. Repeat the process in 4 to review and approve the license agreement.

The install program displays a message when the installation completes.

If you install the software and documentation in the same directory, the *docs* link will be correctly set. For example:

```
% pwd
/home/calibre_istall/ixl_cal_2012.2_17.11
% ls -al

drwxr-xr-x 11 owner group    4096 Oct 10 16:39 .
drwxrwxr-x  9 owner group    4096 Oct 10 16:39 ..
drwxr-xr-x  2 owner group    8192 Oct 10 16:39 bin
drwxr-xr-x  2 owner group    4096 Oct 10 16:38 dev
lrwxrwxrwx  1 owner group   29 Oct 10 16:39 docs -> ..
  /docs_cal_2012.2_17.11/docs
drwxr-xr-x  7 owner group    4096 Oct 10 16:38 etc
drwxr-xr-x  3 owner group    4096 Oct 10 16:38 font_registry
drwxr-xr-x  2 owner group    4096 Oct 10 16:39 lib
drwxr-xr-x 22 owner group   4096 Oct 10 16:39 pkgs
drwxr-xr-x  6 owner group    4096 Oct 10 16:38 registry
drwxr-xr-x 11 owner group    4096 Oct 10 16:38 shared
-rwxr-xr-x  1 owner group 173377 Oct  5 2012 third_party.pdf
lrwxrwxrwx  1 owner group     8 Oct 10 16:39 tmp -> /usr/tmp
drwxr-xr-x  2 owner group    4096 Oct 10 16:38 toolbox
```

If you install the documentation in a different directory or rename the directory, you will need to reset the *docs* link in order for some functionality to work correctly.

Installing Calibre 2012.2 Through 2012.4 for Non-Linux Platforms

When installing the 2012.2 through 2012.4 Calibre software and documentation on a non-Linux platform, you must use the Siemens EDA Install GUI to install the documentation.

Procedure

1. Download the Calibre executables for the software and documentation from Support Center. Place the executable files in the directory in which you want to install the software and documentation.
2. Verify you have permissions to execute the files. If necessary, use the Change Mode command to change the permissions. For example:

```
% chmod a+x ss5_cal_2012.2_17.11_mib.exe
% chmod a+x docs_cal_2012.2_17.11_mib.exe
```

3. At the shell prompt, type the name of the executable file to install the software. For example:

```
% ixl_cal_2012.2_17.11_mib.exe
```

4. Review and approve the license agreement.

Type “D” to display the license agreement and “yes” to accept the license agreement.

The installation program starts and displays a message when the process is complete.
For example:

```
Installation in progress ... Please Wait
0%--10%--20%--30%--40%--50%--60%--70%--80%--90%--100%
|||||||||||||||||||||||||||||||||||||
Installation Completed. Installed to: /home/calibre_install
```

5. At the shell prompt, type the name of the executable file to install the documentation.
For example:

```
% docs_cal_2012.2_17.11_mib.exe
```

6. Repeat the process in step 4 to review and approve the license agreement.

The install program displays a message when the installation completes.

If you install the software and documentation in the same directory, the *docs* link will be correctly set. For example:

```
% pwd
/home/calibre_install/ixl_cal_2012.2_17.11
% ls -al
drwxr-xr-x 11 owner group    4096 Oct 10 16:39 .
drwxrwxr-x  9 owner group    4096 Oct 10 16:39 ..
drwxr-xr-x  2 owner group     8192 Oct 10 16:39 bin
drwxr-xr-x  2 owner group    4096 Oct 10 16:38 dev

1rwxrwxrwx  1 owner group    29 Oct 10 16:39 docs -> ..
      /docs_cal_2012.2_17.11/docs
drwxr-xr-x  7 owner group    4096 Oct 10 16:38 etc
drwxr-xr-x  3 owner group    4096 Oct 10 16:38 font_registry
drwxr-xr-x  2 owner group    4096 Oct 10 16:39 lib
drwxr-xr-x 22 owner group   4096 Oct 10 16:39 pkgs
drwxr-xr-x  6 owner group    4096 Oct 10 16:38 registry
drwxr-xr-x 11 owner group    4096 Oct 10 16:38 shared
-rwxr-xr-x  1 owner group 173377 Oct  5 2012 third_party.pdf
1rwxrwxrwx  1 owner group     8 Oct 10 16:39 tmp -> /usr/tmp
drwxr-xr-x  2 owner group    4096 Oct 10 16:38 toolbox
```

If you install the documentation in a different directory or rename the directory, you will need to reset the *docs* link in order for some functionality to work correctly.

Installing Calibre 2009.3 Through 2012.1

The software and documentation for the Calibre 2009.3 through 2012.1 releases are packaged separately, and there are separate methods for installing each package. You can install Calibre 2009.3 through Calibre 2012.1 software releases on UNIX® platforms using the Siemens EDA Install utility and Siemens EDA Install Bundles (MIB).

- **Software** — Packaged as a Siemens EDA Install Bundle, which is a self-extracting executable with the *_mib.exe* suffix.

- **Documentation** — Packaged as a Siemens EDA Install file, which is installed using the Siemens EDA Install GUI and has the *.mis* suffix.

Procedure

1. Download the Calibre executables for the software and documentation from Support Center. Place the executable files in the directory in which you want to install the software and documentation.

The Siemens EDA Install Bundle installs the CALIBRE_HOME tree into your current working directory by default.

2. Verify you have permissions to execute the files. If necessary, use the Change Mode command to change the permissions. For example:

```
% chmod a+x <file>
```

3. At the shell prompt, type the name of the executable file to install the software. For example:

```
% ./ixl_cal_2012.1_37.24_mib.exe
```

4. Review and approve the license agreement.

Type “D” to display the license agreement and “yes” to accept the license agreement.

The installation program starts and displays a message when the process is complete. For example:

```
Installation in progress ... Please Wait
0%--10%--20%--30%--40%--50%--60%--70%--80%--90%--100%
|||||||||||||||||||||||||||||||||||||||||||||
Installation Completed. Installed to: /home/calibre_install
```

5. Verify your DISPLAY variable is set.

This is required by the Siemens EDA Install GUI, which is used to install the documentation.

6. Execute the following command to install the documentation:

```
% <target_dir>/install.<VCO>/install
    -quicksource <source_dir>/docs_mis_file -tgt <docs_dir>
```

where:

- *<target_dir>* is the installation directory for the software.
- *<VCO>* indicates the platform (ixl for Linux or ss5 for Solaris SPARC).
- *<source_dir>* is the directory path of the docs installation file (*.mis* file).
- *<docs_dir>* is the desired installation directory for the documentation.

Note

 It is recommended that <docs_dir> reside outside of the CALIBRE_HOME tree so that it is not overwritten by update or patch releases. For example (executed from the download directory):

```
% /user/mgc/install.ixl/install -quicksource  
docs_cal_2012.1_19.13.mis -tgt /user/mgc_docs
```

7. After the Siemens EDA Install GUI opens, confirm the documentation installation directory on the Target Selection page; then click **Install**.

Note

 You can specify an alternate target location on the Target Selection page, which overrides the value specified for <docs_dir>.

8. Click **Agree** to accept the license agreement.
9. Click **Done** when the installation is complete.

Note

 When installing the CALIBRE_HOME tree, a symbolic link is automatically created from <target_dir>/calibre_home/docs to <target_dir>/docs_cal_<version>/docs. If you install to a different location, you will need to change the link to the new location or delete the link and set the MGC_DOC_PATH environment variable to the full path of the new location. By default, the install files are placed in \$HOME/mgc. Use “-msiloc <some_other_dir>” to put the install files in a different location.

Installing Calibre 2008.1 Through 2009.2

The software and documentation for the Calibre 2008.1 through 2009.2 releases are packaged separately, and there are separate methods for installing each package. The product documentation is available on Support Center as a separate downloadable Siemens EDA Install (.mis) file and also on the Calibre Documentation CD. You can install both software and documentation for Calibre 2008.1 through Calibre 2009.2 on UNIX® platforms using the Siemens EDA Install utility and Siemens EDA Install Bundles (MIB).

Procedure

1. Download the Calibre executables for the software and documentation from Support Center.
2. Place the executable files in the target installation directory (<target_dir>) in which you want to install the software and documentation.

3. Verify you have permissions to execute the files. If necessary, use the Change Mode command to change the permissions. For example:

```
% chmod a+x <file>
```

4. At the shell prompt, type the name of the executable file to install the software. For example:

```
% ixl_cal_2012.2_17.11_mib.exe
```

5. Review and approve the license agreement.

Type “D” to display the license agreement and “yes” to accept the license agreement.

The installation program starts and displays a message when the process is complete. For example:

```
Installation in progress ... Please Wait
0%--10%--20%--30%--40%--50%--60%--70%--80%--90%--100%
|||||||||||||||||||||||||||||||||||||||||Installation Completed. Installed to: /home/calibre_install
```

6. Verify your DISPLAY variable is set.

This is required by the Siemens EDA Install GUI, which is used to install the documentation.

7. Execute the following command to install the documentation:

```
% <target_dir>/install.<VCO>/install
    -quicksource <source_dir>/docs_mis_file -tgt <docs_dir>
```

where:

- <*target_dir*> is the installation directory for the software.
- <*VCO*> indicates the platform (ixl for Linux or ss5 for Solaris SPARC).
- <*source_dir*> is the directory path of the docs installation file (.mis file).
- <*docs_dir*> is the desired installation directory for the documentation.

Note

 It is recommended that <*docs_dir*> reside outside of the CALIBRE_HOME tree so that it is not overwritten by update or patch releases. For example (executed from the download directory):

```
% /user/mgc/install.ixl/install -quicksource
    docs_cal_2012.1_19.13.mis -tgt /user/mgc_docs
```

8. After the Siemens EDA Install GUI opens, confirm the documentation installation directory on the Target Selection page, then click **Install**.

Note

 You can specify an alternate target location on the Target Selection page, which overrides the value specified for <docs_dir>.

9. Click **Agree** to accept the license agreement.
10. Click **Done** when the installation is complete.

Note

 When installing the CALIBRE_HOME tree, a symbolic link is automatically created from <target_dir>/calibre_home/docs to <target_dir>/docs_cal_<version>/docs. If you install to a different location, you will need to change the link to the new location or delete the link and set the MGC_DOC_PATH environment variable to the full path of the new location. By default, the install files are placed in \$HOME/mgc. Use “-msiloc <some_other_dir>” to put the install files in a different location.

Installing Calibre 2007.2 Through 2007.4

You can install Calibre 2007.2 through Calibre 2007.4 software releases on UNIX® platforms using the Siemens EDA Install utility and Siemens EDA Install Bundles (MIB).

Procedure

1. Download the Calibre executables for the software and documentation from Support Center. Place the executable files in the directory in which you want to install the software and documentation.

The Siemens EDA Install Bundle installs the CALIBRE_HOME tree into your current working directory by default.

2. Verify you have permissions to execute the files. If necessary, use the Change Mode command to change the permissions. For example:

```
% chmod a+x <file>
```

3. At the shell prompt, type the name of the executable file to install the software. For example:

```
% ./ixl_cal_2007.3_<xx.xx>_mib.exe
```

4. Review and approve the license agreement.

Type “D” to display the license agreement and “yes” to accept the license agreement.

The installation program starts and displays a message when the process is complete. For example:

```
Installation in progress ... Please Wait
0%--10%--20%--30%--40%--50%--60%--70%--80%--90%--100%
|||||||||||||||||||||||||||||||||||||  
Installation Completed. Installed to: /home/calibre_install
```

Troubleshooting Installation

The Siemens EDA Install program returns errors if the installation process does not complete, or if the installation process completed but detected certain problems within the installation.

Symptoms

Unable to perform concurrent installations.

Causes

The Siemens EDA Install program does not fully support concurrent installations.

Solution

Perform only one installation at a time regardless of whether you are installing the same or different VCOs.

Chapter 3

Configuration: Setting Up the Environment for Calibre Tools

There are some common tasks an administrator performs after installing and before running the Calibre software.

CALIBRE_HOME Environment Variable	45
Setting the CALIBRE_HOME Environment Variable	46
Using the <code>calibre_vco</code> Utility	47
Running Calibre System Tests	47

CALIBRE_HOME Environment Variable

The CALIBRE_HOME environment variable defines the location of the Calibre software tree.

Note

 If you do not use any Siemens EDA products (other than Calibre), you can set the MGC_HOME environment variable to the location of your Calibre software tree. If you do use other Siemens EDA products (in addition to Calibre), then set the CALIBRE_HOME environment variable to the location of your Calibre software tree and use the MGC_HOME environment variable for other Siemens EDA products.

The precedence for using CALIBRE_HOME and MGC_HOME is as follows:

- If only one variable is set, then its value is used, and both variables are set to that value.
- If both variables are set to the same value, then that value is used.
- If both variables are set to different values, then the value of CALIBRE_HOME is used and a warning is issued. In this case, the MGC_HOME variable is set to the value of CALIBRE_HOME.

When using the MGC_HOME variable, you should be cautious of situations where subprocesses are created, and the value of MGC_HOME is changed by the subprocess. For example, assume MGC_HOME is set to location A and CALIBRE_HOME is not set. When you invoke tclsh (`$MGC_HOME/bin/tclsh`), CALIBRE_HOME is set to the same value as MGC_HOME (location A). Changing the value for MGC_HOME to location B and invoking Calibre (`$MGC_HOME/bin/calibre`) issues a warning message because CALIBRE_HOME (location A) and MGC_HOME (location B) do not match. When this occurs, the value of

CALIBRE_HOME is used and MGC_HOME is set to the same value as CALIBRE_HOME (location A).

Note

 If the location of the software tree is going to be changed within a subprocess, such as within the Tcl shell, do one of the following:

- Change CALIBRE_HOME.
 - Change MGC_HOME and unset CALIBRE_HOME.
 - Set both variables to the same value.
-

Setting the CALIBRE_HOME Environment Variable

Before invoking a Calibre tool, you must set the CALIBRE_HOME environment variable to point to the location of your Calibre software tree.

Prerequisites

- Your Calibre software must be installed.

Procedure

1. Set the CALIBRE_HOME environment variable to the location of a Calibre software tree.

- In a C shell:

```
setenv CALIBRE_HOME path_to_calibre_tree
```

- In a Bourne or Korn shell:

```
CALIBRE_HOME=path_to_calibre_tree  
export CALIBRE_HOME
```

2. Verify the CALIBRE_HOME environment variable is set.

```
echo $CALIBRE_HOME
```

This returns the value assigned to CALIBRE_HOME.

3. Verify the location of CALIBRE_HOME.

```
ls $CALIBRE_HOME
```

This lists the contents of the CALIBRE_HOME directory when the variable is correctly set. You can execute Calibre commands from `$CALIBRE_HOME/bin`.

Related Topics

[Supported Operating Systems and Hardware](#)

[CALIBRE_HOME Environment Variable](#)

Using the `calibre_vco` Utility

Use the `calibre_vco` utility, located in the Calibre software tree, to determine the VCO of your current platform.

Procedure

Run this utility by entering the following command:

```
$CALIBRE_HOME/bin/calibre_vco
```

Results

This utility returns the 3-letter VCO of the platform that identifies the hardware and operating system on which the Calibre software is installed. Refer to “[Supported Operating Systems](#)” on page 22 for information regarding the currently supported operating systems and “[Supported x86-64 Hardware Configurations](#)” on page 23 for information on supported hardware configurations.

Running Calibre System Tests

The Calibre software tree contains a series of system tests you can execute to verify the installation of the Calibre software.

These system tests are located in the `$CALIBRE_HOME/shared/systest` directory. The primary system test, `test.CalibreRelease`, tests the installation of the Calibre software in its entirety. You can either run the `test.CalibreRelease` system test, or you can manually run the appropriate individual system tests for the products you have installed.

Prerequisites

- Your Calibre software must be installed and the `CALIBRE_HOME` environment variable must be set to the location of the Calibre software tree.
- All applicable licenses must be installed and the license file environment variable must point to a valid license file. Refer to “[Licensing: Overview](#)” on page 53 for licensing information.

Procedure

1. Set your working directory to the location of the system tests in the Calibre software tree:

```
cd $CALIBRE_HOME/shared/systest
```

You can list the contents of this directory to view the available system tests.

2. To test the entire installation of the Calibre tools, enter the following command:

```
test.CalibreRelease
```

3. To test the installation of an individual tool, enter the following command:

```
test.Calibre<tool>
```

For example, to test the installation of Calibre nmDRC and Calibre nmDRC-H, enter the following command:

```
test.Calibredrc
```

Related Topics

[Calibre Configuration Information](#)

Chapter 4

Configuration: Managing Calibre Documentation

The Calibre documentation includes user, reference, quick reference cards, and process documentation in both HTML and PDF formats to offer you the various features and benefits of each format. The actual content does not vary between the formats. You can also configure the environment to install and access the documentation from another location, other than the default install location.

Note

 The information provided here is targeted for the Linux platforms because Calibre products only run on these platforms.

Documentation Directory Structure	49
Documentation Setup Options	51

Documentation Directory Structure

The Calibre documentation resides in a self-contained directory structure. All links between documents are relative to each other. This enables you to make a copy of this directory and store it in any location and still have it function as a complete tree.

The installation process automatically installs the documentation at the same level as the Calibre software tree as shown in [Figure 4-1](#). The Calibre software tree (*install_directory/aoi_cal_201x.x_xx*) contains a symbolic link named *docs* that points to the default install location for the documentation.

Note

 As of the 2013.1 release, the Calibre documentation is packaged with the Calibre software tree for both the DVD and software download file.

Figure 4-1. Calibre Documentation Directory Structure

```
--install_directory
  -- docs_cal_201x.x_xx
    |-- docs
      |-- htmldocs
        |-- <handle1>
        |-- <handle2>
        |-- ...
      |-- index.html
      |-- infohubs
        |-- <Calibre_scopes>
        |-- ...
      |-- legal
        |-- <third_party_legal_documentation>
      |-- pdfdocs
        |-- <handle1>.pdf
        |-- <handle2>.pdf
        |-- ...
      |-- videos
        |-- <video1>
        |-- <video2>
        |-- ...
    |-- <VCO>_cal_201x.x_xx
      |-- bin
      |-- dev
      |-- docs -> ../../docs_cal_201x.x_xx/docs
      |-- ...
```

This structure allows you to easily move the documentation directory to a different location that has network access and reset the *docs* link to point to the new location. Maintaining this structure allows the documentation to be automatically accessed by the Calibre software applications.

Table 4-1 lists the naming conventions used for the common *docs_cal_201x.x_xx* directories and files.

Table 4-1. Naming Conventions for the Calibre Doc Tree Directories and Files

Name	Description
<i>htmldocs/<handle></i>	A unique directory name for a manual in HTML format. Most handles consist of <i><prod>_<doctype></i> . For example, the handle for the <i>Standard Verification Rule Format (SVRF) Manual</i> is <i>svrf_ur</i> . The handles are the same for the HTML and PDF version of a manual.

Table 4-1. Naming Conventions for the Calibre Doc Tree Directories and Files

Name	Description
<i>index.html</i>	The initial file used to open the InfoHub.
<i>infohubs/<Calibre_scopes></i>	The scope definitions for the Calibre InfoHub.
<i>legal/<handle>.pdf or legal/<handle>.txt</i>	A unique filename for all Open Source Software (OSS) redistribution licenses used in the Calibre software.
<i>pdfdocs/<handle>.pdf</i>	A unique filename for a PDF manual. Most handles consist of <i><prod>_<doctype></i> . For example, the handle for the <i>Standard Verification Rule Format (SVRF) Manual</i> is <i>svrf_ur.pdf</i> . The handles are the same for the HTML and PDF versions of a manual.
<i>videos/<video></i>	A unique directory name used for a video. The Calibre InfoHub provides links to the available getting started and how-to videos, as well as videos available on Support Center and siemens.com .

Documentation Setup Options

Calibre tools support moving the documentation to an alternate location and also setting a default PDF viewer.

- **Alternate Documentation Location** — The default location for the documentation directory is shown in [Figure 4-1](#) in “[Documentation Directory Structure](#)”. With this structure, you can easily move the *docs_cal_201x.x_xx* documentation directory to a different location that has network access and reset the *docs* symbolic link in the software tree to point to the new location.
- **MGC_PDF_READER** — By default, Adobe Reader is used (*AcroRd32.exe* for Windows or *acroread* for Linux). If you are using a different PDF viewer, you can use the **MGC_PDF_READER** environment variable to specify that PDF viewer. You set this variable to either the full path of a PDF viewer executable, or to the name of the viewer executable that can be found within your PATH environment variable. Regardless of the PDF viewer being used, the executable for the PDF viewer must reside in a directory that is defined in your PATH variable or must be defined using the **MGC_PDF_READER** variable.

Following is an example of setting the **MGC_PDF_READER** environment variable in a C shell environment to point to the Adobe Reader executable. You should check with your system administrator as to the correct location of the Adobe Reader executable, as this location can vary among hosts.

```
setenv MGC_PDF_READER /usr/opt/acrobat/acroread9.10/bin/acroread
```

Note

 The Linux version of Adobe Reader is no longer available for download from Adobe. The Calibre InfoHub and HTML documentation are included with the Calibre software tree as a solution for accessing and searching the documentation set in a Linux environment. The InfoHub and HTML documentation provide both global search and support of linking between documents.

Chapter 5

Licensing: Overview

Calibre provides multiple licensing models, configuration file options, and command line options allowing you to customize licensing to meet the needs of your environment.

Calibre Licensing Requirements	53
Mentor Standard Licensing (MSL)	54
Calibre Licensing Models	56
Calibre Licensing Configuration File	64
Calibre Licensing Command Line Options.....	68
Log Settings for Licensing.....	70
Licensing for Multithreaded Operations.....	71
Simultaneous Multithreading and Licensing Requirements.....	73
Calibre License Reference Description	74

Calibre Licensing Requirements

Calibre tools require the correct versions of both MSL and FLEXnet to manage the administration of licenses.

Starting with the 2019.2 release, Calibre uses MSL (Mentor Standard Licensing) V2018_2 and FLEXnet V11.16 licensing software. For more information about MSL or FLEXnet, refer to the *Mentor Standard Licensing Manual (mgc_licen.pdf)* or the *FLEXnet License Administration Guide (flexnet_lic_admin.pdf)*. These manuals are available in the Calibre documentation tree and on [Support Center](#).

Mentor Standard Licensing (MSL)

The Calibre tools use the Mentor Standard Licensing (MSL) for administering software licenses. MSL is based on FLEXnet licensing software and uses the FLEXnet license file format. MSL enables easy integration of Mentor Graphics licenses into most licensing strategies.

The Calibre tools use an authorization code methodology called Exact Access. Exact Access provides a consistent global authorization code policy and helps you better manage your software assets, standardize and synchronize license administration, and reduce the number of duplicate licenses.

Through this methodology, you receive the authorization codes yearly subsequent to your support contract renewal: each site has one authorization expiration date. New authorization codes are sent automatically prior to the expiration of the old codes.

Exact Access incorporates a version date and provides access to software updates released prior to your support contract expiration date. Refer to *Licensing Mentor Graphics Software* for more information on the Exact Access date.

MSL License File and Environment Variables.....	54
Securing and Releasing Licenses	55
Lost Connection Between Calibre and the MSL License Server	55

MSL License File and Environment Variables

MSL uses license records stored in an ASCII license file and maintained by the system administrator. This license file can be located anywhere in the file system. The system administrator can provide you with the license file pathname. You must provide the pathname when setting the licensing environment variables.

In addition to setting the CALIBRE_HOME environment variable to run Calibre tools (refer to “[Setting the CALIBRE_HOME Environment Variable](#)”), you need to set one of the following licensing variables:

- LM_LICENSE_FILE

This is the FLEXnet Licensing environment variable used by the license server and application to determine the location of license data files.

- MGLS_LICENSE_FILE

This variable is used when multiple products have FLEXnet licensing to allow other vendor products to use the LM_LICENSE_FILE variable, while allowing Mentor Graphics licensing products to get their licenses from the value of MGLS_LICENSE_FILE. Only Mentor Graphics software recognizes this variable; software from other vendors that use FLEXnet licensing ignore it.

Securing and Releasing Licenses

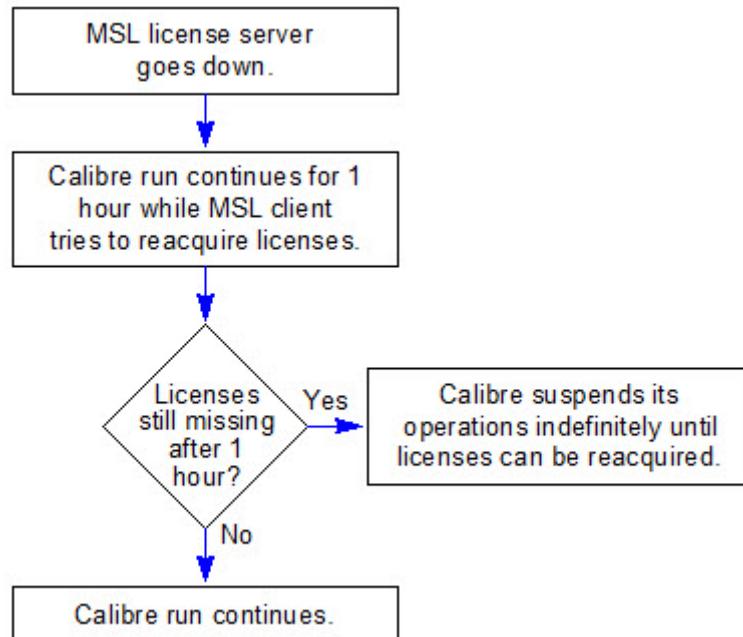
In general, Calibre tools secure any required licenses immediately after invocation and release these licenses after the run completes. However, there are some tools, such as Calibre Connectivity Interface (CCI), where licenses are secured in mid-execution of a session or script.

Lost Connection Between Calibre and the MSL License Server

If the connection between a Calibre tool and the MSL license server is lost, then Calibre re-establishes connections.

Figure 5-1 illustrates the lost connection process for the MSL license server.

Figure 5-1. MSL License Server Lost Connection Process



Calibre Licensing Models

There are two models, Calibre Classic Licensing and Calibre Loop Licensing, that determine how busy licenses are handled by Calibre tools. Calibre Classic Licensing is the default licensing model and supports the ability to queue for and to substitute licenses. Calibre Loop Licensing does not queue for licenses, but rather initiates a looping behavior in an attempt to secure the required (default) or substitute licenses.

Note

 Calibre jobs launched under CalCM use a different licensing model. Refer to “[CalCM and Licensing](#)” in the *Calibre Cluster (CalCM) User’s Manual* for more information.

Calibre Classic Licensing	56
Calibre Loop Licensing	61

Calibre Classic Licensing

Calibre Classic Licensing is the default licensing model and supports the ability to queue for and to substitute licenses. While all Calibre tools use *license queueing*, only selected Calibre tools use *license substitution*.

Calibre classic licensing is enabled and configured by either method:

- Specify the [Retry Mode Classic](#) statement in the license configuration file.
- Specify “-lmretry classic” on the command line. For more information on this argument and other licensing command line arguments, refer to “[Calibre Licensing Command Line Options](#)” on page 68.

[Figure 5-2](#) shows the relationship between license queueing and license substitution.

License Queueing

License queueing allows license requests to be placed in a queue to wait for an available license when the appropriate license is not available. The Calibre classic licensing model automatically enables license queueing when the required license is defined in the license file and the license is currently in use.

For the first license of a product, queueing is on by default and occurs only when licensing the first CPU for a given product. There is no queueing of licenses for additional CPUs. The wait time for a license may be controlled using the -wait or -nowait command line switches.

[Table 5-1](#) compares queueing behavior for single threaded and multithreaded Calibre runs when the first license and additional licenses are not available.

Table 5-1. Single Threaded and Multithreaded License Queueing Comparison

	When the first license is not available...	When additional licenses are not available...
Single Threaded Queueing	Calibre queues for the license.	
Multithreaded Queueing (using -turbo or -turbo_litho) without -turbo_all ¹	Calibre queues for the first occurrence of each license type.	When enough licenses cannot be secured for the requested number of threads, Calibre automatically reduces the number of threads (referred to as <i>licensing dropdown</i>) to match the available licenses. Refer to “ License Dropdown ” for more information.
Multithreaded Queueing (using -turbo or -turbo_litho) with -turbo_all ¹	Calibre queues for the first occurrence of each license type.	When enough licenses cannot be secured for the requested number of threads, Calibre exits with an error.

1. The Calibre licensing system does not provide more sophisticated queueing for groups of licenses in multithreaded runs. This queueing functionality is available from third-party batch queueing systems.

License Substitution

License substitution allows Calibre tools that support license substitution to check the license file for the required licenses. If the required license is not defined, and a substitute license is defined, the tool acquires or queues for the substitute license.

Calibre uses license substitution when a tool encounters the following conditions:

- The license file does not contain the required license for the tool, and
- The license file contains the substitute license for the tool.

Under these circumstances, the tool acquires or queues for the substitute license. Refer to “[License Queueing and Substitution](#)” for a description of the interrelationship between license queueing and license substitution.

The license reference pages in chapters 8 through 12 identify the required and substitute licenses for each Calibre product.

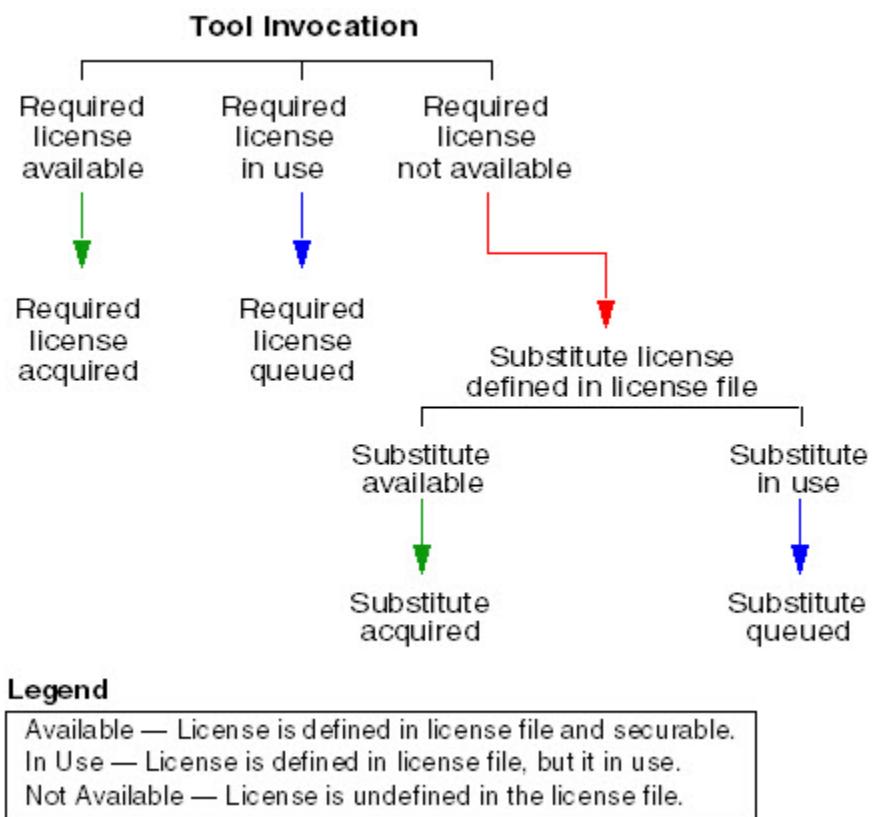
License Queueing and Substitution

Figure 5-2 illustrates the concept of license queueing and substitution for Calibre Classic Licensing. In the figure, the term *required license* refers to the primary license required to run the Calibre tool and *substitute license* refers to an alternate license(s) that may be used when the required license is not available.

Upon invocation, the tool uses the following process to secure a license:

1. If you define the required license in the license file and the license is available, then the tool acquires the required license.
2. If you define the required license in the license file and it is in use, the tool queues for the required license. This occurs even if you have defined the substitute license in the license file.
3. If you omit the required license from the license file but have defined the substitute license, the tool substitutes the required license with the substitute license.

Figure 5-2. Calibre Classic License Queueing and Substitution Scheme



License Dropdown

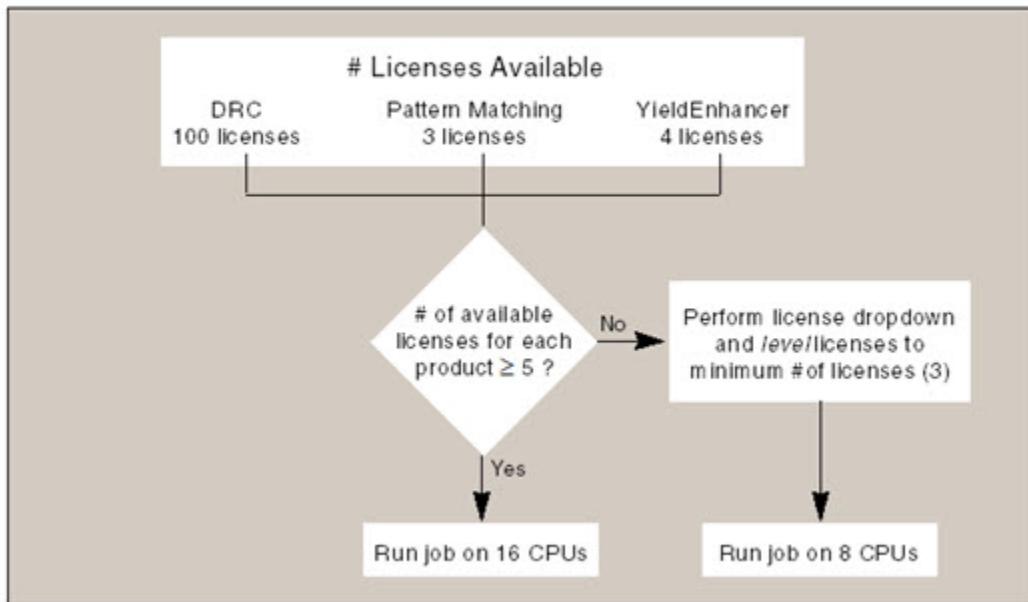
For jobs run in multithreaded mode, when the requested number of licenses cannot be secured, Calibre changes the values specified for the -turbo and -turbo_litho arguments, issues a

warning, and continues the run. This behavior of reducing the number of threads to match the number of available licenses is referred to as *license dropdown*.

When attempting to run multiple tools within the same product group (such as physical verification), a process referred to as *leveling* occurs when there are not enough licenses for one or more tools to run on the number of requested CPUs (see [Figure 5-3](#)). For example, assume you execute a Calibre job to run on 16 CPUs (-turbo 16) that requires DRC, Pattern Matching, and YieldEnhancer licenses. There are 100 DRC, three Pattern Matching, and four YieldEnhancer licenses available. To run on 16 CPUs requires five licenses for each product. Since five licenses are not available for each product, license dropdown occurs, licenses are leveled to three (the number of available Pattern Matching licenses) for all tools, and the job proceeds to run on eight CPUs.

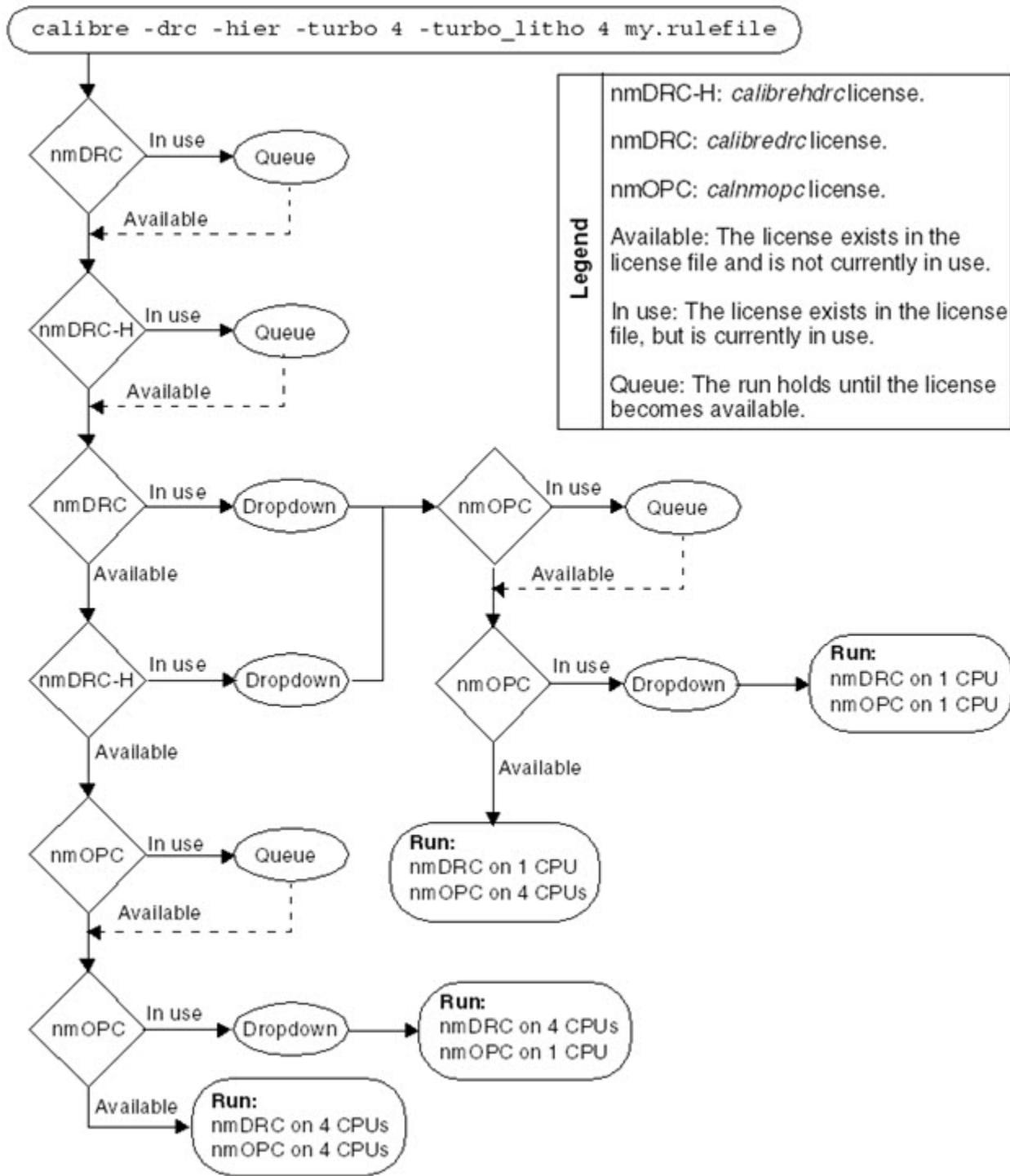
Figure 5-3. License Dropdown With Leveling

Execute Calibre job to run DRC, Pattern Matching,
and YieldEnhancer on 16 CPUs (-turbo 16)
16 CPUs requires 5 licenses for each product



[Figure 5-4](#) illustrates how Calibre attempts to obtain licenses and at which point it either queues for the license or performs licensing dropdown.

Figure 5-4. Queueing and Dropdown Example for 2-CPU OPC Run



Related Topics

[Using License Substitution With Calibre Classic Licensing](#)

Calibre Loop Licensing

Calibre Loop Licensing is an alternative method to Calibre Classic Licensing. With Calibre Loop Licensing, if Calibre is unable to acquire licenses, it continues looping to acquire the necessary licenses for running the Calibre tool. FLEXnet license queueing is never used in this mode. Looping automatically exits when Calibre is unable to acquire licenses because the license server is down or there are problems related to acquiring licenses. Even though this is not the default licensing model, this is the recommended model for new installations.

Loop licensing is enabled and configured with either of the following methods:

- Specify the [Retry Mode Loop](#) command in the license configuration file.
- Specify “-lmretry loop” on the command line. For more information on this argument, refer to [“Calibre Licensing Command Line Options”](#) on page 68.

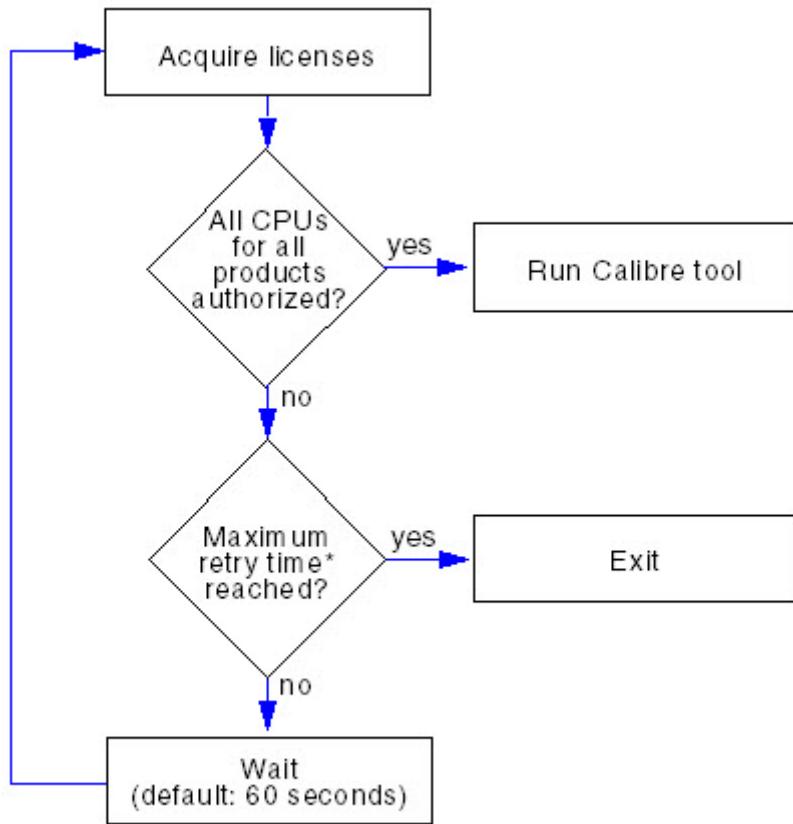
With Calibre Loop Licensing, you can choose to run without dropdown (default) as described in [“Loop Mode Without Dropdown,”](#) or with dropdown capability as described in [“Loop Mode With Dropdown”](#).

When using Calibre Loop Licensing and a required license is in use, it automatically searches for and uses a substitute license if available.

Loop Mode Without Dropdown

By default, Calibre Loop Licensing runs without dropdown capability when you do not specify the “minimum” CPU options available with the Retry Mode Loop command or the “-lmretry loop” command line option. In loop licensing mode, if Calibre cannot acquire licenses for all products, it will release all acquired licenses, wait a specified amount of time, and then rerun licensing. [Figure 5-5](#) illustrates how Calibre Loop Licensing works.

Figure 5-5. Calibre Loop Licensing



* Default retry time is three hours.

Loop Mode With Dropdown

Loop mode with dropdown enables the dropdown capability when there are not enough licenses available to satisfy the initial CPU request. In this mode, you specify the minimum number of CPUs to be licensed using either of the following methods:

- Licensing configuration file
Specify the TURBO_MIN, TURBO_LITHO_MIN, or both arguments with the [Retry Mode Loop](#) command.
- Command line
Specify the TURBO_MIN, TURBO_LITHO_MIN, or both options with the “-lmretry loop” argument. Refer to “[Calibre Licensing Command Line Options](#)” for information on these options.

When loop mode with dropdown is enabled, the behavior is as follows:

- If the initial licensing request can be fully satisfied, then the Calibre run proceeds.

- If the request cannot be fully satisfied but is greater than or equal to the specified minimum number of CPUs, the Calibre run continues with a reduced CPU count.
- If the request cannot be fully satisfied but is less than the specified minimum number of CPUs:
 - The application waits for 60 seconds (default) and attempts to acquire more licenses.
 - During this wait period, the licenses that have been checked out are held and not released (this behavior differs from the default loop mode without dropdown behavior).
 - If the minimum number of CPUs cannot be licensed within the MAXRETRY default of 180 minutes (3 hours), then the Calibre run fails.

Related Topics

[Configuring Calibre Loop Licensing](#)

Calibre Licensing Configuration File

The licensing configuration file is an ASCII file you create to specify the licensing model that is used by Calibre tools.

Calibre Licensing Configuration File Requirements	64
Retry Mode Classic	65
Retry Mode Loop.....	66

Calibre Licensing Configuration File Requirements

You control licensing behavior by specifying statements in a licensing configuration file and configuring the access to this file.

The following syntax conventions apply when creating a licensing configuration file:

- Commands are case insensitive.
- The double slash (//) characters denote comments, where everything between “//” and the end-of-line character is ignored.

To control licensing behavior, you must specify the [Retry Mode Classic](#) or the [Retry Mode Loop](#) statement in the licensing configuration file. These statements determine how licenses are acquired when you invoke a Calibre tool. Command line options are also available that perform the same function as these statements.

If you use a licensing configuration file to control licensing behavior, you must define the method for accessing this file when you invoke a Calibre tool. There are multiple methods available, but you can use only *one* of the following methods to provide access to the file. An error is issued if you attempt to use more than one of these methods.

- Place a site-wide configuration file in `$CALIBRE_HOME/etc/cust/calibre/licensing.conf`. Calibre checks for this file upon tool invocation and reads it if it exists.
- Specify a licensing configuration file using the `-lmconfig` argument on the Calibre command line. Refer to “[Calibre Licensing Command Line Options](#)” on page 68 for information on using this argument.
- Set the `CALIBRE_LM_CONFIG` environment variable to the location of the licensing configuration file.

Related Topics

[Calibre Licensing Command Line Options](#)

Retry Mode Classic

Specifies the Calibre Classic licensing mode and available configuration options.

Usage

RETRY MODE CLASSIC [MAXQUEUE *n*] [NO_DROPDOWN]

Arguments

- **MAXQUEUE *n***

An optional keyword used to limit FLEXnet queueing to a maximum of *n* minutes for any one license. If after *n* minutes Calibre has not acquired the license, it will try the next licensing substitution. The default is infinite. Using this argument is equivalent to using the -wait *n* command line switch.

- **NO_DROPDOWN**

An optional keyword used to disable license dropdown. Setting this option causes Calibre to exit if any product cannot be licensed for all CPUs desired. Using this argument is equivalent to using the -turbo_all command line switch.

Description

In Calibre Classic Licensing (default), Calibre tries to acquire a license for the first CPU for all products and employs FLEXnet license queueing for each substitution. Calibre exits if it cannot acquire a license for the first CPU for all products. After the first CPU for all products is licensed, Calibre attempts to license additional CPUs. If Calibre cannot acquire enough licenses for additional CPUs, it reduces the CPU count for the deficient product(s) to reflect the licenses it was able to acquire.

The “-lmretry classic” command line option can be used instead of specifying the Retry Mode Classic statement in a configuration file. Refer to “[Calibre Licensing Command Line Options](#)” for information on the -lmretry option.

Examples

The following statement specifies that Calibre should run in classic licensing mode with a maximum FLEXnet queueing time of 5 minutes for a single license. This is equivalent to specifying calibre -wait 5 on the command line.

```
//Licensing configuration file
RETRY MODE CLASSIC MAXQUEUE 5
```

Related Topics

[Calibre Licensing Models](#)

Retry Mode Loop

Specifies the Calibre Loop licensing mode and available configuration options.

Usage

```
RETRY MODE LOOP [MAXRETRY {x | UNLIMITED}] [INTERVAL y]  
[RETRY_ON_ERROR] [TURBO_MIN #CPUs] [TURBO_LITHO_MIN #CPUs]
```

Arguments

- MAXRETRY {x | UNLIMITED}

A keyword that specifies the maximum number of minutes (*x*) to retry until licensing succeeds. If licensing does not succeed within the specified amount of time, Calibre exits with an error. The argument, *x*, must be an integer and has a default value of 180 minutes (3 hours). The UNLIMITED argument specifies to wait an unlimited amount of time for licensing to succeed.

- INTERVAL *y*

An optional keyword that specifies the number of seconds to wait between loop iterations. The argument, *y*, must be an integer and has a default value of 60.

- RETRY_ON_ERROR

An optional argument that causes licensing to loop even when the license server cannot be reached or other errors are detected. When this keyword is not provided, looping is only triggered when at least one license is busy.

- TURBO_MIN #CPUs

An optional argument that specifies the minimum number of CPUs that must be licensed for non-litho tools in order for the Calibre run to proceed. The value specified for #CPUs must be an integer that is greater than zero and less than the maximum number of available CPUs.

- TURBO_LITHO_MIN #CPUs

An optional argument that specifies the minimum number of CPUs that must be licensed for litho tools in order for the Calibre run to proceed. The value specified for #CPUs must be an integer that is greater than zero and less than the maximum number of available CPUs.

Description

The Retry Mode Loop command, when used in a licensing configuration file, enables [Calibre Loop Licensing](#) mode. The available arguments allow you to control looping behavior and dropdown capability.

The “-lmretry loop” command line option can be used instead of specifying the Retry Mode Loop statement in a configuration file. Refer to “[Calibre Licensing Command Line Options](#)” for information on the -lmretry option.

Examples

Example 1

This example specifies that Calibre should run in loop licensing mode and attempt to acquire licenses every two minutes (120 seconds) for up to one hour (60 minutes) before exiting.

```
//Licensing configuration file
RETRY MODE LOOP MAXRETRY 60 INTERVAL 120
```

Example 2

In this example, the licensing configuration file named *my_lmconfig.rules* contains the RETRY MODE LOOP statement that defines the following values for the TURBO_MIN and TURBO_LITHO_MIN arguments:

```
RETRY MODE LOOP TURBO_MIN 8 TURBO_LITHO_MIN 48
```

The following command executes Calibre and, if there are not enough licenses available for the initial request of 24 CPUs for non-litho operations and 64 CPUs for litho operations, then the Calibre run proceeds with a minimum of eight CPUs for non-litho operations and 48 CPUs for litho operations.

```
calibre -drc -hier -turbo 24 -turbo_litho 64 -lmconfig my_lmconfig rules
```

Related Topics

[Calibre Licensing Models](#)

Calibre Licensing Command Line Options

Calibre licensing command line options are used to control licensing behavior. These options are not supported for all command line invocations (for example, Calibre® xACT™ digital flow does not support the -wait and -nowait command line options). Use the calibre -help command to determine which Calibre tools support these licensing options.

Usage

```
calibre tool_specific_options [ -nowait | -wait n | -lmretry retry_args ]  
[ -lmconfig licensing_config_file_name ]
```

Arguments

- **-nowait**

An optional argument that causes Calibre to queue only briefly (approximately 10 seconds) before attempting to acquire substitute licenses. This option is equivalent to specifying “-wait 0”.

- **-wait n**

An optional argument used to specify the maximum amount of time (in minutes) for Calibre to queue for a specific license. If the license is unavailable after queueing for *n* minutes, Calibre attempts to acquire any substitute licenses or exits if no suitable substitutions are defined. The maximum value for *n* is 45000. Refer to the section “[Using License Substitution With Calibre Classic Licensing](#)” for more information on substituting licenses.

- **-lmretry retry_args**

An optional argument that specifies whether to use the [Calibre Classic Licensing](#) mode or the [Calibre Loop Licensing](#) mode for the Calibre job. Valid *retry_args* options include:

classic[,MAXQUEUE:*n*][,NO_DROPDOWN] — Specifies to use Calibre Classic Licensing. Refer to the [Retry Mode Classic](#) statement for descriptions of the MAXQUEUE and NO_DROPDOWN keywords, and to [Example 3](#) for an example of this argument.

loop[,MAXRETRY:*x*|UNLIMITED][,INTERVAL:*y*][,RETRY_ON_ERROR]
[,TURBO_MIN:*CPUs*][,TURBO_LITHO_MIN:*CPUs*] — Specifies to use Calibre Loop Licensing. Refer to the [Retry Mode Loop](#) statement for descriptions of the MAXRETRY, UNLIMITED, INTERVAL, RETRY_ON_ERROR, TURBO_MIN, and TURBO_LITHO_MIN keywords, and to [Example 4](#) and [Example 5](#) for examples of these arguments.

The -lmretry argument provides an alternate method to defining the Retry Mode Classic or Retry Mode Loop statement in a license configuration file and then using the -lmconfig argument to specify the license configuration file.

- `-lmconfig licensing_config_file_name`

An optional argument that specifies the path to a license configuration file. The license configuration file contains either the Retry Mode Classic or Retry Mode Loop statement that specifies the licensing model that is used by Calibre tools.

Examples

Example 1

The following example uses the `-nowait` option to fully disable license queueing:

```
calibre -drc -hier -nowait rule_file
```

If a required license is in use, Calibre immediately attempts to acquire the substitute license(s).

Example 2

The following example uses the `-wait` option to queue on a license for five minutes:

```
calibre -drc -wait 5 rule_file
```

If a license does not become available within five minutes, the application exits with the following message:

```
// Queue time specified by -wait switch has elapsed.
```

Example 3

The following example specifies to run Calibre in classic licensing mode and limits FLEXnet queueing to a maximum of five minutes for any license. The `no_dropdown` argument disables dropdown licensing, which causes Calibre to exit if any product cannot be licensed for all desired CPUs.

```
calibre -drc -hier -lmretry classic,maxqueue:5,no_dropdown rules
```

Example 4

The following example specifies to run Calibre in loop licensing mode and attempt to acquire licenses every two minutes (120 seconds) for up to 30 minutes before exiting.

```
calibre -drc -hier -lmretry loop,maxretry:30,interval:120 rules
```

Example 5

The following example uses the `-lmretry` argument to run Calibre loop licensing. If there are not enough licenses available for the initial request of 24 CPUs for non-litho operations and 64 CPUs for litho operations, then the Calibre run continues with a minimum of eight CPUs for non-litho operations and 48 CPUs for litho operations.

```
calibre -drc -hier -turbo 24 -turbo_litho 64  
-lmretry loop,turbo_min:8,turbo_litho_min:48 rules
```

Example 6

The following example uses the -lmconfig argument to specify the path to the licensing configuration file named *licensing.conf*.

```
calibre -drc -hier -turbo -lmconfig <path>/licensing.conf rules
```

Related Topics

[Calibre Licensing Models](#)

[Calibre Licensing Configuration File](#)

Log Settings for Licensing

Calibre Classic Licensing and Calibre Loop Licensing both produce log information that you can configure using the CALIBRE_LM_LOG_LEVEL environment variable. This information is written to the Calibre transcript.

Table 5-2 lists the valid environment variable values and information about the error resulting from each value. For information on the log files associated with the license server, refer to *Licensing Mentor Graphics Software*.

Table 5-2. CALIBRE_LM_LOG_LEVEL Environment Variable Settings

Environment Setting	Prefix	Description	Example
FATAL	// Error:	Fatal errors only. Fatal errors have no possibility for recovery and generally cause Calibre to exit immediately.	Unable to launch mgls_asynch, unable to authorize a product for any CPUs.
WARN	// Warning:	Warnings only. Warnings are errors with the possibility that execution could still succeed in at least a degraded state. Fatal messages are also included when using this setting.	Unable to secure a license. License has been queued.
INFO (default)	//	Informational messages. Fatal and warning messages are also included when using this setting.	CPU counts, products authorized, licenses acquired, licenses released.

Licensing for Multithreaded Operations

A license consumption formula is used by applicable Calibre tools to determine the number of licenses needed for a multithreaded Calibre job.

License Requirements for Distributed Calibre

[Figure 5-6](#) shows the formula that is used to calculate the number of licenses that are required by most Calibre tools when run in multithreaded mode. The number of licenses required is based on the number of CPUs (>1) that are used to run the job. The resulting value is truncated to the nearest integer. Some Calibre products have different licensing requirements, which are described on the respective product license reference page.

Figure 5-6. Standard Multithreaded License Requirements Formula

$$\text{# of Licenses Required} = \frac{(\text{\# of CPUS}) + 3}{4} + 1$$

[Table 5-3](#) provides a quick reference for the number of licenses that are needed based on the number of CPUs (1-116 CPUs) being used for the run.

Table 5-3. License Requirements per # of CPUs

# of CPUs	# of Licenses Required	# of CPUs	# of Licenses Required	# of CPUs	# of Licenses Required
1	1	37-40	11	77-80	21
2-4	2	41-44	12	81-84	22
5-8	3	45-48	13	85-88	23
9-12	4	49-52	14	89-92	24
13-16	5	53-56	15	93-96	25
17-20	6	57-60	16	97-100	26
21-24	7	61-64	17	101-104	27
25-28	8	65-68	18	105-108	28
29-32	9	69-72	19	109-112	29
33-36	10	73-76	20	113-116	30

The following examples show the number of licenses necessary for different tool runs.

- A hierarchical Calibre nmLVS run on 12 CPUs requires the following licenses:
 - 4 *calibrelvs*
 - 4 *calibrehlvs*

- A hierarchical Calibre nmDRC run on six CPUs with a rule file invoking Calibre ORC and Calibre PRINTImage requires the following licenses:
 - 3 *calibredrc*
 - 3 *calibrehdrc*
 - 3 *calibreorc*
 - 3 *calprintimage*

The licensing model you use ([Calibre Classic Licensing](#) or [Calibre Loop Licensing](#)) determines the licensing requirements for a particular Calibre run.

Multithreaded Tool Behavior and License Consumption

For a Calibre nmDRC-H or Calibre nmLVS-H job, the -turbo value option specifies the number of CPUs that Calibre concurrently uses for the job. Remote host CPUs specified in a configuration file are connected to the job, but concurrency is limited to the -turbo value (unless -hyper remote is specified). For example, if you specify “-turbo 8” and your configuration file specifies 16 remote host CPUs, the job only uses eight primary or remote CPUs concurrently even though 24 CPUs are available. If you do not specify a -turbo value, the job uses as many CPUs as are available for concurrent processing.

For a Calibre PERC job, the -turbo value and the number of remote host CPUs specified in a configuration file are aggregated. For example, if you specify “-turbo 8” and your configuration file specifies 16 remote host CPUs, the job connects to and uses 24 CPUs in total. Eight CPUs on the primary host and 16 CPUs on the remote hosts.

In both cases, the license requirements are based on the number of concurrent CPUs that are used by the Calibre tool to process the job.

In some situations, when you have a primary host with multiple CPUs, you may choose to run a Calibre job where the primary host is also specified as a remote host, along with any other specified remote hosts, to process the Calibre job. This is referred to as “overloading” the primary host. In this situation, the number of concurrent CPUs available for use is determined by the job type, -turbo value, and number of remote hosts specified in the configuration file.

Two-CPU Licensing

[Table 5-4](#) shows the Calibre tools and the associated two-CPU license.

Table 5-4. Two-CPU Licenses

Tool	Two-CPU License
Calibre nmDRC-H	<i>calibremt2cpu</i> ¹
Calibre nmLVS-H	<i>calibremt2cpu</i>

1. You can also use the *calibremt2cpu* license for ERC.

Calibre tries securing these special licenses, when available, in addition to the full licenses for the tools. For example, when you invoke Calibre nmDRC-H or Calibre nmLVS-H on a machine containing exactly two CPUs, and use the -turbo argument in either of the following forms:

```
calibre ... -turbo ...
calibre ... -turbo 2 ...
```

After securing the initial nmDRC-H or nmLVS-H license pair, the tool initially tries securing a calibremt2cpu license, rather than securing a second nmDRC-H or nmLVS-H license pair.

If you invoke Calibre nmDRC-H or Calibre nmLVS-H on a machine having more than two CPUs, and use the -turbo argument in the following form:

```
calibre ... -turbo 2 ...
```

After securing the initial nmDRC-H or nmLVS-H license pair, the tool initially attempts securing the calibremt2cpu license, rather than securing a second nmDRC-H or nmLVS-H license pair.

Simultaneous Multithreading and Licensing Requirements

When using Simultaneous Multithreading (SMT) functionality to run a Calibre job using virtual cores, only the physical CPUs are counted toward license requirements. After all physical CPUs are licensed, Calibre automatically launches additional threads for execution on the virtual cores. Calibre does not require additional licenses for the additional threads on the virtual cores.

Table 5-5 provides examples of how licensing is handled when launching a job on hardware that is SMT-enabled. The examples assume the hardware is SMT-enabled with eight physical CPUs and eight virtual cores.

Table 5-5. License Consumption for SMT-Enabled Hardware With Eight CPUs

If you ...	Then Calibre does the following:
Launch Calibre nmDRC and specify -turbo 8	Secures licenses to run on eight physical CPUs. After the eight physical CPUs are licensed, Calibre launches 16 threads for execution.
Launch Calibre nmDRC and specify -turbo 4	Secures licenses to run on four physical CPUs. Because all of the physical CPUs are not licensed, Calibre launches only four threads for execution.
Launch Calibre nmDRC and specify -turbo 10	Limits the -turbo value to match the number of physical CPUs. Runs as if -turbo 8 was specified.

When hyperscaling is used, by default simultaneous multithreading is automatically utilized for pseudo HDBs 1-4.

Related Topics

[Licensing for Multithreaded Operations](#)

[Simultaneous Multithreading With Virtual CPUs](#)

Calibre License Reference Description

The license reference pages use a standard format for providing license information for Calibre products.

The license reference pages are organized into the following product areas:

- [Licensing: Physical Verification Products](#)
- [Licensing: Design for Manufacturability \(DFM\) Products](#)
- [Licensing: Parasitic Extraction Products](#)
- [Licensing: Resolution Enhancement Technology \(RET\) Products](#)
- [Licensing: Mask Data Preparation \(MDP\) Products](#)

Each licensing reference page includes the following information:

- **Required License** — Specifies the primary license required to run the Calibre tool. In some cases, a required license may be a license whose existence is checked for, but it is not checked out and held.
- **Substitute License** — Identifies alternate license(s) that may be used when the primary license is not available. For example, if a *caldopc* license is not available for running Calibre TDopc, then an available Calibre OPCpro (*calopcpro*) license may be substituted in its place.
- **Also Enables** — Identifies additional capabilities that are enabled while running the product. For example, if you run Calibre OPCpro then you are entitled to run Calibre TDopc without consuming any additional licenses.

Note

 While running a Calibre tool may also enable another tool, it does not imply any other licensing relationship. For example, Calibre YieldServer is enabled when running Calibre PERC. However, when running Calibre YieldServer without Calibre PERC, the *calibreperc* license is not a valid substitute license.

- **Considerations and Exceptions** — Provides additional license, MT, Calibre MTflexTM, or Hyperscaling information that you should be aware of when running the Calibre tool.

- **Related Topics** — Provides links to information describing licensing behavior or information that is relevant to the Calibre tool you are running.

Chapter 6

Licensing: Queueing and Substitution

With Calibre classic licensing, options are available for controlling license queuing and license substitution. With Calibre loop licensing, an alternate method to Calibre classic licensing, you can control the acquisition of licenses.

Setting Up Your Licensing Environment	77
Controlling Queueing for Calibre Classic Licensing	78
Disabling Queueing for Calibre Classic Licensing	78
Using License Substitution With Calibre Classic Licensing	79
Configuring Calibre Loop Licensing	80

Setting Up Your Licensing Environment

Depending on your environment, you will need to set either the LM_LICENSE_FILE or the MGLS_LICENSE_FILE environment variable to point to a valid license file or license daemon before invoking a Calibre tool.

Prerequisites

- Your Calibre software must be installed and the CALIBRE_HOME environment variable must be set to the location of the Calibre software tree.
- All applicable licenses must be installed. Refer to the *Mentor Standard Licensing Manual (mgc_licen.pdf)*, located in the *release_documents* directory of your Calibre tree, for information on licensing administration, planning, and customizing.

Procedure

Depending on your shell environment, enter the appropriate command, substituting the actual path to the license file for the *license_file_pathname* variable.

- For C shell:

```
setenv LM_LICENSE_FILE <license_file_pathname>
```

- For Bourne or Korn shell:

```
LM_LICENSE_FILE=<license_file_pathname>
export LM_LICENSE_FILE
```

Controlling Queueing for Calibre Classic Licensing

You can control license queueing for Calibre tools using the `-wait` and `-nowait` switches during the invocation of a Calibre tool. These switches are mutually exclusive.

Prerequisites

- The required license must be defined in the license file.
- The appropriate license variable must be defined (refer to “[Setting Up Your Licensing Environment](#)”).

Procedure

Enter the following command to specify a maximum wait time of five minutes for Calibre to queue for a `calibrehdrc` license.

```
calibre -drc -hier -wait 5 my_rule_file
```

Results

The run proceeds when Calibre tools obtain the requested licenses. The transcript shows the application is waiting for a license during the queueing interval. If the license is still in use after five minutes, then Calibre attempts to acquire the substitute licenses or exits with the following message if no suitable substitutes are defined.

```
// Queue time specified by -wait switch has elapsed.
```

Disabling Queueing for Calibre Classic Licensing

You can disable license queueing for Calibre tools using the `-nowait` switch during the invocation of a Calibre tool.

Prerequisites

- The required license must be defined in the license file.
- The appropriate license variable must be defined (refer to “[Setting Up Your Licensing Environment](#)”).

Procedure

Enter the following command to fully disable license queueing when invoking Calibre:

```
calibre -drc -hier -nowait my_rule_file
```

Results

If a required license is in use, Calibre immediately attempts to acquire the substitute license(s).

Using License Substitution With Calibre Classic Licensing

During license substitution, selected Calibre tools check the license file for available required licenses. If you have not defined the required license but have defined a substitute license, then the Calibre tool acquires or queues for the substitute license or licenses.

Note

 [Figure 5-2](#) on page 58 shows the interrelationship between license queueing and license substitution.

Prerequisites

- The Calibre software must be installed and the CALIBRE_HOME environment variable must be set to the location of the Calibre software tree.
- All applicable licenses must be installed. Refer to the *Mentor Standard Licensing Manual (mgc_licen.pdf)* for information on licensing administration, planning, and customizing.
- The appropriate license variable must be defined (refer to “[Setting Up Your Licensing Environment](#)”).
- Because this example illustrates the use of license substitution, it assumes your rule file contains only Calibre nmDRC and Calibre TDopc commands.

Procedure

1. Create a rule file containing only Calibre nmDRC and Calibre TDopc commands.
2. Create a license file that contains licenses for Calibre nmDRC, Calibre nmDRC-H, and Calibre nmOPC. For example:

```
 SERVER fred 00112F42D493c
 DAEMON mgcld
 INCREMENT calibredrc mgcld 2007.100 1-oct-2007 10
 2DA41ECA103A3A759990 VENDOR_STRING=A23B2C29 SN=5352275
 INCREMENT calibrehdrc mgcld 2007.100 1-oct-2007 10
 6D94BECA7BE9B60691C2 VENDOR_STRING=56BC3C35 SN=5352278
 INCREMENT calibrenmopc mgcld 2007.100 1-oct-2007 10
 5D3531E25B2CFF62E6E7 VENDOR_STRING=EC02D644 SN=5637358
```

3. Invoke Calibre nmDRC-H:

```
calibre -drc -hier -turbo -turbo_litho rules
```

4. Execute the TDopc commands in your rule file.

Results

The tool secures the *calnmopc* licenses because there are no *caltdopc* licenses defined in the license file.

Related Topics

[Calibre Classic Licensing](#)

Configuring Calibre Loop Licensing

In this procedure you use the RETRY MODE LOOP statement to configure your licensing environment for Calibre loop licensing.

Prerequisites

- The Calibre software must be installed and the CALIBRE_HOME environment variable must be set to the location of the Calibre software tree.
- All applicable licenses must be installed. Refer to *Mentor Graphics Standard Licensing Manual (mgc_licen.pdf)* for information on licensing administration, planning, and customizing.

Procedure

1. Create a licensing configuration file by opening an ASCII editor and entering the following:

```
//Calibre Licensing Configuration File
RETRY MODE LOOP MAXRETRY 60 INTERVAL 30
```

2. Save the file as *licensing.conf*.
3. Set the CALIBRE_LM_LOG_LEVEL environment variable as follows:

```
setenv CALIBRE_LM_LOG_LEVEL WARN
```

This variable specifies that only warning and fatal messages should be output to the transcript.

4. Invoke Calibre using the -lmconfig switch on the Calibre command line to specify the licensing configuration file.

```
calibre -drc -hier -turbo -lmconfig <path>/licensing.conf rules
```

Results

The Retry Mode Loop statement in the licensing configuration file specifies that Calibre should attempt to acquire licenses for 60 minutes until licensing succeeds. When a required license is in use, Calibre Loop Licensing automatically searches for and uses a substitute license if available. If the required or substitute licenses are busy, Calibre waits 30 seconds before attempting to acquire licenses again. If licenses are not available after 60 minutes, Calibre exits with an error.

Chapter 7

Licensing: Calibre Platform and General Products

The licensing information for Calibre platform and general products provides information on required and substitute licenses, in addition to any considerations and exceptions.

[Table 7-1](#) lists each Calibre platform and general product and the required and substitute license(s) for that product. Click the Calibre product name to view more details about the licensing requirements.

Table 7-1. Primary License Requirements for Calibre General Products

Calibre Product	Required License(s)	Substitute License(s)
Calibre Cluster Manager (CalCM)	<i>calresourceman</i>	<i>calresourceadv</i>
Calibre Cluster Manager Plus (CalCM+)	<i>calresourceadv</i> <i>calscopeadv</i>	None.
Calibre Dynamic Resource Allocator (DRA)	<i>calresourceman</i>	<i>calresourceadv</i>
Calibre FullScale	<i>calibreproto</i>	None.

Calibre Cluster Manager (CalCM)

Licensing for Calibre® Cluster Manager.

Required License

- *calresourceman*

Substitute License

- *calresourceadv*

Also Enables

- Calibre Dynamic Resource Allocator (DRA)

Considerations and Exceptions

- One *calresourceman* is required for every four CPUs registered with CalCM.

Related Topics

[Calibre Cluster Manager User's Manual](#)

Calibre Cluster Manager Plus (CalCM+)

Licensing for Calibre® Cluster Manager Plus.

Required License

- *calresourceadv*
- *calscopeadv*

Substitute License

- None.

Also Enables

- Calibre Cluster Manager (CalCM)
- Calibre Dynamic Resource Allocator (DRA)

Considerations and Exceptions

- One *calresourceadv* is required for every four CPUs registered with CalCM+.
- One *calscopeadv* is required for every host monitored by CalCM+.

Related Topics

[Calibre Cluster Manager User's Manual](#)

Calibre Dynamic Resource Allocator (DRA)

Licensing for Calibre® Dynamic Resource Allocator.

Required License

- *calresourceman*

Substitute License

- *calresourceadv*

Also Enables

- Calibre Cluster Manager (CalCM)

Considerations and Exceptions

- None.

Related Topics

[Calibre Dynamic Resource Allocator \(DRA\) User's Manual](#)

Calibre FullScale

Licensing for Calibre® FullScale™.

Required License

- *calibrepto*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre Post-Tapeout Flow User's Manual \[Calibre Post-Tapeout Flow User's Manual\]](#)

Chapter 8

Licensing: Physical Verification Products

The licensing information for Calibre Physical Verification products provides information on required and substitute licenses, in addition to any considerations and exceptions.

Table 8-1 lists each Physical Verification product and the required and substitute license(s) for that product. Click the Calibre product name to view more details about the licensing requirements.

Table 8-1. Primary License Requirements for Physical Verification Products

Calibre Product	Required License(s)	Substitute License(s)
Calibre 3DSTACK	<i>calibrehdrc</i> , <i>calibredrc</i> , <i>calibrehlvs</i> , <i>calibrelvls</i> , and <i>caldesignrev</i>	None.
Calibre ADP (Advanced Device Properties)	<i>calibreadp</i>	<ul style="list-style-type: none">• <i>calibrexrc</i>• <i>calibrexccb</i>• <i>calibrexact</i>• <i>calibrexactsoc</i>• <i>calibrexact3d</i>
Calibre AutoFix	<i>calibreautofix</i> and <i>routerautofix</i>	<ul style="list-style-type: none">• <i>calibreinroute</i> for <i>calibreautofix</i>• <i>olympusstation</i> for <i>routerautofix</i>
Calibre Auto-Waivers	<ul style="list-style-type: none">• <i>calwaiver</i>• <i>calpmatch</i> for <i>waive_pattern</i> criteria (see Considerations and Exceptions)• <i>calwaiver</i> and <i>calpmatch</i> when the rule file contains Calibre Pattern Matching rules	<ul style="list-style-type: none">• <i>calibredrc</i> and <i>calibrehdrc</i> for Calibre DRC Auto-Waivers (see Considerations and Exceptions)• <i>calnmdrcgold</i>• <i>calibreperc</i> for Calibre PERC Auto-Waivers

Table 8-1. Primary License Requirements for Physical Verification Products

Calibre Product	Required License(s)	Substitute License(s)
Calibre CB (Cell/Block)	<i>caldrcrvseve</i>	For Calibre nmDRC Cell/Block: <ul style="list-style-type: none">• <i>calibredrc</i>• <i>calibrepvs_s</i>• <i>calibrelfd</i> For Calibre nmLVS Cell/Block: <ul style="list-style-type: none">• <i>calibrelv</i>• <i>calibrepvs_s</i>• <i>calibrexrccb</i> and <i>ictrace</i>• <i>calibrexrc</i> and <i>ictrace</i>
Calibre CI (Connectivity Interface)	<i>calibreci</i>	<ul style="list-style-type: none">• <i>calibrexrc</i>• <i>calibrexrccb</i>• <i>calibreexact</i>• <i>calibreexactsoc</i>• <i>calibreexact3d</i>
Calibre DESIGNrev	<i>caldesignrev</i>	<ul style="list-style-type: none">• <i>calmdpview</i>• <i>calibrelv</i>• <i>calworkbench</i>
Calibre e2lvs	No switch: <ul style="list-style-type: none">• <i>calibrelv</i> Using -cb: <ul style="list-style-type: none">• <i>caldrcrvseve</i> Using -ictrace: <ul style="list-style-type: none">• <i>ictrace</i>	No switch: <ul style="list-style-type: none">• <i>calibreperc</i> Using -cb: <ul style="list-style-type: none">• <i>calibrelv</i>• <i>calibreperc</i> Using -ictrace: <ul style="list-style-type: none">• <i>calibrelv</i>• <i>calibreperc</i>
Calibre Interactive	<i>calinteractive</i>	None.
Calibre Multi-Patterning	<i>calmultipattern</i>	<ul style="list-style-type: none">• <i>calmpgold</i>• <i>calnmdpc</i>• <i>calmfgmpgold</i>• <i>calnmdrcgold</i>

Table 8-1. Primary License Requirements for Physical Verification Products

Calibre Product	Required License(s)	Substitute License(s)
Calibre Multi-Patterning Advanced	<i>calmpgold</i>	<ul style="list-style-type: none"> • <i>calmpadv</i> and <i>calmultipattern</i> • <i>calmfgmpgold</i> • <i>calmfgmpadv</i> and <i>calnmdpc</i>
Calibre nmDRC	<i>calibredrc</i>	<ul style="list-style-type: none"> • <i>calibrepvs_s</i> • <i>calnmdrcgold</i> • <i>calibrelfd</i>
Calibre nmDRCgold	<i>calnmdrcgold</i> (see Considerations and Exceptions)	See Considerations and Exceptions.
Calibre nmDRC-H	<i>calibrehdrc</i> and <i>calibredrc</i>	<ul style="list-style-type: none"> • <i>calibrepvs_s</i> • <i>calnmdrcgold</i> • <i>calibrelfd</i>
Calibre nmLVS	<i>calibrelvs</i>	<ul style="list-style-type: none"> • <i>calibrepvs_s</i> • <i>calibrexrc</i> and <i>ictrace</i>
Calibre nmLVS-H	<i>calibrehlvs</i> and <i>calibrelvs</i>	<i>calibrepvs_s</i>
Calibre nmLVS Advanced	<ul style="list-style-type: none"> • <i>calibrehlvsadv</i> • <i>calibrehlvs</i> and <i>calibrelvs</i> 	<i>calmultipattern</i> for certain MP operations.
Calibre nmLVS Reconnaissance	<i>calreconlvs</i> <i>calibrehlvs</i> and <i>calibrelvs</i> <i>calibreqlb</i>	None.
Calibre Pattern Matching	<i>calpmatch</i>	<ul style="list-style-type: none"> • <i>calnmdrcgold</i> • <i>calibrelfd</i> (GUI-mode only)
Calibre Pattern Matching Manufacturing	<i>calpmatchmfg</i>	None.
Calibre PERC	<ul style="list-style-type: none"> • <i>calibreperc</i> • <i>calibreperc</i> for Calibre PERC LDL • <i>calibrehlvs</i> and <i>calibrelvs</i> for certain LVS applications 	None.

Table 8-1. Primary License Requirements for Physical Verification Products

Calibre Product	Required License(s)	Substitute License(s)
Calibre PERC Advanced	<ul style="list-style-type: none"> • <i>calibrepercadv</i> • <i>calibreperc</i> for Calibre PERC 	<i>calibreperc</i>
Calibre Physical Verification Suite (PVS)	See Considerations and Exceptions	None.
Calibre Query Server	<i>calibreqdb</i> or <i>caldrclvseve</i>	<i>calibrepvss</i>
Calibre RealTime Custom	<i>calrealtime</i>	None.
Calibre RealTime Custom Lite	<i>calrealtimecustomlite</i>	None.
Calibre RealTime Digital	<i>calrealtimedig</i>	None.
Calibre Rule File Analyzer	<i>calibreqdb</i> (see Considerations and Exceptions)	<ul style="list-style-type: none"> • <i>calibredrc</i> • <i>calibrelvss</i>
Calibre RVE	<i>calibreqdb</i>	<ul style="list-style-type: none"> • <i>caldrclvseve</i> • <i>calibrepvss</i>
Calibre SVRFencrypt	<i>calsvrfencrypt</i>	None.
Calibre v2lvs	<p>No switch:</p> <ul style="list-style-type: none"> • <i>calibrelvss</i> <p>Using -cb:</p> <ul style="list-style-type: none"> • <i>caldrclvseve</i> <p>Using -ictrace:</p> <ul style="list-style-type: none"> • <i>ictrace</i> 	<p>No switch:</p> <ul style="list-style-type: none"> • <i>calibreperc</i> <p>Using -cb:</p> <ul style="list-style-type: none"> • <i>calibrelvss</i> • <i>calibreperc</i> <p>Using -ictrace:</p> <ul style="list-style-type: none"> • <i>calibrelvss</i> • <i>calibreperc</i>
Tanner-Calibre One DRC/LVS	<i>caltannerpvss</i>	None.

Calibre 3DSTACK

Licensing for Calibre® 3DSTACK.

Required License

- *calibrehdrc, calibredrc, calibrehlvs, calibrelvls, and caldesignrev*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre 3DSTACK User's Manual](#)

Calibre ADP (Advanced Device Properties)

Licensing for Calibre ADP.

Required License

- *calibreadp*

Substitute License

- *calibrexrc*
- *calibrexrcb*
- *calibrexact*
- *calibrexactsoc*
- *calibrexact3d*

Also Enables

- None.

Considerations and Exceptions

- The Calibre ADP license (*calibreadp*) applies during the circuit extraction phase of a run. You can use the license to enable specialized device extraction functions such as well proximity enclosure functions and device signatures. During a Calibre nmLVS or Calibre nmLVS-H run, these functions produce the necessary parameters for post-layout simulation, including the Calibre xRC tool netlist output. The Calibre ADP licenses are released at the beginning of the LVS comparison phase when connectivity extraction and circuit comparison are performed in the same run.
- A *calibreadp* license is required when running Calibre nmLVS/Calibre nmLVS-H and using the following SVRF constructs:
 - Device SIGNATURE operations.
 - Device TVF functions: TVF_NUMERIC_FUNCTION and TVF_STRING_FUNCTION.
 - Layer operations: [DFM Expand Edge](#), [DFM OR Edge](#), [DFM Shift Edge](#), [DFM Size](#), and LITHO [OPCBIAS](#).
 - Device recognition built-in language functions: enclosure_parallel, enclosure_perpendicular, enclosure_parallel_multifinger, enclosure_perpendicular_multifinger.
 - DFM function: device::enclosure_measurements.

- ANNOTATE keyword: Used in a [Device Layer](#) command.
- [DFM Property](#) functions: SELECT_MERGE_XXY, SORT_MERGE_XXY, RANGE_XXY, and NORMALIZE_XY.
- When using Calibre ADP in multithreaded mode, the license requirements are based on the standard multithreaded consumption formula. If there are fewer Calibre ADP licenses than other Calibre licenses needed for the run, or vice-versa, then the fewer number of licenses governs the number of threads activated.

Calibre AutoFix

Licensing for Calibre® AutoFix™.

Required License

- *calibreautofix* and *routerautofix*

Substitute License

- *calibreinroute* for *calibreautofix*
- *olympusstation* for *routerautofix*

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre AutoFix User's Manual](#)

Calibre Auto-Waivers

Licensing for Calibre® Auto-Waivers™.

Required License

- *calwaiver*
- *calpmatch* for waive_pattern criteria (see Considerations and Exceptions)
- *calwaiver* and *calpmatch* when the rule file contains Calibre Pattern Matching rules

Substitute License

- *calibredrc* and *calibrehdrc* for Calibre DRC Auto-Waivers (see Considerations and Exceptions)
- *calnmdrcgold*
- *calibreperc* for Calibre PERC Auto-Waivers

Also Enables

- None.

Considerations and Exceptions

- The *calwaiver* license is used when running Calibre RVE generation of waiver shapes, the waiver_flow executable, the drc_waiver_asc2gds, or calibre -drc -waiver.
- A *calpmatch* license is required when waivers are applied based on the pattern definition parameters and waive_pattern criteria used in a pattern matching library. Refer to the [Calibre Pattern Matching](#) license reference page for more information.
- The *calibremt2cpu* license is not supported for the waiver_flow executable.
- To disable the substitution of the *calibredrc* and *calibrehdrc* licenses for the *calwaiver* license, you must set the CALIBRE_DRC_WAIVER_SUB_DRC_DISABLE environment variable to 308635.

Related Topics

[Calibre Auto-Waiver User's and Reference Manual](#)

Calibre CB (Cell/Block)

Licensing for Calibre® CB™.

Required License

- *caldrclvseve*

Substitute License

- For Calibre nmDRC Cell/Block:
 - *calibredrc*
 - *calibrepvs_s*
 - *calibrelfd*
- For Calibre nmLVS Cell/Block:
 - *calibrelvs*
 - *calibrepvs_s*
 - *calibrexrcrb* and *ictrace*
 - *calibrexrc* and *ictrace*

Also Enables

- None.

Considerations and Exceptions

- You can run the following tools in a single session (multiple sessions are not supported) with the *caldrclvseve* license:
 - Calibre nmDRC
 - Calibre nmLVS
 - Calibre RVE
 - Calibre Query Server
 - When you perform ERC in flat nmLVS (for example, using the command `calibre -spice`), Calibre does not require any additional licenses.
- To use this license from the command line, you must specify the optional `-cb` argument to the Calibre invocation on the command line.

- To use this license in Calibre Interactive, you can enable or disable Calibre Cell/Block license acquisition for a Calibre Interactive run.

Related Topics

[Calibre Interactive User's Manual](#)

[Calibre RVE User's Manual](#)

Calibre CI (Connectivity Interface)

Licensing for Calibre CI.

Required License

- *calibrecl*

Substitute License

- *calibrexrc*
- *calibrexrcb*
- *calibrexact*
- *calibrexactsoc*
- *calibrexact3d*

Also Enables

- None.

Considerations and Exceptions

- The Calibre Connectivity Interface (CI) is a set of licensed functionality designed for enabling conversion of the Calibre nmLVS SVDB results database into standards-based file formats. You can tailor these formats for downstream tools requiring direct access to nmLVS extraction and comparison results.
- Calibre CI commands require the *calibrecl* license for writing results information. Calibre CI commands run from within Calibre Query Server, which requires the *calibreqlb* license.
- The *calibrecl* license is also required when the “CCI” or “NETLIST” options are specified in the Mask SVDB Directory rule file statement.

Related Topics

[Calibre Connectivity Interface \[Calibre Query Server Manual\]](#)

Calibre DESIGNrev

Licensing for Calibre® DESIGNrev™.

Required License

- Standard mode:
 - 1 *caldesignrev*
- High Capacity (HC) mode:
 - 2 *caldesignrev*

Substitute License

- *calmdpview*
- *calibrelv*
- *calworkbench*

Also Enables

- None.

Considerations and Exceptions

- Calibre DESIGNrev does not require additional licenses for hardware threads.
- You can specify a license timeout interval for Calibre DESIGNrev by setting the MGC_DRV_RELEASE_LICENSE_TIME environment variable to the number of hours in which to release the license.

Related Topics

[Calibre DESIGNrev Layout Viewer User's Manual](#)

[Calibre DESIGNrev Reference Manual](#)

Calibre e2lvs

Licensing for Calibre e2lvs.

Required License

- No switch:
 - *calibreelvs*
- Using -cb:
 - *caldrcrvseve*
- Using -ictrace:
 - *ictrace*

Substitute License

- No switch:
 - *calibreperc*
- Using -cb:
 - *calibreelvs*
 - *calibreperc*
- Using -ictrace:
 - *calibreelvs*
 - *calibreperc*

Also Enables

- None.

Considerations and Exceptions

- You can specify which license the tool uses by adding -cb or -ictrace to the tool's command line invocation.

Related Topics

[E2LVS \[Calibre Verification User's Manual\]](#)

Calibre Interactive

Licensing for Calibre® Interactive™.

Required License

- *calinteractive*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre Interactive User's Manual](#)

[Calibre RVE User's Manual](#)

Calibre Multi-Patterning

Licensing for Calibre® Multi-Patterning.

Required License

- *calmultipattern*

Substitute License

- *calmpgold*
- *calnmdpc*
- *calmfgmpgold*
- *calnmdrcgold*
- *calibrehlvsadv* — For certain Multi-Patterning checks in an LVS run.

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre Multi-Patterning User's and Reference Manual](#)

Calibre Multi-Patterning Advanced

Licensing for Calibre® Multi-Patterning Advanced.

Required License

- *calmpgold*

Substitute License

- *calmpadv* and *calmultipattern*
- *calmfgmpgold*
- *calmfgmpadv* and *calnmdpc*

Also Enables

- Calibre Multi-Patterning

Considerations and Exceptions

- None.

Keywords and Operations Licensed by Calibre Multi-Patterning Advanced

The following keywords and operations require the *calmpgold* license.

- DFM CLUSTER
 - COUNT
 - DISTANCE
 - SELECT_CUT
 - SELECT_ENCLOSE
 - SELECT_INSIDE
 - SELECT_INTERACT
 - SELECT_NOT_CUT
 - SELECT_NOT_ENCLOSE
 - SELECT_NOT_INSIDE
 - SELECT_NOT_INTERACT
 - SELECT_NOT_OUTSIDE

- SELECT_NOT_TOUCH
- SELECT_OUTSIDE
- SELECT_TOUCH
- SPACING
- [DFM MP](#)
 - REDUCED_TARGET_POLYGONS
 - REDUCED_TARGET_SEPARATORS
 - separator_layer [(OPPOSITE | SAME nicety > 0)]
 - stitch_layer (STITCH ...)
- [DFM MPSTITCH](#)
- [RET SIDCUTFILL](#)
- *mp_stitch_gen.tvf*

Related Topics

[Calibre Multi-Patterning User's and Reference Manual](#)

Calibre nmDRC

Licensing for Calibre® nmDRC™.

Required License

- *calibredrc*

Substitute License

- *calibrepvs_s*
- *calnmdrcgold*
- *calibrelfd*

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre Verification User's Manual](#)

Calibre nmDRCgold

Licensing for Calibre® nmDRCgold.

Required License

- *calnmndrcgold* (see Considerations and Exceptions)

Substitute License

- See Considerations and Exceptions.

Also Enables

- None.

Considerations and Exceptions

- You can run any combination of the following with the *calnmndrcgold* license:
 - Calibre nmDRC or Calibre nmDRC-H
 - Calibre Pattern Matching
 - Calibre Multi-Patterning
 - Calibre Auto-Waivers Flow
- *calnmndrcgold* is the preferred license in runs including Calibre nmDRC or Calibre nmDRC-H and two or more of the products listed above. In other runs, individual product licenses are preferred, and you can use the *calnmndrcgold* license as a substitute.

Related Topics

[Calibre Auto-Waiver User's and Reference Manual](#)

[Calibre Multi-Patterning User's and Reference Manual](#)

[Calibre Pattern Matching Reference Manual](#)

[Calibre Pattern Matching User's Manual](#)

[Calibre Verification User's Manual](#)

Calibre nmDRC-H

Licensing for Calibre® nmDRC-H™.

Required License

- *calibrehdrc* and *calibredrc*

Substitute License

- *calibrepvs_s*
- *calnmdrcgold*
- *calibrelfd*

Also Enables

- DBdiff utility (2009.4 and later releases)
- MDP EMBED
- FastXOR
- DFM operations listed in “[DFM Layer Operations Licensed by Calibre nmDRC-H](#)”

Considerations and Exceptions

- When running on a two-CPU machine, you can use the *calibremt2cpu* license to enable the second CPU.

DFM Layer Operations Licensed by Calibre nmDRC-H

The following DFM layer operations require the *calibrehdrc* and *calibredrc* licenses:

- [DFM Circle Analyze](#)
- [DFM Classify](#)
- [DFM Copy](#)
- [DFM DV](#)
- [DFM DV Analyze](#)
- [DFM DV Override](#)
- [DFM Merge](#)
- [DFM NARAC](#)
- [DFM OR](#)

- [DFM Property](#)
- [DFM Property Merge](#)
- [DFM Property Net](#)
- [DFM Property Select Secondary](#)
- [DFM Property Singleton](#)
- [DFM RDB \(non-nodal only\)](#)
- [DFM Read](#)
- [DFM Select By Net](#)
- [DFM Space](#)
- [DFM Stamp](#)
- [DFM Text](#)

Refer to [Calibre YieldAnalyzer](#) for a list of the DFM operations that require a Calibre YieldAnalyzer license and [Calibre YieldEnhancer](#) for a list of the DFM operations that require a Calibre YieldEnhancer license.

Related Topics

[Calibre Verification User's Manual](#)

Calibre nmLVS

Licensing for Calibre® nmLVS™.

Required License

- *calibrelvs*

Substitute License

- *calibrepvs_s*
- *calibrexrc* and *ictrace*

Also Enables

- None.

Considerations and Exceptions

- When you perform ERC in flat Calibre nmLVS, the Calibre tool secures both a flat Calibre nmDRC (*calibredrc*) and a flat Calibre nmLVS (*calibrelvs*) license.

Related Topics

[Calibre Verification User's Manual](#)

Calibre nmLVS-H

Licensing for Calibre® nmLVS-H.

Required License

- *calibrehlvs* and *calibrelvs*

Substitute License

- *calibrepvs_s*

Also Enables

- None.

Considerations and Exceptions

- A *calibrelvs* and *calibrehlvs* license is required for LVS circuit extraction, in cases where a device_layer, pin_layer, or auxiliary_layer in a [Device](#) statement is derived from a [DFM Property](#) operation.
- When you use Calibre nmLVS-H for SPICE extraction (calibre -spice), without any circuit comparison functionality, Calibre nmLVS secures a nmLVS-H license set (*calibrelvs* and *calibrehlvs*).
- When running on a two-CPU machine, Calibre nmLVS-H provides access to the *calibremt2cpu* license, specifically for running on a two-CPU machine.
- When you perform ERC in Calibre nmLVS-H (for example, using the command line calibre -spice), the Calibre tool secures the following licenses: *calibredrc*, *calibrehdrc*, *calibrelvs*, and *calibrehlvs*. The Calibre tool releases the Calibre nmDRC and Calibre nmDRC-H licenses that are in use before the nmLVS-H comparison begins.
- When you perform ERC in nmLVS-H with the -turbo (multithreaded) switch, the Calibre tool secures additional nmDRC and nmLVS license pairs depending on the number of CPUs used.

Related Topics

[Calibre Verification User's Manual](#)

Calibre nmLVS Advanced

Licensing for specialized Calibre nmLVS processes.

Required License

- *calibrehlvsadv*

Substitute License

- None.

Also Enables

- Certain Multi-Patterning checks when substituting for a Calibre Multi-Patterning license.
- Calibre nmLVS Reconnaissance.

Considerations and Exceptions

You can use the Calibre nmLVS Advanced license (*calibrehlvsadv*) to enable certain adjunct features to usual LVS circuit extraction and comparison processes. This license is used in addition to a Calibre nmLVS/Calibre nmLVS-H license pair. The *calibremt2cpu* license does not apply to Calibre nmLVS Advanced license usage.

Specifically, *calibrehlvsadv* is required in the following circumstances:

- LVS comparison on a separate machine when the *-cmp_host* or *-cmp_remotefile* command line option is used. There is no substitution license.

The LVS comparison process counts as one CPU (thread) among the aggregate of processors enabled for the run. Thus, a specific number of CPUs requested for other tasks than LVS comparison is reduced by one to accommodate the LVS comparison thread.

The following table shows examples that assume an 8-CPU primary host and one remote machine.

Table 8-2. Calibre nmLVS Advanced License CPU Allocation

-cmp_host or -cmp_remotefile?	MT Setting	Local CPUs Used	Remote CPUs for Comparison	Total CPUs	License Multiple (per license)
N	-turbo	8	0	8	3
Y	-turbo 8	7	1	8	3
Y	-turbo	8	1	9	4

Table 8-2. Calibre nmLVS Advanced License CPU Allocation (cont.)

-cmp_host or -cmp_remotefile?	MT Setting	Local CPUs Used	Remote CPUs for Comparison	Total CPUs	License Multiple (per license)
Y	none	1	1	2	2

The “[Licensing for Multithreaded Operations](#)” multiplier formula applies to *all licenses* that are acquired, including *calibrehlvsadv*.

- Certain multi-patterning operations are enabled for device extraction or ERC checks with a Calibre Multi-Patterning or Calibre nmLVS Advanced license. The same multi-threaded license acquisition formula used as for other LVS licenses is used in this case.
- The [LVS DB Connectivity Layer](#) statement is used during circuit extraction (calibre -spice).
- Certain advanced device extraction computations when processing more than 15 input layers. Contact your Siemens representative for more details.
- Using a layer derived from a [device::blocked_extent](#) function.

Calibre nmLVS Reconnaissance

Licensing for Calibre® nmLVS stand-alone ERC, short isolation, and soft connection checking.

Required License

- *calreconlvs*
- *calibrelvs* and *calibrehlvs*
- *calibreqdb* (See Considerations and Exceptions)

Substitute License

- *calibrehlvsadv*

Also Enables

- None.

Considerations and Exceptions

- A single *calreconlvs* license is required regardless of whether the run is multithreaded.
- If *calibrehlvsadv* is substituted for *calreconlvs* in a multithreaded run, then the *calibrehlvsadv* licenses are acquired according to the usual multithreaded license scheme.
- The usual multithreaded license scheme is used for *calibrelvs* and *calibrehlvs* license pairs.
- A single *calibreqdb* license is required if *-si=tcl_script* or *-softchk=tcl_script* is specified.

Related Topics

[Calibre nmLVS and Calibre nmLVS-H Command Line \[Calibre Verification User's Manual\]](#)

Calibre Pattern Matching

Licensing for Calibre® Pattern Matching.

Required License

- *calpmatch*

Substitute License

- *calnmdrcgold*
- *calibrelfd* (GUI-mode only)

Also Enables

- Calibre DRC Auto-Waivers Flow (calibre -drc -waiver) when waivers are applied based on the pattern definition parameters and waive_pattern criteria used in a pattern matching library.

Considerations and Exceptions

- To run Calibre Pattern Matching in Calibre nmDRC/nmDRC-H, the respective [Calibre nmDRC](#) and [Calibre nmDRC-H](#) license is required.
- To run pattern capture from [Calibre DESIGNrev](#), [Calibre WORKbench](#), [Calibre MDPview](#), or [Calibre LITHOview](#), a Calibre Pattern Matching license is required in addition to the applicable license for the layout viewer.

Related Topics

[Calibre Pattern Matching Reference Manual](#)

[Calibre Pattern Matching User's Manual](#)

Calibre Pattern Matching Manufacturing

Licensing for Calibre® Pattern Matching Manufacturing.

Required License

- *calpmatchmfg*

Substitute License

- None.

Also Enables

- DFM Data Analysis (DFMDA)

Considerations and Exceptions

- To run Calibre Pattern Matching Manufacturing in Calibre nmDRC/nmDRC-H, the respective [Calibre nmDRC](#) and [Calibre nmDRC-H](#) license is required.
- To run pattern capture from [Calibre DESIGNrev](#), [Calibre WORKbench](#), [Calibre MDPview](#), or [Calibre LITHOview](#), a Calibre Pattern Matching license is required in addition to the applicable license for the layout viewer.

Related Topics

[Calibre Pattern Matching Reference Manual](#)

[Calibre Pattern Matching User's Manual](#)

[DFM Data Analysis User's and Reference Manual](#)

Calibre PERC

Licensing for Calibre® PERC™.

Required License

- *calibreperc* for Calibre PERC
- *calibreperc* for Calibre PERC LDL

Substitute License

- *calibrelvs* and *calibrehlvs* with -lvs_supplement (see Considerations and Exceptions)

Also Enables

- None.

Considerations and Exceptions

- You can use the -lvs_supplement command line option to run circuit extraction on additional CPUs by using Calibre nmLVS/Calibre nmLVS-H license pairs in a one-to-one ratio with Calibre PERC licenses rather than all Calibre PERC licenses. Refer to the *Calibre PERC User's Manual* for more information on using the -lvs_supplement option.
- For Calibre PERC LDL:
 - Use of ldl::svrf, ldl::save, and dfm::new_layer reserves a [Calibre nmDRC/Calibre nmDRC-H](#) license pair.
 - Geometric operations that are used internally to Calibre PERC that require DRC functionality will continue to operate without the need for additional DRC licenses.
 - In the LDL source-based flow, a [Calibre nmLVS/Calibre nmLVS-H](#) license pair is required when performing an LVS comparison to access source-based names and properties. In the source-based flow, a Calibre PERC license can substitute for the Calibre nmLVS/Calibre nmLVS-H license pair.
 - You can use the -hold_lic option to hold the specified licenses for the entirety of the run rather than only when the licenses are required by the tool. Refer to the *Calibre PERC User's Manual* for more information on using the -hold_lic option.
- A [Calibre Auto-Waivers](#) license is required when the [PERC Waiver Path](#) or [PERC LDL Waiver Path](#) specification statement is used.
- This license is required for voltage tracing in Calibre RVE for PERC.

Related Topics

[Calibre PERC User's Manual](#)

[Calibre RVE User's Manual](#)

Calibre PERC Advanced

Licensing for specialized Calibre PERC and Calibre RVE processes.

Required License

- *calibrepercadv* (see Considerations and Exceptions).

Substitute License

- *calibreperc* (see Considerations and Exceptions).

Also Enables

- None.

Considerations and Exceptions

You can use the Calibre PERC Advanced license (*calibrepercadv*) to enable certain adjunct features to usual processes involving Calibre PERC. This license is used in addition to a Calibre PERC license. For multithreaded runs, the “[Licensing for Multithreaded Operations](#)” multiplier formula applies to *all licenses* that are acquired.

A *calibrepercadv* license is required and is not substituted for the following:

- When using [perc_ldl::custom_r0](#).
- When using [dfm::copy_svdb_to_dfmdb](#) to transform an SVDB for use by Calibre PERC.
- When accessing [dfm::copy_svdb_to_dfmdb](#) transformed database output in Calibre PERC and Calibre RVE.

A *calibrepercadv* license is required and is substituted for the following:

- When using [perc::report_base_result -subResultProperty](#).
- When accessing [perc::report_base_result -subResultProperty](#) output using Calibre RVE.

Related Topics

[Calibre PERC User’s Manual](#)

[Calibre RVE User’s Manual](#)

Calibre Physical Verification Suite (PVS)

Licensing for Calibre Physical Verification Suite.

Required License

- See Considerations and Exceptions

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- Calibre PVS is a product bundle that includes both a *calibrepvs_s* and a *calibreintrtl_s* license.
 - If you have a *calibrepvs_s* license, you can run the following tools from a Calibre Interactive session (one instance per license):
 - Calibre ADP
 - Calibre nmDRC/nmDRC-H
 - Calibre nmLVS/nmLVS-H
 - Calibre RVE/Query Server
 - If you have a *calibreintrtl_s* license, you can run the following tools from a Calibre Interactive session (one instance per license):
 - Calibre Interactive
 - Calibre RealTime Custom Lite
- When you perform ERC in flat or hierarchical nmLVS (for example, using the command line `calibre -spice`), Calibre does not require any additional licenses.
- Calibre PVS is not supported by Calibre Cluster Manager (CalCM).

Related Topics

- [Calibre Interactive User's Manual](#)
- [Calibre Query Server Manual](#)
- [Calibre RVE User's Manual](#)
- [Calibre Verification User's Manual](#)

Calibre Query Server

Licensing for Calibre Query Server.

Required License

- *calibrereqdb* or *caldrclvseve*

Substitute License

- *calibrepvs_s*

Also Enables

- None.

Considerations and Exceptions

- When running Query Server without -cb, only the *calibrereqdb* license is required.
- When running Query Server with -cb, Calibre attempts to acquire a *caldrclvseve* license. If a *caldrclvseve* is not available, Calibre attempts to acquire a *calibrereqdb* license.

Related Topics

[Calibre Query Server Manual](#)

Calibre RealTime Custom

Licensing for Calibre® RealTime Custom.

Required License

- *calrealtime*

Substitute License

- None.

Also Enables

- DFM operations listed in “[DFM Layer Operations Licensed by Calibre nmDRC-H](#)”
- Calibre Pattern Matching (batch mode)
- Calibre Multi-Patterning (flat only)

Considerations and Exceptions

- None.

Related Topics

[Calibre RealTime Custom User’s Manual](#)

Calibre RealTime Custom Lite

Licensing for Calibre® RealTime Custom Lite.

Required License

- *calrealtimelite* (see Considerations and Exceptions)

Substitute License

- None.

Also Enables

- DFM operations listed in “[DFM Layer Operations Licensed by Calibre nmDRC-H](#)”.
- Tanner L-Edit integration.

Considerations and Exceptions

- In addition to the required license, Calibre RealTime Custom Lite checks for the existence of a [Calibre RVE](#) license, but does not hold the license.

Related Topics

[Calibre RealTime Custom User’s Manual](#)

Calibre RealTime Digital

Licensing for Calibre® RealTime Digital.

Required License

- *calrealtimedig*

Substitute License

- None.

Also Enables

- DFM operations listed in “[DFM Layer Operations Licensed by Calibre nmDRC-H](#)”
- Calibre Pattern Matching (batch mode)
- Calibre Multi-Patterning (flat only)
- Calibre Reconnaissance (-recon) mode
- DRC Analyze (-analyze)
- [DFM RDB DEF](#)
- [Layout Windel Cell HIER and HALO keywords](#)

Considerations and Exceptions

- None.

Related Topics

[Calibre RealTime Digital User’s Manual](#)

Calibre Rule File Analyzer

Licensing for Calibre Rule File Analyzer.

Required License

- *calibrereqdb* (see Considerations and Exceptions)

Substitute License

- *calibredrc*
- *calibrelhs*

Also Enables

- None.

Considerations and Exceptions

- In addition to the required licenses, Calibre Rule File Analyzer checks for the existence of a *calibrehdrc* or *calibrelhs* license, but it does not hold the license.

Related Topics

[Calibre Verification User's Manual](#)

Calibre RVE

Licensing for Calibre® RVE™.

Required License

- *calibreqdb*

Substitute License

- *caddrclvseve*
- *calibrepvs_s*

Also Enables

- None.

Considerations and Exceptions

- When -cb is specified, the *caddrclvseve* license is preferred instead of the *calibreqdb* license.
- When you start Calibre RVE from a supported layout editor, the -cb argument is set with the “Use Calibre-CB license” checkbox in the opening dialog box for RVE. Additionally, you can specify the Calibre Cell/Block license in Calibre Interactive nmDRC and nmLVS Graphical User Interfaces.
- Some features in Calibre RVE require additional licenses. For example, the following features in Calibre RVE have additional license requirements:
 - DRC HTML reporting — Also requires a [Calibre DESIGNrev](#) or [Calibre MDPview](#) license if layout images are included in the DRC HTML report.
 - Waiver export — Also requires a [Calibre Auto-Waivers](#) license.
 - Interactive short isolation — Also requires a [Calibre nmLVS](#) and [Calibre nmLVS-H](#) license.
 - DFM HTML batch reporting — Also requires a [Calibre nmDRC](#), [Calibre nmDRC-H](#), and [Calibre YieldAnalyzer](#) license.

Related Topics

[Calibre RVE User’s Manual](#)

Calibre SVRFencrypt

Licensing for Calibre SVRFencrypt.

Required License

- *calsvrfencrypt*

Substitute License

- None.

Also Enables

- TVFencrypt
- Caltxtencrypt utility

Considerations and Exceptions

- None.

Related Topics

[Calibre SVRFencrypt User's Manual](#)

Calibre v2lvs

Licensing for Calibre v2lvs.

Required License

- No switch:
 - *calibre**lvs*
- Using -cb:
 - *caldrcrvseve*
- Using -ictrace:
 - *ictrace*

Substitute License

- No switch:
 - *calibre**perc*
- Using -cb:
 - *calibre**lvs*
 - *calibre**perc*
- Using -ictrace:
 - *calibre**lvs*
 - *calibre**perc*

Also Enables

- None.

Considerations and Exceptions

- You can specify which license the tool uses by adding -cb or -ictrace to the tool's command line invocation.

Related Topics

[V2LVS \[Calibre Verification User's Manual\]](#)

Tanner-Calibre One DRC/LVS

Licensing for Tanner-Calibre One DRC/LVS.

Required License

- *caltannerpvs*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- Tanner-Calibre One DRC/LVS is a product bundle that includes both a Tanner-Calibre One DRC/LVS (*caltannerpvs*) and a **Calibre Interactive** (*calinteractive*) license.
- Tanner-Calibre One DRC/LVS license can only be used with the Tanner L-Edit tool.
- Calibre jobs run on only one CPU core.
- You can run one instance of the following with the *caltannerpvs* license:
 - Calibre ADP
 - Calibre nmDRC/nmDRC-H
 - Calibre nmLVS/nmLVS-H
 - Calibre RVE
- When you perform ERC in flat or hierarchical nmLVS (for example, using the command line `calibre -spice`), Calibre does not require any additional licenses.

Related Topics

[Calibre Interactive User's Manual](#)

Chapter 9

Licensing: Design for Manufacturability (DFM) Products

The licensing information for Calibre DFM products provides information on required and substitute licenses, in addition to any considerations and exceptions.

[Table 9-1](#) lists each DFM product and the required and substitute license(s) for that product. Click the Calibre product name to view more details about the licensing requirements.

Table 9-1. Primary License Requirements for Design for Manufacturability (DFM) Products

Calibre Product	Required License(s)	Substitute License(s)
Calibre CMPAnalyzer	<i>calcmpsim, calibreqdb, calyieldserver, calcmpsimconx, calinteractive, and calibredrc/calibrehdrc</i>	None.
Calibre CMP Model Builder	<i>calcmpmb</i> and <i>calworkbench</i>	None.
Calibre LFD	<i>calibrelfd</i>	None.
Calibre YieldAnalyzer	<i>calyieldanalyze</i>	None.
Calibre YieldEnhancer	<i>calyieldenhance</i>	None.
Calibre YieldServer	<i>calyieldserver</i>	<ul style="list-style-type: none">• <i>calibrehdrc</i> and <i>calibredrc</i>• <i>calibrehlvs</i> and <i>calibrelvls</i>

Note

 Some DFM operations require only a [Calibre nmDRC-H](#) license. Refer to “[DFM Layer Operations Licensed by Calibre nmDRC-H](#)” on page 107 for a list of these supported DFM operations.

Calibre CMPAnalyzer

Licensing for Calibre CMPAnalyzer.

Required License

- *calcmpsim, calibrereqdb, calyieldserver, calcmpsiconx, calinteractive, and calibredrc/calibrehdrc*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- If “Enable Smart Distribution for Multi-Threaded Run” for Calibre CMPAnalyzer is selected in the **Simulator Options** tab of the Calibre Interactive Inputs, then one additional license is required when CPUs “All” is specified in the multi-processing flow (parallel flow) compared to the default flow (sequential flow).

Related Topics

[Calibre CMPAnalyzer User’s Manual](#)

Calibre CMP Model Builder

Licensing for Calibre CMP Model Builder.

Required License

- *calcmpmb* and *calworkbench*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- When using the `cmpoptimize` batch command outside of the Calibre WORKbench GUI, only the *calcmpmb* license is required. When not performing simulations and optimizations, only the *calworkbench* license is required.

Related Topics

[Calibre CMP Modeling User's and Reference Manual](#)

Calibre LFD

Licensing for Calibre® LFD™.

Required License

- *calibrelfd*

Substitute License

- None.

Also Enables

- Calibre nmDRC-H
- Calibre OPCpro
- Calibre nmOPC
- Calibre OPCsbar
- Calibre OPCverify
- Calibre Pattern Matching (GUI-mode only)
- Calibre DBdiff

Considerations and Exceptions

- You can substitute Calibre LFD licenses for non-LFD runs. Note that each *calibrelfd* license replaces a single alternate license.
- Calibre LFD requires one license per number of CPUs/8 +1.

Related Topics

[Calibre Litho-Friendly Design User's Manual](#)

Calibre YieldAnalyzer

Licensing for Calibre YieldAnalyzer.

Required License

- *calyieldanalyze*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- None.

DFM Layer Operations Licensed by Calibre YieldAnalyzer

The following DFM layer operations require a *calyieldanalyze* license:

- [DFM Analyze](#)
- [DFM CAF](#)
- [DFM Critical Area](#)
- [DFM GCA](#)
- [DFM Histogram](#)
- [DFM Measure](#)
- [DFM Redundant Vias](#)

Refer to [Calibre nmDRC-H](#) for a list of the DFM operations that require a Calibre nmDRC-H license and [Calibre YieldEnhancer](#) for a list of the DFM operations that require a Calibre YieldEnhancer license.

Related Topics

[Calibre YieldAnalyzer and YieldEnhancer User's and Reference Manual](#)

Calibre YieldEnhancer

Licensing for Calibre YieldEnhancer.

Required License

- *calyieldenhance*

Substitute License

- *calyieldanalyze* (DFM Contour only)

Also Enables

- None.

Considerations and Exceptions

- The Calibre YieldEnhancer PowerVia utility requires the *calyieldenhance* and *caldesignrev* licenses, in addition to a *calibrehdrc* and *calibredrc* license pair. If a *caldesignrev* license is not available, you must specify the LAYOUT_PRECISION variable in the PowerVia *techlib.tcl* file. Additionally, use of the MERGED_OUTPUT variable requires a *caldesignrev* license.

Multithreaded mode license acquisition is based upon the formula given under “[License Requirements for Distributed Calibre](#)” on page 71.

DFM Layer Operations Licensed by Calibre YieldEnhancer

- The following DFM layer operations require a *calyieldenhance* license:
 - [DFM Contour](#)¹
 - [DFM Density](#)
 - [DFM Expand Edge](#)
 - [DFM Expand Enclosure](#)
 - [DFM Fill](#)
 - [DFM Fill Wrap](#)
 - [DFM Grow](#)
 - [DFM Jogclean](#)
 - [DFM Optimize](#)

1. The [Calibre YieldAnalyzer](#) license is substituted if a Calibre YieldEnhancer license is unavailable.

- [DFM OR Edge](#)
- [DFM Partition](#)
- [DFM RDB \(nodal only\)](#)
- [DFM RDB ASCII](#)
- [DFM Rectangles](#)
- [DFM Remove Edge](#)
- [DFM Reshape](#)
- [DFM Segment](#)
- [DFM Shift](#)
- [DFM Shift Edge](#)
- [DFM Size](#)
- [DFM Stripe](#)
- [DFM Transform](#)
- [DFM Transition](#)
- [DFM Via Shift](#)

Refer to [Calibre nmDRC-H](#) for a list of the DFM operations that require a Calibre nmDRC-H license and [Calibre YieldAnalyzer](#) for a list of the DFM operations that require a Calibre YieldAnalyzer license.

Related Topics

[Calibre YieldAnalyzer and YieldEnhancer User's and Reference Manual](#)

Calibre YieldServer

Licensing for Calibre YieldServer.

Required License

- *calyieldserver*

Substitute License

- *calibrehdrc* and *calibredrc*
- *calibrehlvs* and *calibrelvls*
- *calibreperc*

Also Enables

- None.

Considerations and Exceptions

- Depending on the operation you want to perform from YieldServer, you must have the appropriate license. For example, if you want to perform a Calibre nmDRC operation from YieldServer, you must have a [Calibre nmDRC](#) or [Calibre nmDRC-H](#) license.
- After the Calibre nmDRC-H licenses are secured, the licenses are held until the YieldServer session completes. Licenses requirements for other SVRF commands that are executed during the run are released as soon as the operation completes.
- Licenses for SVRF that are executed by `dfm::new_layer` are released as soon as the operation completes.

Related Topics

[Calibre YieldServer Reference Manual](#)

Chapter 10

Licensing: Parasitic Extraction Products

The licensing information for Calibre Parasitic Extraction products provides information on required and substitute licenses, in addition to any considerations and exceptions. For Calibre xRC, you can utilize a unique licensing capability with the License Broker Daemon (LBD).

Table 10-1 lists each Parasitic Extraction product and the required and substitute license(s) for that product. Click the Calibre product name to view more details about the licensing requirements.

Table 10-1. Primary License Requirements for Parasitic Extraction Products

Calibre Product	Required License(s)	Substitute License(s)
Calibre xACT	<i>calibrexact</i>	<ul style="list-style-type: none">• <i>calibrexrc</i> and <i>calibrexrctoact</i>• <i>calibrexactsoc</i>
Calibre xACT 3D	<ul style="list-style-type: none">• <i>calibrexact</i> for PHDB generation and netlist formatting• <i>calibrexact3d</i> for PDB generation	<i>calibrexrc</i> (when running Calibre xACT for PHDB generation and netlist formatting)
Calibre xACTView	<i>calibrexact3d</i> (see Considerations and Exceptions)	None.
Calibre xL	<i>calibrexl</i>	None.
Calibre xRC	<i>calibrexrc</i>	<ul style="list-style-type: none">• <i>calibrexact</i>• <i>calibrexactsoc</i>
Calibre xRC CB	<i>calibrexccb</i>	None.
Calibre xRC to ADVance MS	<i>calibrexcadms</i>	None.
Tanner-Calibre One xRC	<i>caltannerxrc</i>	None.
xCalibrate Rule File Generator	<i>xcalibrate</i>	None.

Calibre License Broker Daemon (LBD) for Calibre xRC

Calibre xRC employs a unique licensing capability utilizing the License Broker Daemon (LBD). With this capability, you can start an LBD server that checks out a specified number of Calibre xRC (*calibrexrc*) licenses from the FLEXnet license server. You can then run one or more Calibre xRC jobs using the LBD as the license server. When the jobs complete, you can stop the LBD server which releases the licenses back to the FLEXnet license server.

Ibd -xrc_flow.....	139
Using the LBD Server for Calibre xRC Jobs	140

lbd -xrc_flow

Starts and stops the LBD server needed to run Calibre xRC jobs.

Usage

```
lbd -xrc_flow {{ -start state_filename [-license_count number_of_licenses] [  
[-log log_filename | stdout] [-fg]} | {-stop state_filename} }}
```

Arguments

- **-start state_filename** [-license_count *number_of_licenses*] [-log *log_filename* | stdout] [-fg]

Starts the LBD server. The *state_filename* value specifies the name of an ASCII file containing the hostname and port on which the LBD server is listening. Calibre xRC jobs use the information in this file to connect to the LBD server. Valid options for starting the LBD server include the following:

- license_count *number_of_license* — Specifies the number of Calibre xRC licenses to check out for running Calibre xRC jobs. The default is 1.
- log *log_filename* | stdout — Specifies the name of a log file into which information regarding the LBD is output. Alternatively, you can specify “stdout” to print the LBD information to stdout. The stdout value is only valid when -fg is also specified.
- fg — Specifies the LBD should run in the foreground, instead of running as a daemon in background mode.

- **-stop state_filename**

Stops the LBD server. The *state_filename* value is the same filename that was specified when starting the LBD server.

Examples

Example 1

This example starts the LBD server and checks out three licenses for the Calibre xRC job.

```
lbd -xrc_flow -start my_xrc_job -license_count 3
```

Example 2

This example stops the LBD server.

```
lbd -xrc_flow -stop my_xrc_job
```

Related Topics

[Using the LBD Server for Calibre xRC Jobs](#)

Using the LBD Server for Calibre xRC Jobs

This procedure describes how to start an LBD license server, run a Calibre xRC job, and then stop the LBD license server.

Prerequisites

- Appropriate licenses for running [Calibre xRC](#).
- A state file containing the hostname and port on which to run the LBD license server.

Procedure

1. Enter the following command to start the LBD server and check out three Calibre xRC licenses. Replace *<state_filename>* with the filename containing the LBD server information:

```
lbd -xrc_flow -start <state_filename> -license_count 3
```

Check the log file for a 0 exit status, which indicates a successful launch of the LBD.

If you receive a non-zero exit status, check the log file for error messages. For example, you may receive an error message and non-zero exit status if there is a problem consuming the xRC licenses.

2. To execute your Calibre xRC jobs, enter the following commands, where *<state_filename>* is the name of the state file you specified when starting the LBD in step 1:

- For PHDB:

```
calibre -xrc -lbd @<state_filename> -phdb
```

- For PDB:

```
calibre -xrc -lbd @<state_filename> -pdb
```

- For Format:

```
calibre -xrc -lbd @<state_filename> -fmt
```

You can execute multiple Calibre xRC jobs simultaneously, as long as the total number of licenses required to run the jobs does not exceed the number of licenses allocated (see step 1) for the job.

If all LBD licenses are busy when starting a job, the job indicates all licenses are in use and exits. The LBD does not support license queueing. If you stop the LBD or a job loses network connectivity with the LBD, the job aborts shortly thereafter.

3. After you finish running your Calibre xRC jobs, enter the following command to shut down the LBD server, replacing *<state_filename>* with the name of the state file you specified in step 1.

```
lbd -xrc_flow -stop <state_filename>
```

Shutting down the LBD server releases the Calibre xRC licenses back to the FLEXnet server.

Related Topics

[lbd -xrc_flow](#)

[Lost Connection Between Calibre and the MSL License Server](#)

Calibre xACT

Licensing for Calibre® xACT™.

Required License

- *calibrexact*
- *calibrexact* and *calibrexrl* for Calibre xACT inductance flow

Substitute License

- *calibrexrc* and *calibrexrctoxact*
- *calibrexactsoc*
- *calibrexact* and *calibrexact3d* for 3D select extraction (see Considerations and Exceptions)

Also Enables

- Calibre xRC

Considerations and Exceptions

- Table 10-2 provides information on the license requirements for a given number of CPUs.

Table 10-2. CPUs Enabled by Calibre xACT Licenses

Number of Licenses	Number of CPUs Enabled
1	4
2	8
3	16
4	24
5	32
6	40

- Calibre xACT does not support the -wait or -nowait command line options.
- When using simultaneous multithreaded processors where Calibre xACT uses all physical CPUs, the virtual cores require no additional licenses.

For example, if you are running on SMT-enabled hardware with eight physical CPUs and eight virtual cores, and you specify -turbo 16, Calibre xACT uses 16 threads and two *calibrexact* licenses. If invoked with -turbo 4, Calibre xACT uses four threads and one *calibrexact* license.

- When running 3D select extraction (-3dselect), Calibre xACT requires both a *calibreexact* license and a *calibreexact3d* license. The *calibreexact3d* license is released as soon as the field solver process completes, while the *calibreexact* license is secured for the entire run.
- For the Calibre xACT inductance flow, a *calibreexact* and *calibrexl* license is required for the entire session when the “I” switch is specified at invocation or when “L” is specified for the MODEL in the PEX XACT CONTROL statement.
- When performing multi-corner extraction with Calibre xACT, licenses are required as shown in [Figure 10-1](#).
- Temperature corners require no additional licenses.
- Double patterning corners require no additional licenses.

Figure 10-1. Calibre xACT License

Number of CPU cores (threads)	Number of simultaneous process corners					
	1	2	3	4	5	Unlimited
1	1 license					
2		2 licenses				
3			3 licenses			
4				3 licenses		
5					3 licenses	
6						3 licenses
7						3 licenses
8						3 licenses
9						3 licenses
10						3 licenses
11						3 licenses
12						3 licenses
13						3 licenses
14						3 licenses
15						3 licenses
16						3 licenses

Related Topics

[Calibre xACT User’s Manual](#)

Calibre xACT 3D

Licensing for Calibre® xACT™ 3D.

Required License

- *calibrexact* for PHDB generation and netlist formatting
- *calibrexact3d* for PDB generation

Substitute License

- *calibrexrc* (when running Calibre xACT for PHDB generation and netlist formatting)

Also Enables

- *fstools* utility (`$_CALIBRE_HOME/bin/fstools`)

Considerations and Exceptions

- [Table 10-3](#) provides information on the licenses requirements for a given number of CPUs.

Table 10-3. CPUs Enabled by License Count

Number of Licenses	Number of CPUs Enabled
1	4
2	8
3	16
4	24
5	32
6	40

Related Topics

[Calibre xACT User's Manual](#)

Calibre xACTView

Licensing for Calibre xACTView.

Required License

- *calibrexact3d* (see Considerations and Exceptions)

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- When invoking Calibre xACTView, the tool checks for the existence of the required or substitute license, but it does not hold the license.

Related Topics

[Calibre xACT User's Manual](#)

Calibre xL

Licensing for Calibre® xL.

Required License

- *calibrexl*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- [Table 10-4](#) shows the license requirements for Calibre xACT inductance extraction, while [Table 10-5](#) shows the license requirements for Calibre xRC inductance extraction when the rule file contains the PEX Inductance Mode PEEC statement.

Table 10-4. CPUs Enabled When Calibre xL Is Launched From Calibre xACT

Number of licenses	Number of CPUs enabled
1	4
2	8
3	16
4	24
5	32
6	40

Table 10-5. CPUs Enabled When Calibre xL Is Launched From Calibre xRC and PEEC Inductance Extraction Mode Is Enabled

Number of xRC licenses	Number of CPUs enabled
1	2
2	4
3	8
4	12
5	16

Related Topics

[Calibre xACT User's Manual](#)

[Calibre xL User's Manual](#)

Calibre xRC

Licensing for Calibre® xRC™.

Required License

- *calibrexrc*

Substitute License

- *calibrexact*
- *calibrexactsoc*

Also Enables

- None.

Considerations and Exceptions

- Table 10-6 provides information on the license requirements for a given number of CPUs.

Table 10-6. CPUs Enabled by Calibre xRC Licenses

Number of Licenses	Number of CPUs Enabled
1	4
2	8
3	16
4	24
5	32
6	40

Related Topics

[Calibre xRC User's Manual](#)

Calibre xRC CB

Licensing for Calibre xRC CB.

Required License

- *calibrexrccb*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- The Calibre xRC CB tool is a separately-licensed parasitic extraction and netlisting product for use on designs containing up to 100,000 transistors.

Related Topics

[Calibre xRC User's Manual](#)

Calibre xRC to ADVance MS

Licensing for Calibre xRC to ADVance MS.

Required License

- *calibrexrcadms*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- [Table 10-7](#) provides information on the license requirements for a given number of CPUs.

Table 10-7. CPUs Enabled by Calibre xRC to ADVance MS Licenses

Number of licenses	Number of CPUs enabled
1	2
2	4
3	8
4	12
5	16

Related Topics

[Calibre xRC User's Manual](#)

Tanner-Calibre One xRC

Licensing for Tanner-Calibre One xRC.

Required License

- *caltannerxrc*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- Tanner-Calibre One xRC license can only be used with the Tanner L-Edit tool.
- You can run one instance of Calibre xRC with the *caltannerxrc* license.
- Calibre xRC jobs run on a maximum of two CPUs.

xCalibrate Rule File Generator

Licensing for xCalibrate™ Rule File Generator.

Required License

- *xcalibrate*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[xCalibrate Batch User's Manual](#)

Chapter 11

Licensing: Resolution Enhancement Technology (RET) Products

The licensing information for Calibre Resolution Enhancement Technology (RET) products provides information on required and substitute licenses, in addition to any considerations and exceptions.

Table 11-1 lists each RET product and the required and substitute license(s) for that product. Click the Calibre product name to view more details about the licensing requirements.

Table 11-1. Primary License Requirements for Resolution Enhancement Technology (RET) Products

Calibre Product	Required License(s)	Substitute License(s)
Calibre Cell Array OPC	<i>calcellarrayopc</i>	None.
Calibre DSA Mask Synthesis	<i>caldsasynth</i>	None.
Calibre DSA Verification	<i>calopcvdsa</i>	None.
Calibre EUV	<i>caleuv</i>	None.
Calibre LITHOview	<i>calibrelv</i>	<i>calworkbench</i>
Calibre LPE	<i>callpe</i>	None.
Calibre nmCLBIAS	<i>calclnmbias</i>	First CPU: <ul style="list-style-type: none">• <i>calopcpro</i>• Two <i>calmtopcpro</i>• <i>calmpcpro</i>• <i>calopcpro</i>• <i>calnmopc</i> Additional CPUs: <ul style="list-style-type: none">• <i>calmtopcpro</i>• <i>calopcpro</i>; enables eight CPUs• <i>calmpcpro</i>• <i>calnmmpc</i>• <i>calnmopc</i>
Calibre nmCLOPC	<i>calclnmopc</i>	None.
Calibre mlOPC	<i>calmlopc</i> , <i>calnmopc</i> , <i>caleuv</i>	None.
Calibre MP Manufacturing	<i>calnmdpc</i>	<i>calmfgmpgold</i>

Table 11-1. Primary License Requirements for Resolution Enhancement Technology (RET) Products (cont.)

Calibre Product	Required License(s)	Substitute License(s)
Calibre MP Manufacturing Advanced	<i>calmfgmpgold</i>	<i>calmfgmpadv</i> and <i>calnmdpc</i>
Calibre nmModelflow	<i>calmdf</i>	None.
Calibre nmOPC	<i>calnmopc</i>	First CPU: <ul style="list-style-type: none">• <i>calopcpro</i> and two <i>calopcvverify</i>; enables four CPUs• <i>calmtopcpro</i> and <i>calopcvverify</i>• <i>calpxopc</i>• <i>calibrelfd</i> Additional CPUs: <ul style="list-style-type: none">• <i>calmtopcpro</i> and <i>calopcvverify</i>• <i>calopcpro</i> and two <i>calopcvverify</i>; enables eight CPUs• <i>calpxopc</i>• <i>calibrelfd</i>
Calibre nmSRAF	<i>calnmsraf</i>	None.
Calibre OPCpro	<ul style="list-style-type: none">• <i>calopcpro</i> for first CPU• <i>calmtopcpro</i> for additional CPUs	First CPU: <ul style="list-style-type: none">• Two <i>calmtopcpro</i>• Two <i>calnmopc</i>• <i>calibrelfd</i> Additional CPUs: <ul style="list-style-type: none">• <i>calopcpro</i>; enables eight CPUs• <i>calnmopc</i>• <i>calibrelfd</i>
Calibre OPCsbar	<i>calopcsbar</i>	<ul style="list-style-type: none">• <i>calnmsraf</i>• <i>calibrelfd</i>
Calibre OPCVerify	<i>calopcvverify</i>	<ul style="list-style-type: none">• <i>calnmopc</i>• <i>calibrelfd</i>
Calibre OPCVerify Classify Plus	<i>calopcvclassifyp</i>	None.

Table 11-1. Primary License Requirements for Resolution Enhancement Technology (RET) Products (cont.)

Calibre Product	Required License(s)	Substitute License(s)
Calibre ORC	<i>calibreorc</i>	First CPU: <ul style="list-style-type: none">• <i>calopcpro</i>• Two <i>calmtopcpro</i>• <i>calopcverify</i> Additional CPUs: <ul style="list-style-type: none">• <i>calmtopcpro</i>• <i>calopcpro</i>, enables eight CPUs• <i>calopcverify</i>
Calibre PRINTimage	<i>calprintimage</i>	<i>calopcverify</i>
Calibre pxOPC	<i>calpxopc</i>	None.
Calibre pxSMO	<i>calsmo</i>	None.
Calibre SONR	<i>calsonr</i>	None.
Calibre TDopc	<i>caltdopc</i>	First CPU: <ul style="list-style-type: none">• <i>calopcpro</i>• Two <i>calmtopcpro</i>• <i>calmpcpro</i>• <i>calopcpro</i>• <i>calnmopc</i> Additional CPUs: <ul style="list-style-type: none">• <i>calmtopcpro</i>• <i>calopcpro</i>; enables eight CPUs• <i>calmpcpro</i>• <i>calnmmpc</i>• <i>calnmopc</i>
Calibre WORKbench	<i>calworkbench</i>	None.

You can include any Calibre RET batch tools in one Calibre nmDRC run. When you do this, you use one set of Calibre nmDRC licenses. For a flat nmDRC run, you use one *calibredrc* license, and for a hierarchical nmDRC run, you use one *calibredrc* and *calibrehdrc* license pair. For multithreaded considerations, refer to “[Licensing for Multithreaded Operations](#)” on page 71.”

For example, the *calibredrc*, *calibrehdrc*, *calibreorc*, and *calprintimage* licenses are required for one run if your rule file contains the following statements:

```
orc_layer = LITHO ORC FILE setup.in poly diff  
p_image_layer = LITHO PRINTIMAGE FILE setup.in poly diff
```

Calibre Cell Array OPC

Licensing for Calibre® Cell Array OPC.

Required License

- First CPU:
 - *calcellarrayopc*
- Additional CPUs:
 - *calcellarrayopc*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre nmOPC User's and Reference Manual](#)

Calibre DSA Mask Synthesis

Licensing for Calibre® DSA Mask Synthesis.

Required License

- *caldsasynth* (see Considerations and Exceptions)

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- You use Calibre nmOPC to access Calibre DSA Mask Synthesis. Refer to the [Calibre nmOPC](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre Directed Self-Assembly User's and Reference Manual](#)

Calibre DSA Verification

Licensing for Calibre® DSA Verification.

Required License

- *calopcvdsa* (see Considerations and Exceptions)

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- You use Calibre OPCverify to access Calibre DSA Verification. Refer to the [Calibre OPCverify](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre Directed Self-Assembly User's and Reference Manual](#)

Calibre EUV

Licensing for Calibre® EUV.

Required License

- *caleuv*
- *calpxopc* (See Considerations and Exceptions)
- *calnmsraf* (See Considerations and Exceptions)

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- To use Calibre EUV capabilities in Calibre nmOPC or Calibre OPCVerify, the respective [Calibre nmOPC](#) or [Calibre OPCVerify](#) license is required.
- A Calibre EUV license is consumed when using an EUV model in Calibre WORKbench, Calibre nmModelflow, or Calibre pxSMO.
- A *calpxopc* license is required only when the EUV option is used with the [RET PXOPC](#) command.
- A *calnmsraf* license is required only when the EUV option is used with the [RET EUV MBSRAF](#) command.

Related Topics

[Calibre pxOPC User's and Reference Manual](#)

[Calibre nmSRAF User's and Reference Manual](#)

Calibre LITHOview

Licensing for Calibre® LITHOview™.

Required License

- *calibrely*

Substitute License

- *calworkbench*

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre WORKbench User's and Reference Manual](#)

Calibre LPE

Licensing for Calibre® LPE.

Required License

- *callpe*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre Local Printability Enhancement User's and Reference Manual](#)

Calibre mlOPC

Licensing for Calibre® mlOPC.

Required License

- *calmopc*
- *calnmopc*
- *caleuv* (See Considerations and Exceptions)

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- The *calmopc* and *calnmopc* licenses are required for execution of the LITHO ML DENSEOPC statement.
- The *calmopc*, *calnmopc*, and *caleuv* licenses are required for execution of the LITHO ML EUV DENSEOPC statement.

Related Topics

[Calibre nmOPC User's and Reference Manual](#)

Calibre MP Manufacturing

Licensing for Calibre® MP Manufacturing.

Required License

- *calnmdpc*

Substitute License

- *calmfgmpgold*

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre Multi-Patterning User's and Reference Manual](#)

Calibre MP Manufacturing Advanced

Licensing for Calibre® MP Manufacturing Advanced.

Required License

- *calmfgmpgold*

Substitute License

- *calmfgmpadv* and *calnmdpc*

Also Enables

- Calibre MP Manufacturing

Considerations and Exceptions

- None.

Keywords and Operations Licensed by Calibre MP Manufacturing Advanced

The following keywords and operations require the *calmfgmpgold* license.

- [DFM MP](#)
 - REDUCED_TARGET_POLYGONS
 - REDUCED_TARGET_SEPARATORS
 - separator_layer [(OPPOSITE | SAME nicety > 0)]
 - stitch_layer (STITCH ...)
- [DFM MPSTITCH](#)
- [RET SIDCUTFILL](#)
- *mp_stitch_gen.tvf*

Related Topics

[Calibre Multi-Patterning User's and Reference Manual](#)

Calibre nmCLBIAS

Licensing for Calibre® nmCLBIAS.

Required License

- *calclnmbias*

Substitute License

- First CPU:
 - *calopcp*
 - *2 calmtopcp*
 - *calmpcp*
 - *calnmmpc*
 - *calnmopc*
- Additional CPUs:
 - *calmtopcp*
 - *calopcp*; enables 8 CPUs
 - *calmpcp*
 - *calnmmpc*
 - *calnmopc*

Also Enables

- Density Convolve

Considerations and Exceptions

- None.

Calibre nmCLOPC

Licensing for Calibre® nmCLOPC.

Required License

- *calclnmopc*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre nmOPC User's and Reference Manual](#)

Calibre nmModelflow

Licensing for Calibre nmModelflow.

Required License

- *calmdf*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- You use Calibre WORKbench to access Calibre nmModelflow. Refer to the [Calibre WORKbench](#) license reference page to determine the licensing requirements for that product.

Calibre nmModelflow uses its own licenses for multithreaded, Calibre MTflex, and job farming operations instead of Calibre WORKbench licenses. The number of Calibre WORKbench licenses used for Calibre nmModelflow is as follows:

- Only one Calibre WORKbench license is required regardless of the number of remotes used by Calibre nmModelflow, even if Calibre WORKbench was invoked with the `-remotefile` option.
- Only one Calibre WORKbench license is required in multithreaded and Calibre MTflex modes.
- In MDF Job Farm mode, one additional Calibre WORKbench license is required per Calibre nmModelflow job that is sent to the dispatcher. For example, dispatching three jobs requires a total of four Calibre WORKbench licenses.
- Invoking the Calibre nmModelflow GUI from Calibre WORKbench uses one *calmdf* license. Additional licenses are checked out according to the following rules.

When launching Calibre nmModelflow from the Calibre WORKbench interface, the tool checks out licenses based on the hardware configuration of the primary host using a formula of $1+((\text{CPU count}+3)/4)$. For example, a 20 CPU machine requires six licenses.

Running calibration jobs in Calibre nmModelflow uses additional licenses beyond the invocation license usage:

- **Calibre MTflex Mode** — When a command uses more than one CPU, Calibre nmModelflow uses the standard license consumption formula as described in the section [“Licensing for Multithreaded Operations”](#) on page 71.

- **MDF Job Farm Mode** — Calibre nmModelflow supports job farming, which is a simultaneous launch of multiple Calibre MTflex jobs. The total number of additional *calmdf* licenses used is the sum of the remotes used by each individual job, calculated independently of each other. For example, if a set of three jobs dispatched together use five remotes, three remotes, and two remotes, the jobs use three, two, and two licenses respectively, for a total of seven licenses, in addition to the ones reserved for invoking the GUI.
- This license is required for Calibre Pattern Generator (CPG) applications. See “[Calibre Pattern Generator Task List](#)” in the *Calibre WORKbench User’s and Reference Manual*.

Related Topics

[Calibre nmModelflow User’s and Reference Manual](#)

Calibre nmOPC

Licensing for Calibre® nmOPC™.

Required License

- *calnmopc*

Substitute License

- First CPU:
 - *calopcpro* and 2 *calopcverify*; enables 4 CPUs
 - *calmtopcpro* and *calopcverify*
 - *calpxopc*
 - *calibrefd*
 - *calcarrayopc*
- Additional CPUs:
 - *calmtopcpro* and *calopcverify*
 - *calopcpro* and 2 *calopcverify*; enables 8 CPUs
 - *calpxopc*
 - *calibrefd*
 - *calcarrayopc*

Also Enables

- TDopc
- OPCverify
- Density Convolve

Considerations and Exceptions

- None.

Related Topics

[Calibre nmOPC User's and Reference Manual](#)

Calibre nmSRAF

Licensing for Calibre® nmSRAF™.

Required License

- *calnmsraf*

Substitute License

- None.

Also Enables

- OPCsbar

Considerations and Exceptions

- None.

Related Topics

[Calibre nmSRAF User's and Reference Manual](#)

Calibre OPCpro

Licensing for Calibre® OPCpro™.

Required License

- *calopcpro* for first CPU
- *calmtopcpro* for additional CPUs

Substitute License

- First CPU:
 - 2 *calmtopcpro*
 - 2 *calnmopc*
 - *calibrelfd*
- Additional CPUs:
 - *calopcpro*; enables 8 CPUs
 - *calnmopc*
 - *calibrelfd*

Also Enables

- TDopc
- Density Convolve
- ORC

Considerations and Exceptions

- None.

Related Topics

[Calibre OPCpro User's and Reference Manual](#)

Calibre OPCsbar

Licensing for Calibre® OPCsbar™.

Required License

- *calopcsbar*

Substitute License

- *calnmsraf*
- *calibrelfd*

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre OPCsbar User's and Reference Manual](#)

Calibre OPCverify

Licensing for Calibre® OPCverify™.

Required License

- *calopcverify*

Substitute License

- *calnmopc*
- *calibrelfd*

Also Enables

- DFM Analyze
- PRINTimage
- ORC

Considerations and Exceptions

- None.

Related Topics

[Calibre OPCverify User's and Reference Manual](#)

Calibre OPCverify Classify Plus

Licensing for Calibre® OPCverify™ Classify Plus.

Required License

- *calopcvcclassifyp* (see Considerations and Exceptions)

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- You use Calibre OPCverify to access Calibre OPCverify Classify Plus. Refer to the [Calibre OPCverify](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre OPCverify User's and Reference Manual](#)

Calibre ORC

Licensing for Calibre® ORC™.

Required License

- *calibreorc*

Substitute License

- First CPU:
 - *calopcpro*
 - *2 calmtopcpro*
 - *calopcverify*
- Additional CPUs:
 - *calmtopcpro*
 - *calopcpro*, enables 8 CPUs
 - *calopcverify*

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre OPCpro User's and Reference Manual](#)

Calibre PRINTimage

Licensing for Calibre® PRINTimage™.

Required License

- *calprintimage*

Substitute License

- *calopcverify*
- *calnmopc*

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre OPCpro User's and Reference Manual](#)

Calibre pxOPC

Licensing for Calibre® pxOPC™.

Required License

- *calpxopc*

Substitute License

- None.

Also Enables

- Calibre nmOPC

Considerations and Exceptions

- None.

Related Topics

[Calibre pxOPC User's and Reference Manual](#)

Calibre pxSMO

Licensing for Calibre® pxSMO™ and RET Selection.

Required License

- *calsmo*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- You use Calibre WORKbench to access Calibre pxSMO. Refer to the [Calibre WORKbench](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre SMO RET Selection User's and Reference Manual](#)

Calibre SONR

Licensing for Calibre® SONR.

Required License

- *calsonr*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- A [Calibre EUV](#) license is required when running Calibre SONR and the EUV keyword is specified with the [SONR Vector Capture](#) command.

Related Topics

[Calibre Sonr User's Manual](#)

Calibre TDopc

Licensing for Calibre® TDopc™.

Required License

- *caltdopc*

Substitute License

- First CPU:
 - *calopcpro*
 - *2 calmtopcpro*
 - *calmpcpro*
 - *calnmmpc*
 - *calnmopc*
- Additional CPUs:
 - *calmtopcpro*
 - *calopcpro*; enables 8 CPUs
 - *calmpcpro*
 - *calnmmpc*
 - *calnmopc*

Also Enables

- Density Convolve

Considerations and Exceptions

- None.

Related Topics

[Calibre Rule-Based OPC User's and Reference Manual](#)

Calibre WORKbench

Licensing for Calibre® WORKbench™.

Required License

- Standard mode:
 - 1 *calworkbench*
- High Capacity (HC) mode:
 - 1 *calworkbench*
 - 1 *caldesignrev*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre WORKbench User's and Reference Manual](#)

Chapter 12

Licensing: Mask Data Preparation (MDP) Products

The licensing information for Calibre Mask Data Preparation (MDP) products provides information on required and substitute licenses, in addition to any considerations and exceptions.

Table 12-1 lists each MDP product and the required and substitute license(s) for that product. Click the Calibre product name to view more details about the licensing requirements.

Table 12-1. Primary License Requirements for Mask Data Preparation (MDP) Products

Calibre Product	Required License(s)	Substitute License(s)
Calibre D2DBDataPrep	<i>cald2dbdataprep</i>	None.
Calibre DefectClassify	<i>calpattnclassify</i> and <i>caldefectsim</i>	None.
Calibre DefectReview	<i>caldefectreview</i>	None.
Calibre FRACTUREc	<i>calfracturec</i>	<i>calfractureall</i>
Calibre FRACTUREh	<i>calfractureh</i>	<i>calfractureall</i>
Calibre FRACTUREi	<i>calfracturei</i>	None.
Calibre FRACTUREj	<i>calfracturej</i>	<i>calfractureall</i>
Calibre FRACTUREm	<i>calfracturem</i>	<i>calfractureall</i>
Calibre FRACTUREn	<i>calfracturen</i>	None.
Calibre FRACTUREp	<i>calfracturep</i>	None.
Calibre FRACTUREt	<i>calfracturet</i>	<i>calfractureall</i>
Calibre FRACTUREv	<i>calfracturev</i>	<i>calfractureall</i>
Calibre Job Deck Editor	<i>caljobdeckedit</i>	None.
Calibre MASKOPT	<i>calmaskopt</i>	<i>calnmmpc</i>
Calibre MDP EMBED	<i>calibrehdrc</i> and <i>calibredrc</i>	None.
Calibre MDP Embedded SVRF	<i>calmdpesvrf</i>	<ul style="list-style-type: none">• <i>calmpcpro</i>• <i>calnmmpc</i>
Calibre MDPAutoClassify	<i>calblankclassify</i>	None.
Calibre MDPDefectAvoidance	<i>caleuvda</i>	None.
Calibre MDPmerge	<i>calmdpmerge</i>	None.

Table 12-1. Primary License Requirements for Mask Data Preparation (MDP) Products (cont.)

Calibre Product	Required License(s)	Substitute License(s)
Calibre MDPstat	Appropriate FRACTURE license (see Considerations and Exceptions)	<ul style="list-style-type: none"> • <i>calfracturem</i> • <i>calfracturej</i> • <i>calfracturet</i> • <i>calfractureh</i> • <i>calfracturev</i> • <i>calfracturec</i> • <i>calfractureall</i>
Calibre MDPverify	Appropriate FRACTURE license (see Considerations and Exceptions)	<ul style="list-style-type: none"> • <i>calfracturem</i> • <i>calfracturej</i> • <i>calfracturet</i> • <i>calfractureh</i> • <i>calfracturev</i> • <i>calfracturec</i> • <i>calfractureall</i>
Calibre MDPview	<i>calmdpview</i>	<ul style="list-style-type: none"> • <i>calibrelv</i> • <i>calworkbench</i>
Calibre Metrology API (MAPI)	<i>calmdpmapi</i>	None.
Calibre MPCpro	<i>calmpcpro</i> (see Considerations and Exceptions)	<i>calnmmpc</i>
Calibre MPCverify	<i>calmpcverify</i>	<i>calnmmpc</i>
Calibre nmCLMPC	<i>calibreclmpc</i>	None.
Calibre nmMPC	<i>calnmmpc</i>	None.

Calibre D2DBDataPrep

Licensing for Calibre® D2DBDataPrep.

Required License

- *cald2dbdataprep*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre DefectClassify

Licensing for Calibre® DefectClassify™.

Required License

- *calpattnclassify* and *caldefectsim*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- You use Calibre DefectReview to access Calibre DefectClassify. Refer to the [Calibre DefectReview](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre DefectClassify User's Manual](#)

Calibre DefectReview

Licensing for Calibre® DefectReview™.

Required License

- *caldefectreview*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre DefectReview User's Manual](#)

Calibre FRACTUREc

Licensing for Calibre® FRACTUREc™.

Required License

- *calfracturec*

Substitute License

- *calfractureall*

Also Enables

- Calibre MDPverify
- Calibre MDPstat

Considerations and Exceptions

- You use Calibre MDP Embedded SVRF when you embed SVRF commands with a FRACTURE statement. Refer to the [Calibre MDP Embedded SVRF](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre FRACTUREh

Licensing for Calibre® FRACTUREh™.

Required License

- *calfractureh*

Substitute License

- *calfractureall*

Also Enables

- Calibre MDPverify
- Calibre MDPstat

Considerations and Exceptions

- You use Calibre MDP Embedded SVRF when you embed SVRF commands with a FRACTURE statement. Refer to the [Calibre MDP Embedded SVRF](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre FRACTUREi

Licensing for Calibre® FRACTUREi™.

Required License

- *calfracturei*

Substitute License

- None.

Also Enables

- Calibre MDPverify
- Calibre MDPstat

Considerations and Exceptions

- You use Calibre MDP Embedded SVRF when you embed SVRF commands with a FRACTURE statement. Refer to the [Calibre MDP Embedded SVRF](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre FRACTUREj

Licensing for Calibre® FRACTUREj™.

Required License

- *calfracturej*

Substitute License

- *calfractureall*

Also Enables

- Calibre MDPverify
- Calibre MDPstat

Considerations and Exceptions

- You use Calibre MDP Embedded SVRF when you embed SVRF commands with a FRACTURE statement. Refer to the [Calibre MDP Embedded SVRF](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre FRACTUREm

Licensing for Calibre® FRACTUREm™.

Required License

- *calfracturem*

Substitute License

- *calfractureall*

Also Enables

- Calibre MDPverify
- Calibre MDPstat

Considerations and Exceptions

- You use Calibre MDP Embedded SVRF when you embed SVRF commands with a FRACTURE statement. Refer to the [Calibre MDP Embedded SVRF](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre FRACTUREn

Licensing for Calibre® FRACTUREn™.

Required License

- *calfracturen*

Substitute License

- None.

Also Enables

- Calibre MDPverify
- Calibre MDPstat

Considerations and Exceptions

- You use Calibre MDP Embedded SVRF when you embed SVRF commands with a FRACTURE statement. Refer to the [Calibre MDP Embedded SVRF](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre FRACTUREp

Licensing for Calibre® FRACTUREp™.

Required License

- *calfracturep*

Substitute License

- None.

Also Enables

- Calibre MDPverify
- Calibre MDPstat

Considerations and Exceptions

- You use Calibre MDP Embedded SVRF when you embed SVRF commands with a FRACTURE statement. Refer to the [Calibre MDP Embedded SVRF](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre FRACTUREt

Licensing for Calibre® FRACTUREt™.

Required License

- *calfracturet*

Substitute License

- *calfractureall*

Also Enables

- Calibre MDPverify
- Calibre MDPstat

Considerations and Exceptions

- You use Calibre MDP Embedded SVRF when you embed SVRF commands with a FRACTURE statement. Refer to the [Calibre MDP Embedded SVRF](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre FRACTUREv

Licensing for Calibre® FRACTUREv™.

Required License

- *calfracturev*

Substitute License

- *calfractureall*

Also Enables

- Calibre MDPverify
- Calibre MDPstat

Considerations and Exceptions

- You use Calibre MDP Embedded SVRF when you embed SVRF commands with a FRACTURE statement. Refer to the [Calibre MDP Embedded SVRF](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre Job Deck Editor

Licensing for Calibre Job Deck Editor.

Required License

- *caljobdeckedit*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- You use Calibre WORKbench or Calibre MDPview to access Calibre Job Deck Editor. Refer to the [Calibre WORKbench](#) or [Calibre MDPview](#) license reference page to determine the licensing requirements for these products.

Related Topics

[Calibre Job Deck Editor User's Manual](#)

Calibre MASKOPT

Licensing for Calibre® MASKOPT™.

Required License

- *calmaskopt*

Substitute License

- *calnmmpc*

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre MDP EMBED

Licensing for Calibre MDP EMBED.

Required License

- *calibrehdrc* and *calibredrc*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- Additional product licenses may be required for the SVRF commands embedded within the MDP EMBED command.
- The SVRF commands that are embedded within an MDP EMBED statement are executed simultaneously and, as a result, must run on the same number of CPUs. The number of CPUs utilized by each MDP EMBED statement is determined by the embedded application licensed to run on the fewest CPUs.

For example, assume you have an MDP EMBED statement that includes SVRF commands that invoke four different applications. The application with the least number of licensed CPUs (MT or Calibre MTflex) determines the number of CPUs utilized for all of the commands embedded within the MDP EMBED command.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre MDP Embedded SVRF

Licensing for Calibre® MDP Embedded SVRF™.

Required License

- *calmdpesvrf*

Substitute License

- *calmpcpro*
- *calnmmpc*

Also Enables

- None.

Considerations and Exceptions

- Additional product licenses may be required for the SVRF commands within the embedded SVRF block of a FRACTURE or MDPVERIFY statement.
- The embedded SVRF commands are executed simultaneously and, as a result, must run on the same number of CPUs. The number of CPUs utilized by each FRACTURE or MDPVERIFY statement is determined by the embedded application licensed to run on the fewest CPUs.

For example, assume you have a FRACTURE or MDPVERIFY statement that includes SVRF commands that invoke four different applications. The application with the least number of MT or Calibre MTflex licensed CPUs determines the number of CPUs utilized for all of the commands embedded within the FRACTURE or MDPVERIFY statement.

- For licensing information related to FRACTURE-free embedded SVRF (MDP EMBED), refer to [Calibre MDP EMBED](#).

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre MDPAutoClassify

Licensing for Calibre® MDPAutoClassify™.

Required License

- *calblankclassify*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- You use Calibre MDPDefectReview to access Calibre MDPAutoClassify. Refer to the [Calibre DefectReview](#) license reference page to determine the licensing requirements for this product.

Related Topics

[Calibre MDPAutoClassify User's Manual](#)

Calibre MDPDefectAvoidance

Licensing for Calibre® MDPDefectAvoidance™.

Required License

- *caleuvda*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- You use Calibre MDPview to access Calibre MDPDefectAvoidance. Refer to the [Calibre MDPview](#) license reference page to determine the licensing requirements for this product.
- Some features in Calibre MDPDefectAvoidance require additional licenses. For example:
 - HTML reporting requires a Calibre RVE license and Calibre DESIGNrev or Calibre MDPview license to generate an HTML report.
 - Using Calibre RVE to view the visualization results database requires a Calibre RVE license.

Related Topics

[Calibre MDPDefectAvoidance User's Manual](#)

Calibre MDPmerge

Licensing for Calibre® MDPmerge™.

Required License

- *calmdpmerge*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre MDPstat

Licensing for Calibre® MDPstat™.

Required License

- Appropriate FRACTURE license (see Considerations and Exceptions)

Substitute License

- *calfracturem*
- *calfracturej*
- *calfracturet*
- *calfractureh*
- *calfracturev*
- *calfracturec*
- *calfractureall*

Also Enables

- None.

Considerations and Exceptions

- The [Calibre MDPstat](#) operation requires a substitute license only if you *omit* FRACTURE operations from your SVRF rule file.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre MDPverify

Licensing for Calibre® MDPverify™.

Required License

- Appropriate FRACTURE license (see Considerations and Exceptions)

Substitute License

- *calfracturem*
- *calfracturej*
- *calfracturet*
- *calfractureh*
- *calfracturev*
- *calfracturec*
- *calfractureall*

Also Enables

- None.

Considerations and Exceptions

- The Calibre **MDPVERIFY** operation requires a substitute license only if you *omit* FRACTURE operations from your SVRF rule file.
- For licensing information when embedding SVRF commands within the MDPVERIFY statement, refer to [Calibre MDP Embedded SVRF](#).

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre MDPview

Licensing for Calibre® MDPview™.

Required License

- *calmdpview*

Substitute License

- *calibrelv*
- *calworkbench*

Also Enables

- None.

Considerations and Exceptions

- Calibre MDPview does not require additional licenses for hardware threads.

Related Topics

[Calibre MDPview User's and Reference Manual](#)

Calibre Metrology API (MAPI)

Licensing for Calibre® Metrology™ API.

Required License

- *calmdpmapi*

Substitute License

- None.

Also Enables

- None.

Considerations and Exceptions

- You use Calibre WORKbench or Calibre MDPview to access Calibre MAPI. Refer to the [Calibre WORKbench](#) or [Calibre MDPview](#) license reference page to determine the licensing requirements for these products.

Related Topics

[Calibre Metrology API \(MAPI\) User's and Reference Manual](#)

Calibre MPCpro

Licensing for Calibre® MPCpro™.

Required License

- *calmpcpro* (see Considerations and Exceptions)

Substitute License

- *calnmmpc*

Also Enables

- Calibre MDP Embedded SVRF
- TDopc

Considerations and Exceptions

- The *calmpcpro* license enables the Density Convolve and MDP Mapsize operations.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre MPCverify

Licensing for Calibre® MPCverify.

Required License

- *calmpcverify*

Substitute License

- *calnmmpc*

Also Enables

- None.

Considerations and Exceptions

- None.

Related Topics

[Calibre Mask Data Preparation User's and Reference Manual](#)

Calibre nmCLMPC

Licensing for Calibre® nmCLMPC.

Required License

- *calibreclmpc*

Substitute License

- None.

Also Enables

- Calibre nmMPC

Considerations and Exceptions

- None.

Related Topics

[Calibre nmMPC and Calibre nmCLMPC User's and Reference Manual](#)

[DFM Data Analysis User's and Reference Manual](#)

Calibre nmMPC

Licensing for Calibre® nmMPC™.

Required License

- *calnmmpc*

Substitute License

- None.

Also Enables

- MPCpro
- MDP EMBED
- MPCverify
- MASKOPT

Considerations and Exceptions

- None.

Related Topics

[Calibre nmMPC and Calibre nmCLMPC User's and Reference Manual](#)

Chapter 13

Distributed Calibre: Overview

The multithreaded, distributed processing capability of Calibre provides a high performance data architecture that utilizes CPUs on and remote hosts for faster processing of full chip and other large designs. With this data architecture, you can take advantage of large SMP servers and low-cost network processors, such as rack-mount systems.

The Calibre tools produce the same output regardless of whether you use the multithreaded capability. The difference is you can achieve faster run times on machines containing multiple CPUs. When using the multithreaded capability, total CPU time is generally greater but real time is faster.

Refer to “[Distributed Calibre: Tool-Specific Support Information](#)” on page 259 for information on distributed Calibre tool support.

Distributed Calibre Recommendations and Concepts	213
Multithreaded Processing Modes	216
Data Processing Modes	234
Configuration Requirements and Guidelines for Distributed Calibre	236
Calibre and Distributed Resource Management Tools	251
Known Issues	257

Distributed Calibre Recommendations and Concepts

There are some recommended steps to help you get started using the Calibre distributed processing capabilities. In addition, understanding some basic concepts about scaling, scalability, and SMT-enabled hardware will help you get the best results in a distributed Calibre run.

1. Familiarize yourself with distributed Calibre terms described in the “[Glossary](#)” on page 421 and with the distributed Calibre concepts described in this section.
2. Review the tool-specific information (“[Distributed Calibre: Tool-Specific Support Information](#)” on page 259) to identify any tool-specific requirements.
3. Configure your hardware and network for running distributed Calibre (“[Configuration Requirements and Guidelines for Distributed Calibre](#)” on page 236).
4. Determine your licensing requirements for running distributed Calibre (“[Licensing for Multithreaded Operations](#)” on page 71).

5. Create a configuration file (“[Distributed Calibre: Creating a Configuration File](#)” on page 271) using statements described in the Reference Dictionary (“[Distributed Calibre: Reference Dictionary](#)” on page 309).

Calibre Scaling and Scalability

The Calibre transcript provides information you can use to calculate the scaling and scalability of a job to determine whether you are getting the best results in a distributed Calibre run.

- Scaling — The number of CPUs ($<N>$) that are used for the Calibre job. Scaling is generally calculated by dividing the transcribed CPU time by the REAL time ($<CPU_time>/<REAL_time>$).
- Scalability — The percentage of scaling on $<N>$ cores. Scalability is always represented as a percentage value (0-100 percent).

A scalable system demonstrates performance improvement proportional to the hardware capacity. Scalability quantifies the benefit of using multiple CPUs on a specific set of hosts and is a desirable property of a Calibre system. While hardware acceleration can improve single-threaded efficiency, it does not directly impact scalability. You can use the following formula to determine scalability on the primary host and remotes:

$<CPU_time> / <REAL_time> / <CPUs>$

- $<CPU_time>$ and $<REAL_time>$ time are reported at the end of the transcript. For example:

```
--- TOTAL CPU TIME = 25020 + 8809081 REAL TIME = 78471
```

In this example, the first time represents the CPU time for the primary host (25020) and the second CPU time represents the remote time (8809081).

- $<CPUs>$ is reported in the Calibre MTflex section of the transcript. For example:

```
MTflex CPU resources: 16 , 120/120 Remote
```

For Calibre jobs run in MT mode, numbers are only available for the primary host.

For Calibre jobs run in Calibre MTflex mode, primary and remote scaling should be computed separately because they are unrelated. Use the appropriate CPU time and CPU count for primary and remote hosts, as needed. A single scaling number is computed for all remote hosts, which is the average across all remotes. Use the upper (numerator) remote CPU count because this is the number of CPUs connected by Calibre MTflex.

Primary host scaling for Calibre MTflex jobs is expected to be low and does not indicate a problem because the primary host acts primarily as a task dispatcher. For this reason, the CPU load on Calibre MTflex primary hosts is ideally less than 75 percent to ensure the primary host is not impeding performance.

The following are examples of calculating primary and remote host scalability:

Primary host scalability = $25020 / 78471 / 16 = 2$ percent

Remote host scalability = $8809081 / 78471 / 120 = 94$ percent

SMT-Enabled Hardware

For non-hyperscaling jobs on SMT-enabled hardware, the number of connected cores is doubled if all physical CPUs are licensed for use. Beginning with the Calibre 2009.3 release, simultaneous multithreading on remote hosts appears in the transcript as twice the number of requested physical CPUs. For example:

```
MTflex CPU resources: 16 , 240/120 Remote
```

The remote numbers are the values for ns/np , where ns is the number of remote connections and np is the number of physical CPUs. Either number may be used to compute scalability depending on the context. Scalability values greater than 100 percent are expected when using the physical CPU count. CPU seconds and scalability measurements on SMT-enabled processors do not compare to processors that are not SMT-enabled and should be used with caution.

Additional information on scaling and scalability is available in the [*Calibre Post-Tapeout Flow User's Manual*](#).

Multithreaded Processing Modes

There are several different multithreaded processing modes available for distributing your design data or Calibre processes across multiple CPUs in order to speed up the processing of a Calibre job.

Table 13-1. Overview of Multithreaded Processing Modes

Distributed Calibre Mode	Description	Recommended Operations Invocation Options
Multithreaded (MT) Processing	The design is executed across multiple CPUs on one host, referred to as the Primary Host . MT executes threads on CPUs that share memory, referred to as primary CPUs.	Calibre nmDRC, Calibre nmLVS, Calibre xRC, Calibre nmOPC, Calibre MDP, Calibre DFM, Calibre LFD: <ul style="list-style-type: none">• -turbo [#CPUs]• -turbo_litho [#CPUs]
Calibre MTflex Processing	The design is distributed across multiple CPUs on multiple networked hosts. Calibre MTflex executes threads on CPUs that do not share memory, referred to as remote CPUs. A communication path, such as a LAN or switch, is required between primary and remote hosts.	Calibre nmDRC, Calibre nmLVS, Calibre nmOPC, Calibre MDP, Calibre DFM, Calibre LFD: <ul style="list-style-type: none">• -turbo [#CPUs] -remote• -turbo [#CPUs] -remotefile• -turbo_litho [#CPUs] -remote• -turbo_litho [#CPUs] -remotefile
Multithreaded (MT) Processing with Hyperscaling	Both the design and SVRF operations are executed on multiple CPUs on the primary host.	Calibre nmDRC, Calibre nmLVS, Calibre MDP, Calibre nmOPC ¹ : <ul style="list-style-type: none">• -turbo [#CPUs] -hyper• -turbo_litho [#CPUs] -hyper
Calibre MTflex Processing with Hyperscaling	Both the design and SVRF operations are distributed across multiple CPUs on multiple networked hosts. Pseudo hierarchical databases are created only on the primary host.	Calibre nmDRC, Calibre nmLVS, Calibre MDP, Calibre nmOPC ¹ : <ul style="list-style-type: none">• -turbo [#CPUs] -remote -hyper• -turbo [#CPUs] -remotefile -hyper• -turbo_litho [#CPUs] -remote -hyper• -turbo_litho [#CPUs] -remotefile -hyper

Table 13-1. Overview of Multithreaded Processing Modes (cont.)

Distributed Calibre Mode	Description	Recommended Operations Invocation Options
Calibre MTflex Processing, Hyperscaling, and Remote Data Server	Both the design and SVRF operations are distributed across multiple CPUs on multiple networked hosts. Data is distributed across remote hosts thereby reducing primary host memory requirements. Data previously stored on the primary host is distributed to “remote data servers” on the remote host.	Calibre nmDRC: <ul style="list-style-type: none"> • -turbo [#CPUs] -remote -hyper -remotedata • -turbo [#CPUs] -remotefile -hyper -remotedata • -turbo_litho [#CPUs] -remote -hyper -remotedata • -turbo_litho [#CPUs] -remotefile -hyper -remotedata
Calibre MTflex Processing, Hyperscaling, Remote Data Server, and Hyper Remote	Distributes the design and SVRF operations across multiple CPUs on multiple networked hosts. The Hyper Remote capability enables the distribution of pseudo hierarchical databases across remote hosts and must be used in conjunction with the Remote Data Server capability.	Calibre nmDRC: <ul style="list-style-type: none"> • -turbo [#CPUs] -remote -hyper remote -remotedata [-recoveroff -recoverremote] • -turbo [#CPUs] -remotefile -hyper remote -remotedata [-recoveroff -recoverremote] • -turbo_litho [#CPUs] -remote -hyper remote -remotedata [-recoveroff -recoverremote] • -turbo_litho [#CPUs] -remotefile -hyper remote -remotedata [-recoveroff -recoverremote]
Calibre FullScale	Calibre FullScale is a multi-processing platform designed to improve scalability and runtime for typical post-tapeout runs.	Refer to the Calibre Post-Tapeout Flow User's Manual for information on operations and invocation options.

1. Hyperscaling has no direct benefit for OPC operations and increases primary and remote host memory requirements. Only OPC jobs with significant pre- or post-OPC SVRF operations should be considered as candidates for hyperscaling and then only if sufficient free memory is available.

Note

All SVRF operations run with MT, Calibre MTflex, and hyperscaling. You do not need to make any changes to your rule file in order to run MT, Calibre MTflex, and hyperscaling. When you invoke a Calibre job with hyperscaling, the operations that can leverage MT or Calibre MTflex with hyperscaling functionality do so automatically. There are SVRF operations that, by their nature, cannot be distributed to remote CPUs and run only on the primary host in Calibre MTflex mode.

The Calibre tools with *no* support for MT and Calibre MTflex functionality include Calibre nmDRC flat, Calibre nmLVS flat, and Calibre xL. Also, only the circuit extraction phase of Calibre nmLVS-H supports multithreading, the comparison phase does not.

Multithreaded (MT) Processing

MT refers to multithreaded CPU technology that runs more than one thread of execution within an operating system. This technology speeds up the processing of a Calibre job by leveraging the multiple CPUs available on one host.

For MT mode, you use the following command line options in addition to other Calibre invocation options:

- `-turbo [number_of_cpus]`

This option instructs Calibre to use multithreaded parallel processing for all stages except RET and MDP operations. The optional *number_of_cpus* argument is a positive integer that specifies the number of CPUs to use for processing. By default, Calibre runs on the maximum number of CPUs available when *number_of_cpus* is not specified or if you specify a value that is greater than the maximum available. For best performance, it is recommended that you avoid specifying a value for *number_of_cpus*.

- `-turbo_litho [number_of_cpus]`

This option instructs Calibre to use multithreaded parallel processing when performing RET and MDP operations. The optional *number_of_cpus* argument is a positive integer that specifies the number of CPUs to use for RET and MDP processes. If you do not specify *number_of_cpus*, Calibre runs on the maximum number of CPUs available for which you have licenses, regardless of the -turbo setting.

You can specify the -turbo and -turbo_litho options concurrently in a single command line and the respective *number_of_cpus* can vary between the two options.

The following are the requirements for running MT:

- Hierarchical execution (-hier)
- Adequate licenses for the tools and number of threads you intend to run

- Host with multiple CPUs

Refer to [Table 13-8](#) on page 236 for information on processing requirements and recommendations.

Calibre MTflex Processing

Calibre MTflex functionality provides a parallel processing data architecture for running multithreaded Calibre tools on distributed networked computers. When you run a tool with Calibre MTflex functionality, Calibre creates threads and processes as needed across the network to remote computers.

Calibre MTflex executes threads on the following hosts:

- [Primary Host](#) using the CPUs that share memory.
- [Remote Host](#) using the remote CPUs that do not share memory.

Additionally, Calibre MTflex provides control and data transfer mechanisms to support remote execution.

For Calibre MTflex mode, you specify the [-remote](#) or [-remotefile](#) argument in addition to any of the MT arguments ([-turbo](#) and [-turbo_litho](#)).

A Calibre MTflex server is set up using the host list (specified by [-remote](#)) or reading the contents of the configuration file (specified by [-remotefile](#)). Licensing requirements are determined once the remote hosts are connected and the number of (and remote) CPUs are known.

The requirements for running Calibre MTflex include the following:

- Hierarchical execution (-hier).
- Licenses for running the Calibre tools. Refer to “[Licensing for Multithreaded Operations](#)” for more information.
- Communication path between the and remote CPUs, typically a LAN or switch.
- Configuration file. Refer to “[Distributed Calibre: Creating a Configuration File](#)” for more information on the configuration file required for running Calibre MTflex.
- Depending on the tools that are used during the Calibre run, there may be tool-specific considerations that need to be considered. Refer to “[Tool-Specific Information for Distributed Calibre](#)” for more details.
- All platforms must run the same version of Calibre tools.

Note

 A Calibre job running on remotes in a mixed operating system (non-homogeneous) environment will generate warning messages identifying the operating systems that are in use. In addition, there may be differences in output (refer to “[Output Differences](#)” on page 257 for more information).

Refer to [Table 13-8](#) on page 236 for information on minimum hardware guidelines for and remote hosts. For information on Calibre MTflex and the SHARED memory statistic, refer to “[Shared Memory Guidelines for Distributed Calibre](#)” on page 250.

A log file named *CalibreRRLog.<logtag>[.<hdbid>]* is automatically created and saved if Calibre MTflex generates any warnings or errors during a run. You can set the **CALIBRE_MTFLEX_SAVE_LOGS** environment variable to a non-null value to save this log file along with the Remote Compute Server (RCS) log files at the end of the Calibre run.

Hyperscaling

Multithreading (MT) and Calibre MTflex provide a high performance data architecture for faster processing of full chip and other large designs, while hyperscaling provides an architecture for distributing SVRF operations. Hyperscaling can be used with both MT and Calibre MTflex runs, providing designers significant performance improvements while taking simultaneous advantage of large SMP (Shared-memory Processor) servers and low-cost network processors, such as rack-mount systems.

While multithreading distributes computations across multiple CPUs, hyperscaling enables the concurrent parallel execution of SVRF operations and is only valid in hierarchical tools that support hyperscaling. Some SVRF operations, such as those that perform geometric layer operations, can be hyperscaled, while others cannot. Both hyperscaled and non-hyperscaled SVRF operations can reside in the same rule file. During the run, the SVRF operations that do not support hyperscaling will not run concurrently. Pseudo hierarchical databases are created only on the primary host, unless -hyper remote is specified. Hyperscaling can perform rule checks in an order different than what appears in the rule file. Non-hyperscaling runs follow the rule file order.

Hyperscaling counts hierarchical objects differently from objects in a traditional Calibre run. Specifically, hyperscaling selectively flattens internally-recognized, very small cells, such as vias and contacts. Hyperscaling also selectively flattens internally-recognized top-level cells that are identified based upon your Layout Base Layer statement (or Layout Top Layer statement, if applicable). This means object counts can differ for the same job run in hyperscaling mode versus a traditional Calibre run.

Note

 To reduce memory requirements, any operations in which the inputs are exclusively flat are restricted to HDB 0. As a result, excessive flattening can significantly impact scaling.

Only use hyperscaling, enabled with the -hyper command line option, in an MT or Calibre MTflex environment. Do not use hyperscaling for single CPU runs. This option can only be specified with the -turbo, -turbo_litho, or both options, which are supported in MT and Calibre MTflex modes.

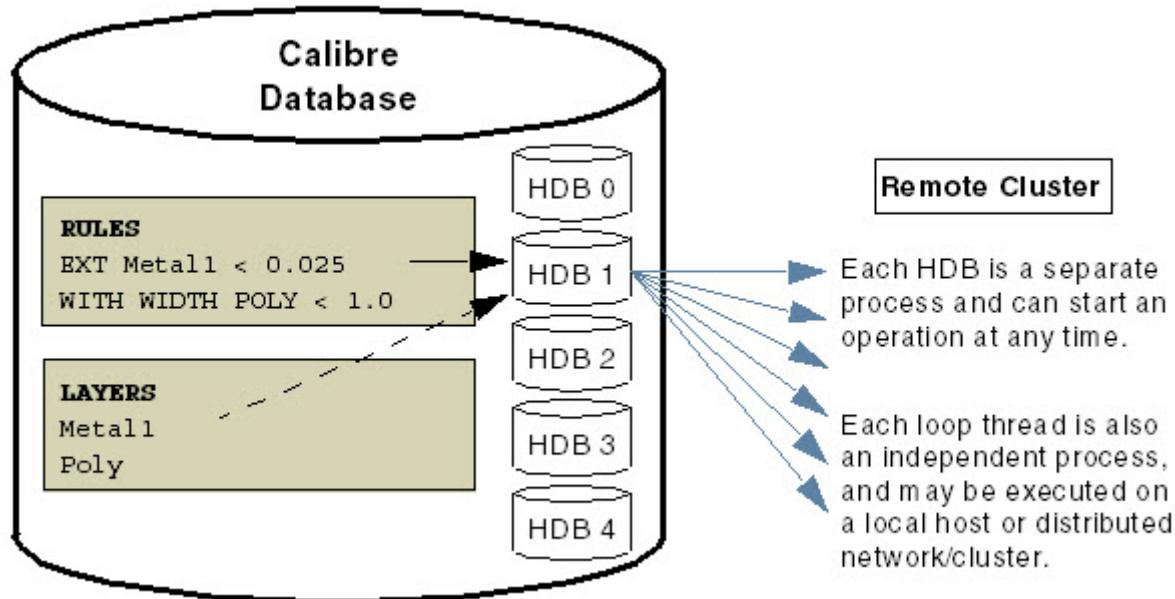
Note

The -hyper switch should not be specified for a single-threaded run. If you specify -hyper and do not specify -turbo, Calibre exits with an error message. If you specify -hyper and -turbo 1, Calibre issues a warning and continues:

```
WARNING: HYPERSCALING requested but only 1 cpu available, continuing without it.
```

[Figure 13-1](#) illustrates how hyperscaling distributes both the design and operations across multiple CPUs through the use of pseudo hierarchical databases, or pseudo HDBs. For more information on using hyperscaling, refer to “[Running MT and Calibre MTflex with Hyperscaling](#).” Refer to [Table 13-8](#) on page 236 for information on processing requirements and recommendations. For information on hyperscaling and the SHARED memory statistic, refer to the section on “[Shared Memory Guidelines for Distributed Calibre](#).”

Figure 13-1. Distribution of a Design and SVRF Operations in Hyperscaling Mode



Hyperscaling creates four pseudo HDBs as shown in [Figure 13-1](#). The number four has been found to provide optimal performance across a wide variety of inputs.

[Table 13-2](#) identifies the number of HDBs created when running hyperscaling in multithreaded mode and a value is not specified for -turbo. The same numbers apply when you specify a value for -turbo that is greater than the number of CPU cores. When specifying both -turbo and

-remote, only four pseudo HDBs are created on the primary host except in the case when -hyper remote is specified.

Table 13-2. HDBs Created for Hyperscaling Runs in Multithreaded Mode

CPU Cores	HDBs Created
< 32	Defaults to -hyper 5 (primary + 4 pseudo HDBs)
= 32 < 60	Defaults to -hyper 7 (primary + 6 pseudo HDBs)
= 60	Defaults to -hyper 9 (primary + 8 pseudo HDBs)

On small Calibre MTflex clusters, Calibre Remote Compute Server (RCS) processes are connected to each pseudo HDB: four additional RCS processes on each remote CPU. On larger Calibre MTflex clusters, the total number of RCS processes is limited as shown in [Table 13-3](#).

Table 13-3. Remote Compute Servers (RCS) per CPU for Calibre MTflex Clusters Using Hyperscaling

CPU Cores	Number of Additional RCS Processes
< 200	Each CPU is attached to all pseudo HDBs (4).
= 200 < 400	Each CPU is attached to half of the pseudo HDBs (2).
= 400	Each CPU is attached to one quarter of the pseudo HDBs (1).

Simultaneous Multithreading With Virtual CPUs

Simultaneous multithreading (SMT) improves CPU efficiency by treating each physical CPU as two logical CPUs (one physical and one virtual). The processor causes the operating system to behave as if there are twice as many CPUs, so the operating system schedules twice as many processes to run.

While you can use simultaneous multithreading and hyperscaling (-hyper) to improve runtime performance, they are different in that simultaneous multithreading is a hardware feature of specific processors while hyperscaling is a software feature internal to Calibre and available on all processor types.

The run-time benefit of SMT is less than 2X, despite the 2X increase in the number of CPUs. [Table 13-4](#) provides some general guidelines of the run-time benefit for the different Calibre job classes (there is a noticeable variation in benefits from job to job).

Table 13-4. Calibre Runtime Benefit From Simultaneous Multithreading

Calibre Job Class	Run-time Benefit
DRC	15 percent
MDP	20 percent
OPC	25 percent

Caution

 Enabling SMT may double the memory requirement to run Calibre. Do not use SMT with Calibre unless sufficient physical memory is available. This constraint is primarily an issue for Calibre LITHO OPC, where the memory requirement scales linearly with CPU count.

The following conditions must be met in order to leverage SMT capabilities on your hardware when running Calibre jobs:

- SMT must be enabled in the BIOS.

You can view the information in the */proc/cpuinfo* file to determine whether SMT is enabled. By default, Calibre uses SMT when it is enabled in the BIOS.

- You must use all physical CPUs on the machine to run the Calibre job.

You must specify *-turbo* with no value so that it defaults to running on all physical CPUs, or specify *-turbo #*, where *#* is the total number of physical CPUs on the primary host. Specifying a *-turbo* value that is less than the number of physical CPUs effectively ignores SMT, as the criteria to license all cores has not been met. Use of *-turbo_all* with a *-turbo* value greater than the number of physical CPUs will result in an error as this condition cannot be met. The same constraint applies to Calibre MTflex remotes. You must ensure that all physical CPUs on a remote host are licensed in order to leverage SMT capabilities on that host. Refer to the [REMOTE HOST](#) reference page for more information on this statement.

- Optionally, there are environment variables and configuration file statements you can set to fully leverage SMT capabilities. Refer to “[SMT Environment Variable and Configuration File Settings](#)” on page 224 for more information.

To maximize the benefit of SMT, IBM® Spectrum LSF (formerly IBM Platform LSF), and Univa Grid Engine, you should take the following steps:

- Configure available execution slots based on the number of physical CPUs, not total logical cores. This configuration is best for all applications because net throughput is much less than 2X for all logical cores, so assigning jobs to all logical cores will overload servers (hosts) and reduce net throughput for all applications. See “[Calibre and Distributed Resource Management Tools](#)” for details.
- To increase the odds that all physical CPUs are licensed by a single Calibre job, configure the allocation rules to fill all available slots on each server before looking for additional slots on other servers. For example, use the fewest number of servers (hosts) instead of distributing across the maximum number of servers. With Univa Grid Engine this is accomplished via the *\$fill_up* allocation rule. Use of such allocation rules by most jobs usually maximizes the number of hosts benefiting from SMT.

The Calibre transcript provides additional information for jobs that run with SMT-enabled hardware. In the following example, the transcript header indicates the number of physical CPUs and that SMT is enabled. The remote host section indicates there are five remote hosts,

each with eight physical CPUs (np), eight virtual cores (nv), and 16 total logical cores (ns). The run summary indicates all physical and virtual cores were used for the job.

- Transcript header information

```
// CPU Info: Cores = 40, SMT enabled with 40 additional virtual
processors
...
```

- Remote host information

```
// Running on 40 CPUs (pending licensing)
// List of connected remote hosts:
// REMOTE HOST node1: np = 8, nv = 8, ns = 16
// REMOTE HOST node2: np = 8, nv = 8, ns = 16
// REMOTE HOST node3: np = 8, nv = 8, ns = 16
// REMOTE HOST node4: np = 8, nv = 8, ns = 16
// REMOTE HOST node5: np = 8, nv = 8, ns = 16
// MTflex CPU resources: 64 , 80/40 Remote
//
// LITHO operations available for use on 40 CPUs (pending licensing)
```

- Run summary information

```
--- PROCESSOR COUNT = 40 + 40 (virtual)
```

SMT Environment Variable and Configuration File Settings

When SMT is enabled, you have the option of using an environment variable or configuration file statement to fully leverage SMT capabilities. The defaults for these configuration options are such that, in most cases, running a Calibre job on an SMT-enabled system can automatically leverage the SMT capabilities.

Table 13-5 summarizes the different settings you can use to enable or disable SMT when running a Calibre job with or without hyperscaling. Refer to the sections “[With Hyperscaling](#)” and “[Without Hyperscaling](#)” for more information.

Table 13-5. SMT Environment Variable and Configuration File Setting Summary

Mode	Operations	Default	To Enable or Disable
With Hyperscaling			
MT	DRC	ON	Always ON.
	LITHO	ON	Set the CALIBRE_HYPER_SMTFACTOR environment variable.
MTflex	DRC	ON	Always ON.
	LITHO	OFF	Add the HYPER SMTFACTOR statement to the configuration file.
Without Hyperscaling			

Table 13-5. SMT Environment Variable and Configuration File Setting Summary (cont.)

Mode	Operations	Default	To Enable or Disable
MT	DRC or LITHO	ON	Set the CALIBRE_MT_SMT_FACTOR environment variable.
MTflex	DRC or LITHO	ON	Set the CALIBRE_MT_SMT_FACTOR environment variable. Or, add the SMTFACTOR keyword to the LAUNCH AUTOMATIC , LAUNCH CLUSTER , or LAUNCH MANUAL statement in the configuration file. ¹

1. The [CALIBRE_MT_SMT_FACTOR](#) environment variable setting overrides the [LAUNCH AUTOMATIC|CLUSTER|MANUAL](#) SMTFACTOR setting.

With Hyperscaling

When hyperscaling is used for a Calibre run, you can enable the appropriate environment variable or configuration file statement so that SMT can be utilized for HDB 0, which is beneficial in post tape-out flows where much of the job time is spent solely on HDB 0. By default, SMT is automatically utilized for pseudo HDBs 1-4 when hyperscaling is used. It is not necessary to specify the following environment variable or configuration file statement to utilize SMT for pseudo HDBs 1-4.

- **For Calibre runs in MT mode with hyperscaling** — Use the [CALIBRE_HYPER_SMTFACTOR](#) environment variable to launch two Calibre processes per physical CPU for operations running on HDB 0 (as is done for jobs without hyperscaling). The default is 1 (enabled). Set the environment variable to 0 to limit operations running on HDB 0 to one process per physical CPU, thereby preventing the utilization of SMT.
- **For Calibre runs in MTflex mode with hyperscaling** — Use the SMTFACTOR keyword for the [HYPER](#) statement to launch two Calibre processes per physical CPU for operations running on HDB 0 (as is done for jobs without hyperscaling). The default is 0 (disabled), which limits operations running on HDB 0 to one process per physical CPU, thereby preventing the utilization of SMT.

Without Hyperscaling

When hyperscaling is not used for a Calibre run, you can set the appropriate environment variable or configuration file statement to enable or disable SMT processing.

- **For Calibre runs in MT mode without hyperscaling** — Use the [CALIBRE_MT_SMT_FACTOR](#) environment variable to enable SMT processing. The default is 1 (enabled).

Note

 If set to 0, this variable overrides any other SMT-related settings (such as the SMTFACTOR keyword for the LAUNCH statements).

- **For Calibre runs in MTflex mode without hyperscaling** — Use the SMTFACTOR keyword in the [LAUNCH AUTOMATIC](#), [LAUNCH CLUSTER](#), or [LAUNCH MANUAL](#) to enable SMT processing. The default is 1 (enabled).

Remote Data Server

Remote Data Server (RDS) technology is designed to reduce primary host memory requirements and improve Calibre MTflex performance by distributing data across remote hosts. It is supported for 64-bit Calibre and remote host processes. Because of the distributed nature of the data, every remote host used in a Calibre MTflex run must be able to communicate with every other remote host in the run.

RDS technology is intended for large designs requiring multiple hours to complete. A distributed computing environment is required and a minimum of 24 remote CPUs is preferred. Remote counts of 40-60 CPUs are typically the optimal operating range. While hyperscaling is the suggested Calibre execution mode with `-remotedata`, it is not required.

Requirements for running RDS:

- Distributed Calibre
- 64-bit primary host
- 64-bit remote hosts
- Multi-CPU remote hosts

Recommendations for running RDS:

- Hyperscaling
- Minimum 24 remote CPUs
- 2 GB memory per remote CPU

Refer to the section “[Shared Memory Guidelines for Distributed Calibre](#)” for information on RDS and the SHARED memory statistic.

RDS technology is triggered by appending the `-remotedata` argument to the [-remote](#) or [-remotefile](#) statements. There are two recovery modes available with the `-remotedata` argument:

- `-recoveroff`

If a remote fails then the Calibre job fails. This mode is recommended for best performance and is the default. For networks of uncertain reliability, use -recoverremote.

- -recoverremote

Creates a copy of the data on the remotes. If you use -remotedata, this argument is recommended for runs having many remote hosts.

If requested, one backup RDS is assigned for each RDS on the remote host to allow for recovery in case of a failure on a remote host. If -recoveroff is specified, either explicitly or by default, then no backup RDS is assigned.

Total distributed memory space available for primary host offload is summarized in lines like the following:

```
// MTflex RDS resources: <RDS Size>GB, with recovery [on|off]
```

For example, specifying the -recoveroff option reports 20GB of usable remote memory with 20 RDS resources:

```
// MTflex RDS resources: 20GB, with recovery off
```

Usable remote memory does not include the backup for -recoverremote, which means “usable” is half the total “consumed” RDS space on each remote host. You can increase the total amount of usable remote memory by using more total remote CPUs or by increasing the amount of usable memory per CPU.

RDS memory consumption includes the backup for -recoverremote, so the values are twice the usable space. Also, references to RDS memory and REMOTE memory in the Calibre transcript will include the backup for -recoverremote when appropriate, and may also represent double the usable space.

```
CPU TIME = 3 + 345  REAL TIME = 13  LVHEAP = 16809/66820/67098  SHARED = 0/  
0  REMOTE = 6197/243136  OPS COMPLETE = 22851 OF 22857  ELAPSED TIME =  
18866
```

Remote Host Summary:

```
REMOTE LVHEAP node6070: RCS = 29377 RDS = 8446 PHDB = 20476 TOTAL = 58299
```

You can use the SHARED memory reporting to manage RDS usable memory space. SHARED has different meanings when running with and without remotedata. Without -remotedata, SHARED is the portion of LVHEAP that can be off-loaded from the primary host to the remotes (the remainder of LVHEAP cannot be off-loaded). With -remotedata, a non-zero SHARED value indicates that all of the available RDS space has been used and space has overflowed back onto LVHEAP on the primary host. Non-zero SHARED memory with -remotedata is associated with poor run-time performance and indicates the job should be reconfigured to increase the amount of usage RDS space.

Both SHARED and REMOTE report memory usage as current/peak. On runs using -remotedata with non-zero SHARED (all RDS space is used), the peak REMOTE memory will be equal to the total usable memory or double that amount using -recoverremote.

Refer to [Table 13-8](#) on page 236 for information on processing requirements and recommendations.

Hyper Remote

The hyper remote functionality enables Pseudo Hierarchical Databases (HDBs) to be distributed across remote hosts, thereby reducing memory requirements on the primary host and improving Calibre MTflex performance.

The hyper remote functionality is supported only when using -hyper in conjunction with Remote Data Server capability ([-remotedata](#)). Hyper remote automatically configures the number of Pseudo HDBs and Remote Data Servers (RDS) created on a physical remote host based on the number of remote CPUs on the host assigned to Calibre. The equation for determining the number of HDB and RDS processes created depends on the number of CPUs in the remote cluster. The RDS multiple is always one per CPU, with one RDS and one backup RDS created for every remote CPU. [Table 13-6](#) lists the multiples that are used, based on the number of CPUs, for determining the number of Pseudo HDBs.

Table 13-6. RDS and HDB Multipliers for Hyper Remote

Number of CPUs	-remotedata -hyper remote	
	RDS Multiple	Pseudo HDB Multiple
Less than 24 (small)	1	2
24 to 63 (medium)	1	4
64 and greater (large)	1	8

For example, in a medium size remote cluster (24 to 63 CPUs), Calibre assigns one Pseudo HDB for every four CPUs on a remote and eight RDS for every four CPUs on a remote (one RDS plus one backup RDS for each CPU on the remote host).

If the number of remote CPUs assigned to Calibre is less than the multiple, one Pseudo HDB is assigned to Calibre. When the number of remote CPUs is not evenly divisible by the multiple, an additional Pseudo HDB is not added. For example, in a configuration of 26 remote CPUs, Calibre assigns six Pseudo HDBs to the run.

If RDS -recoveroff is specified, the memory requirements are reduced by 1 GB for every RDS on each remote CPU.

The following command is used to execute the operation shown in [Figure 13-2](#) and [Figure 13-3](#):

```
$ calibre -drc -hier -turbo -hyper remote -remotedata
    -remotefile remote.config my_rules
```

Figure 13-2. Hyper Remote Configuration of Pseudo HDBs and Remote Data Servers on a Medium Size (24-63 CPUs) Remote Cluster

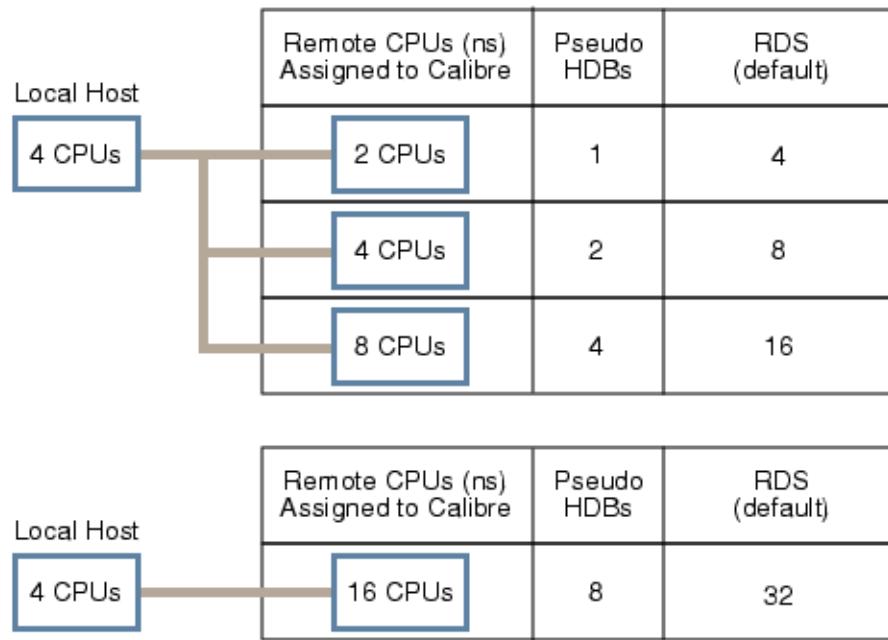
Local Host	Remote CPUs (ns*) Assigned to Calibre	Pseudo HDBs	RDS (default)
4 CPUs	2 CPUs	1	4
	4 CPUs	1	8
	8 CPUs	2	16
	16 CPUs	4	32

* ns refers to the number of Calibre servers assigned to the remote

Following is the transcript that is generated for the configuration shown in [Figure 13-2](#):

```
// MTflex initialization for pseudo HDBs completed.
// Running on 30 CPUs (pending licensing)
// List of connected remote hosts:
//  REMOTE HOST xyz101: np = 2, ns = 2
//  REMOTE HOST xyz102: np = 4, ns = 4
//  REMOTE HOST xyz201: np = 8, ns = 8
//  REMOTE HOST xyz301: np = 16, ns = 16
// MTflex CPU resources: 4 , 30/30 Remote
```

Figure 13-3. Hyper Remote Configuration of Pseudo HDBs and Remote Data Servers on a Small (<24) Remote Cluster



Following is the transcript that is generated for the configuration shown in [Figure 13-3](#):

```
// MTflex initialization for pseudo HDBs completed.  
// Running on 14 CPUs (pending licensing)  
// List of connected remote hosts:  
// REMOTE HOST xyz101: np = 2, ns = 2  
// REMOTE HOST xyz102: np = 4, ns = 4  
// REMOTE HOST xyz201: np = 8, ns = 8  
// MTflex CPU resources: 4 , 14/14 Remote  
  
// MTflex initialization for pseudo HDBs completed.  
// Running on 14 CPUs (pending licensing)  
// List of connected remote hosts:  
// REMOTE HOST xyz301: np = 16, ns = 16  
// MTflex CPU resources: 4 , 16/16 Remote
```

Hyper remote is compatible with IBM Spectrum LSF. If you run IBM Spectrum LSF and are not using the full set of remotes, hyper remote configures the remotes based on the remote CPUs allocated to Calibre. For more information, refer to the section on “[Calibre and Distributed Resource Management Tools](#).”

Figure 13-4. Comparison Running Hyper Remote on a Small and Medium Remote Cluster

Remote CPUs Assigned to Calibre	Pseudo HDBs	
	Small Remote Cluster (< 24)	Medium Remote Cluster (24-63)
Using 2 CPUs on a 4 CPU remote host	1	1
Using 3 CPUs on an 8 CPU remote host	1	1
Using 6 CPUs on an 8 CPU remote host	3	1
Medium Remote Cluster only: Using 16 CPUs on a 16 CPU remote host		4

Following is the transcript that is generated for the medium remote cluster in [Figure 13-4](#):

```
// MTflex initialization for pseudo HDBs completed.
// Running on 27 CPUs (pending licensing)
// List of connected remote hosts:
// REMOTE HOST xyz101: np = 4, ns = 2
// REMOTE HOST xyz201: np = 8, ns = 3
// REMOTE HOST xyz202: np = 8, ns = 6
// REMOTE HOST xyz301: np = 16, ns = 16
// MTflex CPU resources: 4 , 27/36 Remote
```

The requirements for running hyper remote include the following:

- 64-bit primary host
- Distributed Run
 - 64-bit remote hosts.
 - Multi-CPU remote hosts.
- Hyperscaling (-hyper).
 - Minimum four Pseudo HDBs.
- Every remote host used in the run must be able to communicate with every other remote host used in the run.
- Remotedata (-remotedata [-recoveroff | -recoverremote]).

The recommendations for running hyper remote include the following:

- 4 GB memory per remote CPU.
- No scatter bar rule files.
- Hyper remote should be used when running large designs on large remote configurations.

- Hyper remote should be used when it is important to reduce memory usage on the primary host. For example, you may want to reduce memory usage on the primary host with small designs and small remote configurations.

A remote host is automatically disqualified from a hyper remote run if the host is 32-bit or has only one CPU. In addition, if the number of Pseudo HDB server candidates is less than four, Calibre automatically reverts to running the job in hyperscaling mode. The transcript contains information about any remote hosts that were disqualified from the run. For more information, refer to the description of the [-hyper](#) remote keyword.

Refer to [Table 13-8](#) on page 236 for additional information on processing requirements and recommendations.

Hyper Remote Transcripts

The hyper remote transcript provides information about the remote hosts that are disqualified from a run, as well as the remote hosts that are used in the run. There are some common transcript messages you might encounter when running hyper remote.

In [Figure 13-5](#), the transcript indicates there were not enough remote hosts available to run as pseudo HDB server candidates. The job reverts to running in hyperscaling mode on the primary host only.

Figure 13-5. Executing Hyper Remote With Less Than Four Pseudo HDBs

```
// Determining Remote Pseudo HDBs.  
// Insufficient remote hosts found for remote Pseudo HDBs, reverting to  
// non-remote hyper mode  
  
// HYPERSCALING ENABLED with 4 pseudo HDBs.
```

In [Figure 13-6](#), the transcript indicates there are five remote hosts available for running Pseudo HDBs. Calibre proceeds to connect to the five pseudo HDBs and run the job in hyperscaling mode on the five remote servers.

Figure 13-6. Hyper Remote Transcript Running With Sufficient Resources

```
// Determining Remote Pseudo HDBs.  
// REMOTE HOST xyz1 used for Remote Pseudo HDB  
// REMOTE HOST xyz2 used for Remote Pseudo HDB  
// REMOTE HOST xyz3 used for Remote Pseudo HDB  
// REMOTE HOST xyz4 used for Remote Pseudo HDB  
// REMOTE HOST xyz5 used for Remote Pseudo HDB  
// Connected to Remote Pseudo HDB 1 on remote host xyz1  
// Connected to Remote Pseudo HDB 2 on remote host xyz2  
// Connected to Remote Pseudo HDB 3 on remote host xyz3  
// Connected to Remote Pseudo HDB 4 on remote host xyz4  
// Connected to Remote Pseudo HDB 5 on remote host xyz5  
// HYPERSCALING ENABLED with 5 pseudo HDBs.
```

In [Figure 13-7](#), the transcript indicates that the “xyz3” remote host cannot be used as a remote pseudo HDB because “xyz3” is a 32-bit remote host. Calibre connects to the remaining four pseudo HDBs and runs the job in hyperscaling mode on the four remote servers.

Figure 13-7. Hyper Remote Transcript Running With a 32-Bit Remote Host

```
// Determining Remote Pseudo HDBs.  
// REMOTE HOST xyz3 not used for Remote Pseudo HDB, must use 64-bit  
// Calibre  
// REMOTE HOST xyz1 used for Remote Pseudo HDB  
// REMOTE HOST xyz2 used for Remote Pseudo HDB  
// REMOTE HOST xyz4 used for Remote Pseudo HDB  
// REMOTE HOST xyz5 used for Remote Pseudo HDB  
// Connected to Remote Pseudo HDB 1 on remote host xyz1  
// Connected to Remote Pseudo HDB 2 on remote host xyz2  
// Connected to Remote Pseudo HDB 3 on remote host xyz4  
// Connected to Remote Pseudo HDB 4 on remote host xyz5  
// HYPERSCALING ENABLED with 4 pseudo HDBs.
```

Calibre FullScale

Calibre FullScale is a multi-processing platform designed to improve scalability and runtime for typical post-tapeout runs. Calibre FullScale supports a limited set of SVRF operations, setlayer commands, and models.

Refer to the section “[Calibre FullScale Platform](#)” in the *Calibre Post-Tapeout Flow User’s Manual* for information on using Calibre FullScale, in addition to details on supported operations, commands, and models. The *Calibre Post-Tapeout Flow User’s Manual* also includes information on the transcript generated by a Calibre FullScale run.

Data Processing Modes

Calibre provides data processing modes that optimize the input hierarchy by partitioning data into a uniform and hierarchical structure of bins. These modes are primarily used for jobs that have little or no incoming hierarchy (for example, post-Litho operations) and Calibre jobs that cause significant hierarchical degradation during a run (for example, Litho or fill operations). These cases generally contain pre- or post-Litho SVRF operations.

An overview of the available data processing modes is shown in [Table 13-7](#).

Table 13-7. Overview of Calibre Data Processing Modes

Data Processing Mode	Description	Recommended Operations Invocation Options
“ULTRAflex Data Processing — LAYOUT ULTRA FLEX”	Partitions data by creating uniform bins of data with an empty cell hierarchy above that is used during distributed processing. Calibre ULTRAflex changes almost all of the input hierarchy, except original cells that exist within the lowest level of the binned structure.	Calibre nmOPC, Calibre MDP: <ul style="list-style-type: none">• -turbo [#CPUs] -remote• -turbo [#CPUs] -remotefile
“TURBOflex Data Processing — LAYOUT TURBO FLEX”	Partitions data by creating uniform bins of data across the extent of the hierarchy that is used during distributed processing. Calibre TURBOflex changes some of the input hierarchy, keeping some original cells throughout the multiple levels of the binned structure.	Calibre nmOPC, Calibre MDP: <ul style="list-style-type: none">• -turbo [#CPUs] -remote• -turbo [#CPUs] -remotefile

TURBOflex Data Processing — LAYOUT TURBO FLEX

Calibre TURBOflex is a data processing option that can achieve highly scalable parallel processing on large, multi-CPU servers and distributed compute systems (clusters). TURBOflex is most useful in a post-tapeout flow that significantly modifies the database hierarchy during processing, and should be used with extreme caution in traditional DRC flows on highly hierarchical databases.

Refer to “[TURBOflex Data Construction](#)” in the *Calibre Post-Tapeout Flow User’s Manual* for more information on using this capability.

ULTRAflex Data Processing — LAYOUT ULTRA FLEX

Calibre ULTRAflex is a data processing option that can achieve highly scalable parallel processing on large, multi-CPU servers and distributed compute systems (clusters). ULTRAflex is most useful in a post-tapeout flow and should be used with extreme caution in traditional DRC flows on highly hierarchical databases. The overall runtime for ULTRAflex is typically slower than TURBOflex.

Refer to “[ULTRAflex Data Construction](#)” in the *Calibre Post-Tapeout Flow User’s Manual* for more information on using this capability.

Configuration Requirements and Guidelines for Distributed Calibre

There are some hardware, network, licensing, and shared memory configuration requirements and guidelines for using distributed Calibre.

Hardware Configuration Guidelines	236
Network Configuration Guidelines	240
Licensing Requirements for Distributed Calibre	249
Shared Memory Guidelines for Distributed Calibre	250

Hardware Configuration Guidelines

There are some minimum recommended hardware configuration guidelines for the primary and remote hosts you use for Calibre jobs. In general, when running Calibre MTflex on clusters of more than 400 remote CPUs, consult with your IT department to ensure that adequate system resources are available for your target operating system.

Table 13-8. Recommended Hardware Specifications for and Remote Hosts

Feature	Primary Host	Remote Host
OS	64-bit OS.	
SW Version	All servers must run the same version of Calibre tools.	
Network	10 Gigabit/second for and remote hosts with ping time <0.500 ms. (1 Gigabit/second is sufficient for remotes with <16 CPUs).	
Disk	Fast network filer recommended to reduce file transfer times. Primary disks (if used) should be RAID0 striped to provide >600 MB/s sustained throughput.	Single small disk for temp files only.

Table 13-8. Recommended Hardware Specifications for and Remote Hosts

Feature	Primary Host	Remote Host
CPU General Guidelines	<p>For Calibre MTflex the primary host acts primarily as a dispatcher so performance is less critical. The number of CPUs should scale with the number of concurrent large Calibre MTflex jobs; a minimum of 16 CPUs is recommended for concurrent large Calibre MTflex jobs, but 32 CPUs is preferred.</p> <p>For MT jobs the CPU speed on the primary host impacts Calibre performance, so it is recommended to run MT jobs on the machine having the latest CPUs and fastest CPU clock speeds.</p>	The most recent processors will provide better throughput per clock. Calibre performance scales with CPU clock speed. Simultaneous multithreading (SMT) can provide additional throughput at no additional cost for jobs that are license constrained (that is, the number of licensed CPUs is less than the number of threads available to run).
CPU (manufacturing flows)	You may use either dual-processor with additional memory or quad-processor primary hosts, as needed. Dual-processor primary hosts reduce cost and have sufficient CPUs for most uses, but reduce maximum memory to <768GB. Quad-processor primary hosts have sufficient CPUs for dedicated Calibre MTflex usage or mixed MT / MTflex.	Faster remote CPU clock speeds are preferred, which typically means fewer CPUs per remote host. Maximum remote scaling varies from site to site but can be >10,000 CPUs for large litho-flat jobs.
CPU (design flows)		See the chart in Figure 13-8 on page 238 for typical CPU counts by technology node. Maximum remote scaling is typically 200 to 300 CPUs for 20nm and older technology nodes, and up to 1000 CPUs for large 7nm jobs.
RAM General Guidelines	The amount of primary host memory is data dependent and varies from site to site. You are encouraged to track memory trends over time to project future needs. The numbers shown in the RAM Manufacturing and RAM Design rows are typical. Calibre MTflex primary hosts should not be allowed to disk swap.	Consult Siemens EDA technical support if memory usage is higher than expected. Calibre memory usage can often be reduced by changing setup or configuration options. The numbers shown in the RAM Manufacturing and RAM Design rows are per physical CPU (not counting SMT virtual cores).
RAM (manufacturing flows) ¹	<ul style="list-style-type: none"> • 45nm: 64GB • 32nm: 128GB • 28nm: 256GB • 20nm: 0.5TB • <7nm: 3TB 	Refer to “ Recommended Setting for tilemicens per Node ” in the <i>Calibre nmOPC User’s and Reference Manual</i> for details. <ul style="list-style-type: none"> • All nodes: 8GB/CPU²

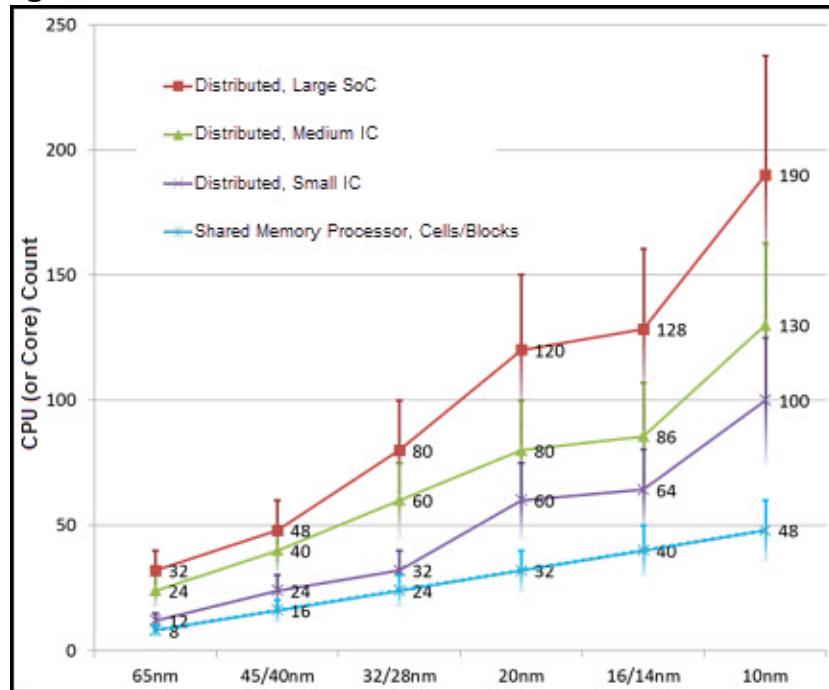
Table 13-8. Recommended Hardware Specifications for and Remote Hosts

Feature	Primary Host	Remote Host
RAM (design flows)	<ul style="list-style-type: none"> • 45nm: 64GB • 32nm: 128GB • 28nm: 256GB • 20nm: 0.5TB • <7nm: 1TB minimum (2TB recommended) 	<ul style="list-style-type: none"> • >32nm: 4GB/CPU • 28nm: 8GB/CPU • 20nm: 16GB/CPU • 16nm: 24GB/CPU • <16nm: 32GB/CPU

1. For advanced process node and other hardware recommendations, consult Siemens EDA technical support.
2. CPU and logical core are interchangeable.

Figure 13-8 shows the typical minimum CPU counts by technology node needed to achieve overnight run times for design-side (DRC) layouts of various sizes. The numbers are typical examples only. Actual run times vary from customer to customer depending on the layout data and the speed of the remote hosts for distributed jobs. Peak whole-job average scaling can be as high as 1000 CPUs for full chip DRC jobs at 7nm using “-hyper remote”.

Figure 13-8. Calibre nmDRC/nmDRC-H Customer Trends



Primary Host

For distributed Calibre jobs, the minimum configuration includes a primary host with at least eight CPUs. The primary host should have enough RAM to read in the layout and to perform hierarchical database optimization for your largest design.

If hyperscaling is used, a 16 CPU primary host is more effective than an eight CPU machine in MT or Calibre MTflex mode. The hardware configuration also depends on the data and rules. Smaller configurations work well with smaller data sets.

In a Calibre MTflex run, the primary host performs the following tasks:

- I/O management
- Multithreaded computation
- Near zero computation (operations that take ~0 seconds to complete)
- Current computation from failed remotes

For operations that are not optimized for Calibre MTflex, a faster primary host may significantly improve turnaround time.

For best performance, you should do the following:

- Let the primary host manage its CPUs, preferably on an eight to 16 CPU machine.
- Do not use any CPUs on the primary host as remote CPUs.

Remote Host

As Calibre scales with CPU speed, use the fastest CPUs that are available on remote hosts. Calibre with hyperscaling usually uses all CPUs on the remote host.

The remote hosts for distributed Calibre jobs should have sufficient RAM as described in [Table 13-8](#). For a homogeneous environment, the remote hosts must use the same OS and address space as the primary host. Additionally, the primary and remote hosts must use the same version of Calibre.

Network Configuration Guidelines

The network configuration is an important consideration for a distributed Calibre environment. While the data being transmitted is small, the frequency of transmission between the and remote hosts is high. As a result, a dedicated, high-speed network between the and remote hosts is recommended.

Calibre MTflex Network Setup	240
Calibre MTflex Network Management	243
Calibre MTflex Network Monitoring	244
Network Routing Examples	245
Linux Network Settings.....	245

Calibre MTflex Network Setup

The recommended Calibre MTflex network setup, testing, and performance validation is intended to support all Calibre applications that run in an MTflex environment, including DRC, OPC, and MDP tools. Refer to the applicable user’s manual for product-specific details about using the tool in a distributed environment.

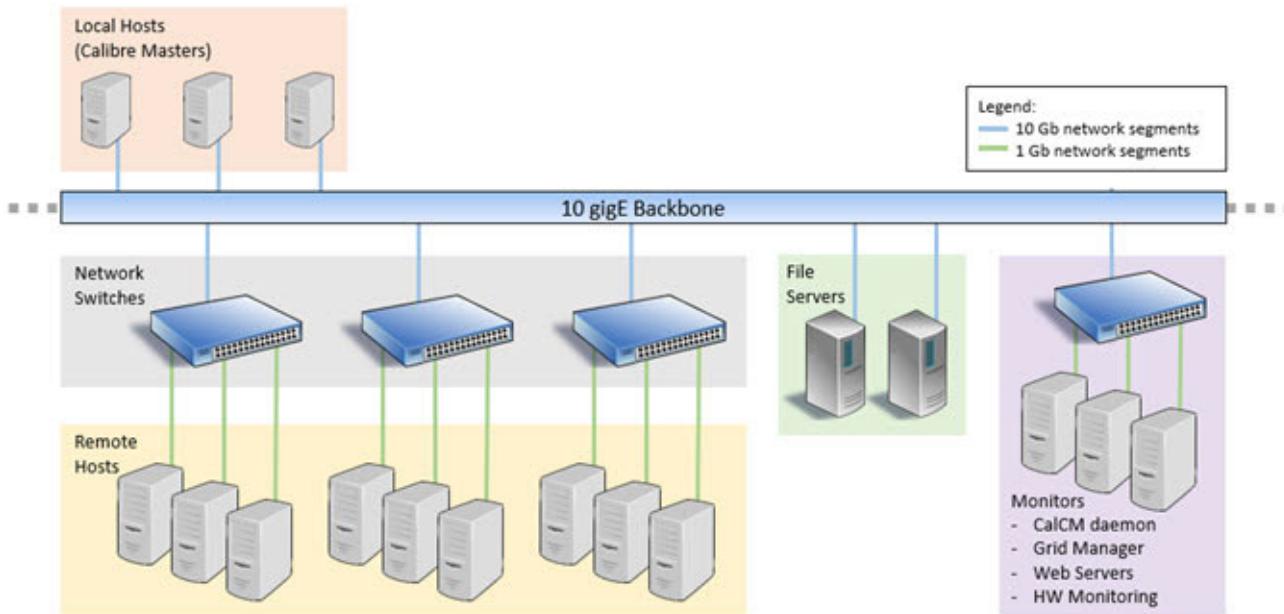
The information provided in this section is targeted at 32nm to 20nm process nodes. Minor adjustments to the specifications may be recommended for 15nm to 14nm and beyond.

Older network topologies such as “all gigE” are still supported and provide reasonable performance for smaller Calibre jobs, but the benefit of having faster networking and file servers increases with the number of remote hosts and the size of the input and output files.

Topology

Calibre MTflex is designed to be compatible with standard customer compute farm technology with minimal special requirements. For maximum flexibility and best performance, Calibre MTflex should be run on a “flat” network topology with connectivity from any host to any remote host. Connectivity is defined using the rsh (or ssh) and ping capabilities. The minimum recommended network bandwidth is 1 gigabit Ethernet (gigE) for remote hosts and 10 gigabit Ethernet (10 gigE) between primary hosts, network switches, and file servers as shown in [Figure 13-9](#).

Figure 13-9. Recommended Network Topology



Special Considerations

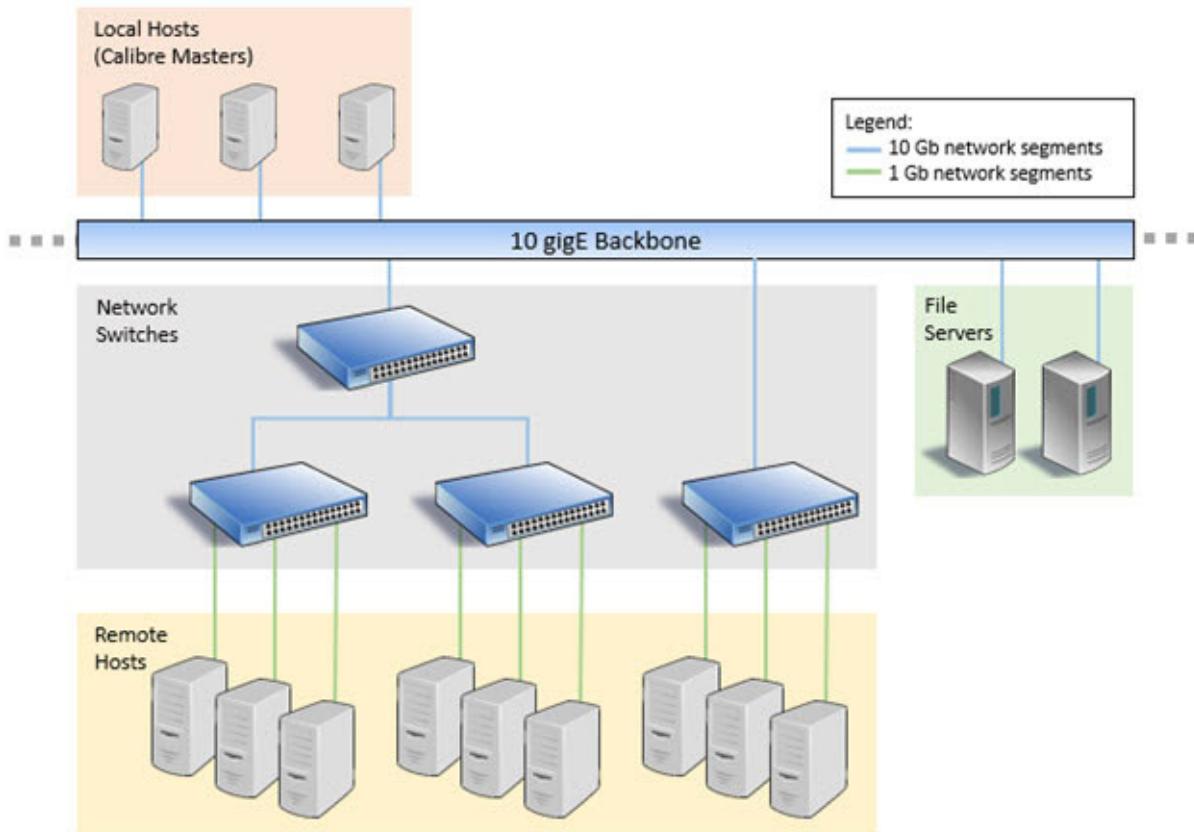
There are some special networking considerations to take into account for the recommended network topology:

- For small DRC jobs using less than 80 remote CPU cores, gigE is sufficient for the primary host.
- Faster networks, such as 40 gigE or QDR (40/32 Gbps) Infiniband, do not provide a significant performance benefit.
- File server bandwidth and latency are important for large MDP jobs and for large post-OPC files for “Cumulative RDB” time.
- Peer-to-peer connectivity between remote hosts is typically not needed except for Calibre jobs using “[Remote Data Server](#)” technology (-remotedata).
- Isolated clusters with sets of remote hosts dedicated to individual primary hosts can be used for single Calibre MTflex jobs but are not compatible with Calibre Cluster Manager (CalCM), which requires that remote hosts are reassignable between primary hosts.
- Wide Area Network (WAN) connectivity is not recommended during Calibre job execution due to insufficient bandwidth, latency, or both. However, WAN connectivity is often used for uploading and downloading a file prior to or following Calibre job execution.
- Use of 10 gigE primary hosts obviates the need for RDS network offload.

- Peak network traffic per-remote-host is low for DRC and very low for OPC. Peak traffic for MDP FRACTURE and MDPverify is up to 35 MB/sec for 8 and 12 CPU remote hosts available in 2011. Peak MDP traffic for 16/32 CPU remote hosts available in 2012 may reach 70 MB/sec, which is more than half of gigE bandwidth.
- Network switches should be high quality, non-blocking, with preferably four (minimum of two) 10 gigE ports. A typical switch might have 24 gigE ports for remote hosts and four 10 gigE ports: three for Calibre MTflex primary hosts and one for connection to the network backbone.

Tiered network switches with gigE interconnects as shown in [Figure 13-10](#) are not recommended for MDP jobs and should be used with caution for other Calibre applications. For non-MDP jobs, no more than 80 total remote CPUs should be aggregated on a single gigE link. For example, this would be ten 8-CPU remote hosts maximum per switch.

Figure 13-10. Recommended Network Topology - Tiered Switches



Network Setup Testing and Performance Validation

Network setup testing and performance validation is recommended any time hardware changes are made to the network, and periodically thereafter (for example, once a quarter). For best results, network benchmarking should be run on idle systems as part of a routine maintenance cycle.

Calibre provides a network benchmark tool for this purpose. To measure network performance, include the “[MONITOR LOCAL IO](#)” statement in a Calibre MTflex remote configuration file and use a dummy or an invalid rule filename. Upon executing a Calibre run using this setup, Calibre connects to all remotes specified in the configuration file, runs the network test (which takes several minutes), and then exits because of an invalid rule file. This test should be run when the hosts are idle.

MONITOR LOCAL IO tests the network performance between the primary host and each remote host, in addition to measuring bandwidth and latency. Bandwidth is expressed in Mbit/s. Latency is expressed in the number of packets-per-second, which is the inverse of ping time. Refer to the section “[Monitoring Remote Processes](#)” for an example of the output from the MONITOR LOCAL IO statement.

Typical MONITOR LOCAL IO bandwidth from each primary host to each remote host is 920 Mbit/s for a 1 Gbit/s network or 9000 Mbit/s for a 10 Gbit/s network. Typical ping latency is 0.200 ms, which translates to 5,000 packets/s in the MONITOR LOCAL IO output. Calibre performance is not sensitive to network performance, but numbers significantly lower than the typical values may indicate problems requiring further investigation.

Other network benchmark tools, such as Netperf and Iperf, can be used to perform similar network performance benchmarks. You can also perform a quick bandwidth test using the ping command (for example, ping -s 32000 <hostname>). File system benchmark tools such as IOZone, Bonnie++, and SIO (from NetApp®) can also be used to extend the basic network testing to include file server bandwidth and latency.

Private Network Settings

To run Calibre using Calibre MTflex functionality across a private network, where the remote hosts can only recognize the primary host and not the wider network, you must alter your configuration. When used on a “shared” network system, you should ensure that no “hub” points exist between the primary and remote hosts.

- Install Calibre on the primary host, and set your CALIBRE_HOME environment variable to that location. If the remote hosts are different platforms than the host, then you must install Calibre for each different platform on the host.
- Use the optional NAME keyword on the [LAUNCH AUTOMATIC](#) or [LAUNCH CLUSTER](#) statement to specify the name of the primary host or its IP number from the /etc/hosts file. If the primary host has multiple IP numbers or names, you must specify an IP address or name that is visible to the remote host.

Calibre MTflex Network Management

You can use scripts (similar to “cluster health” scripts) to minimize Calibre MTflex issues. Use of “cluster health” scripts enables the use of the MINCOUNT argument in the LAUNCH AUTOMATIC, LAUNCH CLUSTER, and LAUNCH MANUAL statements, which prevents

unexpectedly long Calibre run times due to insufficient allocation of remote hosts. Unsupported pseudo-code is available on request. These scripts run in conjunction with a reliable job scheduler or Distributed Resource Management System (DRMS).

By following the preventative measures described in [Table 13-9](#), MINCOUNT failures are rare. Automatic removal of unusable hosts is required to prevent flushing of the job submittal queue.

Table 13-9. Calibre MTflex Network Management Preventative Measures

Task	When	Description
Periodic reboot and housecleaning	Every ten days to three months when systems are idle	Clean up temporary space on disk drives, reboot, and perform post-reboot checks.
Unauthorized process check	Every ten minutes	Check for and kill any unauthorized processes (those not started by the job scheduler) to minimize contention.
Pre-flight check	Prior to launching any Calibre processes	Confirm valid paths, file system mounts, and disk full indications. On fatal errors, abort Calibre launch, remove host from job scheduler pool, automatically submit a ticket to IT, and substitute another host in the remote pool to maintain Calibre cluster size.

Calibre MTflex Network Monitoring

Network hardware monitoring greatly reduces Calibre MTflex troubleshooting time and, in some cases, can prevent long running jobs from failing by allowing problems to be corrected in a production environment before they become fatal.

Web-based network monitoring has the additional benefit that it does not expose a customer's Intellectual Property (IP), so the network monitoring data can be made available to Calibre Customer Support using screen capture or VPN technology, unlike proprietary test cases and log files.

A wide variety of commercially supported and free network monitoring tools are available for this purpose. The following is the recommended data set to monitor:

- Disk swap
- CPU load
- Load average
- Network traffic

Internally, Siemens EDA uses Nagios for detection and notification of alarm conditions and Ganglia for time-series charts of cluster load, but other equivalent tools are available.

Network Routing Examples

There are some network routing best practices and routing commands for determining the routing setup.

Network Routing Best Practices

As a best practice, each remote should be able to access (ping) every other remote. If your remotes are on separate network segments, do not over restrict them with the subnet mask (CIDR).

For example, consider nodes from 192.168.1.1 through 192.168.10.254:

```
Node A: 192.168.0.1
netmask 255.255.0.0 -> OK (more open than necessary)
netmask 255.255.245.0 -> GOOD (exact match to segments listed)
netmask 255.255.255.0 -> BAD (only open to 192.168.0.x segment)

Node1: 192.168.10.1 CIDR
192.168.1.0/16 -> OK (same as 255.255.0.0 - more open than necessary)
192.168.1.0/20 -> GOOD (equal to 255.255.240.0, allowing
    102.168.[0-16].x)
192.168.10.1/24 -> BAD (same as 255.255.255.0, only allowing 192.168.10.x
    segment)
```

The overly restricted (BAD) segments could still work, but would require a router between isolated segments.

Routing Commands

Use the following commands on each remote to determine the routing setup. While a direct response is ideal (indicates good routing setup), a second hop is okay if the latency remains at or below 0.2ms (indicates a router between hosts).

```
% /sbin/ifconfig
eth0 inet      addr:192.168.10.1  Bcast:255.255.0.0  Mask: 255.255.0.0

% traceroute Master
1  Master(192.168.1.1)  0.2ms  0.2ms  0.2ms
```

Linux Network Settings

Generally, Calibre MTflex works well with default Linux kernel and driver settings. A few changes may be necessary for large Calibre MTflex clusters that include many hundreds of remote CPUs.

Current ulimit settings can be displayed in the C shell using **limit**. For example:

```
% limit
cputime unlimited
filesize unlimited
datasize unlimited
stacksize 10240 kbytes
coredumpsize unlimited
memoryuse unlimited
vmmemoryuse unlimited
descriptors 65535
memorylocked 64 kbytes
maxproc 2048
```

They can be displayed with **ulimit -a** in Bourne shells. The **limit** and **ulimit** commands can also be used to configure settings of interest. You may need root-level privileges to do this.

Settings can be configured in */etc/security/limits.conf* or in the following files:

RHEL7: */etc/security/limits.d/20-nproc.conf*

RHEL6 and 5: */etc/security/limits.d/90-nproc.conf*

These settings are important for the performance of the machine when running Calibre. Certain runtime warnings are issued if these settings are not optimal, as discussed in the following table.

Table 13-10. Linux Network Kernel and Driver Recommended Settings

Class	Recommended Settings
ulimit settings limit name (ulimit name)	<p>descriptors (open files) — Calibre MTflex and CalCM use a file descriptor for each remote process (TCP socket), so this value should be larger than the total number of CPUs in the available pool, plus additional descriptors as needed for file I/O. Set this value to 65535 on hardware having <=32 GB of memory and to 350402 on hardware having >32 GB memory to allow for future growth.</p> <p>If the value is < 65535, then the Calibre transcript contains this warning:</p> <p>WARNING: Please increase descriptors limit for best performance</p> <p>If this warning is not mitigated, then multithreaded runs with large numbers of remote hosts could abort.</p> <p>maxproc (max user processes) — Set this value to the maximum process limit configured for the kernel. Refer to Table 13-13 on page 258 for guidelines on calculating the maxproc setting.</p> <p>If the maxproc setting is too low, then the Calibre transcript contains this warning:</p> <p>WARNING: Please increase max process limit for best performance</p> <p>Failure to mitigate the warning can result in Calibre aborting.</p> <p>coredumpsize (core file size) — Core files are used to debug crashes, which are rare with Calibre. This value may be set as needed based on IT practices because production Calibre builds do not produce useful core files. In the unlikely event that CSD recommends debugging a crash at a customer site, a special debug version of the Calibre executable will be provided and coredumpsize should be set to “unlimited” for the duration of the test.</p>

Table 13-10. Linux Network Kernel and Driver Recommended Settings (cont.)

Class	Recommended Settings
RSH Limits	<p>Linux limits the number of concurrent RSH processes as a protection against “denial of service” attacks. This can cause LAUNCH CLUSTER and/or CalCM to fail when connecting to certain remotes on large clusters.</p> <p>Workaround: If you encounter problems connecting to >500 remote CPUs on large jobs, use SSH instead of RSH or increase the <i>xinetd.conf</i> CPS (incoming connections per second) parameter from the default of 50 to some larger value.</p> <p>For the RHEL6 OS, add the following to <i>/etc/sysctl.conf</i> to prevent remote host connection failures on large clusters:</p> <pre>net.ipv4.tcp_tw_recycle = 1</pre> <p>This statement tells the OS to recycle the tcp ports faster, which alleviates most of the failing connections.</p> <p> Note: For the RHEL/CentOS 8 operating system, you must use SSH instead of RSH.</p>
NIC Settings	No specific performance settings are required. Follow the manufacturer’s recommendations for the best performance and check periodically for driver updates.
Internet Address Resolution Protocol (ARP) Table	The ARP cache tables and garbage collection (gc) thresholds should be large enough to retain the network addresses for all hosts in the available pool. Network ARP table garbage collection thresholds must be larger than the number of nodes (not CPUs) in the cluster to ensure adequate performance. The Linux default of 128 for gc_thresh1 is too low for clusters with more than 512 remote CPUs, assuming four CPUs per remote node. (See <i>/proc/sys/net/ipv4/neigh/default/gc_thresh</i> .)
Network Driver/TCP Settings	<p>Network (driver, TCP) settings should be tuned for high performance, large data block transfers, and OS kernel offload.</p> <p>When Calibre initially connects to a large number of remotes, the incoming connection requests are cached in the Linux system queue, which is called the TCP listen backlog queue. On RHEL5 and RHEL6 systems, the default size of this queue is 128, which is too small for Calibre MTflex. The small queue size can cause Calibre to be very slow when initializing the connection. The small queue size can also cause remotes to be dropped before connection to the primary host is established. For releases prior to Calibre 2017.2, the recommendation is to increase this queue size on primary hosts to 1024 by adding the following line to <i>/etc/sysctl.conf</i>.</p> <pre>net.core.somaxconn = 1024</pre>

Table 13-10. Linux Network Kernel and Driver Recommended Settings (cont.)

Class	Recommended Settings
Systems with Large Disks	For systems with disks that are one TB or larger, bump the number of NFSD to 16. The variable to set in the <i>/etc/sysconfig/nfs</i> file, is RPCNFSDCOUNT for RH and USE_KERNEL_NFSD_NUMBER for SLES.
OS Vendor Updates and Patches	Periodically check for OS Vendor networking updates and patches. Fixes to xinetd, ipv4, and tcp are often important to prevent random Calibre MTflex connection failures.
Disk Swap Space	<p>Remote hosts — Disk swap should be enabled on remote hosts and disk swap space should be equal to available memory (RAM). Remote host performance for some applications (such as OPC) is relatively insensitive to small amounts of remote disk swap up to 50 percent of RAM. Allowing for swapping may prevent jobs from failing unnecessarily.</p> <p>Primary hosts — Disk swap should be enabled on Calibre MTflex primary hosts, but should be avoided because primary host disk swap interferes with network performance and severely reduces Calibre MTflex performance. The amount of disk swap space is left to your discretion, but should be sufficient to allow user intervention (usually by killing one or more jobs) before swap free space goes to zero, which will cause the OS to become unresponsive.</p>
Name Service Caching	The Linux name service cache daemon has been reported to cause network failures. Unless this service is required, disabling the nscd may improve reliability.
Virtual Memory Zone Reclaim Mode	Set zone_reclaim_mode to one to ensure Calibre application data has priority over disk cache data. A value of zero (0) in this variable can significantly increase Calibre run times under certain circumstances. <pre>\$ echo 1 > /proc/sys/vm/zone_reclaim_mode</pre>

Licensing Requirements for Distributed Calibre

There are no requirements for unique licenses in order to run Calibre in a distributed environment. When you invoke a Calibre tool to run in a distributed environment, Calibre queries the operating system to determine the number of CPUs and recognizes multiple cores or hardware threads as individual CPUs.

Prior to running any Calibre tools, you should ensure the LM_LICENSE_FILE or MGLS_LICENSE_FILE environment variable is set to a valid license file or license daemon. Refer to “[MSL License File and Environment Variables](#)” on page 54 for more information.

For details regarding licensing requirements for distributed Calibre, refer to “[Licensing for Multithreaded Operations](#)” on page 71 and “[Simultaneous Multithreading and Licensing Requirements](#)” on page 73.

Shared Memory Guidelines for Distributed Calibre

When Calibre runs an operation, it needs some memory for computation and some memory to hold data. When hyperscaling is used, sometimes an operation on one HDB uses the same memory as an operation on a separate HDB. To reduce the overall memory footprint in such cases, data is transferred from the first HDB to the second HDB, and the amount of data transferred is written to the SHARED component. The SHARED statistic output to the transcript is the amount of memory in megabytes that was transferred between HDBs for a particular operation. For example:

LVHEAP = 4/6/6 SHARED = 1/1

The first number is the shared memory for the current step, while the second number is the maximum shared memory. This is a subset of the LVHEAP statistics.

When running Calibre in Multithreaded (MT) mode, the SHARED statistic is not an indicator of any sort of efficiency.

In Calibre MTflex mode, running with RDS (Remote Data Server) and without -hyper remote, SHARED numbers other than 0/0 can indicate there is not enough remote memory available. This typically indicates that improvement is possible with the addition of remote memory.

When using -hyper remote with RDS, it is important to have a SHARED memory statistic of 0/0, as memory sharing in this configuration degrades performance. If RDS is not enabled and you are seeing large SHARED numbers, this means you can see improvement by enabling RDS.

Calibre and Distributed Resource Management Tools

You can use distributed resource management tools, such as IBM Spectrum LSF (formerly IBM Platform LSF) or Univa Grid Engine, to manage workload processing for compute- and data-intensive applications. These tools are useful for scheduling and tracking the completion of Calibre jobs in a distributed environment.

IBM Spectrum LSF and Univa Grid Engine can be used with Calibre in one of the following execution modes:

- Single CPU run
- Multithreaded (MT) on SMP (shared memory multi-processor) machine run
- Distributed Calibre MTflex run

IBM Spectrum LSF and Univa Grid Engine can both be configured to maximize the advantages of simultaneous multithreading (SMT). Refer to “[Simultaneous Multithreading With Virtual CPUs](#)” on page 222 for more information.

Note

 The rcalibre command includes an argument that is commonly used with IBM Spectrum LSF. Refer to the [rcalibre](#) command reference page for more information.

Job Checkpointing	251
Job Suspension	251
IBM Spectrum LSF Commands	253
Univa Grid Engine Commands	255

Job Checkpointing

Calibre does not support the job checkpointing capabilities in IBM Spectrum LSF and Univa Grid Engine.

Job Suspension

Calibre supports suspending and resuming jobs, typically employed using the bstop and bresume commands for IBM Spectrum LSF and the qmod command for Univa Grid Engine. When a job is suspended, it remains in memory but Calibre releases all licenses and suspends operation. When the job is resumed, Calibre checks out the license(s) and continues the run.

Calibre may also be suspended outside of the IBM Spectrum LSF or Univa Grid Engine using the TSTP signal (Ctrl-z) and resumed using the CONT signal (fg command).

IBM Spectrum LSF may implement policies to automatically pause Calibre (SIGSTOP) under conditions of high load. Calibre is designed to run at high loads for extended periods of time to achieve best performance.

Caution

-  You should disable IBM Spectrum LSF policies to automatically pause Calibre under high load conditions. These conditions will negatively impact performance.
-

IBM Spectrum LSF Commands

Some commonly used IBM Spectrum LSF commands include bsub, bjobs, and bqueues. For more information on these and other IBM Spectrum LSF commands, refer to your IBM Spectrum LSF documentation.

Table 13-11 provides an alphabetical listing of some of the more commonly used IBM Spectrum LSF commands. For examples of these commands, refer to “[Executing Calibre Using IBM Spectrum LSF](#).”

Table 13-11. Commonly Used IBM Spectrum LSF Commands

Command	Description	Example
bjobs	Displays job information. By default, this command displays information about the user who invoked the job.	Displays the jobs for all users: <code>bjobs -u all</code> Displays details about the submitted job: <code>bjobs -l</code>
bkill	Cancels pending batch jobs and kills jobs that are currently running.	Kills job 4321: <code>bkill 4321</code>
bqueues	Displays information about available queues.	Displays queues that accept jobs from <username>: <code>bqueues -u <username></code> Displays queues that run jobs from <hostname>: <code>bqueues -m <hostname></code>
bresume	Resumes one or more suspended jobs.	Resume job 8765: <code>bresume 8765</code>
bstop	Suspends one or more unfinished jobs.	Suspend job 8765: <code>bstop 8765</code>

Table 13-11. Commonly Used IBM Spectrum LSF Commands (cont.)

Command	Description	Example
bsub	Used to submit a job for batch execution. A unique ID is assigned to the job. Some of the valid bsub options include the following: <ul style="list-style-type: none">• <code>-o file_path</code> redirects the standard output into the location specified by <i>file_path</i>.• <code>-n [min_proc[,max_proc]]</code> specifies the number of processors required to run the job.• <code>-R "resource_reqt_file"</code> runs the job on a host that meets the resource requirements specified in the resource requirements file.• <code>-n cpus</code> instructs LSF to select the specified number of CPUs for running Calibre.	Submit my_job to the default queue: <code>bsub my_job</code>
lsgrun	Submits an interactive task to a group of hosts.	
lshosts	Displays static resource information about hosts in the cluster.	
lsinfo	Displays the resources available in your cluster, including resource names and meanings, host types and models, and associated CPU factors known to the system.	

Univa Grid Engine Commands

Some commonly used Univa Grid Engine commands include qhost, qstat, and qsub. For more information on these and other Univa Grid Engine commands, refer to your Univa Grid Engine documentation.

Table 13-12 provides an alphabetical listing of some of the more commonly used Univa Grid Engine commands.

Table 13-12. Commonly Used Univa Grid Engine Commands

Univa Grid Engine Command	Description	Example
qconf -sc	Displays the current complex configuration.	Display the current complex configuration: <code>qconf -sc</code>
qconf -sql	Displays a list of all currently configured cluster queues.	Display a list of configured cluster queues: <code>qconf -sql</code>
qdel	Cancels a job that is running or waiting to execute.	Deletes job 4321: <code>qdel 4321</code>
qhost	Displays host information (for example, architecture, number of processors, and load).	Display host information: <code>qhost</code> Display host information, including information about specific jobs running on each host: <code>qhost -j</code>
qmod -sj	Suspends one or more grid jobs.	Suspend job 8765: <code>qmod -sj 8765</code>
qmod -usj	Resumes one or more grid jobs.	Resume job 8765: <code>qmod -usj 8765</code>
qstat	Displays information about the jobs in the Univa Grid Engine queues.	Displays a list of all running and waiting jobs: <code>qstat</code> Displays summary information on all queues and the queued job list: <code>qstat -f</code>

Table 13-12. Commonly Used Univa Grid Engine Commands (cont.)

Univa Grid Engine Command	Description	Example
qsub	<p>Submits a job to the Univa Grid Engine for batch execution and assigns the job a unique ID. Some of the valid qsub options include the following:</p> <ul style="list-style-type: none">• -cwd directs Univa Grid Engine to run the job in the same directory from which you submitted it.• -v <i>variable</i> passes an environment variable to the job.• -o <i>path</i> redirects stdout from the Univa Grid Engine script.• -l <i>resources</i> specifies a list of resources required for your job.	<p>Submits a job to the default queue and specifies stdout is output to <i>drc.log</i>. Instructs the Univa Grid Engine to use an 8 CPU Linux host with at least 2 Gb of free memory, and uses the <i>run_calibre.csh</i> script to switch Calibre trees and execute Calibre:</p> <pre>qsub -o drc.log -l "num_proc=8, archlx24x86, mem_free>2G" run_calibre.csh</pre>

Known Issues

There are some known issues related to using distributed Calibre.

Output Differences.....	257
Exceeding a Shell's File Descriptor Limit.....	257
NFSD Error “Too Many Open TCP Sockets”	258
Potential Network Errors	258

Output Differences

Due to the inherent differences in floating-point calculations, different hardware platforms show differences in Calibre output. These differences are typically limited to one database unit or one step size for any given Calibre operation. These differences are considered within an acceptable accuracy tolerance for Calibre.

Calibre output differences due to floating-point calculation disparities may also appear in heterogeneous (mixed operating systems) Calibre MTflex runs. The Calibre MTflex thread scheduler determines which thread is to be executed on which machine and CPU at run time to best utilize the machine resources. Because of the dynamic nature of Calibre MTflex runs, any particular calculation could be run on a different platform from run to run. These run configuration differences may result in output inconsistencies, even in runs using the same Calibre version and same hardware configuration.

Model-based OPC results can also change because of this behavior. However, because there is no unique answer for model-based OPC in any case, it is recommended that you always run ORC at the end to verify the results. Usually, one step-size level variations do not cause significant impact, but it is best to use this practice of ORC verification to ensure the variations due to the floating-point computation inconsistency are indeed insignificant.

Exceeding a Shell's File Descriptor Limit

When performing multithreaded or Calibre MTflex runs, you can encounter situations where you run out of available licenses for the Calibre software run. Running out of licenses causes the Calibre application either to exit with a warning or to run with fewer licenses.

If you have the required number of licenses and encounter this problem, then you should check your operating system's file descriptor limit, specifically the limit in the shell used to invoke the Calibre application. To adjust the shell's file descriptor limit, consult the documentation for your particular operating system.

The Mentor Standard Licensing (MSL) system requires the following shell file descriptors per each invocation:

- Four descriptors for each license server in the LM_LICENSE_FILE variable

- One descriptor for each license that is checked out
- One descriptor for each remote process
- Small number of descriptors for other overhead

Refer to “[Licensing for Multithreaded Operations](#)” for information on licensing in a distributed Calibre environment.

NFSD Error “Too Many Open TCP Sockets”

In some cases, when running Calibre MTflex where a large number of remotes are concurrently writing back to a primary host, an error message is generated.

Symptoms

For example:

```
kernel: nfsd: too many open TCP sockets, consider increasing the number of
nfsd threads
```

Solution

Increase the number of nfsd threads to 16 as follows:

- For RHEL 5, modify the RPCNFSDCOUNT value in */etc/sysconfig/nfs*.
- For SLES 11, modify the USE_KERNEL_NFSD_NUMBER value in */etc/sysconfig/nfs*.

Potential Network Errors

Running a job in Calibre MTflex and Hyperscaling modes (-turbo -hyper) may occasionally generate a “PTHREAD INIT Failure” or “Hyper data storage failure” error message.

You can prevent these errors by increasing the user limits (maxproc). The maxproc setting limits the total number of threads that can be used for a job. [Table 13-13](#) provides some guidelines for the maxproc setting based on the number of physical remote CPUs that are to be used for a Calibre job.

Table 13-13. Guidelines for Calculating the maxproc Setting

Physical Remote CPUs (N)	Total Threads on Primary Host
< 200	$N + 4N = 5N$
$\geq 200 < 400$	$N + 2N = 3N$
≥ 400	$N + N = 2N$

Chapter 14

Distributed Calibre: Tool-Specific Support Information

Some Calibre tools take advantage of the multithreaded (MT) or Calibre MTflex capability and, when possible, also leverage hyperscaling capabilities.

Calibre Physical Verification Product Support for Multithreaded and Hyperscaling Modes	260
Calibre DFM Product Support for Multithreaded and Hyperscaling Modes	262
Calibre PEX Product Support for Multithreaded and Hyperscaling Modes	263
Calibre RET Product Support for Multithreaded and Hyperscaling Modes	264
Calibre MDP Product Support for Multithreaded and Hyperscaling Modes	266
Tool-Specific Information for Distributed Calibre.....	268

Calibre Physical Verification Product Support for Multithreaded and Hyperscaling Modes

Some Calibre Physical Verification products support multithreaded (MT) or Calibre MTflex with or without hyperscaling.

Table 14-1 provides a list of Calibre Physical Verification products and identifies whether the product supports running a Calibre job using multithreaded (MT) or Calibre MTflex with or without hyperscaling mode.

Table 14-1. Calibre Physical Verification Product Support for Multithreaded and Hyperscaling Modes

Calibre Product	Multithreaded or Calibre MTflex	Hyperscaling
Calibre 3DSTACK	Yes	Yes
Calibre ADP	Yes	Yes
Calibre AutoFix	No	No
Calibre Auto-Waivers	Yes	No ¹
Calibre CB	Yes	No
Calibre Connectivity Interface	Yes ²	No
Calibre DESIGNrev	Yes	No
Calibre e2lvs	No	No
Calibre Interactive	No	No
Calibre Multi-Patterning	Yes	Yes
Calibre Multi-Patterning Advanced	Yes ²	Yes
Calibre nmDRC	No	No
Calibre nmDRCgold	Yes	Yes
Calibre nmDRC-H	Yes	Yes
Calibre nmLVS	No	No
Calibre nmLVS-H	Yes	Yes
Calibre nmLVS Advanced	Yes	Yes
Calibre nmLVS Reconnaissance	Yes	Yes
Calibre Pattern Matching	Yes	Yes
Calibre PERC	Yes ³	Yes ⁴
Calibre PERC Advanced	Yes ³	Yes ⁴

Table 14-1. Calibre Physical Verification Product Support for Multithreaded and Hyperscaling Modes (cont.)

Calibre Product	Multithreaded or Calibre MTflex	Hyperscaling
Calibre Physical Verification Suite (PVS)	Yes	Yes
Calibre Query Server	Yes ²	No
Calibre RealTime Custom	No	No
Calibre RealTime Custom Lite	No	No
Calibre RealTime Digital	No	No
Calibre Rule File Analyzer	No	No
Calibre RVE	Yes ⁵	No
Calibre SVRFencrypt	No	No
Calibre v2lvs	No	No
Tanner-Calibre One DRC/LVS	No	No

1. Hyperscaling is not available when running Calibre RVE generation of waiver shapes or using the waiver_flow executable. The hyperscaling (-hyper) option is available when using calibre -drc -waiver.
2. Multithreaded parallel processing is supported for some operations performed in Calibre RVE and Calibre Query Server, such as Interactive Short Isolation in Calibre RVE and Query Server for LVS and DRC operations executed from a Calibre YieldServer script or the **Layer Browser** tab in Calibre RVE for DFM.
3. Refer to the [Calibre PERC User's Manual](#) for information on running Calibre PERC in multithreaded or Calibre MTflex mode.
4. Only applies when using the CALIBRE::HIERARCHICAL CIRCUIT EXTRACTOR module.
5. Multithreaded parallel processing is supported for some operations performed in Calibre RVE and Calibre Query Server, such as Interactive Short Isolation in Calibre RVE and Query Server for LVS and DRC operations executed from a Calibre YieldServer script or the Layer Browser tab in Calibre RVE for DFM.

Related Topics

[Multithreaded \(MT\) Processing](#)

[Calibre MTflex Processing](#)

[Hyperscaling](#)

[Multithreaded Parallel Processing for Calibre RVE Operations \(-turbo Option\) \[Calibre RVE User's Manual\]](#)

Calibre DFM Product Support for Multithreaded and Hyperscaling Modes

Some Calibre Design for Manufacturability (DFM) products support multithreaded (MT) or Calibre MTflex with or without hyperscaling.

[Table 14-2](#) provides a list of Calibre DFM products and identifies whether the product supports running a Calibre job in multithreaded (MT) or Calibre MTflex mode and in hyperscaling mode.

Table 14-2. Calibre Design for Manufacturability (DFM) Product Support for Multithreaded and Hyperscaling Modes

Calibre Product	Multithreaded or Calibre MTflex	Hyperscaling
Calibre CMPAnalyzer	Yes	Yes
Calibre CMP Model Builder	Yes ¹	No
Calibre LFD	Yes	No
Calibre YieldAnalyzer	Yes	Yes
Calibre YieldEnhancer	Yes	Yes
Calibre YieldServer	Yes	Yes

1. Many stages in cmpoptimize are not multithreaded. Because of the additional load when running distributed, runtime may actually increase when working on smaller data sets.

Related Topics

[Multithreaded \(MT\) Processing](#)

[Calibre MTflex Processing](#)

[Hyperscaling](#)

Calibre PEX Product Support for Multithreaded and Hyperscaling Modes

Some Calibre Parasitic Extraction (PEX) products support multithreaded (MT) or Calibre MTflex with or without hyperscaling.

Table 14-3 provides a list of Calibre PEX products and identifies whether the product supports running a Calibre job in multithreaded (MT) or Calibre MTflex mode and in hyperscaling mode.

Table 14-3. Calibre Parasitic Extraction (PEX) Product Support for Multithreaded and Hyperscaling Modes

Calibre Product	Multithreaded or Calibre MTflex	Hyperscaling
Calibre xACT	Yes	No
Calibre xACT 3D	Yes (-pdb and -phdb only)	No
Calibre xACT 3D Reference	No	No
Calibre xACTView	No	No
Calibre xL	No	No
Calibre xRC	Yes (-pdb and -phdb only) ¹	No
Calibre xRC CB	No	No
Calibre xRC to ADVance MS	Yes (-pdb only)	No
Tanner-Calibre One xRC	No	No
xCalibrate Rule File Generator	Yes	No

1. By default (without specifying -turbo), Calibre xRC runs on two CPUs using one *calibrexrc* license. To use only one CPU, specify “-turbo 1” during Calibre xRC invocation.

Related Topics

- [Multithreaded \(MT\) Processing](#)
- [Calibre MTflex Processing](#)
- [Hyperscaling](#)
- [Calibre xRC Distributed Operations](#)

Calibre RET Product Support for Multithreaded and Hyperscaling Modes

Some Calibre Resolution Enhancement Technology (RET) products support multithreaded (MT) or Calibre MTflex with or without hyperscaling.

Table 14-4 provides an alphabetized list of Calibre RET products and identifies whether the product supports running a Calibre job in multithreaded (MT) or Calibre MTflex mode and in hyperscaling mode.

Table 14-4. Calibre Resolution Enhancement Technology (RET) Product Support for Multithreaded and Hyperscaling Modes

Calibre Product	Multithreaded or Calibre MTflex	Hyperscaling
Calibre Cell Array OPC	Yes	No
Calibre DSA Mask Synthesis	Yes	No
Calibre DSA Verification	Yes	No
Calibre EUV	Yes	No ¹
Calibre LITHOview	Yes	No
Calibre LPE	Yes	No
Calibre mlOPC	Yes	No
Calibre MP Manufacturing	Yes	No
Calibre MP Manufacturing Advanced	Yes	No
Calibre nmModelflow	Yes	No
Calibre nmOPC	Yes	No ¹
Calibre nmSRAF	Yes	No
Calibre OPCpro	Yes	No ¹
Calibre OPCsbar	Yes	No ¹
Calibre OPCverify	Yes	No ¹
Calibre OPCverify Classify Plus	Yes	No ¹
Calibre ORC	Yes	No ¹
Calibre PRINTimage	Yes	No ¹
Calibre pxOPC	Yes	No
Calibre pxSMO	Yes	No

Table 14-4. Calibre Resolution Enhancement Technology (RET) Product Support for Multithreaded and Hyperscaling Modes (cont.)

Calibre Product	Multithreaded or Calibre MTflex	Hyperscaling
Calibre SONR	Yes	No
Calibre TDopc	Yes	No ¹
Calibre WORKbench	Yes	No

1. Hyperscaling has no direct benefit for OPC operations and increases local and remote host memory requirements. Only OPC jobs with significant pre- or post-OPC SVRF operations should be considered as candidates for hyperscaling and then only if sufficient memory is available.

Related Topics

[Multithreaded \(MT\) Processing](#)

[Calibre MTflex Processing](#)

[Hyperscaling](#)

[Calibre OPCpro Distributed Operations](#)

Calibre MDP Product Support for Multithreaded and Hyperscaling Modes

Some Calibre Mask Data Preparation (MDP) products support multithreaded (MT) or Calibre MTflex with or without hyperscaling.

Table 14-5 provides an alphabetized list of Calibre MDP products and identifies whether the product supports running a Calibre job in multithreaded (MT) or Calibre MTflex mode and in hyperscaling mode.

Table 14-5. Calibre Mask Data Preparation (MDP) Product Support for Multithreaded and Hyperscaling Modes

Calibre Product	Multithreaded or Calibre MTflex	Hyperscaling ¹
Calibre DefectClassify	Yes	No
Calibre DefectReview	Yes	No
Calibre FRACTUREc	Yes	No
Calibre FRACTUREh	Yes	No
Calibre FRACTUREi	Yes	No
Calibre FRACTUREj	Yes	No
Calibre FRACTUREm	Yes	No
Calibre FRACTUREn	Yes	No
Calibre FRACTUREp	Yes	No
Calibre FRACTUREt	Yes	No
Calibre FRACTUREv	Yes	No
Calibre Job Deck Editor	No	No
Calibre MASKOPT	Yes	No
Calibre MDP EMBED	Yes	No
Calibre MDP Embedded SVRF	Yes	No
Calibre MDPAutoClassify	Yes	No
Calibre MDPDefectAvoidance	Yes	No
Calibre MDPmerge	Yes	No
Calibre MDPstat	Yes	No
Calibre MDPverify	Yes	No
Calibre MDPview	Yes	No
Calibre Metrology API (MAPI)	No	No

Table 14-5. Calibre Mask Data Preparation (MDP) Product Support for Multithreaded and Hyperscaling Modes (cont.)

Calibre Product	Multithreaded or Calibre MTflex	Hyperscaling ¹
Calibre MPCpro	Yes	No
Calibre MPCverify	Yes	No
Calibre nmMPC	Yes	No

1. Refer to “[Calibre MDP Applications and Hyperscaling](#)” on page 270 for information on using hyperscaling with Calibre MDP applications.

Related Topics

[Multithreaded \(MT\) Processing](#)

[Calibre MTflex Processing](#)

[Hyperscaling](#)

[Calibre FRACTURE Distributed Operations](#)

[Calibre MDPmerge Distributed Operations](#)

[Calibre MDP Applications and Hyperscaling](#)

Tool-Specific Information for Distributed Calibre

Any SVRF operation may appear in a rule file used for distributed Calibre runs. While not all SVRF operations are distributed, the operations that can scale will do so automatically. Those operations that do not participate in distributed processing run only on the primary host. Some Calibre tools may have limitations or restrictions for running in distributed Calibre mode.

Refer to “[Distributed Calibre: Reference Dictionary](#)” on page 309 for statement, variable, and command line reference information.

Calibre xRC Distributed Operations	268
Calibre OPCpro Distributed Operations	268
Calibre FRACTURE Distributed Operations	269
Calibre MDPmerge Distributed Operations	269
Calibre MDP Applications and Hyperscaling	270

Calibre xRC Distributed Operations

Distributed Calibre functionality is available with Calibre® xRC™ when creating the Parasitic Database (PDB).

Calibre OPCpro Distributed Operations

By default, only the CPU usage for the primary host is provided in the transcript when running Calibre OPCpro in Calibre MTflex mode. Full CPU time statistics for both the local and remote hosts are available after the OPCpro run has completed and the total runtime information for each command group is printed to the log file.

In order to get status messages for remote CPUs written to the transcript during the cell processing loop of Calibre OPCpro runs in Calibre MTflex mode, you must set the LITHO_REPORT_REMOTE_CPU_TIME variable prior to invoking Calibre. For example:

```
setenv LITHO_REPORT_REMOTE_CPU_TIME 1
```

When this variable is not enabled (default), the final log still contains full CPU time statistics including those of remote processes and total time by command group.

Note

 This reporting may impact performance because of the much larger number of writes to the output, so this is recommended primarily for debugging problems with remote runs.

Calibre FRACTURE Distributed Operations

The Calibre FRACTURE tools include: FRACTUREc, FRACTUREh, FRACTUREi, FRACTUREj, FRACTUREm, FRACTUREn, FRACTUREp, FRACTUREt, and FRACTUREv. These tools, when run with distributed Calibre, create temporary files on remote hosts during FRACTURE operations. You can use the LOCAL HOST DIR statement to specify a directory on the primary host where these tools can write temporary files.

The following example shows a **LOCAL HOST DIR** statement used in a configuration file:

```
#Configuration file example
REMOTE HOST linuxone 2 MGC_HOME /$INSTALL_DIR/aoi/mgc_home
LOCAL HOST DIR /net/machine/dir1/tmp_location
```

If the directories specified by the REMOTE HOST or LOCAL HOST DIR statements are not accessible by a remote host when it first tries to write one of these temporary files, Calibre disconnects the remote host and issues the following error message:

```
ERROR: MTflex: OPEXEC failed
WARNING: Problem with CPU on remote host: node<#>, it will no longer be
used
```

Calibre continues the run without using the remote host.

Calibre MDPmerge Distributed Operations

During Calibre MTflex processing, remote processes directly access the chip files referenced in the input and output rule files. To ensure these files are visible to remote hosts, Calibre MDPmerge places restrictions on the input and output parsing of directory data.

- Input directory:
 - Specify a single directory name, not a path.
 - The value specified for the **LOCAL HOST DIR** *path* statement is automatically prefixed to the directory you specify.
 - If you specify a path of any type, Calibre MDPmerge reads the last directory in the path and automatically prefixes LOCAL HOST DIR to the remaining directory. Consequently, you must place the input rule file directly inside the LOCAL HOST DIR *path* directory.
- Output directory:
 - No support for specifying absolute paths.
 - Calibre MDPmerge prefixes the LOCAL HOST DIR *path* to the complete output_dir argument.

- You may specify any relative path if it is relative to LOCAL HOST DIR *path* and points to a location accessible from both local and remote hosts.

During normal execution, Calibre prints the user specified paths in the transcript. The actual paths when running Calibre MDPmerge in Calibre MTflex mode (after applying the input and output directory restrictions) are not printed in the transcript except in error messages.

Calibre MDP Applications and Hyperscaling

You can use hyperscaling with Calibre MDP applications. There is currently a limitation that prohibits the use of commands that depend on each other. External file input cannot be verified during the run and hence a Calibre MDPverify application relying on the output of the Calibre FRACTURE operations in the same run may fail due to a missing input file or an incomplete file.

Currently, only MDP commands of the same type should be used when hyperscaling mode is targeted or MDP commands of variable type can be used when the input files are available at the beginning of the run.

The commands with dependencies include Calibre FRACTURE, Calibre MDPverify, and Calibre MDPstat.

Chapter 15

Distributed Calibre: Creating a Configuration File

You control Calibre MTflex functionality through the use of a configuration file.

About the Configuration File	272
Configuration File Statements	274
Using Environment Variables in a Configuration File.....	275

About the Configuration File

A configuration file is an ASCII file that specifies information used for launching remote hosts to run a job in Calibre MTflex mode. The term *launch* refers to invoking Calibre on a remote host as a Calibre MTflex server and connecting to that server. The normal mode of operation is to launch a Calibre MTflex server automatically, but you can also manually invoke the server on a remote host and connect to the primary host.

You create a configuration file using the statements documented in “[Configuration File Reference Dictionary](#)” on page 310, while adhering to the following syntax conventions:

- “//” style comments, where everything between “//” and the end-of-line character is ignored.
- All arguments to the configuration file statements support single-quote (‘ ’) or double-quote (“ ”) syntax. All characters between matching quotation marks are accumulated as the argument. A quoted value is terminated by an end-of-line.
- Arguments that specify a path should not contain relative references, links, shell variables, or environment variables that require dereferencing to resolve. Paths specified for MGC_HOME or CALIBRE_HOME, and the current working directory must be fully qualified paths.

The following is an example of a configuration file, which typically resides on the primary host.

Example 15-1. Configuration File

```
LAUNCH CLUSTER MINCOUNT 4
REMOTE COMMAND qsub_script.csh ARGUMENTS [ -o rem1.log -j y -t 1-4 -l
    "num_proc=2,arch=*" "run_cluster" %H %P ]
```

You specify the path to the configuration file when invoking Calibre using the *-remotefile filename* argument. For example, if you create a configuration file named *mtflex.config* you would specify it at runtime using the *-remotefile* switch, as follows:

calibre -drc -hier -turbo -remotefile mtflex.config -hyper my_rule_file	
MGC_HOME Definition in the Remote Host Statement.....	272
xCalibrate Configuration File Requirements	273

MGC_HOME Definition in the Remote Host Statement

You can change the default location of the MGC_HOME tree for each remote host from the configuration file. Specifying the location of the MGC_HOME tree is required if the remote hosts use a different platform from the primary host.

For example:

```
REMOTE HOST linuxone 2 MGC_HOME /$INSTALL_DIR/aoi/mgc_home
REMOTE HOST linuxtwo 2 MGC_HOME /$INSTALL_DIR/aoi/mgc_home
REMOTE HOST linuxthree 2 MGC_HOME /$INSTALL_DIR/aoi/mgc_home
REMOTE HOST linuxfour 4 MGC_HOME /$INSTALL_DIR/aoi/mgc_home
```

Related Topics

[Using Environment Variables in a Configuration File](#)

xCalibrate Configuration File Requirements

The configuration file for xCalibrate uses the same syntax conventions and configuration file statements as used with other Calibre tools, with the exception of the REMOTE COMMAND statement.

Note

 As of the 2021.1 release, the VER and DIR configuration file header statements are no longer required or supported. xCalibrate uses the same configuration file syntax as Calibre.

The **REMOTE COMMAND** for xCalibrate must specify an executable script that calls the **rxcalibrate** (remote xcalibrate) utility to initiate a Calibre MTflex run for xCalibrate on remote hosts. See “[Executing xCalibrate With Calibre MTflex Using IBM Spectrum LSF](#)” on page 299 for an example.

Configuration File Statements

There are certain statements you can specify in your configuration file to control distributed Calibre operations.

Refer to “[Distributed Calibre: Reference Dictionary](#)” for detailed information about these statements.

Table 15-1. Configuration File Statements

Statement	Description
HYPER	Overrides parameters used to determine remote pseudo HDB (PHDB) allocation for hyperscaling.
LAUNCH AUTOMATIC	Sets parameters for the automatic connection of the primary host to remote hosts. You can also set this statement using the CALIBRE_MTFLEX_LAUNCH environment variable.
LAUNCH CLUSTER	Sets parameters for the connection of the primary host to the remote hosts. You can also set this statement using the CALIBRE_MTFLEX_LAUNCH environment variable.
LAUNCH MANUAL	Sets parameters for the user-initiated, manual connection of the primary host to the remote hosts. You can also set this statement using the CALIBRE_MTFLEX_LAUNCH environment variable.
LOCAL HOST COMPUTE	Specifies the number of CPUs on the primary host to use for computational tasks. You can also set values for this statement with the CALIBRE_MTFLEX_LOCAL_HOST environment variable.
LOCAL HOST DIR	Specifies where Calibre stores temporary files, and input and output job decks. You can also define this same information using the CALIBRE_MTFLEX_LOCAL_HOST_DIR environment variable.
MONITOR LOCAL	Enables monitoring of Calibre MTflex operations on the primary host. You can also set this statement using the CALIBRE_MTFLEX_MONITOR_LOCAL environment variable.
MONITOR REMOTE	Enables monitoring of Calibre MTflex operations on the remote host. You can also set this statement using the CALIBRE_MTFLEX_MONITOR_REMOTE environment variable.
MONITOR SYSTEM	Enables the monitoring and reporting of primary and remote hosts by collecting system information, such as memory and swap usage.

Table 15-1. Configuration File Statements (cont.)

Statement	Description
REMOTE COMMAND	Provides information on launching a Calibre Remote Protocol (CRP) server on a remote host.
REMOTE HOST	Provides information for launching a remote host when using LAUNCH AUTOMATIC.

Using Environment Variables in a Configuration File

You can define an environment variable in your operating system environment and subsequently use that variable in your Calibre MTflex configuration file.

Procedure

1. In your operating system environment define the environment variable. For example:

C Shell:

```
% setenv MYCOUNT 5
```

Bourne/Korn Shell:

```
$ set MYCOUNT=5
$ export MYCOUNT
```

2. Create a configuration file named *mtflex.config* and enter the LAUNCH CLUSTER statement along with the \$MYCOUNT variable. For example:

```
LAUNCH CLUSTER COUNT $MYCOUNT
REMOTE COMMAND bsub ARGUMENTS [<LSF_options>]
```

3. Save and close the configuration file.
4. Execute Calibre and specify the name of the configuration file. For example:

```
% calibre -drc -hier -turbo -remotefile mtflex.config <rulefile>
```

The transcript outputs results similar to the following:

```
// $MYCOUNT resolved to 5
// Performing cluster launch of remote hosts: WAIT 600 COUNT 5 ...
```

Examples

You can optionally use Linux utilities to create the configuration file by using a template configuration file and the sed command. (Sed is a utility available in all Linux environments.) The template contains known placeholders that the sed command replaces with environment-

specific values. The placeholders begin with “@” because the template is coded that way to make them more visible. For example:

- Use an ASCII editor to create the following template configuration file named *template-mtflex.conf*:

```
# This is named template-mtflex.conf
LAUNCH CLUSTER MINCOUNT @job_size
REMOTE COMMAND @work_dir/qsub_script.csh \
    ARGUMENTS ["@work_dir/arguments"]
```

- Save and close the file.
- Execute the following sed command to substitute the @job_size and @work_dir placeholder values in *template-mtflex.conf* with the values of the \$JOB_SIZE and \$WORK_DIR environment variables:

```
sed -e "s^@job_size^$JOB_SIZE^g" -e "s^@work_dir^$WORK_DIR^g" \
    template-mtflex.conf > mtflex.conf
```

- The results of the sed command are output to the file *mtflex.conf*, using the values of 512 and /home/me for \$JOB_SIZE and \$WORK_DIR, respectively:

```
LAUNCH CLUSTER MINCOUNT 512
REMOTE COMMAND /home/me/qsub_script.csh ARGUMENTS ["/home/me/arguments"]
```

Chapter 16

Distributed Calibre: Executing and Monitoring

Some of the more common scenarios for running a distributed Calibre job involve using the multithreaded (MT) or Calibre MTflex capabilities, with or without hyperscaling mode. Monitoring utilities allow you to track how the job is distributed along with the performance of the local and remote hosts.

Refer to “[Distributed Calibre: Reference Dictionary](#)” on page 309 for statement, variable, and command line reference information.

Running MT Mode	277
Running Calibre MTflex in a Homogeneous Environment.....	279
Running Calibre MTflex in a Heterogeneous Environment	281
Running MT and Calibre MTflex with Hyperscaling	285
Monitoring Remote Processes.....	288
Monitoring Distributed Calibre	292
Executing Calibre Using IBM Spectrum LSF.....	294
Executing xCalibrate With Calibre MTflex Using IBM Spectrum LSF	299
Comparing Calibre MTflex Runs	301
Debugging Performance Issues Using a Calibre Log File	305

Running MT Mode

Multithreaded (MT) mode enables multithreaded parallel processing functionality, which distributes a design across multiple CPUs on the same primary host. This procedure describes how to run a Calibre job in MT mode using the -turbo and -turbo_litho command line argument.

To run a Calibre job in MT mode (except for Litho operations), you must specify the -turbo argument and an optional positive integer that specifies the number of CPUs to use during the processing. To run a Calibre job in MT mode for Litho operations, you must specify the -turbo_litho argument and an optional positive integer that specifies the number of CPUs to use during the processing of Litho operation.

If you do not specify an integer for the -turbo or -turbo_litho argument, then Calibre runs on the maximum number of CPUs available on the primary host for which you have licenses. If an integer value is specified that is greater than the maximum number of CPUs available, Calibre runs only on the number of available CPUs. You can specify the -turbo and -turbo_litho

operations concurrently in a single command line and the respective number of CPUs can vary between the two options.

You can use the -turbo_all argument to specify that Calibre must secure the number of CPUs specified by -turbo or -turbo_litho in order to run the job. The -turbo_all argument causes Calibre to exit with an error when the requested number of threads cannot be secured.

Prerequisites

- The CALIBRE_HOME environment variable must be set to the location of the Calibre software tree on the primary host.
- LM_LICENSE_FILE or MGLS_LICENSE_FILE must be set to the location of a license file.

Procedure

Execute Calibre to run in MT mode using one of the following methods:

If you want to...	Enter the following command:
Run Calibre nmDRC-H using the maximum number of CPUs available on the primary host for which you have licenses.	calibre -drc -hier -turbo rules
Run Calibre nmDRC-H using three CPUs on the primary host.	calibre -drc -hier -turbo 3 rules
Run Calibre nmDRC-H and Calibre RET on four CPUs.	calibre -drc -hier -turbo 4 -turbo_litho 4 rules
Run Calibre nmDRC-H on two CPUs and Calibre RET on four CPUs.	calibre -drc -hier -turbo 2 -turbo_litho 4 rules
Run Calibre nmDRC-H and Calibre RET on all available CPUs.	calibre -drc -hier -turbo -turbo_litho rules
Ensure you secure eight CPUs to run Calibre nmDRC-H or else exit.	calibre -drc -hier -turbo 8 -turbo_all rules

The occurrence of a line in your transcript similar to the following indicates that the hyper throttle is active:

```
// Initializing MT on HDB 0: 12
```

The turbo number is less than the total number of CPUs (12 in this example), so Calibre actively manages the number of active jobs rather than let a physical limit enforce it.

Running Calibre MTflex in a Homogeneous Environment

To run a design in Calibre MTflex mode, you must specify the -remote or -remotefile argument during Calibre tool invocation. In this procedure, you create a configuration file that specifies the primary host, remote hosts, and launch method for running a job in Calibre MTflex mode. You then invoke Calibre, using the -remotefile argument to specify the path to the configuration file.

Prerequisites

- A homogeneous environment where the Calibre software tree resides on the primary host and the CALIBRE_HOME definition for all remote hosts points to the same Calibre software tree.
- The CALIBRE_HOME environment variable must be set to the location of the Calibre software tree on the primary host.
- LM_LICENSE_FILE or MGLS_LICENSE_FILE must be set to the location of a license file.

Procedure

1. Create a configuration file named *mtflex.config* and specify the launch method (LAUNCH AUTOMATIC) and the names of the remote hosts to be used for running the Calibre job.

```
LAUNCH AUTOMATIC
# three remote hosts are launched automatically
REMOTE HOST machine1
REMOTE HOST machine2
REMOTE HOST machine3
```

2. Save and close the configuration file.
3. Invoke Calibre.

- Calibre nmDRC-H:

```
calibre -drc -hier -turbo -turbo_litho \
-remotefile mtflex.config rules
```

The -turbo and -turbo_litho arguments instruct Calibre nmDRC-H to use multithreaded parallel processing and use all available CPUs on the remote hosts.

- Calibre xACT 3D:

```
calibre -xact -3d -rcc -xcell xcell_list -turbo 24 \
-remotefile mtflex.config rules
```

The -turbo argument instructs Calibre xACT 3D to use 24 CPUs on the local and remote hosts.

All of the remote CPUs specified in the configuration file are connected to the primary host for the run.

Examples

An alternative method to defining remote hosts in a configuration file is to use the -remote argument to specify the remote hosts names in the command line when invoking Calibre. For example, enter the following command to invoke Calibre nmDRC-H and specify machine1 and machine2 as the remote hosts.

```
calibre -drc -hier -turbo -turbo_litho -remote machine1,machine2 \
rules
```

To invoke Calibre xACT 3D and specify machine1 and machine2 as the remote hosts, enter the following command:

```
calibre -xact -3d -rcc -xcell xcell_list -turbo 24 \
-remote machine1,machine2 rules
```

Running Calibre MTflex in a Heterogeneous Environment

A heterogeneous environment is a network of local and remote hosts that include different processors.

Refer to “[Mixed Operating Systems Environment](#)” on page 23 for more information on running Calibre in a heterogeneous environment.

There are some configuration file statements you can use to run Calibre MTflex in a heterogeneous environment.

Table 16-1. Options for Running Calibre MTflex in a Heterogeneous Environment

Options for Running Calibre MTflex in a Heterogeneous Environment	Description
Using the LAUNCH CLUSTER and REMOTE COMMAND statements	You invoke a Calibre tool and use the LAUNCH CLUSTER statement to set parameters for the connection of the local computer to the remote computers. You then use the REMOTE COMMAND statement to start a Calibre MTflex server on a remote host. See “ Using the LAUNCH CLUSTER and REMOTE COMMAND Statements ” on page 281.
Using the LAUNCH MANUAL statement and calibre -mtflex command	You invoke a Calibre tool and use the LAUNCH MANUAL statement to enable the manual execution of the Calibre MTflex functionality. This example shows how to perform six manual connections, with a mandatory minimum of two connections. See “ Using the LAUNCH MANUAL Statement and calibre -mtflex Command ” on page 283.

Using the LAUNCH CLUSTER and REMOTE COMMAND Statements

You can create a configuration file in which you define the LAUNCH CLUSTER and REMOTE COMMAND statements. The LAUNCH CLUSTER statement sets parameters for the connection of the primary host to the remote hosts, while the REMOTE COMMAND statement starts a Calibre MTflex server on a remote host. You then execute Calibre using the –remotefile argument to specify the path to a configuration file.

Prerequisites

- The CALIBRE_HOME environment variable must be set to the location of a Calibre software tree. This software tree must be accessible by the remote hosts and the same version of Calibre must be used throughout the network.
- The LM_LICENSE_FILE or MGLS_LICENSE_FILE must be set to the location of a license file.
- Ability to run rsh or ssh (required for RHEL/CentOS 8 operating system) on all remote hosts. The *launch_remote.sh* script as described in the procedure uses the standard rsh OS utility for executing processes on the remote shell. Any other remote shell utility, such as ssh, may be substituted for rsh.
- A heterogeneous environment where the Calibre MTflex configuration file is used to communicate with the primary host to identify the path to the appropriate CALIBRE_HOME tree.

Procedure

1. Create a file named *args* and add the names of your remote hosts. For example:

```
node91
node92
node93
node94
node95
node90
...
```

2. Create a file named *launch_remote.sh* that includes the following:

```
#!/bin/sh
n=0
for node in `cat $1`
do
    rsh $node $CALIBRE_HOME/bin/rcalibre "/scratch1" 1 \
        $CALIBRE_HOME $VCO_HOME -mtflex $CALIBRE_REMOTE_CONNECTION &
    rsh $node $CALIBRE_HOME/bin/rcalibre "/scratch1" 1 \
        $CALIBRE_HOME $VCO_HOME -mtflex $CALIBRE_REMOTE_CONNECTION &
    n=`expr $n + 2`;
    if [ $n -eq $CALIBRE_REMOTE_COUNT ]
    then break;
    fi
done
```

This script does the following:

- Outputs the contents of the *args* file from Step 1 as the argument list for the loop.
- Sets the CALIBRE_HOME environment variable to \$VCO_HOME for each CPU. The variable \$VCO_HOME needs to be defined in the environment.
- Specifies the log file directory as */scratch1*. This assumes each remote host has a mounted partition named “*/scratch1*”.

- At the end of the loop, the number of connections made by the script are counted and compared against the values specified by the LAUNCH CLUSTER statement (Step 3) for maximum and minimum connections. Execution of the **REMOTE COMMAND** assigns values to the CALIBRE_REMOTE_COUNT for the child process.
 - Uses the values assigned to the CALIBRE_REMOTE_CONNECTION variable to set and communicate the port and process ID for each individual connection between the primary and remote hosts. Execution of the **REMOTE COMMAND** assigns a value to the CALIBRE_REMOTE_CONNECTION variable for the child process.
3. Create a Calibre MTflex configuration file named *my_config_file* and include the LAUNCH CLUSTER and REMOTE COMMAND statements. The COUNT argument for the LAUNCH CLUSTER statement specifies the primary host should expect to connect to a total of six CPUs on the remote hosts, and the MINCOUNT argument specifies the minimum number of remote CPU connections to expect before the run fails. The *launch_remote.sh* file is discussed in Step 2; the *args* file is discussed in Step 1.
- ```
LAUNCH CLUSTER COUNT 6 MINCOUNT 2
// script must have execution privileges
// script and argument list must be accessible from primary host
REMOTE COMMAND /user/jsmith/bin/launch_remote.sh \
 ARGUMENTS ["/user/jsmith/remote/args"]
```
4. Invoke Calibre nmDRC-H, using the -remotefile argument to specify the path to the Calibre MTflex configuration file.

```
calibre -drc -hier -turbo -remotefile my_config_file rules
```

The Calibre command line -remotefile argument is used to specify a path to a configuration file that includes information about the primary and remote hosts, how to partition the run, and the launch method. This argument is required when running Calibre MTflex in a heterogeneous environment.

## Using the LAUNCH MANUAL Statement and **calibre -mtflex** Command

You can create a configuration file that includes a LAUNCH MANUAL statement. You then execute the Calibre job and use the **calibre -mtflex** command to manually connect each remote host to the primary host.

### Procedure

1. Create a configuration file named *mtflex.config* that includes the following LAUNCH MANUAL statement.

```
LAUNCH MANUAL WAIT 120 COUNT 6 MINCOUNT 2
```

This statement specifies:

- A wait time of 120 seconds that the primary host will wait for a connection to be made.
  - The primary host will connect to a total of six remote hosts.
  - The minimum number of remote hosts that must connect before the WAIT time expires.
2. Save and close the configuration file.
  3. Define the **CALIBRE\_MTFLEX\_LOCAL\_HOST\_DIR** environment variable, specifying the path to the directory where Calibre can store temporary log files created by the remote hosts.
    - For C shell:

```
setenv CALIBRE_MTFLEX_LOCAL_HOST_DIR "/scratch/logfiles/"
```
    - For Bourne/Korn shell:

```
CALIBRE_MTFLEX_LOCAL_HOST_DIR="/scratch/logfiles/"
export CALIBRE_MTFLEX_LOCAL_HOST_DIR
```

When you invoke **calibre -mtflex** on a remote host, a log file is created in your working directory (default location) at invocation time using the following syntax:

```
CalibreRemoteLog.<localhost_name>.<localhost_pid>.
<YYYY>_<MMDD>_<HHMM>.<remotehost_name>.<remotehost_pid>
```

This file is automatically deleted upon successful termination of the server. You can set the **CALIBRE\_MTFLEX\_LOCAL\_HOST\_DIR** environment variable to override this default location.

4. Invoke Calibre on the primary host:

```
calibre -drc -hier -turbo -remotefile mtflex.config rules
```

The primary host displays a message similar to:

```
// Calibre Remote Connection 172.16.11.22:12345
```

5. Login to the remote host and set **CALIBRE\_HOME**:

```
rlogin node120
setenv CALIBRE_HOME "calibre_path"
```

6. Connect the remote host to the primary host using the following command line syntax on the remote host:

```
calibre -mtflex <host_inet_address>:<host_port_number>
```

where:

`<host_inet_address>:<host_port_number>` is required and specifies the address and port number of the primary host. Calibre displays this information when you begin the run on the primary host.

You can then substitute the address and port number (displayed in step 3) to connect the remote host to the primary host, as follows:

```
calibre -mtflex 172.16.11.22:12345
```

If your remote host is a multi-processor machine, you must invoke `calibre -mtflex` once for each processor.

## Running MT and Calibre MTflex with Hyperscaling

Hyperscaling is a mode of running Calibre in either Multithreaded (MT) or Multithreaded Distributed (Calibre MTflex) environments. Hyperscaling mode enables the parallel execution of SVRF operations, allowing operations to execute concurrently thereby increasing the utilization of CPUs and improving scalability. Hyperscaling is not used for single-threaded environments.

As an internal optimization, hyperscaling flattens very small cells like contacts and vias. It also flattens top-level cells that it can identify after processing the [Layout Base Layer](#) (or [Layout Top Layer](#)) statements. Due to these optimizations, the hierarchical object counts that appear in the transcript may differ between a run that uses hyperscaling versus a traditional MT or Calibre MTflex run.

---

### Note

 When using hyperscaling, always use a correctly-specified [Layout Base Layer](#) statement in the rule file. ([Layout Top Layer](#) may be used, but Layout Base Layer is preferred in most situations.) Failure to do so often results in reduced performance, which is true for any Calibre nmDRC-H run.

---

### Prerequisites

- The CALIBRE\_HOME environment variable must be set to the location of a Calibre software tree that is accessible by the remote hosts.
- LM\_LICENSE\_FILE or MGLS\_LICENSE\_FILE must be set to the location of a license file.

### Procedure

1. Invoke Calibre with the -remotefile argument. For example, to invoke nmDRC-H:

```
calibre -drc -hier -turbo 4 -hyper -remotefile mtflex.cfg rules
```

Calibre automatically determines the appropriate hyperscaling number to assign based on the specified -turbo number.

2. Review the header section for information on the hyperscaling operations:

```
Running Calibre: calibre -remotefile mtflex.cfg -drc -hier -turbo 4
-hyper drc.header
...
// Starting time: Fri Jan 18 14:11:40 2013
//
...
// HYPERSCALING ENABLED with 4 pseudo HDBs.
// Initializing MTflex on pseudo HDB 1
// Calibre Remote Connection <host>:12345
// Launching Calibre Remote server on: node1234
// Waiting for launch of remote hosts
// Launching Calibre Remote server on: node1234
// Connected to CPU on remote host node1234, logfile = /tmp/
CalibreRemoteLog.<host>.1234.1111_2222_3333.node1234.4444
...
// Initializing MTflex on pseudo HDB 2
...
// Initializing MTflex on pseudo HDB 3
...
// Initializing MTflex on pseudo HDB 4
...
// MTflex initialization for pseudo HDBs completed.
// Running on 4 CPUs (pending licensing)
// List of connected remote hosts:
// REMOTE HOST node1234: np = 8, nv = 8, nc = 2, ns = 8
// MTflex CPU resources: 8 Local, 8/8 Remote
```

3. Review the Constructing Hierarchical Database section at the end of the Calibre Layout Data Input Module:

```

----- CALIBRE LAYOUT DATA INPUT MODULE -----

...
CONSTRUCTING HIERARCHICAL DATABASE
...
PSEUDO HDB 3 CONSTRUCTED. CPU TIME = 0 REAL TIME = 0 LVHEAP = 2/3/3
PSEUDO HDB 2 CONSTRUCTED. CPU TIME = 0 REAL TIME = 0 LVHEAP = 2/3/3
PSEUDO HDB 1 CONSTRUCTED. CPU TIME = 0 REAL TIME = 0 LVHEAP = 2/3/3
PSEUDO HDB 4 CONSTRUCTED. CPU TIME = 0 REAL TIME = 0 LVHEAP = 2/3/3
PSEUDO HDB SYSTEM CONSTRUCTION COMPLETE: 12 CPU TIME = 0
REAL TIME = 0 LVHEAP = 5/11/11 SHARED = 1/32
HIERARCHICAL DATABASE CONSTRUCTOR COMPLETE.
...
```

This information means the pseudo databases used by the CPUs are ready to store data.

This section also lists the following two algorithms that are only used with hyperscaling:

```
FLATTENING SELECTED VERY SMALL CELL PLACEMENTS
FLATTENING SELECTED TOP LAYER CELL PLACEMENTS
```

These are internal optimizations needed for a hyperscaling run. These algorithms affect object counts that appear in the run transcripts. Object counts can differ between a hyperscaling run and a non-hyperscaling run, which is an expected behavior.

4. Review the summary section located near the end of the transcript:

```
Cumulative ONE-LAYER BOOLEAN Time: CPU = 0/0 + 0/2 REAL = 0/3
Cumulative TWO-LAYER BOOLEAN Time: CPU = 0/0 + 0/0 REAL = 0/1
Cumulative POLYGON TOPOLOGICAL Time: CPU = 0/0 + 0/2 REAL = 0/5
Cumulative POLYGON MEASUREMENT Time: CPU = 0/0 + 0/0 REAL = 0/0
Cumulative SIZE Time: CPU = 0/1 + 0/2 REAL = 0/2
Cumulative EDGE TOPOLOGICAL Time: CPU = 0/0 + 0/4 REAL = 0/3
Cumulative EDGE MEASUREMENT Time: CPU = 0/0 + 0/0 REAL = 0/0
Cumulative ONE-LAYER DRC Time: CPU = 0/0 + 0/2 REAL = 0/2
Cumulative TWO-LAYER DRC Time: CPU = 0/0 + 0/1 REAL = 0/1
Cumulative MISCELLANEOUS Time: CPU = 0/0 + 0/1 REAL = 0/1
Cumulative CONNECT Time: CPU = 0/0 + 4/0 REAL = 1/0
Cumulative RDB Time: CPU = 0/0 + 0/0 REAL = 0/0
```

The REAL times are reported as follows:

primary host/remote hosts (aggregate)

5. The summary section also includes data similar to the following that reports statistics for both the primary and remote hosts:

```
HDB 0: CPU TIME = 0 + 4 LVHEAP = 18
HDB 1: CPU TIME = 0 + 2 LVHEAP = 7 RX = 3 TX = 1
HDB 2: CPU TIME = 1 + 5 LVHEAP = 8 RX = 6 TX = 1
HDB 3: CPU TIME = 1 + 6 LVHEAP = 7 RX = 6 TX = 1
HDB 4: CPU TIME = 1 + 5 LVHEAP = 7 RX = 7 TX = 1
HDB 0-4 TOTAL LVHEAP = 47
```

The primary host CPU numbers are reported before the + sign, and the remote host numbers are reported after it, as follows:

primary HDB 0/HDB 1-n + remote HDB 0/HDB 1-n

For CPU TIME, the format is primary host + remote hosts (aggregate).

6. At the end of the transcript, statistics are reported for the remote hosts by HDB and per CPU used for data processing. For example:

```
MTflex HDB 0
 REMOTE CPU node1: CPU TIME = 0, STATUS = OK(0), LVHEAP = 1/4/4,
 RX = 0, TX = 0
 REMOTE CPU node2: CPU TIME = 1, STATUS = OK(0), LVHEAP = 1/20/20,
 RX = 0, TX = 0
 REMOTE CPU node3: CPU TIME = 0, STATUS = OK(0), LVHEAP = 1/4/4,
 RX = 0, TX = 0
 ...
 ...
```

## Monitoring Remote Processes

The monitoring utilities allow you to monitor the load and memory usage on individual machines in a cluster, as well as the input and output characteristics between the primary and remote hosts. The following steps describe how to use these capabilities, in addition to providing some debugging methodologies while using Calibre in a cluster environment.

To use the monitoring capabilities, add the **MONITOR LOCAL** and **MONITOR REMOTE** statements to your configuration file. Each statement can be specified once, for the primary or remote hosts. If any monitoring is enabled on the remote hosts, the log files are not deleted when the process completes.

### Prerequisites

- The CALIBRE\_HOME environment variable must be set to the location of a Calibre software tree that is accessible by the remote hosts.
- LM\_LICENSE\_FILE or MGLS\_LICENSE\_FILE must be set to the location of a license file.
- You must perform steps 1-3 in “[Using the LAUNCH CLUSTER and REMOTE COMMAND Statements](#).”

### Procedure

1. Create a configuration file that includes the following statements:

```
MONITOR LOCAL CONNECT LOAD MEMORY IO
MONITOR REMOTE CONNECT MEMORY IO
```

The MONITOR LOCAL statement enables:

- Verbose output relating to the primary host and remote host connection process (CONNECT)
- Internal task monitoring on the primary host (LOAD)
- LVHEAP memory monitoring on the primary host (MEMORY)
- Network IO monitoring on the primary and remote hosts (IO)

2. Save and close the configuration file.
3. Execute the Calibre run. For example:

```
calibre -drc -hier -turbo -remotefile mtflex.config rules
```

4. Review the transcript results. The CONNECT option enables more verbose output relating to the primary and remote connections. This option is useful for debugging connection problems. Following is a sample output of a remote connection with this option enabled. The report shows the version used and the exact Calibre invocation on the remote host.

```
// Connecting to CPU on remote host node19...
Creating child process to invoke remote process
Child process executing following command:
/usr/bin/rsh node19 -n /cal_home/bin/rcalibre /tmp 1 CALIBRE_HOME
 /cal_home -mtflex <host>:12345
Remote process invocation started successfully
Remote process invocation completed, now waiting for socket
 connection
Done.
VERIFY: remote calibre_version = v2019.1_0.0
```

When monitoring is not enabled, only the first line is displayed in the transcript.

When monitoring is enabled and a connection problem is identified, you should perform the following tasks to resolve the issue:

- Ensure you can specify the remote host in an rsh or ssh (required for the RHEL/CentOS operating system) command on the primary host.
  - Make sure rcalibre can be invoked on the remote host using the path specified for the rcalibre command.
  - Verify the specified TCP/IP port (58422 in this example) is working correctly.
5. The LOAD option enables internal task monitoring of Calibre MTflex operations on the primary host. The output includes information relating to the internal thread management and remote operation behavior. The monitoring information is enclosed in angle brackets (<>) and contains the string “MON” at the beginning of each line. For

details on the output, refer to the “[MONITOR LOCAL LOAD Transcript](#)” reference page. The following is a sample output generated by the LOAD option:

```
drop::TMP<3> = reference_layer INTERACT check_layer != 1

<MON:B 3219 O 14>
<MON:S 3219>
<MON:L 3224 T 116 0 Q 1678 0 R 116>
<MON:L 3229 T 116 0 Q 2015 0 R 113>
<MON:L 3234 T 116 0 Q 1827 0 R 116>
<MON:L 3239 T 116 0 Q 1497 0 R 116>
<MON:L 3244 T 116 0 Q 1053 0 R 116>
<MON:L 3249 T 116 0 Q 1147 0 R 116>
...
<MON:F 3887 W 668 L 235 R 9410>
<MON:E 3887 O 14>
drop::TMP<3> (HIER-FMF TYP=1 CFG=0 HGC=0 FGC=0 HEC=0 FEC=0 VHC=F
drop::TMP<3> (HIER-FMF TYP=1 CFG=0 HGC=0 FGC=0 HEC=0 FEC=0 VHC=F
VPC=F)
CPU TIME = 235 + 9410 REAL TIME = 668 LVHEAP = 2885/10598/10599
OPS COMPLETE = 7 OF 32
```

6. The MEMORY option enables the reporting of LVHEAP memory on the primary or remote nodes. Following is an example of the transcript generated by the Boolean NOT operation running on eight remote nodes:

```
E = C NOT B

<MON:B 3869 O 12>
<MON:S 3870>
<MON:L 3931 T 8 0 Q 606 0 R 8>
<MON:M 3932 H 3228 3853 3853>
<MON:L 3995 T 8 0 Q 400 0 R 8>
<MON:M 3999 H 3390 3853 3853>
<MON:L 4055 T 8 0 Q 109 0 R 8>
<MON:M 4056 H 3517 3853 3853>

<MON:L 4117 T 8 0 Q 13 0 R 8>
<MON:M 4119 H 3728 3853 3853>
<MON:F 4193 W 323 L 6 R 2072>
<MON:E 4193 O 12>
E (HIER TYP=1 CFG=1 HGC=14991864 FGC=99064435 HEC=1002467482
FEC=3500465063 VHC=F VPC=F)
CPU TIME = 6 + 2072 REAL TIME = 324 LVHEAP = 3843/3867/3867
OPS COMPLETE = 9 OF 29
```

The monitoring information is enclosed in angle brackets and the format is as follows:

```
<MON:M timestamp H n1 n2 n3>
```

The *n1*, *n2*, and *n3* values represent the currently used memory, the currently allocated memory, and the maximum allocated memory in megabytes (MB). This is similar to the LVHEAP information provided at the end of each operation.

7. The IO option enables network input/output monitoring. When enabled for the primary host, Calibre performs a series of tests (read, write, and latency) on each remote CPU after the connection is completed. Following is an example of the results generated by this option:

```
Performing sequential IO characterization to remote CPU: node2
Socket Info: RCVBUF = 87380, SNDBUF = 16384, MAXSEG = 1460,
NODELAY = 1
WRITE 0: 10000 16K blocks, TIME = 1.38838, Mb/s = 944.064
WRITE 1: 10000 16K blocks, TIME = 1.38508, Mb/s = 946.312
WRITE 2: 10000 16K blocks, TIME = 1.3879, Mb/s = 944.392
READ 0: 10000 16K blocks, TIME = 1.38245, Mb/s = 948.115
READ 1: 10000 16K blocks, TIME = 1.38118, Mb/s = 948.989
READ 2: 10000 16K blocks, TIME = 1.38107, Mb/s = 949.061
PACKETS 0: 10000 16 byte packets, TIME = 1.29852,
packets/s = 7701.07
PACKETS 1: 10000 16 byte packets, TIME = 1.26893,
packets/s = 7880.69
PACKETS 2: 10000 16 byte packets, TIME = 1.25407,
packets/s = 7974.06
```

The read and write performance tests are composed by sending 10,000 16K blocks three times to the remote CPU. The reported speed should be close to the actual network speed. For this example, the network is a 1 Gb/sec network and the average read and write speed is 946.82 Mbit/s.

The latency of the connection is measured by sending 10,000 16-byte packets three times to the remote CPU. For the remote, the IO monitoring option reports the number of bytes read and written by the remote. The format that appears in the remote host log file is:

<MON:I *timestamp* R *n1* W *n2*>

The *n1* and *n2* value reports the IO counts. *n1* is the bytes (in millions) read by the remote and *n2* is the number of bytes (in millions) written by the remote.

## Monitoring Distributed Calibre

---

The MONITOR SYSTEM command when used with the RRDGRAPH argument reports system level information using RRD graph functions. The MONITOR SYSTEM command generates a script named *CalibreRRDGraph.sh* in the output directory. This script reads RRD files produced and updated by Calibre at every time interval and writes out PNG files.

The PNG files can be viewed in a web browser or any supporting application. Each file is a chart of load averages for a local or remote host as shown in [Figure 17-1](#) and [Figure 17-2](#). The following procedures describe options for viewing PNG files:

|                                                     |            |
|-----------------------------------------------------|------------|
| <b>Viewing PNG Files Using a Web Server .....</b>   | <b>292</b> |
| <b>Viewing PNG Files Without a Web Server .....</b> | <b>293</b> |

## Viewing PNG Files Using a Web Server

In this procedure, you use a web server to view the PNG files produced by the *CalibreRRDGraph.sh* script.

### Prerequisites

- RRDtool must be set up correctly.
- For Procedure 1, a web server must be installed and configured. For Linux platforms, the most commonly used web server is Apache which can be downloaded from:

<http://www.apache.org>

### Procedure

1. Create a configuration file and include the following MONITOR SYSTEM commands:

```
MONITOR SYSTEM rrdgraph width 700
MONITOR SYSTEM rrdgraph height 300 timespan 10hour
MONITOR SYSTEM interval 120
MONITOR SYSTEM report rrd /usr/local/rrdtool/bin/rrdtool rrdscript
MONITOR SYSTEM outdir /user/test
```

2. Create a link from the MONITOR SYSTEM output directory (*/user/test*) to the web server. For example, on a Linux platform:

```
cd apache-server-dir/htdocs
ln -s /user/test .
```

3. Copy the CGI file (available in the \$CALIBRE\_HOME tree) to the MONITOR SYSTEM output directory:

```
cd /user/test
cp $CALIBRE_HOME/shared/pkgs/icv/tools/misc/cms.cgi .
```

4. Open a web browser and enter the following

<http://<web-server-host-machine>/user/test/cms.cgi>

5. To view updated contents, you must reload the page.

## Viewing PNG Files Without a Web Server

In this procedure, you view the PNG files produced by the *CalibreRRDGraph.sh* script without using a web server.

### Prerequisites

- RRDtool must be set up correctly.

### Procedure

1. Create a file named *cms.sh* in the *monitor-system-outdir* directory and enter the following:

```
#!/bin/sh
echo "<html> <head> <title>Remotes</title> </head> <body> >
 cms.html
source CalibreRRDGraph.sh >> cms.html
echo "</body> </html>" >> cms.html
```

2. Execute the *cms.sh* file:

```
source cms.sh
```

3. Use a web browser to view the *cms.html* web page generated in step 2.
4. To view the updated network status, you must repeat step 2 and reload the *cms.html* page.

## Executing Calibre Using IBM Spectrum LSF

IBM Spectrum LSF is software for managing workload processing for compute and data-intensive applications. With IBM Spectrum LSF, you can schedule and track the completion of batch workloads across a distributed environment.

The following procedures provide examples using the IBM Spectrum LSF bsub command to submit and run a Calibre job in different environments:

|                                                                                                |            |
|------------------------------------------------------------------------------------------------|------------|
| <b>Executing Calibre Through IBM Spectrum LSF in MT Mode Using SMPs on the Same Host .....</b> | <b>294</b> |
| <b>Executing Calibre Through IBM Spectrum LSF in a Calibre MTflex Environment..</b>            | <b>295</b> |
| <b>Executing Calibre Through IBM Spectrum LSF for a Single CPU Run .....</b>                   | <b>297</b> |
| <b>Executing Calibre xRC Through IBM Spectrum LSF Using bsub Commands .....</b>                | <b>298</b> |

## Executing Calibre Through IBM Spectrum LSF in MT Mode Using SMPs on the Same Host

This procedure illustrates how to execute Calibre through IBM Spectrum LSF in Multithreaded (MT) mode using Shared Memory Multi-Processors (SMP) on the same host. In this situation, you instruct IBM Spectrum LSF to execute Calibre nmDRC-H on eight CPUs on the same host.

### Prerequisites

- IBM Spectrum LSF must be installed and configured for your network.
- The CALIBRE\_HOME environment variable must be set to the location of a Calibre software tree that is accessible by the remote hosts.
- LM\_LICENSE\_FILE or MGLS\_LICENSE\_FILE must be set to the location of a license file.

### Procedure

1. Create a shell script named *run\_calibre*. For example:

```
#!/bin/csh
setenv CALIBRE_HOME /calibre_tree/linux_tree
Execute Calibre
$CALIBRE_HOME/bin/calibre $*
```

2. Execute the script using the bsub command:

```
bsub -o drc.log -n 8 -R "span[hosts=1]
select[type==LINUX && maxmem>2000]"
run_calibre -drc -hier -turbo -hyper rule.file
```

Table 16-2 provides a description of the bsub commands used in this example.

**Table 16-2. bsub Command Line Arguments**

| bsub Arguments | Description                                                     |
|----------------|-----------------------------------------------------------------|
| -o drc.log     | Redirects the Calibre standard output to drc.log.               |
| -n 8           | Instructs LSF to select any 8 CPUs for running Calibre.         |
| hosts=1        | Specifies that the 8 CPUs must be on the same host.             |
| type==LINUX    | The selected host must be a Linux platform                      |
| maxmem>2000    | The host must have at least two Gbit of memory                  |
| run_calibre    | A shell script that sets the Calibre tree and executes Calibre. |

## Executing Calibre Through IBM Spectrum LSF in a Calibre MTflex Environment

In Calibre MTflex mode, Calibre runs in a “primary-remote” mode, where the primary host manages input data and distributes data to each host and each remote host gets data from the primary host, runs a specific SVRF operation, and returns the results back to the primary host.

It is required in this mode to submit the Calibre job first to a primary host with multiple CPUs. This job then submits IBM Spectrum LSF jobs for running each remote host. In the shell, the primary host job has a configuration file that specifies a hostname for each remote.

In this procedure, the script and configuration files you create depend on how your IBM Spectrum LSF grid is configured. If the IBM Spectrum LSF parameter JOB\_ACCEPT\_INTERVAL is set to 0, you can use job arrays and IBM Spectrum LSF will put as many remotes as possible on the best host for the job, then get more remotes from other hosts, if needed. If JOB\_ACCEPT\_INTERVAL is set to 1, a job array is not ideal for licensing and performance reasons.

---

### Note

 The default for JOB\_ACCEPT\_INTERVAL is 1.

---

### Prerequisites

- IBM Spectrum LSF must be installed and configured for your network.
- The CALIBRE\_HOME environment variable must be set to the location of a Calibre software tree that is accessible by the remote hosts.

- LM\_LICENSE\_FILE or MGLS\_LICENSE\_FILE must be set to the location of a license file.
- To run this procedure with the RHEL/CentOS 8 operating system, you must substitute rsh with ssh.

## Procedure

1. Create an executable shell script named *run\_cluster*. The script you create depends on how your IBM Spectrum LSF grid is configured.

- If LSF\_JOB\_ACCEPT\_INTERVAL is set to 1:

```
#!/bin/csh
lsgrun -p -m "$LSB_HOSTS" calibre -mtflex \
$CALIBRE_REMOTE_CONNECTION -f
```

In this script, the lsgrun command is used to execute a task on a set of hosts defined by the LSB\_HOSTS variable, which is set by IBM Spectrum LSF. The CALIBRE\_REMOTE\_CONNECTION variable defines a host and port ID for the connection between the primary and remote host.

- If LSF\_JOB\_ACCEPT\_INTERVAL is set to 0:

```
#!/bin/csh
setenv CALIBRE_HOME /calibre_tree/linux_tree \
$CALIBRE_HOME/bin/rcalibre "/tmp" 1 \
-mtflex $CALIBRE_REMOTE_CONNECTION -f
rsh $node $CALIBRE_HOME/bin/rcalibre "/scratch1" 1 \
CALIBRE_HOME $VCO_HOME -mtflex $CALIBRE_REMOTE_CONNECTION &
```

This script sets the value of CALIBRE\_HOME and uses the rcalibre command to connect to the host machine using the IP and port specified by the CALIBRE\_REMOTE\_CONNECTION environment variable.

---

### Note

 For the RHEL/CentOS 8 operating system, you must use ssh instead of rsh.

---

2. Create a configuration file named *config\_lsf*. The type of configuration file you create depends on how your IBM Spectrum LSF grid is configured.

- If LSF\_JOB\_ACCEPT\_INTERVAL is set to 1:

```
LAUNCH CLUSTER WAIT 120 COUNT 8 MINCOUNT 6
REMOTE COMMAND bsub ARGUMENTS [-n 8 -J "MTFlexJob" \
-o MTFlexJob_%J.log -R "select[type==LINUX64 && hostname!='%H']" \
/<run_cluster_path>/run_cluster]
```

- If LSF\_JOB\_ACCEPT\_INTERVAL is set to 0:

```
LAUNCH CLUSTER MINCOUNT 8
REMOTE COMMAND bsub ARGUMENTS [-J "JobArray[1-8]" \
-R "span[hosts=1] select[hostname!='%H' && type==any]" \
/<run_cluster_path>/run_cluster"]
```

The LAUNCH CLUSTER statement sets the parameters for connection of the primary host to the remote hosts. The MINCOUNT keyword causes the Calibre run to fail if the number of remote connections is less than the specified number of minimum remote connections after the WAIT time expires.

The REMOTE COMMAND statement specifies information for launching remote hosts. The first argument to the REMOTE COMMAND specifies the name of the Linux command or utility to execute. In this example, the REMOTE COMMAND executes the bsub command and the following arguments inside the brackets ([ ]):

- -J is an option to the bsub command and is used to specify the job name and to repeat the execution of the bsub command if an array is specified. The value [1-8] causes the bsub command to execute eight times. For a successful Calibre run, the value specified for MINCOUNT must be the same as the number of executed LSF jobs.
- -R is an option to the bsub command used to specify resource requirements for the job. In this example, the filter contains the following filter criteria:
  - span[hosts=1] specifies that the CPUs used for the job must be on the same host.
  - select[hostname!=%H && type==any] specifies none of the machine processors on the primary host be used as a remote processor and the job can run on any remote hosts. The %H represents the hostname portion of the Calibre remote connection.
- *<run\_cluster\_path>/run\_cluster* specifies the path to the script created in Step 1.

3. Execute the Calibre run in a distributed, multithreaded environment using IBM Spectrum LSF as follows:

- ```
bsub -o drc.log -n 4 $CALIBRE_HOME/bin/calibre -drc \
      -hier -turbo -remotefile config_lsf rule.file
```
- -o redirects the Calibre standard output to the file named *drc.log*.
 - -n instructs LSF to select any four CPUs for running Calibre.

Executing Calibre Through IBM Spectrum LSF for a Single CPU Run

In this procedure, you execute Calibre through IBM Spectrum LSF for a single CPU run.

The command for executing Calibre in a shell (outside of IBM Platform LSF) is:

```
$CALIBRE_HOME/bin/calibre -drc -hier rule.file > drc.log
```

Prerequisites

- IBM Spectrum LSF must be installed and configured for your network.

- The CALIBRE_HOME environment variable must be set to the location of a Calibre software tree that is accessible by the remote hosts.
- LM_LICENSE_FILE or MGLS_LICENSE_FILE must be set to the location of a license file.

Procedure

1. Create a shell script named *run_calibre* to set the CALIBRE_HOME variable and execute Calibre. For example:

```
#!/bin/csh
setenv CALIBRE_HOME /calibre_tree/linux_tree
# Execute Calibre
$CALIBRE_HOME/bin/calibre $*
```

2. Execute the script using the bsub command:

```
bsub -o drc.log run_calibre -drc -hier rule.file
```

Executing Calibre xRC Through IBM Spectrum LSF Using bsub Commands

In this procedure, you use bsub commands in IBM Spectrum LSF to execute Calibre xRC, PDB, and FMT.

Prerequisites

- IBM Spectrum LSF must be installed and configured for your network.
- The CALIBRE_HOME environment variable must be set to the location of a Calibre software tree that is accessible by the remote hosts.
- LM_LICENSE_FILE or MGLS_LICENSE_FILE must be set to the location of a license file.

Procedure

1. Create a file named *script* that includes the following commands:

```
#!/bin/sh
bsub $1 >&! k1.log
bsub $2 >&! k2.log
bsub $3 >&! k3.log
```

2. Enter the following in Queue command:

```
./script %k1 %k2 %k3
```

3. Run PEX to execute xRC, PDB, and FMT.

Executing xCalibrate With Calibre MTflex Using IBM Spectrum LSF

You can invoke the xCalibrate tool and use the REMOTE COMMAND statement to start a Calibre MTflex server on a remote host. This example shows how to use the LAUNCH CLUSTER statement to connect a primary host to one or more remote hosts and how to run xCalibrate using the -remotefile argument.

Prerequisites

- The CALIBRE_HOME environment variable must be set to the location of a Calibre software tree. This software tree must be accessible by the remote hosts and the same version of Calibre must be used throughout the network.
- The LM_LICENSE_FILE or MGLS_LICENSE_FILE environment variable must be set to the location of a license file.
- The ability to run ssh (required for RHEL/CentOS 8 operating system) on all remote hosts.
- A heterogeneous environment where the Calibre MTflex configuration file is used to communicate with the primary host to identify the path to the appropriate CALIBRE_HOME tree.
- IBM Spectrum LSF must be installed and configured for your network. Refer to “[IBM Spectrum LSF Commands](#)” on page 253 for information on some of the commonly used LSF commands.

Procedure

1. Create an executable shell script file named *launch_remote.sh* that calls the *rxcalibrate* utility. For example:

```
#!/bin/bash
echo "Remote Hosts selected by the bsub command: $LSB_HOSTS"
echo "Need to connect them to the master referenced by:
$CALIBRE_REMOTE_CONNECTION"
echo "Checking $MGC_HOME is set like the master: $MGC_HOME"
for host in $LSB_HOSTS
do
    #Usage: rxcalibrate <rundir> <env-var-count> <env_var_1>
    #<env_var_value> <env_var_2> <env_var_value> ... [calibre_switches]
    #${MGC_HOME}/bin/rxcalibrate "/tmp" 1 $MGC_HOME ${MGC_HOME} -mtflex
    $CALIBRE_REMOTE_CONNECTION -64 -f &
done
```

This script uses the rxcalibrate utility to connect to the host machine using the IP and port specified by the CALIBRE_REMOTE_CONNECTION environment variable. The LSB_HOSTS variable defines the set of hosts for use by IBM Spectrum LSF. The CALIBRE_REMOTE_CONNECTION variable defines a host and port ID for the connection between the primary and remote host.

2. Create a Calibre MTflex configuration file called *remote_config* and specify the LAUNCH CLUSTER and REMOTE COMMAND statements. For example:

```
LAUNCH CLUSTER WAIT 60 COUNT 20 MINCOUNT 20
REMOTE COMMAND bsub
    ARGUMENTS [ -J "remotes[1-20]" -R "select[hostname!='%H' &&
defined(linux) && (ostype=='CENT6.5' || ostype=='CENT6.6' ||
ostype=='RHEL6.6')]" -o remotes.log ./launch_remote.sh ]
```

In this example, the COUNT keyword for the LAUNCH CLUSTER statement specifies the number of remote connections that must be established before the run begins. The MINCOUNT keyword specifies the minimum number of remote connections that must be established before the specified WAIT time expires in order for the Calibre run not to fail.

The REMOTE COMMAND executes the *launch_remote.sh* shell script.

3. Invoke xCalibrate using the -remotefile argument to specify the path to the Calibre MTflex configuration file you created in step 2.

```
xcalibrate -exec -remotefile remote_config test.mipt
```

The -remotefile command line argument specifies a path to the configuration file that includes information about the primary and remote hosts, how to partition the run, and the launch method.

Results

In this procedure, the job completes successfully on 20 cores as requested.

Related Topics

[LAUNCH CLUSTER](#)

[REMOTE COMMAND](#)

[CALIBRE_REMOTE_CONNECTION](#)

[rxcalibrate](#)

[-remotefile](#)

Comparing Calibre MTflex Runs

You will likely notice differences in the reporting of progress and performance when comparing Calibre transcripts from the same run performed twice, once using Calibre MTflex and once using Calibre MTflex with hyperscaling. These differences result from the hyperscaling optimizations, and do not indicate a variation in the expected outcome of the operations.

You can use the following procedures to understand the transcript information that is generated by a Calibre MTflex run:

Running Calibre MTflex Without Hyperscaling [301](#)

Running Calibre MTflex With Hyperscaling [302](#)

Running Calibre MTflex Without Hyperscaling

When you run a Calibre job in Calibre MTflex mode, your transcript contains additional information regarding the Calibre MTflex operations.

Prerequisites

- None. For comparison purposes, it is recommended that you have a transcript from a Calibre job that was run in Calibre MTflex mode without hyperscaling.

Procedure

1. Calibre echoes the configuration file to the run transcript before rule file compilation.
2. The number of remote connections is most commonly determined by the commands in your configuration file. Here is an example of a typical transcript describing the CPUs used for the Calibre application during a Calibre MTflex run:

```
// Calibre Remote Connection 172.16.11.22:12345
// Performing automatic launch of remote hosts
// Connected to CPU on remote host my_machine, logfile = ...
...
// List of connected remote cpus:
// REMOTE CPU my_machine: pid = 21305, os = "Red Hat Enterprise
// Linux Workstation release 6.4 (<primary_host>)"
...
// Running on 8 CPUs
// List of connected remote hosts:
// REMOTE HOST my_machine: np = 4, nc = 2, ns = 4
...
// MTflex CPU resources: 16 Local, 8/8 Remote
```

This example shows the OS information for the connected remote hosts, in addition to resource usage for a command line that contained -turbo 8 and a reference to 8 remote CPUs. The number of licenses Calibre requires is dependent on the number of CPUs shown in the “Running on” line.

Another scenario is when you specify -turbo with no argument. This instructs Calibre to use the maximum number of available CPUs for which you have licenses. In this case, the transcript shows the following for a run where there are two CPUs on the host machine and 12 CPUs on the available remote machines, but the job is only running on 12 CPUs due to license limitations:

```
// Calibre Remote Connection 172.16.11.22:12345
// Performing automatic launch of remote hosts
// Connecting to CPU on remote host my_machine... Done.
...
// Running on 12 CPUs
// MTflex CPU resources: 2 Local, 12 Remote
```

3. Calibre reports usage statistics for CPU time and real time for each Calibre operation, in the Calibre Layout Data Input Module, and at the end of the transcript. Calibre also reports the number of processors used for each Calibre tool run from the rule file.

```
--- TOTAL CPU TIME = 609 + 3822 REAL TIME = 569
REMOTE CPU node1: CPU TIME = 42, STATUS = OK, LVHEAP = 0/1/1,
    RX = 0, TX = 0
...
--- PROCESSOR COUNT = 4
--- PROCESSOR COUNT FOR <TOOL NAME> = 4
--- PROCESSOR COUNT FOR <TOOL NAME> = 2
```

- Total CPU Time reports the cumulative number of CPU seconds for the primary host (first number) and the remote hosts (second number) that remained connected until then end of the run.
- The CPU-specific statistics RX and TX report, in millions, the number of received and transmitted bytes, respectively, and LVHEAP reports data on memory usage.

Running Calibre MTflex With Hyperscaling

The following is another example of the transcript results generated as a result of running a Calibre job in Calibre MTflex mode with hyperscaling.

Prerequisites

- None. For comparison purposes, it is recommended that you have a transcript from a Calibre job that was run in Calibre MTflex mode with hyperscaling.

Procedure

1. Total number of pseudo HDBs created (4 in this example):

```
// 64 bit virtual addressing enabled
// Starting time: Sun May 28 17:34:18 2006
// HYPERSCALING ENABLED with 4 pseudo HDBs.

// Reading remote host configuration file: ./remote.config
LAUNCH AUTOMATIC name eagle
```

2. Connection made to each processor by HDB0:

```
// Calibre Remote Connection eagle:34674
// Performing automatic launch of remote hosts
// Connecting to CPU on remote host condor17 ... Done.
// Connecting to CPU on remote host condor17 ... Done.
// Connecting to CPU on remote host condor18 ... Done.
// Connecting to CPU on remote host condor18 ... Done.
// ...
```

3. Connection made for each pseudo HDB:

```
// Initializing MTflex on pseudo HDB 1
// Connecting to CPU on remote host condor17 ... Done.
// Connecting to CPU on remote host condor17 ... Done.
// ...
// Initializing MTflex on pseudo HDB 4// ...
```

4. Number of remote CPUs being used (the values shown, 32 in this example, for the number of CPUs and Calibre MTflex CPU resources should be equal):

```
// ...
// MTflex initialization for pseudo HDBs completed.
// Running on 32 CPUs
// MTflex CPU resources: 8 Local, 32 Remote//
// Initializing MT on pseudo HDB 1
// Initializing MT on pseudo HDB 2
// Initializing MT on pseudo HDB 3
// ...
// Initializing MT on pseudo HDB n
--- CALIBRE:::DRC-H - Sun May 28 17:36:23 2006
```

5. HDB used to execute the current operation and the status of the current memory for HDB1 on the primary host CPU:

```
PX6:::<1> = PX6:::M6 INTERACT PX6:::M6 >= 2
-----
Operation EXECUTING on HDB 1 (T44001 S0 E8 H0 A169 C5)
PX6:::<1> = PX6:::M6 INTERACT PX6:::M6 >= 2
-----
PX6:::<1> (HIER TYP=1 CFG=1 HGC=0 FGC=0 HEC=0 FEC=0 VHC=F VPC=F)
CPU TIME = 0 + 0 REAL TIME = 1 LVHEAP = 2074/3620/3633 DBLOCK =
    724/6400 OPS COMPLETE = 2913 OF 3017
Operation COMPLETED on HDB 1 LVHEAP = 273/1452/1453
```

6. Summary section reports statistics for both the primary and remote hosts. The semantics of this section are similar to the statistics in an MT hyperscaling run (except MT does not include remote host statistics).

```
Cumulative ONE-LAYER BOOLEAN Time: CPU = 0/0 + 0/1 REAL = 0/2
Cumulative TWO-LAYER BOOLEAN Time: CPU = 0/0 + 0/0 REAL = 0/1
Cumulative POLYGON TOPOLOGICAL Time: CPU = 0/0 + 0/2 REAL = 0/2
Cumulative POLYGON MEASUREMENT Time: CPU = 0/0 + 0/0 REAL = 0/0
Cumulative SIZE Time: CPU = 0/1 + 0/2 REAL = 0/2
Cumulative EDGE TOPOLOGICAL Time: CPU = 0/0 + 0/11 REAL = 0/4
Cumulative EDGE MEASUREMENT Time: CPU = 0/0 + 0/0 REAL = 0/0
Cumulative ONE-LAYER DRC Time: CPU = 0/0 + 0/1 REAL = 0/0
Cumulative TWO-LAYER DRC Time: CPU = 0/0 + 0/1 REAL = 0/0
Cumulative MISCELLANEOUS Time: CPU = 0/0 + 0/0 REAL = 0/1
Cumulative CONNECT Time: CPU = 0/0 + 3/0 REAL = 1/0
Cumulative RDB Time: CPU = 0/0 + 0/0 REAL = 0/0
```

For the CPU times, the primary host numbers are reported before the + sign and the remote host numbers are reported after it as follows:

primary HDB 0/HDB 1-n + remote HDB 0/HDB 1-n

The REAL times are reported as follows:

primary host/remote hosts (aggregate)

7. Memory used on the primary host per HDB:

```
HDB 0: CPU TIME = 6956 + 31007 LVHEAP = 9217
HDB 1: CPU TIME = 3808 + 26421 LVHEAP = 2903
HDB 2: CPU TIME = 4244 + 15935 LVHEAP = 1062
HDB 3: CPU TIME = 3472 + 17284 LVHEAP = 1067
HDB 4: CPU TIME = 3212 + 32044 LVHEAP = 1056
HDB 0-4 TOTAL LVHEAP = 15305
```

For CPU TIME, the format is this: primary host + remote hosts (aggregate).

8. Total memory used on the primary host:

```
--- CALIBRE::DRC-H EXECUTIVE MODULE COMPLETED. CPU TIME = 21694 +
122693 REAL TIME = 13648
--- TOTAL RULECHECKS EXECUTED = 1272
--- TOTAL RESULTS GENERATED = 200244 (3063467)
--- DRC RESULTS DATABASE FILE = ./results/drc.db (ASCII)
--- CALIBRE::DRC-H COMPLETED - Wed Jun 7 14:17:48 2006
--- TOTAL CPU TIME = 23897 + 122693 REAL TIME = 15243
...
```

9. Total memory used on the remote hosts by HDB and per CPU used for data processing:

```
...
MTflex HDB 0
REMOTE CPU condor17: CPU TIME = 453, STATUS = OK(0), LVHEAP =
7/338/430, RX = 2553, TX = 422
REMOTE CPU condor17: CPU TIME = 416, STATUS = OK(0), LVHEAP =
7/344/433, RX = 2468, TX = 404
REMOTE CPU condor18: CPU TIME = 2039, STATUS = OK(0), LVHEAP =
7/1188/1279, RX = 2951, TX = 979
REMOTE CPU condor18: CPU TIME = 577, STATUS = OK(0), LVHEAP =
7/336/428, RX = 2585, TX = 516
...
MTflex HDB 1
REMOTE CPU condor17: CPU TIME = 424, STATUS = OK(0), LVHEAP =
5/132/132, RX = 1094, TX = 469
REMOTE CPU condor17: CPU TIME = 447, STATUS = OK(0), LVHEAP =
5/202/202, RX = 1270, TX = 635
REMOTE CPU condor18: CPU TIME = 460, STATUS = OK(0), LVHEAP =
5/98/98, RX = 1090, TX = 495
REMOTE CPU condor18: CPU TIME = 415, STATUS = OK(0), LVHEAP =
5/142/142, RX = 1072, TX = 441
...
MTflex HDB 2
REMOTE CPU condor17: CPU TIME = 309, STATUS = OK(0), LVHEAP =
5/82/82, RX = 644, TX = 318
REMOTE CPU condor17: CPU TIME = 186, STATUS = OK(0), LVHEAP =
5/68/68, RX = 557, TX = 244
REMOTE CPU condor18: CPU TIME = 205, STATUS = OK(0), LVHEAP =
5/60/60, RX = 597, TX = 271
REMOTE CPU condor18: CPU TIME = 178, STATUS = OK(0), LVHEAP =
5/88/88, RX = 545, TX = 255
...
MTflex HDB 3
...
```

10. Total processor count:

```
--- PROCESSOR COUNT = 32
--- SUMMARY REPORT FILE = ./reports/drc.sum
```

Debugging Performance Issues Using a Calibre Log File

The Calibre log file includes a considerable amount of information that can be helpful in debugging performance issues encountered during a Calibre run. The following procedure identifies specific items to look for in the log file that may help improve performance for a subsequent run.

Prerequisites

None. It is recommended that you have a log file from a Calibre run that you can refer to for comparison purposes.

Procedure

1. Review the header information in the log file.

```
Calibre version → // Calibre <version> <timestamp>
// Calibre Utility Library <version> <timestamp>
// Litho Libraries <version> <timestamp>
//
// Copyright Mentor Graphics Corporation 1996-2016
// All Rights Reserved.
// THIS WORK CONTAINS TRADE SECRET AND PROPRIETARY INFORMATION
// WHICH IS THE PROPERTY OF MENTOR GRAPHICS CORPORATION
// OR ITS LICENSORS AND IS SUBJECT TO LICENSE TERMS.
//
// Mentor Graphics software executing under <architecture>
//
Hardware configuration → // Running on <OS> <hostname> <version> <build>.
//
// Entries in /proc/meminfo:
...
// CPU Info: Cores = 16, SMT enabled with 16 additional virtual processors
// Max file descriptors: 65535
// 64 bit virtual addressing enabled
Command line → // Running <version>/pkgs/icv/pvt/calibre -drc -hier -turbo -hyper drc.control
// Process ID: 17707
//
// Starting time: <timestamp>
//
...
MONITOR UTILITY
MONITOR SYSTEM REPORT ASCII
// Calibre Remote Log tag = ...
// Calibre Remote Connection ...
// Performing cluster launch of remote hosts: WAIT 900 COUNT 4 MINCOUNT 4 PARALLEL
// Starting execution of remote commands
// Waiting for cluster launch of remote hosts
// Connected to CPU on remote host <host>, logfile = <filename>, time = 4
...
// List of connected remote cpus:
//   REMOTE CPU <node>; pid = <id>...
...
// Calibre Controller Connection ...

// HYPERSCALING ENABLED with # pseudo HDBs.

// Running on # CPUs (pending licensing)
...
// MTflex CPU resources: 16 Local, 4/4 Remote
// LITHO operations available for use on 4 CPUs (pending licensing)
//
```

Be sure to note the Calibre version, hardware configuration, and Calibre command line. If using remotes, you should confirm that all Calibre versions match.

2. Verify the final LVHEAP is within the machine's capacity.

```
...
HDB 0-4 TOTAL LVHEAP = 9386
...
```

In this example, the LVHEAP number is reporting the maximum memory requirement of the application.

The LVHEAP numbers, which appear with all derived layer statistics in the log file, report approximate current memory usage in MBytes for Calibre tools. The format of the memory usage report is:

LVHEAP = memory currently used / memory allocated / maximum memory allocated

3. Check whether the hardware is fully utilized by searching the log file for HDB and PROCESSOR COUNT.

```
HDB 0: CPU TIME = 3  LVHEAP = 44
HDB 1: CPU TIME = 6  LVHEAP = 24  RX = 6  TX = 1
HDB 2: CPU TIME = 6  LVHEAP = 14  RX = 6  TX = 1
HDB 3: CPU TIME = 6  LVHEAP = 24  RX = 4  TX = 1
HDB 4: CPU TIME = 6  LVHEAP = 26  RX = 7  TX = 1
HDB 0-4 TOTAL LVHEAP = 132
```



HDB usage

To save remote host log files, which are normally deleted after the run, set the **CALIBRE_MTFLEX_SAVE_LOGS** environment variable to a non-null character. For example:

```
setenv CALIBRE_MTFLEX_SAVE_LOGS 1
```

4. Search for the keyword SHARED.

```
HIERARCHICAL DATABASE CONSTRUCTOR COMPLETE.
CPU TIME = 1 + 0  REAL TIME = 1  LVHEAP = 5/11/11  SHARED = 1/32
```

SHARED data is a subset of the LVHEAP data and, of the LVHEAP memory used, represents the amount in the SHARED section that was shared with other HDB(s).

When the available RDS (Remote Data Server) memory is filled then overflow data is stored on the local (server) host. Shared memory is undesirable when running RDS and Hyper Remote. Reducing memory by using RDS is preferred over using shared memory. As a result, if you are running RDS and getting significant shared memory, it can indicate you are not using enough memory from the remotes. Getting large amounts of shared memory while not using RDS can indicate that you will see an improvement by using RDS.

5. Review the CELL AND PLACEMENT SUMMARY.

CELL AND PLACEMENT SUMMARY			
CELL TYPE	CELLS	PLACEMENTS	FLAT PLACEMENTS
USER	2328	3095277	75000240
VERY SMALL	0	0	0
TOP LAYER	10	54	312
VERY SMALL	0	0	0
PSEUDO	5132	2407677	13406674
TOTAL	7460	5502954	88406914

A high number of user cells (2328 in this example) with respect to the TOTAL (7460) indicates poor re-use of cells. A ratio of hierarchical and flat placement counts less than 1:10 may indicate a poor hierarchy. The hierarchy in this example is okay since the ratio is 1:16 (5502954:88406914).

In cases where the number of pseudo cells (5132 in this example) is high relative to the number of user cells (2328), this can indicate that Calibre is having to move too much data around and had to create an intermediate hierarchy to store the results internally. In turn, this can mean the original hierarchy has a very high percentage of cell instances that are overlapping or that the Layout Base Layer or Layout Top Layer statements need adjusting.

6. Look for HIGH-COST HCELLS.

```
HIGH-COST HCELLS
  cap_dac (EXPANDING DENSE OVERLAPS: 18.0)
  NMOS_VTL_23 (EXPANDING DENSE OVERLAPS: 12.6)
HIERARCHICAL DATABASE CONSTRUCTOR COMPLETE.
```

It may be possible to improve extraction performance by removing these hcells from the hcell list for future runs. Beginning with the 2009.4 Calibre release, you can try using the LVS Auto Expand Hcells PRESET statement.

7. Look for flattened or merged layers that are used in downstream operations.
8. Look for other operations that may cause the overall job to run slow, such as Layout Magnify or Magnify.
9. Search for lines in the log file that provide runtime or memory usage information. You can use grep and sort to extract the desired string and perform a sort based on the number of columns.

```
grep 'OPS COMPLETE' log > ops
sort -nk 8 ops > ops.s
```

You can then use the results of this operation to identify strings or OPERATION numbers to search for in the log file in order to identify the actual statement that was run.

Results

You should now have an idea of items to look for in the Calibre log file that can help in debugging performance issues.

Chapter 17

Distributed Calibre: Reference Dictionary

You can control distributed Calibre modes through the use of configuration file statements, environment variables, and command line options.

Refer to “[Syntax Conventions](#)” on page 19 for information on syntax semantics.

Configuration File Reference Dictionary	310
Transcript Reference.....	348
Distributed Calibre Environment Variables.....	354
Command Line Options Reference Dictionary.....	377
Utilities	397
Errors and Warnings.....	402

Configuration File Reference Dictionary

You can use configuration file statements in a configuration file to control launching and connecting to Calibre MTflex servers.

Refer to “[About the Configuration File](#)” on page 272 for information on creating a configuration file.

HYPER	311
LAUNCH AUTOMATIC	313
LAUNCH CLUSTER	316
LAUNCH MANUAL	320
LOCAL HOST DIR	323
LOCAL HOST COMPUTE	324
MONITOR LOCAL	325
MONITOR REMOTE	329
MONITOR SYSTEM	332
MONITOR UTILITY	340
REMOTE COMMAND	341
REMOTE HOST	345

HYPER

Sets parameters that define the setup information for using hyperscaling with Calibre MTflex.

Usage

HYPER [SMTFACTOR [0 | 1]] [LICENSE ALL] [RDSMULTIPLE [0 | 1]]

Arguments

- **SMTFACTOR [0 | 1]**

An optional keyword that specifies whether to create additional RCS processes on remote CPUs connected only to HDB0 and that support SMT processing. All remote CPUs must be used for the job and licensed to enable this capability. Valid values are:

- 0 — Disables the ability to launch additional threads on the virtual CPU cores (default).
- 1 — Enables the ability to launch additional threads on the virtual CPU cores.

This feature should be used for Litho flows to provide optimal performance. For Litho flows, a warning is issued if this argument is not specified for jobs running with hyperscaling on systems where SMT processing is enabled. The default is 0, which disables this feature.

- **LICENSE ALL**

An optional keyword that instructs Calibre MTflex to license all remote CPUs on a remote host if at least one Calibre MTflex server is started on the remote host. This argument is typically not needed.

Normally Calibre MTflex only licenses remote CPUs that are “connected” to the primary host. With the -hyper option, connecting to less than all CPUs on a remote host does not provide optimal performance. Using less Calibre MTflex servers on a remote host can minimize RAM resources while still allowing hyperscaling to use all CPUs for computation.

- **RDSMULTIPLE [0 | 1]**

An optional keyword that specifies whether to add one RDS for every remote CPU during a Calibre job. This option is useful when working with a job scheduler (such as IBM Spectrum LSF or Univa Grid Engine), where there may be an uneven distribution of CPUs per remote host, causing the RDS space to be hard to predict. Valid values are:

- 0 — Disables the ability to add one RDS for every remote CPU (default).
- 1 — Enables the ability to add one RDS for every remote CPU.

Description

This statement overrides the parameters used to determine setup information for remote Pseudo HDB (PHDB) when using hyperscaling.

Examples

```
HYPER LICENSE ALL SMTFACTOR 1
```

Related Topics

[CALIBRE_HYPER_RDSMULTIPLE](#)

[CALIBRE_HYPER_SMTFACTOR](#)

[CALIBRE_MTFLEX_HYPER](#)

[-remotedata](#)

LAUNCH AUTOMATIC

Sets parameters for the remote hosts and connecting with those servers.

Usage

```
LAUNCH AUTOMATIC [PORT port_number] [WAIT n_seconds]  
[MINCOUNT min_connects] [NAME {IPaddress | hostname}] [SMTFACTOR {0 | 1}]  
[REMOTE POLL n_seconds] [RSH shell_path] [FAILCOUNT int]
```

Arguments

- PORT *port_number*

An optional keyword set specifying the TCP port number to use for connecting to the primary host.

- WAIT *n_seconds*

An optional keyword set specifying the amount of time (in seconds) that the primary host waits for the remote host server to connect. The argument *n_seconds* must be an integer and has a default value of 600. If the remote host does not connect within this time, the primary host will time out and then attempt to connect to the remote host specified by the next REMOTE HOST statement. Once a connection is established with a remote host, the WAIT time is reset to the full *n_seconds* value when attempting to connect to the next remote host.

- MINCOUNT *min_connects*

An optional keyword set specifying the minimum number of remote connections that must be established before the WAIT time expires in order for the Calibre run not to fail. The default is 1. The value specified for *min_connects* must be an integer.

- NAME {*IPaddress* | *hostname*}

An optional keyword set specifying the hostname (or IP address) of the primary host. By default, remote hosts use the primary hostname() returned by the operating system as the connection name. This keyword set should not be used unless it is required for some reason, such as forcing the use of a specific network interface other than the default, as hard-coded names make system administration more difficult to manage.

- SMTFACTOR {0 | 1}

An optional keyword for enabling or disabling the use of virtual CPU cores on SMT-enabled remote hosts. The default is 1 when hyperscaling (-hyper) is not used. To use this capability with hyperscaling, you must specify the [HYPER](#) SMTFACTOR statement in your configuration file.

0 — Instructs Calibre not to use virtual CPU cores on remote hosts when SMT processing is enabled in the BIOS.

1 — Instructs Calibre to use virtual CPU cores on remote hosts when SMT processing is enabled in the BIOS. This is the default when hyperscaling (-hyper) is not used.

- **REMOTE POLL *n_seconds***

An optional keyword set specifying an amount of time (in seconds) for the remote hosts to check if the socket connection has closed because the controlling process on the primary host died. The default is 60 seconds.

- **RSH *shell_path***

An optional keyword set that specifies the path to a remote shell (RSH) or secure shell (SSH) command. If specified, this value can be overridden using the RSH keyword in the REMOTE HOST statement.

Note



The RHEL/CentOS 8 operating systems only support the use of SSH.

- **FAILCOUNT *int***

An optional keyword pair specifying the number of remote failures that will cause the entire run to abort.

Description

The LAUNCH AUTOMATIC statement sets parameters for the connection of the primary host to the remote hosts. In automatic mode, Calibre uses the REMOTE HOST statements in the configuration file to specify information for launching remote hosts, with the primary host performing the launches. You can specify the LAUNCH AUTOMATIC statement only once in the configuration file. You can use the CALIBRE_MTFLEX_LAUNCH environment variable to define values for the LAUNCH AUTOMATIC arguments.

Note

 LAUNCH AUTOMATIC is the default run mode and is automatically assumed when the LAUNCH statement is used without specifying the AUTOMATIC, CLUSTER, or MANUAL keyword.

Examples

The following configuration file illustrates the use of the LAUNCH AUTOMATIC statement and specifying the location of remote hosts using the REMOTE HOST statement:

```
// Mtflex configuration file -- automatic
LAUNCH AUTOMATIC
REMOTE HOST linuxone 2 CALIBRE_HOME /$INSTALL_DIR/aoi/calibre_home
REMOTE HOST linuxtwo 2 CALIBRE_HOME /$INSTALL_DIR/aoi/calibre_home
REMOTE HOST linuxthree 4 CALIBRE_HOME /$INSTALL_DIR/aoi/calibre_home
```

Related Topics

[HYPER](#)

[CALIBRE_MTFLEX_LAUNCH](#)

[LAUNCH CLUSTER](#)

LAUNCH MANUAL

REMOTE HOST

LAUNCH CLUSTER

Sets parameters for connection of the primary host to the remote hosts.

Usage

```
LAUNCH CLUSTER [PORT port_number] [WAIT n_seconds] [COUNT n_connections]
[MINCOUNT min_connects] [NAME {IPaddress | hostname}] [SMTFACTOR {0 | 1}]
[REMOTE POLL n_seconds] [FAILCOUNT int]
```

Arguments

- PORT *port_number*

An optional keyword set that specifies the port number to use for connecting to the primary host.

- WAIT *n_seconds*

An optional keyword set that specifies the amount of time (in seconds) that the primary host will wait for a REMOTE COMMAND statement to execute or for a remote connection to be established. The time is measured from the initial invocation of Calibre or the last successful connection. After this time expires, the primary host assumes all remotes are configured and continues. The argument *n_seconds* must be an integer and has a default value of 600.

The WAIT argument accepts an environment variable as input as shown in this example:

```
setenv MYWAIT 60
```

In the configuration file specify the environment variable:

```
LAUNCH CLUSTER WAIT $MYWAIT SMTFACTOR 1
```

The command terminates with a fatal error if the environment variable cannot be resolved.

- COUNT *n_connections*

An optional keyword set that specifies the number of remote connections that are expected by the primary host before the run begins. The value specified for *n_connections* must be an integer and has a default value of zero (0). Once the primary host establishes *n_connections*, it will not wait for additional connections. If the primary host does not establish *n_connections* before the specified WAIT time expires, it will start the run with less than the specified number of hosts. When used with MINCOUNT, the value specified for COUNT must be greater than or equal to the value specified for MINCOUNT.

The COUNT argument accepts an environment variable as input as shown in this example:

```
setenv MYCOUNT 16
```

In the configuration file specify the environment variable:

```
LAUNCH CLUSTER COUNT $MYCOUNT SMTFACTOR 1
```

The command terminates with a fatal error if the environment variable cannot be resolved.

- **MINCOUNT *min_connects***
An optional keyword set that specifies the minimum number of remote connections that must be established before the WAIT time expires in order for the Calibre run not to fail. The value specified for *min_connects* must be an integer. The default is 1.
- **NAME {*IPaddress* | *hostname*}**
An optional keyword specifying the hostname (or IP address) of the primary machine. The connection name used by the remote hosts defaults to the primary `hostname()` returned by the operating system. The NAME keyword should not be used unless it is required for some reason, such as forcing use of a specific network interface other than the default, as hard-coded names make system administration more difficult to manage.
- **SMTFACTOR {0 | 1}**
An optional keyword used for jobs running on remote hosts with SMT enabled. If specified, the number of RCS processes created is calculated by multiplying the value specified for *smtfactor* by the number of remote cores connected to HDB0. The default is 1 when hyperscaling (-hyper) is not used. Refer to the SMTFACTOR argument for the HYPER statement when using hyperscaling.
 - 0 — Instructs Calibre not to use virtual CPU cores on remote hosts when SMT processing is enabled in the BIOS.
 - 1 — Instructs Calibre to use virtual CPU cores on remote hosts when SMT processing is enabled in the BIOS. This is the default when hyperscaling (-hyper) is not used.
- **REMOTEPOOLL *n_seconds***
An optional keyword set that specifies for remote hosts to poll every *n_seconds* to check if the socket connection is closed due to the primary host exiting. The default is 60 seconds.
- **FAILCOUNT *int***
An optional keyword set that specifies the number of remote failures that will cause the entire run to abort.

Description

The LAUNCH CLUSTER statement sets parameters for the connection of the primary host to the remote hosts. In cluster mode, Calibre uses the REMOTE COMMAND statements in the configuration file to specify information for launching remote hosts. Each REMOTE COMMAND statement is executed in the order they appear in the configuration file. You can specify the LAUNCH CLUSTER statement only once in the configuration file. You can use the CALIBRE_MTFLEX_LAUNCH environment variable to define values for the LAUNCH CLUSTER arguments.

Note

 LAUNCH AUTOMATIC is the default run mode and is automatically assumed when the LAUNCH statement is used without specifying the AUTOMATIC, CLUSTER, or MANUAL keyword.

Examples

The following configuration file illustrates the use of the LAUNCH CLUSTER and REMOTE COMMAND statements. This example uses the rcalibre utility, which is described in more detail in “[rcalibre](#)” on page 398.

This example uses the rsh utility, which is not supported with the RHEL/CentOS 8 operating systems. If running this example on RHEL/CentOS 8, you must substitute ssh for rsh.

Configuration File *config.lc*:

```
LAUNCH CLUSTER COUNT 6 MINCOUNT 2
// script must have execution privileges
// script and argument list must be accessible from primary host
REMOTE COMMAND $MGC_EXAMPLES/calbr_admin_gd/launch_remote.sh
    ARGUMENTS ["$MGC_EXAMPLES/calbr_admin_gd/remote_args.txt"]
```

\$MGC_EXAMPLES/calbr_admin_gd/launch_remote.sh:

```
# In the following script the user has specified the CALIBRE_HOME tree
# and an aoi_home tree as environment variables.
# Calibre controls and accesses the env var CALIBRE_REMOTE_CONNECTION
# to set and communicate the port and process ID for each individual
# connection between the primary host and the remote host(s).
# This script assumes that the remotes all have a mounted primary
partition
# named "/scratch1" to use for log files. Note the default location for
# the log data on the remote hosts is /tmp.
# n is used to count how many connections are made by the script,
# and compared to the values set on the LAUNCH CLUSTER statement for
# maximum and minimum connects.
# In the for loop below, 'cat $1' takes the ARGUMENTS specified as
# a file name in the configuration file "config.lc" and outputs the
# contents as an argument list for the "for" loop.
# This script is using the standard OS utility rsh for the remote shell.
# Any other remote shell utility, such as ssh, may be substituted for rsh.
# In this script, the CALIBRE_REMOTE_COUNT is checked at the end of the
# loop to stop connecting remotes once the limit is met.
# Note the CALIBRE_REMOTE_MINCOUNT is not used here; the primary host
# process keeps track of the number of successful connections and
# terminates the entire Calibre run if the number of successful
# connections is less than the specified CALIBRE_REMOTE_MINCOUNT.
#!/bin/sh
n=0
for node in `cat $1`
do
    rsh $node $CALIBRE_HOME/bin/rcalibre "/scratch1" 1 CALIBRE_HOME \
        $aoi_home -mtflex $CALIBRE_REMOTE_CONNECTION &
    rsh $node $CALIBRE_HOME/bin/rcalibre "/scratch1" 1 CALIBRE_HOME \
        $aoi_home -mtflex $CALIBRE_REMOTE_CONNECTION &
    n=`expr $n + 2`;
    if [ $n -eq $CALIBRE_REMOTE_COUNT ]
    then break;
    fi
done
```

remote_args.txt:

```
node91
node92
node93
node94
node95
node90
```

Related Topics

[CALIBRE_MTFLEX_LAUNCH](#)

[HYPER](#)

[LAUNCH MANUAL](#)

[LAUNCH AUTOMATIC](#)

[REMOTE COMMAND](#)

[REMOTE HOST](#)

[Using the LAUNCH CLUSTER and REMOTE COMMAND Statements](#)

LAUNCH MANUAL

Sets parameters for connection of the primary host to the remote hosts.

Usage

Manual Mode (-remotefile) Syntax

```
LAUNCH MANUAL [WAIT n_seconds] [PORT port_number] [COUNT n_connections]  
[MINCOUNT min_connects] [SMTFACTOR {0 | 1}] [REMOTEPOOLL n_seconds]  
[FAILCOUNT int]
```

LVS Compare (-cmp_remotefile) Syntax

```
LAUNCH MANUAL [WAIT n_seconds]
```

Arguments

- **WAIT *n_seconds***
An optional keyword set that specifies the amount of time (in seconds) the primary host should wait for a manual connection to a remote host. The time is measured from the initial invocation of Calibre or the last successful connection. After this time expires, the primary host assumes all remote hosts are configured and continues with the run. The *n_seconds* value must be a positive integer. The default WAIT time is 600 seconds when this keyword set is not specified.
- **PORT *port_number***
An optional keyword set that specifies the port number to use for connecting to the primary host.
- **COUNT *n_connections***
An optional keyword set that specifies the number of remote connections (*n_connections*) expected by the primary host. The *n_connections* value must be a positive integer. The default is zero (0) when this keyword set is not specified.
Once the primary host establishes *n_connections*, it will not wait for additional connections. If the primary host does not establish *n_connections* before the specified WAIT time expires, it will start the run with less than the specified number of hosts. When used with MINCOUNT, the value specified for COUNT must be greater than or equal to the value specified for MINCOUNT.
If COUNT is not specified, but MINCOUNT is, then COUNT is set to MINCOUNT.
- **MINCOUNT *min_connects***
An optional keyword set that specifies the minimum number of remote connections that must be established before the WAIT time expires in order for Calibre to start the run. The Calibre run will fail if the number of remote hosts is less than the specified *min_connects*. The default is 1 when this keyword set is not specified.

- **SMTFACTOR {0 | 1}**

An optional keyword used for jobs running on remote hosts with SMT enabled. If specified, the number of RCS processes created is calculated by multiplying the value specified for *smtfactor* by the number of remote CPUs connected to HDB0. The default is 1 when hyperscaling (-hyper) is not used. Refer to the SMTFACTOR argument for the **HYPER** statement when using hyperscaling.

- 0 — Instructs Calibre not to use virtual CPU cores on remote hosts when SMT processing is enabled in the BIOS.
- 1 — Instructs Calibre to use virtual CPU cores on remote hosts when SMT processing is enabled in the BIOS. This is the default when hyperscaling (-hyper) is not used.

- **REMOTEPOOLL *n_seconds***

An optional keyword set that specifies for remote hosts to poll every *n_seconds* to check if the socket connection is closed due to the primary host exiting. If there is no reply from primary host, the process on the remote hosts can then be killed. The default is 60 seconds when this keyword and value are not specified.

- **FAILCOUNT *int***

An optional keyword set that specifies the number of remote failures that will cause the entire run to abort.

Description

The LAUNCH MANUAL statement sets parameters for the connection of the primary host to the remote hosts. In manual mode, Calibre ignores all REMOTE statements and relies on the user to launch remote hosts using the Calibre command line -mtflex argument. You can specify this statement only once in the configuration file. This command is often used when primary and remote hosts are not part of the same subnetwork. You can use the **CALIBRE_MTFLEX_LAUNCH** environment variable to define values for the LAUNCH MANUAL arguments.

Examples

Example 1

The following configuration file shows how to use the LAUNCH MANUAL statement to run Calibre MTflex in manual mode on four remote CPUs:

```
// Mtflex configuration file -- manual
// Informing primary host that a minimum of four connections must be made
// before the default WAIT time of 600 seconds expires in order for
// Calibre to start the run
LAUNCH MANUAL MINCOUNT 4
```

Example 2

Refer to [Example 7](#) in the “Calibre nmLVS and Calibre nmLVS-H Command Line” topic of the *Calibre Verification User’s Manual* for an example that uses the LAUNCH MANUAL statement to launch an LVS compare process on a remote host.

Related Topics

[CALIBRE_MTFLEX_LAUNCH](#)

[HYPER](#)

[LAUNCH CLUSTER](#)

[LAUNCH AUTOMATIC](#)

[REMOTE HOST](#)

[-mtflex](#)

LOCAL HOST DIR

Specifies a directory on the primary host for storing temporary files that are created on remote hosts by some applications.

Usage

LOCAL HOST DIR *path*

Arguments

- *path*

A required argument specifying the path to a directory where Calibre can store temporary files created on remote hosts by some applications, such as Calibre FRACTURE and Calibre MDPmerge. The specified directory must be accessible by the primary host.

When the LOCAL HOST DIR statement is not specified in the configuration file, the default is to use the current working directory for -remote and -remotefile Calibre MTflex modes.

Refer to “[Calibre FRACTURE Distributed Operations](#)” and “[Calibre MDPmerge Distributed Operations](#)” for more information.

Description

The LOCAL HOST DIR statement controls where temporary files that are created by some applications are stored on the primary host. You can specify this statement *only once* in the configuration file. Calibre echoes the value of LOCAL HOST DIR to the transcript with the [-remote](#) or [-remotefile](#) command line options.

When using the -remote command line option, you can use the [CALIBRE_MTFLEX_LOCAL_HOST](#) environment variable to define a value for the LOCAL HOST DIR statement. When using the -remote or -remotefile command line options, you can use the [CALIBRE_MTFLEX_LOCAL_HOST_DIR](#) environment variable to specify where to store temporary remote host files on the primary host.

Examples

The following example uses the LOCAL HOST DIR statement to specify a directory for storing temporary files created by the remote hosts:

```
LOCAL HOST DIR "/net/machine/dir1/tmp_location"
```

Related Topics

[CALIBRE_MTFLEX_LOCAL_HOST](#)

[CALIBRE_MTFLEX_LOCAL_HOST_DIR](#)

[Calibre FRACTURE Distributed Operations](#)

[Calibre MDPmerge Distributed Operations](#)

LOCAL HOST COMPUTE

Specifies the number of CPUs on the primary host to use for computational tasks.

Usage

LOCAL HOST COMPUTE *num_cpus*

Arguments

- ***num_cpus***

A required argument specifying the number of CPUs on the primary host to perform computational tasks. The value specified for ***num_cpus*** must be a positive integer that is less than or equal to the local CPU count. When this statement is enabled, the operations consume licenses for the local CPU count (treated like remote RCSs). The default is “<local CPU count>/2”.

Typically the primary host CPUs are used only for managing computational tasks on the remote CPUs and operations that are not multithreaded. If the primary host has more CPUs than are needed for managing remote tasks, you can use this argument to improve overall real time performance.

When using hyperscaling ([-hyper](#)), the value specified for ***num_cpus*** is per HDB. That is, the number of primary host threads used to perform computational tasks is equal to the ***num_cpus*** value multiplied by the HDB count, which is five by default. With hyperscaling, you should not specify a ***num_cpus*** value other than “1” AND only when the primary host has at least four CPUs. When the number of primary host CPUs is exactly four, they can be slightly overdriven.

Description

The LOCAL HOST COMPUTE statement controls primary host CPU usage. You can specify this statement only once in the configuration file. When using [-remote](#) to execute a Calibre job, you can use the [CALIBRE_MTFLEX_LOCAL_HOST](#) environment variable to define values for the LOCAL HOST COMPUTE statement.

Examples

Example 1

The following example is the recommended usage when executing a Calibre job in hyperscaling (-hyper) mode, which should only be used when the primary host has at least four CPUs:

```
LOCAL HOST COMPUTE 1
```

Example 2

The following example specifies that four CPUs are used on the primary host. (Do not use this form for hyperscaling mode.)

```
LOCAL HOST COMPUTE 4
```

MONITOR LOCAL

Enables monitoring of Calibre MTflex operations on the primary host.

Usage

```
MONITOR LOCAL [INTERVAL seconds] [CONNECT] [LOAD]
[MEMORY [virtual lithoshared]] [IO] [MINBW num_bw]
```

Arguments

- INTERVAL *seconds*

An optional keyword set that specifies the minimum time in seconds between monitor reports. The default is 60 seconds. Intervals less than 10 seconds are not recommended as they unnecessarily clutter the log file and increase run time.

This INTERVAL is used for the MON:L (LOAD) and MON:M (MEMORY) statements only and is not intended for fine resolution of load or scaling. MON:F is printed for each cell processing loop independent of the INTERVAL time. CPU and REAL time are printed for each completed SVRF operation. These mechanisms are provided for fine resolution of scaling in place of short MON:L intervals.

Following are some recommended INTERVAL settings. INTERVAL settings shorter than those recommended should be avoided except for special needs.

For nmDRC:

```
MONITOR LOCAL LOAD INTERVAL 60          // once per minute
```

For nmOPC:

```
MONITOR LOCAL LOAD INTERVAL 300         // once every five minutes
```

- CONNECT

An optional keyword that enables verbose output relating to the primary host and remote host connection process. You should use this keyword when debugging connection problems between the primary and remote hosts.

- LOAD

An optional keyword that enables internal task monitoring on the primary host. When you specify this keyword, Calibre writes the task monitoring information to the transcript using the following formats:

<MON:S *timestamp*>

This MON statement marks the beginning of the parallel threaded loop. The value of *timestamp* is the number of seconds since process invocation.

<MON:L *timestamp* T *n1 n2* Q *n3 n4* R *n5* C *n6 n7*>

This MON statement monitors the state of the parallel loop during execution where:

- *n1* is the number of active cell threads.

- $n2$ is the number of active stripe threads.
- $n3$ is the cell queue length.
- $n4$ is the stripe queue length.
- $n5$ is the number of active threads executing remote operations.
- $n6$ is the number of cells that have completed processing.
- $n7$ is the total number of cells to be processed.

<MON:F *timestamp* W $n1$ L $n2$ R $n3$ >

This MON statement marks the end of the parallel threaded loop where:

- $n1$ is the number of real time seconds since the start of the loop.
- $n2$ is the total CPU seconds on the primary host since the start of the loop.
- $n3$ is the sum of all remote CPU seconds since the start of the loop.

Note

 In the absence of a configuration file, the MONITOR LOCAL LOAD functionality may be enabled for Calibre jobs run in MT mode by setting the CALIBRE_MT_MONITOR_LOAD environment variable to the desired interval in seconds. In the following example, the LOAD associated MON statements are written to the Calibre transcript once per minute.

```
setenv CALIBRE_MT_MONITOR_LOAD 60
```

- **MEMORY** [virtual lithoshared]

An optional keyword that enables LVHEAP memory monitoring on the primary host. The MEMORY argument writes LVHEAP information to the transcript using the following format:

<MON:M *timestamp* H $n1$ $n2$ $n3$ >

Where:

- $n1$ is the current amount of memory used in MB.
- $n2$ is the current amount of memory allocated in MB.
- $n3$ is the maximum amount of memory allocated in MB.

The MEMORY argument supports the use of the following keyword:

virtual lithoshared — An optional keyword set used in conjunction with the LITHO_SOCS KERNELS_SHARED environment variable to enable and report shared memory for post-tapeout runs. When this keyword is specified, the log file includes a virtual memory section marked with V and a shared litho section marked with LS to the MON:M entries as shown in this example:

```
<MON:M 25449 H 3 322 471 V 1043 808 767 LS 224 56 224>
```

Refer to [LITHO_SOCS KERNELS_SHARED](#) in the *Calibre Post-Tapeout Flow User's Manual* for more information.

- **IO**

An optional keyword that enables network IO monitoring on either or both the primary and remote hosts. The results characterize the network performance to each connected remote host CPU after obtaining network connection, and reports the current IO counts. The results are reported to the transcript using the following format:

<MON:I *timestamp* R *n1* W *n2*>

Where:

- *n1* is the number of bytes in millions read by the remote host.
- *n2* is the number of bytes in millions written by the remote host.

- **MINBW *num_bw***

An optional keyword used to specify the minimum bandwidth performance (in MB/sec) for each remote host. This keyword can be specified only with the IO keyword. The standard IO tests are run and the results are compared against the value specified for *num_bw*. A remote host is removed from the pool if it fails the test.

Description

The MONITOR LOCAL statement is optional and enables monitoring of Calibre MTflex operations on the primary host. You can specify the MONITOR LOCAL statement once in the configuration file. Refer to “[Monitoring Remote Processes](#)” for additional information on usage and expected results for this statement.

When using Calibre MTflex functionality, you can monitor the progress of your Calibre run by specifying the MONITOR LOCAL or MONITOR REMOTE statements in your Calibre MTflex configuration file. In general, you should activate the monitor functionality if you are performing a CPU-intensive Calibre run. Additionally, the monitor output is useful for analysis of poor-performing runs. This can potentially save running an extra job just to collect the monitor output when you encounter a problem.

You can use the CALIBRE_MTFLEX_MONITOR_LOCAL environment variable as an alternate method of defining values for the MONITOR LOCAL statement.

Examples

The following MONITOR LOCAL LOAD statement enables task monitoring on the primary host:

```
MONITOR LOCAL LOAD INTERVAL 60
```

The LOAD argument identifies the start of a parallel threaded loop (MON:S), the state of the parallel loop during execution (MON:L), and the end of the parallel threaded loop (MON:F) as shown in this example transcript:

```
<MON:S 3117>
<MON:L 3177 T 6 114 Q 3324 0 R 120 C 101 5603>
<MON:L 3237 T 2 118 Q 3056 0 R 120 C 487 5603>
<MON:L 3297 T 3 117 Q 3038 0 R 120 C 571 5603>
<MON:L 3357 T 1 119 Q 2771 0 R 120 C 927 5603>
<MON:L 3417 T 8 112 Q 2616 0 R 120 C 1114 5603>
<MON:L 3477 T 7 113 Q 1290 0 R 120 C 3028 5603>
<MON:L 3537 T 3 117 Q 1128 0 R 120 C 3273 5603>
<MON:L 3597 T 0 120 Q 1208 0 R 120 C 3619 5603>
<MON:L 3657 T 0 120 Q 753 0 R 120 C 4279 5603>
<MON:F 3769 W 652 L 39 R 72336>
```

Related Topics

[CALIBRE_MTFLEX_MONITOR_LOCAL](#)

[MONITOR REMOTE](#)

[Monitoring Remote Processes](#)

[MONITOR SYSTEM](#)

[LITHO_SOCS KERNELS_SHARED \[Calibre Post-Tapeout Flow User's Manual\]](#)

MONITOR REMOTE

Enables monitoring of Calibre MTflex operations on the remote hosts.

Usage

MONITOR REMOTE [INTERVAL *seconds*] [CONNECT] [MEMORY [virtual lithoshared]]
[COMMAND *command_path*]

Arguments

- INTERVAL *seconds*

An optional keyword set that specifies the minimum time in seconds between monitor reports. The default is 60 seconds. Intervals less than 10 seconds are not recommended as they unnecessarily clutter the log file and increase run time.

- CONNECT

An optional keyword that enables verbose output relating to the primary host and remote host connection process. You should use this keyword when debugging connection problems between the primary and remote hosts.

- MEMORY [virtual lithoshared]

An optional keyword that enables LVHEAP memory monitoring on the remote host. The MEMORY argument writes LVHEAP information to the transcript using the following format:

<MON:M <*timestamp*> H <*n1*> <*n2*> <*n3*>>

Where:

- *n1* is the current amount of memory used in MB.
- *n2* is the current amount of memory allocated in MB.
- *n3* is the maximum amount of memory allocated in MB.

The MEMORY argument supports the use of the following keyword:

virtual lithoshared — An optional keyword set used in conjunction with the LITHO_SOCS KERNELS_SHARED environment variable to enable and report shared memory for post-tapeout runs. When this keyword is specified, the log file includes a virtual memory section marked with V and a shared litho section marked with LS to the MON:M entries as shown in this example:

<MON:M 25449 H 3 322 471 V 1043 808 767 LS 224 56 224>

Refer to [LITHO_SOCS_KERNELS_SHARED](#) in the *Calibre Post-Tapeout Flow User's Manual* for more information.

- **COMMAND** *command_path*

An optional keyword specifying the path to a file that contains a command to be executed on each remote host by “*/bin/sh*”. This keyword is useful for debugging remote connections by having a command, such as ping, run on every remote host after connecting.

Description

The MONITOR REMOTE statement is optional and enables monitoring of Calibre MTflex operations on the remote host. You can specify the MONITOR REMOTE statement once in the configuration file. Refer to “[Monitoring Remote Processes](#)” for additional information on usage and expected results for this statement.

When using Calibre MTflex functionality, you can monitor the progress of your Calibre run by specifying the MONITOR LOCAL or MONITOR REMOTE statements in your Calibre MTflex configuration file. In general, you should activate the monitor functionality if you are performing a CPU-intensive Calibre run. Additionally, the monitor output is useful for analysis of poor-performing runs. This can potentially save running an extra job just to collect the monitor output when you encounter a problem.

You can use the CALIBRE_MTFLEX_MONITOR_REMOTE environment variable as an alternate method of defining values for the MONITOR REMOTE statement.

Note

 By default, Calibre removes the log files on the remote host after successfully completing the Calibre MTflex operation. If you use the monitor functionality, then the remote host logs remain on the remote host. In general, you can find these log files in the remote host’s *tmp* directory, for example */net/remote_host/tmp*.

Examples

In the following example, the MONITOR REMOTE statement prints the current memory usage for each remote host into a log file every 10 minutes and prevents the deletion of the log file at the end of the run.

```
MONITOR REMOTE INTERVAL 600 MEMORY
```

The MEMORY argument enables LVHEAP memory monitoring (MON:M) on the primary host as shown in this example transcript:

```
<MON:M 3932 H 3228 3853 3853>
```

Related Topics

[CALIBRE_MTFLEX_MONITOR_REMOTE](#)

[MONITOR LOCAL](#)

[Monitoring Remote Processes](#)

MONITOR SYSTEM

LITHO_SOCS KERNELS_SHARED [Calibre Post-Tapeout Flow User's Manual]

MONITOR SYSTEM

Enables system level monitoring of the primary and remote hosts.

Usage

```
MONITOR SYSTEM [ INTERVAL num_seconds ]
[REPORT [STDOUT | ASCII | FILE filename | RRDSCRIPT | RRD rrdtool_path]]
[HEARTBEAT n_factor [PRINT] [EMAIL]]
[MEMORY limit [PRINT] [EMAIL] [KILL | KILLONE]]
[LOCALMEMORY limit [PRINT] [EMAIL]]
[SWAP limit [PRINT] [EMAIL] [KILL | KILLONE]]
[LOCALSWAP limit [PRINT] [EMAIL]]
[EMAILLIST “name1, name2, ...”] [EMAILPATH path] [OUTDIR outdir_path]
[RRDGRAPH [TIMESPAN time_spec] [WIDTH num_width] [HEIGHT num_height]]
```

Arguments

- **INTERVAL *num_seconds***

An optional keyword set that specifies the amount of time in seconds between collecting information from the primary and remote hosts. Note that specifying a small value can slow down the Calibre run. The default is 300 seconds (5 minutes). The minimum value allowed is 60 seconds.

- **REPORT [STDOUT | ASCII | FILE *filename* | RRDSCRIPT | RRD *rrdtool_path*]**

An optional keyword that specifies the output format for the report.

STDOUT — Data is collected based on the INTERVAL time and appended to a temporary file named *CalibreCrpNetworkMonitorUnreported*. The data is output to the stdout of the primary host process using the following format and the temporary file is deleted.

```
<SYSMON: timestamp HOST name TOTAL_MEMORY n1 FREE_MEMORY n2 TOTAL_SWAP n3
FREE_SWAP n4 LOADAVG n5 n6 n7>
```

ASCII — Data is collected based on the INTERVAL time and appended to an ASCII comma separated value (CSV) file named *CalibreNetworkMonitorLog.\$primary.\$pid.\$time*, where *\$primary* is the name of the primary host, *\$pid* is the process ID of the Calibre run, and *\$time* is the date of output.

FILE *filename* — Data is collected based on the INTERVAL time and appended to the specified *filename*.

RRDSCRIPT — Creates a file named *CalibreRRDGraph.sh* which contains the RRD commands to graph the RRD data. The RRDGRAPH keyword can be used to control the graph functions. This keyword also creates script files that can be used to create and update RRD files. These files are named *CalibreRRD.<nodename>.sh* for each node and *CalibreRRD.summary.sh*.

RRD *rrdtool_path* — Creates a file named *CalibreRRDGraph.sh* which contains the RRD commands to graph the RRD data. The RRDGRAPH keyword can be used to control the graph functions. The value specified for *rrd_path* should be a valid path to

the RRD tool. This keyword uses the *rrdtool_path* to perform the create and update functions, and to create RRDs named *CalibreRRD.<nodename>.rrd* and *CalibreRRD.summary.rrd*.

- HEARTBEAT *n_factor* [PRINT] [EMAIL]

An optional keyword set that specifies a value to use as a multiplier of the value specified by INTERVAL *num_seconds*. Calibre uses the following formula to calculate a response time between the primary and remote host:

$$n_factor * \text{INTERVAL } num_seconds = response_time$$

When a remote host is determined to be unresponsive after the calculated time period, the connection between the primary and remote host is dropped by closing the socket on the primary host and removing the remote host from current or future use. For example, using the default INTERVAL *num_seconds* of 300 and specifying a HEARTBEAT *n_factor* value of 3 causes a remote host to be dropped when there is no response after 900 seconds (15 minutes).

PRINT — Prints an error message to stdout when there is no response from a remote host during the calculated time period. For example:

```
"SYSTEM NETWORK ERROR: <remote_hostname> IS NOT RESPONSIVE !" every
<num_seconds> seconds
WARNING: Machine <remote_hostname> is not responsive in the last <n>
pings. Calibre stops using it.
```

EMAIL — Sends an error message in an email when there is no response from a remote host during the calculated time period.

Note

 The HEARTBEAT keyword is useful and recommended in environments with longer running jobs and a higher probability of network failures. In some cases, remote hosts that crash or are rebooted do not close the TCP connection to the primary host. When this happens, if the HEARTBEAT keyword is not specified, it can take up to two hours for the operating system to inform the primary host of this failure giving the appearance that a Calibre job is hung. The HEARTBEAT keyword can promptly detect and send notification when a remote host is unresponsive.

- MEMORY *limit* [PRINT] [EMAIL] [KILL | KILLONE]

An optional keyword set that specifies a limit for the total memory consumed on a remote host and, optionally, the action to be taken when the *limit* value is exceeded. The value specified for *limit* can be in megabytes (MB) or as a percentage of the total memory. To specify a percentage of the total memory, you must append a “p” to the value.

PRINT — Prints to stdout when the specified *limit* value is exceeded. The PRINT option is recommended with the KILL or KILLONE keyword, as PRINT provides additional information about why a process was killed.

EMAIL — Generates an email when the specified *limit* value is exceeded.

KILL | KILLONE

KILL — Kills all processes on the remote host that exceed the specified *limit* for the memory.

KILLONE — Kills a single process on the remote host that exceeds the specified *limit* for the memory.

Refer to “[KILL and KILLONE Options for the MEMORY and SWAP Arguments](#)” on page 336 for more information on these options.

Note

 If the configuration file contains multiple entries for the MONITOR SYSTEM MEMORY statement, only the value specified in last entry is used.

- LOCALMEMORY *limit* [PRINT] [EMAIL]

An optional keyword set that specifies a limit for the total memory consumed on a primary host, and the action to be taken when the *limit* value is exceeded. The value specified for *limit* can be in megabytes (MB) or as a percentage of the total memory. To specify a percentage of the total memory, you must append a “p” to the value.

PRINT — Prints to stdout when the specified *limit* value is exceeded.

EMAIL — Generates an email when the specified *limit* value is exceeded.

- SWAP *limit* [PRINT] [EMAIL] [KILL | KILLONE]

An optional keyword set that specifies a limit for the total swap used on a remote host, and the action to be taken when the *limit* value is exceeded. The value specified for *limit* can be in megabytes (MB) or as a percentage of the total swap. To specify a percentage of the total swap, you must append a “p” to the value. The default *limit* is 0.

PRINT — Prints to stdout when the specified *limit* is exceeded. The PRINT option is recommended with the KILL or KILLONE option, as PRINT provides additional information about why a process was killed.

EMAIL — Generates an email when the specified *limit* is exceeded.

KILL | KILLONE

KILL — Kills all processes on the remote host that exceed the specified *limit* for the swap.

KILLONE — Kills a single process on the remote host that exceeds the specified *limit* for the swap.

Refer to “[KILL and KILLONE Options for the MEMORY and SWAP Arguments](#)” on page 336 for more information on these options.

Note

 If the configuration file contains multiple entries for the MONITOR SYSTEM SWAP argument, only the value specified in last entry is used.

- **LOCALSWAP *limit* [PRINT] [EMAIL]**

An optional keyword set that specifies a limit for the total swap used on the local host, and the action to be taken when the *limit* value is exceeded. The value for *limit* can be in megabytes (MB) or as a percentage of the total swap. To specify a percentage of the total swap, you must append a “p” to the value. The default *limit* is 0.

PRINT — Prints to stdout when the specified *limit* value is exceeded.

EMAIL — Generates an email when the specified *limit* value is exceeded.

- **EMAILLIST “*name1, name2, ...*”**

An optional keyword used to specify one or more email addresses to receive reports from the specified MONITOR SYSTEM options, such as HEARTBEAT or MEMORY. Email addresses must be separated by a comma or a space. The username for the master process is used when a value is not specified for this keyword.

- **EMAILPATH *path***

An optional keyword set that specifies the path to the email command to be used. The default is */bin/mail*.

- **OUTDIR *outdir_path***

An optional keyword set that specifies an output directory path to be used for writing report files. The default is the current working directory of the primary host process.

- **RRDGRAPH [TIMESPAN *time_spec*] [WIDTH *num_width*] [HEIGHT *num_height*]**

An optional keyword used to control the RRD graph functions. The graph functions create PNG files for viewing. This argument is ignored if either REPORT RRD or REPORT RRDSCRIPT is not specified.

The difference between RRD and RRDGRAPH is Calibre creates and updates RRD files at every time interval when RRD is specified. When RRDSCRIPT is specified, Calibre writes out scripts which can be executed to create and update RRD files.

TIMESPAN — Specifies the time window that is displayed.

WIDTH — Specifies the width of the graph with actual data in pixels.

HEIGHT — Specifies the height of the graph with actual data in pixels.

Description

This statement enables system level monitoring, by collecting the following information from the primary host and each remote host:

- **MEMORY_TOTAL**
- **MEMORY_FREE**
- **SWAP_TOTAL**
- **SWAP_FREE**

- LOAD_1, LOAD_5, LOAD_15 (machine loads for the last 1, 5, and 15 minutes)

This information is automatically collected during the run. You can specify the information to be reported at a specific time interval and printed to STDOUT, ASCII, or RRD.

RRD is the acronym for Round Robin Database. RRDtool is an open source tool useful for collecting and reporting time sampled data. RRDtool can be freely downloaded and compiles on a number of different Linux platforms. Calibre does not provide RRDtool but produces data in RRD format that can be executed by RRDtool. Any combination of reporting options is allowed. For more information, refer to:

<http://oss.oetiker.ch/rrdtool/index.en.html>

KILL and KILLONE Options for the MEMORY and SWAP Arguments

The KILL and KILLONE options for the MEMORY and SWAP arguments try to prevent unresponsive remote hosts by reducing resource consumption (memory or swap) on the remote host. Resource consumption is reduced on a remote host by killing RCS processes. In order to kill RCS processes, there must be communication between the primary and remote host. Killing an RCS process terminates the Calibre MTflex connection by closing the remote end of the socket. The primary host is notified when the socket is closed and removes the remote from any future work assignment.

The purpose of KILLONE is to increase the available memory or swap for other processes on a remote host by killing a single process that exceeds a specified limit. KILLONE is intended to be used in flows such as OPC, where memory usage is nearly uniform across all remote hosts. Rather than killing all remote hosts, as tends to happen with KILL, the KILLONE option gradually reduces compute scaling while gradually increasing available memory per remote process. The result is optimal scaling and memory usage across remote hosts in the case where the initial configuration exceeds available memory limits.

With the KILLONE option, the INTERVAL argument determines the frequency (every other interval) at which a remote process is killed. This ensures the operating system has time to respond to the previously killed remote process. Subsequent processes are killed only if, for example, the *limit* value for the SWAP parameter is still exceeded and active disk swapping is still occurring. If the *limit* value is exceeded but no additional disk swap is detected, additional processes are not be killed (warning message only). This feature is included because some operating systems do not reliably reduce the swap used value as memory is freed for other uses.

The behavior of the KILL and KILLONE options differs from the HEARTBEAT argument, which closes the socket on the primary host end.

Note

 KILL and KILLONE have no effect on remote data servers (-remotedata). Remote data servers will continue to run on the remote host until the primary host process terminates.

Examples

Example 1

The following statement generates a report to stdout every five seconds:

```
MONITOR SYSTEM REPORT stdout ascii rrd <path>/bin/rrdtool
```

Output to stdout:

```
<SYSMON 5 HOST machineOne TOTAL_MEM 7978 FREE_MEM 7294 TOTAL_SWAP 16386
FREE_SWAP 16181 LOADAVG 0.44 0.25 0.09>
```

Output to ascii:

```
5,opt2cal3,7978,7294,16386,16181,0.440000,0.250000,0.090000
```

Example 2

The following example causes one process to be killed when the swap reaches 40 percent of the total swap space:

```
MONITOR SYSTEM INTERVAL 300 SWAP 40p KILLONE
```

Example 3

The following example monitors the memory on the primary and remote hosts and kills processes that exceed the specified memory limit of 2000 MB:

```
MONITOR SYSTEM MEMORY 2000 PRINT EMAIL KILL
```

In addition, the following message is printed to stdout and emailed:

```
WARNING: Killed 4 processes on machine <name> because it used memory 2110
MB out of total 3965 MB, exceeding memory limit 2000 MB
```

Example 4

The following example kills a single process on the remote host that exceeds a specified swap limit of 1000:

```
MONITOR SYSTEM SWAP 1000 KILLONE PRINT
```

The PRINT option outputs the following results:

```
[Calibre hostname:1117:1257186584] KILLED: SWAP: remote_host 1000+ MB
(1920 MB out of 30725 MB)
WARNING: Killed 1 process on machine remote_host because it used swap
memory 1920 MB out of total 30725 MB, exceeding swap limit 1000 MB
```

Example 5

The following example generates a report every 120 seconds (2 minutes) to stdout and an ASCII file. In addition, a file named *CalibreRRDGraph.sh* is created containing the RRD commands to graph the RRD data. The report files are output to */user/test*. The PNG file generated by the

graph functions is 700 x 300 pixels and displays a time window of 10 hours. An email is automatically generated if the memory or swap usage reaches 5000 MB. If no response is received from a remote host in three seconds, a report is output to stdout and also to the email addresses a@b.com and c@d.com.

```
MONITOR SYSTEM rrdgraph width 700
MONITOR SYSTEM rrdgraph height 300 timespan 10hour
MONITOR SYSTEM interval 120
MONITOR SYSTEM report stdout ascii rrd /usr/local/rrdtool/bin/rrdtool \
    rrdscript
MONITOR SYSTEM memory 5000 email print
MONITOR SYSTEM swap 5000 email print
MONITOR SYSTEM heartbeat 3 print email
MONITOR SYSTEM emaillist "a@b.com c@d.com"
MONITOR SYSTEM outdir /user/test
```

An example of the output generated by using the RRDGRAPH capabilities of the MONITOR SYSTEM statement is shown in [Figure 17-1](#) (primary host) and [Figure 17-2](#) (remote machine).

Figure 17-1. Example RRDGRAPH Output of the Load Average on a Primary Host

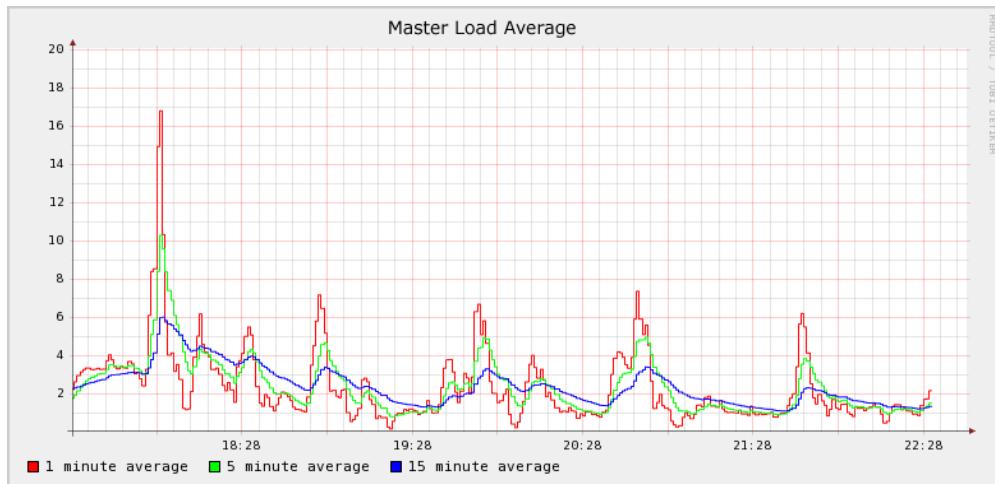
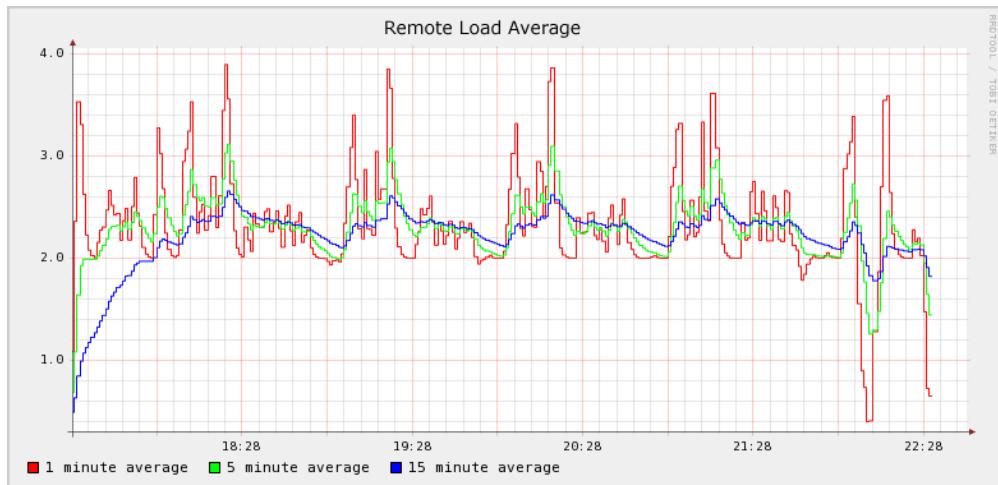


Figure 17-2. Example RRDGRAPH Output of the Load Average on a Remote Machine



Related Topics

[MONITOR LOCAL](#)

[Monitoring Distributed Calibre](#)

[MONITOR REMOTE](#)

MONITOR UTILITY

Mandatory for Calibre DRA and Calibre Cluster Manager (CalCM). This statement is not required for any other distributed Calibre runs.

Note

 The MONITOR UTILITY statement is mandatory for correct Calibre DRA usage. It instructs Calibre to compute the CPU demand for the job, which is reported when using the **calibre_dra -status** command. For more information on MONITOR UTILITY, refer to “Running Calibre DRA” in the *Calibre Dynamic Resource Allocator (DRA) User’s Manual*.

REMOTE COMMAND

Specifies a command and arguments that provide information about how to launch a Calibre MTflex server or an LVS compare process on a remote host.

Usage

REMOTE COMMAND *command_path* [ARGUMENTS '[' *arg1* ... *argn* '']]

Arguments

- ***command_path***

A required keyword specifying the path to an executable file on the primary host that contains commands for launching a Calibre MTflex server or an LVS compare process on a remote host.

For xCalibrate jobs, the executable file must specify the rxcalibrate utility to initiate an xCalibrate MTflex run on remote hosts.

- **ARGUMENTS '['*arg1* ... *argn*'']**

An optional keyword and parameters that provide arguments to the executable file specified by ***command_path***. The arguments must be enclosed in brackets ([]).

Description

The REMOTE COMMAND statement specifies a command and arguments that provide information about how to launch a Calibre MTflex server or an LVS compare process on a remote host. There may be one or more REMOTE COMMAND statements, which are used only in LAUNCH CLUSTER mode. LAUNCH CLUSTER mode executes each REMOTE COMMAND in the order they appear in the configuration file.

The argument strings support a fixed set of variable substitution characters. The substitution variables are denoted by a preceding percent (%) character, and can be disabled by escaping the percent (%) character with a backslash (\). The set of supported variables include:

- %H — The host name portion of the calibre remote connection.
- %P — The port number portion of the calibre remote connection.
- %C — The count from the LAUNCH CLUSTER statement.

Execution of the REMOTE COMMAND consists of the following:

- Executes the fork() function to start a child process
- Sets the following environment variables in the child process:
 - CALIBRE_REMOTE_CONNECTION = <host>:<port>
 - CALIBRE_REMOTE_COUNT = <count> from LAUNCH CLUSTER command

- CALIBRE_REMOTE_MINCOUNT = <min_count> from LAUNCH CLUSTER command
- Executes the execvp() function on the command and any optional arguments in the child process

Examples

Example 1

The following configuration file illustrates the use of the LAUNCH CLUSTER and REMOTE COMMAND statements. Examples of the supporting **command_path** and ARGUMENTS file are also provided. This example uses the rcalibre utility, which is described in more detail in “[rcalibre](#)” on page 398.

Configuration File *config.txt*:

```
LAUNCH CLUSTER COUNT 6 MINCOUNT 2
// MINCOUNT is optional
// script must have execution privileges
// script and argument list must be accessible from primary host
REMOTE COMMAND /user/jsmith/bin/launch_remote.sh
    ARGUMENTS ["/user/jsmith/remote/args"]
```

/user/jsmith/bin/launch_remote.sh:

```
# This script provides an example of an executable script that contains
# commands for launching a Calibre MTflex server on a remote host.

# The CALIBRE_HOME is set to the value of the $aoi_home environment
# variable. Calibre controls and accesses the CALIBRE_REMOTE_CONNECTION
# env var to set and communicate the port and process ID for each
# individual connection between the primary host and the remote host(s).

# This script assumes that the remotes all have a mounted local partition
# named "/scratch1" to use for log files. Note the default location for
# the log data on the remote hosts is /tmp.

# n is used to count how many connections are made by the script,
# and compared to the values set by the LAUNCH CLUSTER statement for
# maximum and minimum connects.
```

```

# In the for loop below, 'cat $1' takes the ARGUMENTS specified as
# a filename in the configuration file "config.lc" and outputs the
# contents as an argument list for the "for" loop.

# This script is using the standard OS utility rsh for the remote shell.
# Any other remote shell utility, such as ssh, may be substituted for rsh.

# In this script, the CALIBRE_REMOTE_COUNT is checked at the end of the
# loop to stop connecting remotes once the limit is met.

# Note the CALIBRE_REMOTE_MINCOUNT is not used here; the primary host
# process keeps track of the number of successful connections and
# terminates the entire Calibre run if the number of successful
# connections is less than the specified CALIBRE_REMOTE_MINCOUNT.

#!/bin/sh
n=0
for node in 'cat $1'
do
    rsh $node $CALIBRE_HOME/bin/rcalibre "/scratch1" 1 CALIBRE_HOME \
        $aoi_home -mtflex $CALIBRE_REMOTE_CONNECTION &
    rsh $node $CALIBRE_HOME/bin/rcalibre "/scratch1" 1 CALIBRE_HOME \
        $aoi_home -mtflex $CALIBRE_REMOTE_CONNECTION &
    n='expr $n + 2';
    if [ $n -eq $CALIBRE_REMOTE_COUNT ]
    then break;
    fi
done

```

/user/jsmith/remote/args:

```

node91
node92
node93
node94
node95
node90

```

Executing the following command calls the configuration file, which sets up the Calibre MTflex run using the preceding files:

```
calibre -drc -hier -turbo ... -remotefile config.txt ...
```

Example 2

Refer to [Example 7](#) in the “Calibre nmLVS and Calibre nmLVS-H Command Line” topic of the *Calibre Verification User’s Manual* for an example that uses the REMOTE COMMAND statement to launch an LVS compare process on a remote host.

Related Topics

[LAUNCH CLUSTER](#)

[CALIBRE_REMOTE_CONNECTION](#)

[CALIBRE_REMOTE_COUNT](#)

[**CALIBRE_REMOTE_MINCOUNT**](#)

[**rcalibre**](#)

[**Using the LAUNCH CLUSTER and REMOTE COMMAND Statements**](#)

[**Executing Calibre Using IBM Spectrum LSF**](#)

[**rxcalibrate**](#)

REMOTE HOST

Provides information for launching a remote host when using LAUNCH AUTOMATIC.

Usage

```
REMOTE HOST machineName [cpuCount] [MGC_HOME mgcPath] [DIR directory]  
[MAXMEM limit] [LAUNCHNAME name] [MGC_LIB_PATH lib_path]  
[RSH shell_path]
```

Arguments

- ***machineName***

A required argument specifying the network name of a remote host to be used for launching a remote server. A ***machineName*** can only appear once in a configuration file.

- ***cpuCount***

An optional argument specifying the number of CPUs that Calibre should use on the remote host. You should specify this value when you want to use fewer CPUs on the machine than are available. The default is for Calibre to use all CPUs. Specifying a -turbo value overrides this argument. For example, if you specify a *cpuCount* of 16 and -turbo 2, the job will run on two CPUs.

- **MGC_HOME *mgcPath***

An optional keyword set that specifies the path to the Calibre executable for use by the remote host. You must specify this keyword if the remote host is a different platform from the primary host. The default behavior is to use the same Calibre executable as the primary host.

- **DIR *directory***

An optional keyword set that specifies the directory in which to write the remote log file. The log file is normally deleted when the run completes without Calibre MTflex errors. Only the remote host specified by ***machineName*** needs to access this directory. The default directory is */tmp*.

The MONITOR REMOTE statement can be used to print the remote host current memory usage at specified intervals and is not deleted at the end of the run.

- **MAXMEM *limit***

An optional keyword set that specifies a value by which to limit the LVHEAP for each remote process. You express *limit* in units of MB (for example, MAXMEM 1000).

The MAXMEM *limit* you specify is per process. For example, if you are using two remote CPUs, then you should set the *limit* to half of the total RAM of the remote hosts.

When using hyperscaling, the recommended MAXMEM *limit* setting is:

$$\textit{limit} = (\text{Total RAM of remote hosts} / \text{Total CPU count of remote hosts}) / 5$$

This is a conservative number and prevents the remote hosts from going into swap memory.

When using MAXMEM and the remote memory usage exceeds MAXMEM, then the application executes the remote job locally and re-initializes the remote host. If this happens, then the application reports this information similar to the following in the transcript:

```
Note: Memory limit exceeded on remote host: host_inet_address,  
executing task on primary host  
...  
Note: Reinitialized remote host: host_inet_address
```

- **LAUNCHNAME *name***

An optional keyword set that specifies the name of the primary host to connect to. This can be useful for specifying unique IP names when the primary host has multiple network connections. If you omit this keyword, then Calibre uses the value you specify in the LAUNCH AUTOMATIC statement.

- **MGC_LIB_PATH *lib_path***

An optional keyword set that specifies the search path for Calibre API libraries, such as MAPI. It defaults to the value on the primary host.

- **RSH *shell_path***

An optional keyword set that specifies the path to a remote shell (RSH) or secure shell (SSH) command. If specified, this value can be overridden using the RSH keyword in the REMOTE HOST statement.

Note

 The RHEL/CentOS 8 and subsequent versions only support the use of SSH.

Description

The REMOTE HOST statement provides information for launching a remote server on a remote host when using the **LAUNCH AUTOMATIC** command. You can specify this statement any number of times in the same configuration file but only one per line, and a **machineName** can only appear once in a configuration file.

Examples

Example 1

The following example illustrates the differences in the Calibre MTflex summary at the end of the application's transcript when using MAXMEM:

```
REMOTE CPU 192.168.1.2: CPU TIME = 2855, STATUS = OK(0),  
LVHEAP = 1/1001/1001, ...
```

where:

- **STATUS = OK(*n*)**

Specifies the number of times (*n*) the application re-initialized the remote host.

- LVHEAP = 1/1001/1001

Specifies the *actual* memory consumption on the remote host.

Example 2

The following configuration file illustrates the use of the LAUNCH AUTOMATIC statement and specifying the location of remote hosts using the REMOTE HOST statement:

```
// MTflex configuration file -- automatic

LAUNCH AUTOMATIC
REMOTE HOST linuxone 2 MGC_HOME /$INSTALL_DIR/aoi/Mgc_home
REMOTE HOST linuxtwo 2 MGC_HOME /$INSTALL_DIR/aoi/Mgc_home
REMOTE HOST linuxthree 4 MGC_HOME /$INSTALL_DIR/aoi/Mgc_home
```

Related Topics

[LAUNCH AUTOMATIC](#)

[MONITOR REMOTE](#)

[CALIBRE_MTFLEX_REMOTE_MAXMEM](#)

[Running Calibre MTflex in a Homogeneous Environment](#)

Transcript Reference

The Calibre transcript contains useful information about the local and remote hosts when Calibre is run in a distributed mode.

MONITOR LOCAL LOAD Transcript	349
Calibre MTflex Transcript	352

MONITOR LOCAL LOAD Transcript

Output for: MONITOR LOCAL LOAD statement

The LOAD option for the MONITOR LOCAL statement enables internal task monitoring of Calibre MTflex operations on the primary host. This reference page describes the parameters that can be found in the transcript output when you use the MONITOR LOCAL LOAD statement.

Format

Monitoring information in the transcript can be identified by the following:

- The information is enclosed in angle brackets (<>).
- The information contains the string “MON” at the beginning of each line.

Parameters

- <MON:B *timestamp* O *n1*>

The MON:B statement indicates the start of the SVRF command and includes the following information:

timestamp — Identifies the number of seconds since process invocation.

O *n1* — Identifies the type of operation. For example, a value of 14 identifies the INTERACT command.

MON:B and MON:E occur in pairs, representing the beginning and end of each SVRF command, with matching operation codes (*n1*).

- <MON:S *timestamp*>

The MON:S statement marks the beginning of a parallel threaded loop and includes the following information:

timestamp — Identifies when the parallel threaded loop started.

MON:S and MON:F occur in pairs, representing the beginning and end of the parallel threaded loop.

- <MON:L *timestamp* T *n1 n2* Q *n3 n4* R *n5* C *n6 n7*>

The MON:L statement provides information about the state of the parallel loop during execution and includes the following information:

timestamp — Identifies when the snapshot of the parallel loop was taken.

T *n1 n2* — Identifies the number of active cells (*n1*) and active stripe threads (*n2*).

Q *n3 n4* — Identifies the cell queue length (*n3*) and stripe queue length (*n4*).

R *n5* — Identifies the number of active threads executing remote operations. You can determine if there are some threads running on the primary host when the number of active threads executing on a remote host (*n5*) is much less than the number of active

threads ($n1$). This increases the load on the primary host and degrades the scalability. The ideal situation is to have all threads executing remotely.

C $n6\ n7$ — The $n7$ value represents the total number of cells to be processed in the parallel loop and $n6$ represents the number of cells that have completed processing. For MDP, FRACTURE, and MPC operations, the $n6$ and $n7$ values represent sections instead of cells.

For non-FRACTURE operations, the total number of cells is typically reported in the “CELL AND PLACEMENT SUMMARY” of the transcript. Each cell loop processes all of the cells and is marked by a MON:S - MON:F pair in the log file. If the cell loop takes longer than the MON:L interval, then MON:L statements will start printing intermediate progress reports. Short cell loops have no MON:L intermediate data, but instead provide average scaling for the whole loop, which is R/W where R=Remote CPU time and W=Wall or REAL time from the MON:F statement.

- <MON:F *timestamp* W $n1$ L $n2$ R $n3$ >

The MON:F statement marks the end of the parallel threaded loop.

timestamp — Identifies when the parallel threaded loop ended.

W $n1$ — Identifies the REAL time for the loop.

L $n2$ — Identifies the primary CPU time for the loop.

R $n3$ — Identifies the total remote CPU times for the loop.

It is not uncommon for some SVRF operations to have more than one parallel loop. The summation of the times from all loops may be used to compute the scaling for the associated SVRF operation. Scaling is defined as the ratio of the CPU time to the REAL time $(n2+n3)/n1$.

- <MON:E *timestamp* O $n1$ >

The MON:E statement indicates the end of the SVRF command.

O $n1$ — Identifies the type of operation.

MON:B and MON:E occur in pairs, representing the beginning and end of each SVRF command, with matching operation codes ($n1$).

Examples

CELL AND PLACEMENT SUMMARY			
CELL TYPE	CELLS	PLACEMENTS	FLAT PLACEMENTS
...			
USER	4699	325559	5418406
VERY SMALL	44	15755	463995
TOP LAYER	16	1242	131781
VERY SMALL	13	1214	130097
PSEUDO	6230	71648	2851958
TOTAL	10929	397207	8270364

```

Completed cell 'ICV_5547'.  TIMESTAMP 22903.  LVHEAP = 1249/1494/1494.
<MON:L 23557 T 0 4 Q 0 0 R 4 C 10926 10929>
Starting cell 'ICV_6230'.  Finished 10926 of 10929.  TIMESTAMP 23744.
Completed cell 'ICV_5191'.  TIMESTAMP 23744.  LVHEAP = 1195/1494/1494.
Cell 'ICV_6230': 400 tiles
<MON:L 24160 T 0 25 Q 0 0 R 25 C 10927 10929>
<MON:L 24792 T 0 7 Q 0 0 R 7 C 10927 10929>
<MON:L 25438 T 0 12 Q 0 0 R 12 C 10927 10929>
<MON:L 26056 T 0 10 Q 0 0 R 10 C 10927 10929>
<MON:L 26684 T 0 16 Q 0 0 R 16 C 10927 10929>
<MON:L 27324 T 0 8 Q 0 0 R 8 C 10927 10929>
<MON:L 28049 T 0 6 Q 0 0 R 6 C 10927 10929>
Starting cell 'chip100'.  Finished 10927 of 10929.  TIMESTAMP 28344.
Completed cell 'ICV_6230'.  TIMESTAMP 28344.  LVHEAP = 1286/1494/1494.
Cell 'chip100': 1681 tiles
<MON:L 28650 T 0 83 Q 0 0 R 83 C 10928 10929>
<MON:L 29269 T 0 25 Q 0 0 R 25 C 10928 10929>
<MON:L 29872 T 0 14 Q 0 0 R 14 C 10928 10929>
<MON:L 30485 T 0 15 Q 0 0 R 15 C 10928 10929>
<MON:L 31091 T 0 9 Q 0 0 R 9 C 10928 10929>
<MON:L 31696 T 0 5 Q 0 0 R 5 C 10928 10929>
<MON:L 32559 T 0 4 Q 0 0 R 4 C 10928 10929>
<MON:L 33428 T 0 5 Q 0 0 R 5 C 10928 10929>
<MON:L 34483 T 0 3 Q 0 0 R 3 C 10928 10929>
...
<MON:F 3887 W 668 L 235 R 9410>
<MON:E 3887 O 14>
drop::TMP<3> (HIER-FMF TYP=1 CFG=0 HGC=0 FGC=0 HEC=0 FEC=0 VHC=FVPC=F)
CPU TIME = 235 + 9410 REAL TIME = 668 LVHEAP = 2885/10598/10599
OPS COMPLETE = 7 OF 32

```

Related Topics

[MONITOR LOCAL](#)

Calibre MTflex Transcript

Output for: Calibre jobs run in Calibre MTflex mode

This reference page describes the remote host parameters that can be found in the transcript output from a running a Calibre job in Calibre MTflex mode.

Format

Remote host information in the transcript can be identified by the “//” characters followed by the string “REMOTE HOST *hostname*:”, where *hostname* is the name of the remote host about which the subsequent information is being reported.

Parameters

- np
The number of physical CPUs available on the remote host. When used in the Calibre documentation, the term “CPU” refers to a physical CPU core unless stated otherwise.
- nv
The number of virtual cores on the remote host. This value only appears in the transcript if the processor is Simultaneous Multithreading (SMT) capable. SMT processors are indicated in the Calibre log file by the presence of “nv =” and terminology such as “SMT enabled”.
- nc
The number of “chip socket packages” on the remote host. This value, which is currently unused by Calibre, is typically “2” for remote hosts. Large Calibre primary hosts may have “4” or “8” chip packages. Remote hosts with more than “2” packages are not recommended for OPC operations due to poor performance. The number of packages may be unknown on older systems, in which case this value defaults to “1”.
- ns
The number of connected logical CPU cores on the remote host. This number is smaller than np when all CPUs are not assigned to the Calibre job due to a -turbo limit or a configuration file setting for Calibre MTflex. This number is also equal to the number of Calibre remote compute server (RCS) processes running on the remote host prior to any -hyper replication of RCS.

Examples

Example 1

The following transcript indicates there are eight CPUs on the remote host (np = 8), two chip sockets on the remote host (nc = 2), and all eight CPUs are assigned to the Calibre job (ns = 8).

```
...
// MTflex initialization for pseudo HDBs completed.
// Running on 8 CPUs (pending licensing)
// List of connected remote hosts:
// REMOTE HOST node1234: np = 8, nc = 2, ns = 8
// MTflex CPU resources: 16 Local, 8/8 Remote
...
```

Example 2

The following transcript is for a Calibre job running with hyperthreading enabled. There are eight CPUs on the remote host (np = 8), eight virtual cores on the remote host (nv = 8), two chip sockets on the remote host (nc = 2), and 16 logical CPU cores assigned to the Calibre job (ns = 16).

```
...
// MTflex initialization for pseudo HDBs completed.
// Running on 200 CPUs (pending licensing)
// List of connected remote hosts:
// REMOTE HOST node4321: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4322: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4323: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4324: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4325: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4326: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4327: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4328: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4329: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4330: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4331: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4332: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4333: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4334: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4335: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4336: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4337: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4338: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4339: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4340: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4341: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4342: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4343: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4344: np = 8, nv = 8, nc = 2, ns = 16
// REMOTE HOST node4345: np = 8, nv = 8, nc = 2, ns = 16
// MTflex CPU resources: 16 Local, 400/200 Remote
//
// LITHO operations available for use on 200 CPUs (pending licensing)
//
...
```

Distributed Calibre Environment Variables

You can use environment variables to set parameters for multithreaded (MT) and Calibre MTflex runs.

CALIBRE_FLATTEN_LIKE_HYPER	355
CALIBRE_HDBFLEX_MIN	356
CALIBRE_HYPER_SMTFACTOR	357
CALIBRE_HYPER_RDSMULTIPLE	359
CALIBRE_HYPERFLEX_SAVE_LOGS.....	360
CALIBRE_LVS_REMOTE_CONNECTION	361
CALIBRE_MT_MONITOR_LOAD	362
CALIBRE_MT_MONITOR_MEMORY	363
CALIBRE_MT_SMT_FACTOR.....	364
CALIBRE_MTFLEX_HYPER	365
CALIBRE_MTFLEX_LAUNCH.....	366
CALIBRE_MTFLEX_LOCAL_HOST.....	367
CALIBRE_MTFLEX_LOCAL_HOST_DIR	368
CALIBRE_MTFLEX_MONITOR_LOCAL	369
CALIBRE_MTFLEX_MONITOR_REMOTE	370
CALIBRE_MTFLEX_REMOTE_DIR.....	371
CALIBRE_MTFLEX_REMOTE_MAXMEM.....	372
CALIBRE_MTFLEX_SAVE_LOGS	373
CALIBRE_REMOTE_CONNECTION	374
CALIBRE_REMOTE_COUNT	375
CALIBRE_REMOTE_MINCOUNT.....	376

CALIBRE_FLATTEN_LIKE_HYPER

Flattens selected very small cells and selected top layer cells.

Usage

CALIBRE_FLATTEN_LIKE_HYPER YES

Arguments

- **YES**

Specifies that very small cells and selected top layer cells should be flattened.

Description

This variable flattens selected very small cells and selected top layer cells. In general, it improves Calibre MTflex performance because these cells are better processed as geometries inside their containing cells rather than remaining as cells. The algorithm that selects the cells to flatten is identical to one used when hyperscaling (-hyper) is performed.

This variable is ignored when hyperscaling (-hyper) is specified.

Examples

C Shell

```
% setenv CALIBRE_FLATTEN_LIKE_HYPER YES
```

Bourne/Korn Shell

```
$ CALIBRE_FLATTEN_LIKE_HYPER=YES
$ export CALIBRE_FLATTEN_LIKE_HYPER
```

CALIBRE_HDBFLEX_MIN

Specifies the behavior for the MINCOUNT argument when HDBFLEX is enabled.

Usage

CALIBRE_HDBFLEX_MIN [0 | 1 | 2]

Arguments

- 0
Ignores the MINCOUNT value.
- 1
Generates a warning when an issue occurs and ignores the MINCOUNT value (default).
- 2
Aborts the Calibre run when an issue occurs.

Description

Use the CALIBRE_HDBFLEX_MIN environment variable to control the behavior of MINCOUNT when enabling HDBFLEX by specifying calibre -drc -turbo -hdbflex. With -hdbflex, remotes are not connected early in the Calibre process, so fatal connections to remotes are not known until after HDB construction. The MINCOUNT argument (for [LAUNCH AUTOMATIC](#), [LAUNCH CLUSTER](#), or [LAUNCH MANUAL](#)) determines the minimum number of connections that must be established in order for the Calibre run not to fail. You can use the CALIBRE_HDBFLEX_MIN variable to control the behavior of MINCOUNT.

Refer to “[Calibre nmDRC and Calibre nmDRC-H Command Line](#)” in the *Calibre Verification User’s Manual* for more information on the -hdbflex argument.

Examples

C Shell

```
% setenv CALIBRE_HDBFLEX_MIN 2
```

Bourne/Korn Shell

```
$ CALIBRE_HDBFLEX_MIN=2
$ export CALIBRE_HDBFLEX_MIN
```

CALIBRE_HYPER_SMTFACTOR

Launches additional threads on remote CPUs connected only to HDB0 and that support SMT processing.

Usage

CALIBRE_HYPER_SMTFACTOR [0 | 1]

Arguments

- 0
Disables the ability to launch additional threads on the virtual CPU cores (default).
- 1
Enables the ability to launch additional threads on the virtual CPU cores.

Description

This variable is used to enable or disable the ability to launch additional threads on CPUs connected only to HDB0 and that support SMT processing. This feature should be used in Litho flows to provide optimal performance for non-hyperscaled operations that run only on HDB0. All remote CPUs must be used for the job and licensed for this feature to work.

For Calibre jobs running in multithreaded (MT) mode with hyperscaling on an SMT-enabled system, set this variable to 1 in order to launch two Calibre processes per physical CPU for operations running on HDB 0. If running in Calibre MTflex mode with hyperscaling, you can enable this functionality by using the HYPER statement with the SMTFACTOR keyword in a configuration file or by setting the CALIBRE_MTFLEX_HYPER environment variable when there is no configuration file.

For Litho flows, a warning is issued if this variable is not specified for jobs running with hyperscaling on SMT-enabled systems. When this variable is set, the virtual cores are reported in the Calibre transcript.

It is recommended that you set this variable to 0 for jobs with unusually high memory requirements, such as Litho jobs, when the available memory is insufficient to support the additional threads.

Examples

C Shell

```
% setenv CALIBRE_HYPER_SMTFACTOR 1
```

Bourne/Korn Shell

```
$ CALIBRE_HYPER_SMTFACTOR=1
$ export CALIBRE_HYPER_SMTFACTOR
```

Related Topics

[HYPER](#)

[CALIBRE_MTFLEX_HYPER](#)

[Simultaneous Multithreading With Virtual CPUs](#)

CALIBRE_HYPER_RDSMULTIPLE

Enables or disables the addition of one Remote Data Server (RDS) for every remote CPU during a Calibre job.

Usage

CALIBRE_HYPER_RDSMULTIPLE [0 | 1]

Arguments

- 0

Disables the ability to add one RDS for every remote CPU (default).

- 1

Enables the ability to add one RDS for every remote CPU.

Description

This variable is used to enable or disable the ability to add one RDS for every remote CPU during a Calibre job. This option is useful when working with a job scheduler (such as IBM Spectrum LSF or Univa Grid Engine), where there may be an uneven distribution of CPUs per remote host, causing the RDS space to be hard to predict.

Examples

C Shell

```
% setenv CALIBRE_HYPER_RDSMULTIPLE 1
```

Born/Korn Shell

```
$ CALIBRE_HYPER_RDSMULTIPLE=1
$ export CALIBRE_HYPER_RDSMULTIPLE
```

Related Topics

[HYPER](#)

[-remotedata](#)

CALIBRE_HYPERFLEX_SAVE_LOGS

Specifies that PHDB log files are saved (instead of deleted) at the end of a run.

Usage

CALIBRE_HYPERFLEX_SAVE_LOGS *value*

Arguments

- *value*

An optional value that when set to a non-null value saves the PHDB log files generated by a hyperscaled (-hyper) run.

Description

An environment variable used to specify that PHDB log files are saved rather than deleted at the end of a run. By default, PHDB log files are automatically deleted when Calibre successfully terminates from the server.

Examples

C Shell

```
% setenv CALIBRE_HYPERFLEX_SAVE_LOGS yes
```

Bourne/Korn Shell

```
$ CALIBRE_HYPERFLEX_SAVE_LOGS=yes
$ export CALIBRE_HYPERFLEX_SAVE_LOGS
```

Related Topics

[Hyperscaling](#)

CALIBRE_LVS_REMOTE_CONNECTION

Specifies the primary host and port information for a remote host to connect with the primary host when launching a separate LVS compare process on a remote host. This variable is set internally by the tool.

Usage

CALIBRE_LVS_REMOTE_CONNECTION *host:port*

Arguments

- ***host***
A required value specifying the hostname of the primary host.
- ***port***
A required value specifying the port number of the primary host.

Description

An environment variable specifying the name and port information of the primary host. This information is used by the remote host to connect with the primary host. You reference this variable with the [-par_cmp_remote](#) argument when you want to launch LVS comparison on a remote machine using the [-cmp_remotefile](#) command line argument. These options are discussed in the *Calibre Verification User's Manual*.

This variable is available to scripts that run as a subprocess of “calibre”, that is, the ***command_path*** argument of [REMOTE COMMAND](#). The value of this variable is not available to the [-cmp_remotefile](#) argument, as the file specified by [-cmp_remotefile](#) does not do variable substitution. However, [-cmp_remotefile](#) does do substitution of %H:%P constructs as part of a [REMOTE COMMAND](#) specification.

Examples

Refer to “[Example 7](#)” in the “Calibre nmLVS and Calibre nmLVS-H Command Line” topic of the *Calibre Verification User's Manual*.

CALIBRE_MT_MONITOR_LOAD

Specifies a time interval for writing monitoring (MON) statements to the transcript.

Usage

CALIBRE_MT_MONITOR_LOAD *interval*

Arguments

- *interval*

Specifies the time in seconds between writing the LOAD associated MON statements (MON:L) to the transcript.

Description

When running in multithreaded (MT) mode, this variable specifies the time between writing the LOAD associated MON statements (MON:L) to the transcript. You use this variable in the absence of a configuration file.

Examples

C Shell

```
% setenv CALIBRE_MT_MONITOR_LOAD 60
```

Bourne/Korn Shell

```
$ CALIBRE_MT_MONITOR_LOAD=60
$ export CALIBRE_MT_MONITOR_LOAD
```

Related Topics

[MONITOR LOCAL](#)

CALIBRE_MT_MONITOR_MEMORY

Specifies a time interval for writing LVHEAP memory information to the transcript.

Usage

CALIBRE_MT_MONITOR_MEMORY *interval*

Arguments

- *interval*

Specifies a time in seconds between writing LVHEAP memory information to the transcript.

Description

When running in multithreaded (MT) mode, this variable specifies the time between writing LVHEAP memory information to the transcript.

Examples

C Shell

```
% setenv CALIBRE_MT_MONITOR_MEMORY 60
```

Bourne/Korn Shell

```
$ CALIBRE_MT_MONITOR_MEMORY=60
$ export CALIBRE_MT_MONITOR_MEMORY
```

CALIBRE_MT_SMT_FACTOR

Enables or disables support for SMT processing when running in multithreaded (MT) mode.

Usage

CALIBRE_MT_SMT_FACTOR [0 | 1]

Arguments

- 0
Disables support for SMT processing when it is enabled in the BIOS.
- 1
Enables support for SMT processing when it is enabled in the BIOS (default).

Description

This variable is used to enable or disable support for SMT processing when running in MT mode. Setting this variable to 0 overrides any other SMT-related settings.

Examples

C Shell

```
% setenv CALIBRE_MT_SMT_FACTOR 0
```

Bourne/Korn Shell

```
$ CALIBRE_MT_SMT_FACTOR=0
$ export CALIBRE_MT_SMT_FACTOR
```

Related Topics

[Simultaneous Multithreading With Virtual CPUs](#)

CALIBRE_MTFLEX_HYPER

Assigns values to arguments for the HYPER statement.

Usage

CALIBRE_MTFLEX_HYPER [LICENSE ALL] [SMTFACTOR [0 | 1]]

Arguments

Refer to the corresponding argument descriptions for the HYPER statement.

Description

This variable provides a mechanism for supplying the same set of information as defined by the HYPER statement. It is valid only when used with the -remote command line option and is ignored if a configuration file is used.

Examples

C Shell

```
% setenv CALIBRE_MTFLEX_HYPER "smtfactor 1"
```

Bourne/Korn Shell

```
$ CALIBRE_MTFLEX_HYPER="smtfactor 1"  
$ export CALIBRE_MTFLEX_HYPER
```

Related Topics

[HYPER](#)

CALIBRE_MTFLEX_LAUNCH

Assigns values to arguments for the LAUNCH AUTOMATIC, LAUNCH MANUAL, or LAUNCH CLUSTER statements.

Usage

CALIBRE_MTFLEX_LAUNCH

```
“[AUTOMATIC [RSH shell_path] | MANUAL | CLUSTER]
[WAIT n_seconds] [PORT port_number] [COUNT n_connections]
[MINCOUNT min_connects] [NAME {IPaddress | hostname}]”
```

Arguments

Refer to the corresponding argument descriptions for the LAUNCH AUTOMATIC, LAUNCH CLUSTER, and LAUNCH MANUAL statements.

The quotation marks (“ ”) are required.

Description

This variable provides a mechanism for supplying the same set of information as defined by the LAUNCH AUTOMATIC, LAUNCH MANUAL, or LAUNCH CLUSTER statement. It is valid only when used with the -remote command line option and is ignored if a configuration file is used.

The list of remote hosts (specified by -remote) is ignored if MANUAL or CLUSTER mode is specified.

Examples

C Shell

```
% setenv CALIBRE_MTFLEX_LAUNCH "AUTOMATIC NAME 192.168.1.14 WAIT 30 \
RSH /usr/bin/ssh"
```

Bourne/Korn Shell

```
$ CALIBRE_MTFLEX_LAUNCH="AUTOMATIC NAME 192.168.1.14 WAIT 30 \
RSH /usr/bin/ssh"
$ export CALIBRE_MTFLEX_LAUNCH
```

Related Topics

[LAUNCH AUTOMATIC](#)

[LAUNCH CLUSTER](#)

[LAUNCH MANUAL](#)

CALIBRE_MTFLEX_LOCAL_HOST

Assigns values to arguments for the LOCAL HOST DIR and LOCAL HOST COMPUTE statements.

Usage

CALIBRE_MTFLEX_LOCAL_HOST “[DIR path] [WORKER num_workers]”

Arguments

Refer to the corresponding argument descriptions for the LOCAL HOST DIR or LOCAL HOST COMPUTE statements. The quotation marks (“ ”) are required.

Description

Use this variable to specify values for the local (primary) host that can otherwise be defined by the LOCAL HOST DIR or LOCAL HOST COMPUTE configuration file statements. It is valid only when used with the -remote command line option and is ignored if a configuration file is used.

Examples

C Shell

```
% setenv CALIBRE_MTFLEX_LOCAL_HOST "DIR /user/name/files/"
```

Bourne/Korn Shell

```
$ CALIBRE_MTFLEX_LOCAL_HOST="DIR /user/name/files/"  
$ export CALIBRE_MTFLEX_LOCAL_HOST
```

Related Topics

[LOCAL HOST DIR](#)

[CALIBRE_MTFLEX_LOCAL_HOST](#)

[CALIBRE_MTFLEX_LOCAL_HOST_DIR](#)

[-remote](#)

CALIBRE_MTFLEX_LOCAL_HOST_DIR

Specifies a path on the primary host where Calibre can store temporary files created by some applications on remote hosts.

Usage

CALIBRE_MTFLEX_LOCAL_HOST_DIR “*path*”

Arguments

- “*path*”

Specifies a path on the primary host for storing temporary files created on remote hosts by some tools, such as Calibre FRACTURE and Calibre MDPmerge. The specified directory must be accessible by the primary host. The quotation marks (“ ”) are required.

Description

Use this variable to specify a path on the primary host where Calibre can store temporary files created on remote hosts by some tools, such as Calibre FRACTURE and Calibre MDPmerge.

This variable provides a mechanism for supplying the same set of information as defined by the LOCAL HOST DIR statement. It is valid when used with the -remote or -remotefile command line option.

Examples

C Shell

```
% setenv CALIBRE_MTFLEX_LOCAL_HOST_DIR "/user/name/files/"
```

Bourne/Korn Shell

```
$ CALIBRE_MTFLEX_LOCAL_HOST_DIR="/user/name/files/"  
$ export CALIBRE_MTFLEX_LOCAL_HOST_DIR
```

Related Topics

[LOCAL HOST DIR](#)

[CALIBRE_MTFLEX_LOCAL_HOST](#)

CALIBRE_MTFLEX_MONITOR_LOCAL

Enables monitoring of Calibre MTflex operations on the primary host.

Usage

CALIBRE_MTFLEX_MONITOR_LOCAL “[INTERVAL *seconds*] [CONNECT] [LOAD]
[MEMORY] [IO]”

Arguments

Refer to the corresponding argument descriptions for the MONITOR LOCAL statement. The quotation marks (“ ”) are required.

Description

This variable provides a mechanism for supplying the same set of information as defined by the MONITOR LOCAL statement. It is valid only when used with the -remote command line option and is ignored if a configuration file is used.

Examples

C Shell

```
% setenv CALIBRE_MTFLEX_MONITOR_LOCAL "IO CONNECTION"
```

Bourne/Korn Shell

```
$ CALIBRE_MTFLEX_MONITOR_LOCAL="IO CONNECTION"  
$ export CALIBRE_MTFLEX_MONITOR_LOCAL
```

Related Topics

[MONITOR LOCAL](#)

[-remote](#)

[About the Configuration File](#)

CALIBRE_MTFLEX_MONITOR_REMOTE

Enables monitoring of Calibre MTflex operations on the remote host.

Usage

CALIBRE_MTFLEX_MONITOR_REMOTE “[INTERVAL *seconds*] [CONNECT]
[MEMORY]”

Arguments

Refer to the corresponding argument descriptions for the MONITOR REMOTE statement. The quotation marks (“ ”) are required.

Description

This variable provides a mechanism for supplying the same set of information as defined by the MONITOR REMOTE statement. It is valid only when used with the -remote command line option and is ignored if a configuration file is used.

Examples

C Shell

```
% setenv CALIBRE_MTFLEX_MONITOR_REMOTE "INTERVAL 60 MEMORY"
```

Bourne/Korn Shell

```
$ CALIBRE_MTFLEX_MONITOR_REMOTE="INTERVAL 60 MEMORY"  
$ export CALIBRE_MTFLEX_MONITOR_REMOTE
```

Related Topics

[MONITOR REMOTE](#)

[-remote](#)

[About the Configuration File](#)

CALIBRE_MTFLEX_REMOTE_DIR

Specifies a directory on the remote host for writing log files.

Usage

CALIBRE_MTFLEX_REMOTE_DIR *path*

Arguments

- *path*

Specifies the path where the remote log file is written. Only the remote host needs access to this directory. By default, remote log files are written to */tmp*.

Description

This variable is used to specify a directory on the remote host for writing log files. This variable is valid only when using the **-remote** command line option and is ignored if a configuration file is used.

Examples

C Shell

```
% setenv CALIBRE_MTFLEX_REMOTE_DIR /calibre/logs
```

Bourne/Korn Shell

```
$ CALIBRE_MTFLEX_REMOTE_DIR=/calibre/logs
$ export CALIBRE_MTFLEX_REMOTE_DIR
```

Related Topics

[-remote](#)

[About the Configuration File](#)

CALIBRE_MTFLEX_REMOTE_MAXMEM

Specifies an LVHEAP limit to use on a remote host.

Usage

CALIBRE_MTFLEX_REMOTE_MAXMEM *limit*

Arguments

- *limit*

Specifies a value (in MB) used to set the LVHEAP limit on a remote host. The value you specify is per process. For example, if you are using two remote CPUs, then you should set the *limit* to half of the total RAM of the remote hosts.

When using hyperscaling, the recommended setting for *limit* is:

limit = (Total RAM of remote hosts / Total CPU count of remote hosts) / 5

This is a conservative number and prevents the remote hosts from going into swap memory.

Description

This environment variable can be set on each remote host to specify an LVHEAP limit. This variable is equivalent to setting the MAXMEM argument for the REMOTE HOST statement. The units are in megabytes (MB). This variable has no effect on the primary host and is ignored if a configuration file is used.

Examples

C Shell

```
% setenv CALIBRE_MTFLEX_REMOTE_MAXMEM 1000
```

Bourne/Korn Shell

```
$ CALIBRE_MTFLEX_REMOTE_MAXMEM=1000
$ export CALIBRE_MTFLEX_REMOTE_MAXMEM
```

Related Topics

[REMOTE HOST](#)

[About the Configuration File](#)

CALIBRE_MTFLEX_SAVE_LOGS

Specifies whether the Remote Compute Server (RCS) log files and CalibreRRLog file are saved at the end of a run.

Usage

CALIBRE_MTFLEX_SAVE_LOGS *value*

Arguments

- *value*

An optional value that when set to a non-null value causes the Remote Compute Server log files and CalibreRRLog file to be saved rather than deleted at the end of the run.

Description

An environment variable used to specify that the log files from the Remote Compute Server (RCS) processes are saved rather than deleted at the end of a run. Hyperscaling creates pseudo HDBs that are connected to Remote Compute Server processes on each remote CPU. By default, log files are automatically deleted on Remote Compute Servers when Calibre successfully terminates from the server.

If enabled, this environment variable also saves the CalibreRRLog file, which is created when Calibre MTflex warnings or errors are generated.

Examples

C Shell

```
% setenv CALIBRE_MTFLEX_SAVE_LOGS 1
```

Bourne/Korn Shell

```
$ CALIBRE_MTFLEX_SAVE_LOGS=1
$ export CALIBRE_MTFLEX_SAVE_LOGS
```

Related Topics

[Hyperscaling](#)

CALIBRE_REMOTE_CONNECTION

Specifies the primary host and port information for the remote host to use to connect with the primary host.

Usage

CALIBRE_REMOTE_CONNECTION *host:port*

Arguments

- ***host***
A required value specifying the hostname of the primary host.
- ***port***
A required value specifying the port number of the primary host.

Description

An environment variable used to specify the primary host and port information for the remote host to use in order to connect with the primary host. This variable can be used when launching jobs using the rcalibre or rxcalibrate utilities.

The REMOTE COMMAND statement sets the specified ***host:port*** values in a child process.

Examples

C Shell

```
% setenv CALIBRE_REMOTE_CONNECTION linuxzone:1564
```

Bourne/Korn Shell

```
$ CALIBRE_REMOTE_CONNECTION=linuxzone:1564
$ export CALIBRE_REMOTE_CONNECTION
```

Related Topics

[LAUNCH CLUSTER](#)

[REMOTE COMMAND](#)

[rcalibre](#)

[rxcalibrate](#)

[Using the LAUNCH CLUSTER and REMOTE COMMAND Statements](#)

[Executing Calibre Through IBM Spectrum LSF in a Calibre MTflex Environment](#)

CALIBRE_REMOTE_COUNT

Specifies a value for the COUNT argument used in the LAUNCH CLUSTER or LAUNCH MANUAL statement.

Usage

CALIBRE_REMOTE_COUNT

Arguments

None.

Description

An environment variable that is set in a child process by the REMOTE COMMAND statement. Execution of the REMOTE COMMAND automatically sets the CALIBRE_REMOTE_COUNT variable to the value of the COUNT parameter specified in the LAUNCH CLUSTER or LAUNCH MANUAL command.

This variable is used in scripts, for example to stop connecting remotes once the value specified by the COUNT parameter is met.

Examples

The following script uses the CALIBRE_REMOTE_COUNT variable to stop connecting remotes once the COUNT parameter is met.

```
#!/bin/sh
n=0
for node in `cat $1`do
    rsh $node $CALIBRE_HOME/bin/rcalibre "/scratch1" 1 $CALIBRE_HOME \
        $aoi_home -mtflex $CALIBRE_REMOTE_CONNECTION &
    n=`expr $n + 2`;
    if [ $n -eq $CALIBRE_REMOTE_COUNT ]
    then break;
    fi
done
```

For a complete example of how this variable is used, refer to the example provided with the LAUNCH CLUSTER or REMOTE COMMAND statement.

Related Topics

[LAUNCH CLUSTER](#)

[LAUNCH MANUAL](#)

[REMOTE COMMAND](#)

CALIBRE_REMOTE_MINCOUNT

Specifies a value for the MINCOUNT argument used in the LAUNCH CLUSTER statement.

Usage

CALIBRE_REMOTE_MINCOUNT

Arguments

None.

Description

An environment variable that is set in a child process by the REMOTE COMMAND statement. The value of this variable is equal to the MINCOUNT parameter specified by the LAUNCH CLUSTER statement. This variable is used in scripts, for example to terminate a Calibre run if the number of successful connections is less than the specified CALIBRE_REMOTE_MINCOUNT.

Examples

For information on how this variable might be used, refer to the example provided with the LAUNCH CLUSTER or REMOTE COMMAND statement.

Related Topics

[LAUNCH CLUSTER](#)

[REMOTE COMMAND](#)

Command Line Options Reference Dictionary

You can use command line options to control distributed Calibre modes.

Calibre Command Line	377
-hyper	378
-mtflex	381
-remote	383
-remotecmd	385
-remotedata	387
-remotefile	391
-turbo	393
-turbo_all	394
-turbo_litho	395

Calibre Command Line

Enter calibre -help at the shell prompt to display the command line options that are available for each Calibre tool. In the following usage section, *tool_specific_options* is used to refer to options such as -drc or -xact. Refer to the appropriate user documentation for detailed descriptions of the command line options associated with a specific tool. Note that the distributed command line options are not available for all Calibre tools.

Usage

```
calibre { tool_specific_options
  [ { { -turbo [ #CPUs ] [ -turbo_all ] }
    || { -turbo_litho [ #CPUs ] }
    || { -turbo [ #CPUs ] [ -turbo_all ] -turbo_litho [ #CPUs ]
    } } [ { -remote host[,host,...] || -remotefile filename
      || -remotecmd filename count }
    [ -remotedata [ { -recoveroff | -recoverremote } ] ] ]
    [ -hyper [ remote ] ] ]
  } rule_file_name || { -mtflex host_connection options }
```

-hyper

Enables hyperscaling mode, which enables the concurrent, parallel execution of SVRF operations.

Usage

calibre [-lfd]

-hyper

calibre [-drc]

-hyper [connect] [remote]

Note

 You can also use the **-hdbflex** or **-hdbflex_acquire** argument instead of the **-hyper** argument when invoking calibre **-drc -hier**. Refer to the *Calibre Verification User's Manual* for more information on these arguments.

The calibre **-drc -hier** command supports the use of the **-hdbflex** or **-hdbflex_acquire** arguments.

calibre [-dfm | -perc]

-hyper [connect] [remote]

calibre [-lvs]

-hyper [pathchk] [remote] [cmp]

Note

 Refer to “[Calibre nmLVS and Calibre nmLVS-H Command Line](#)” in the *Calibre Verification User's Manual* for more information on the pathchk and cmp arguments.

Arguments

- connect

An optional argument that enables connectivity-related operations to be processed by hyperscaling. This argument can be particularly beneficial for antenna checks but can increase memory usage.

- remote

An optional argument that triggers remote pseudo hierarchical database technology. Specifying this argument enables the **-remotedata** argument by default, which enables the Remote Data Server (RDS) technology. The number of parallel operations increases when **-hyper remote** is used in conjunction with the RDS technology.

The hyper remote functionality distributes pseudo hierarchical databases (HDBs) across remote hosts, thereby reducing memory requirements on the primary host and improving Calibre MTflex performance. This functionality is intended to be used on large designs that take multiple hours to run. A distributed computing environment is required with a

minimum of four remote hosts, while 24 remote CPUs are recommended for best performance.

The primary host can be as small as a two CPU machine with as much memory as necessary to complete the job. While a four CPU primary host is preferred, two CPUs is sufficient and has minimal performance penalties. Refer to the “[Hyper Remote](#)” section for more information on this capability.

Description

The -hyper argument specifies hyperscaling mode, which enables the concurrent, parallel execution of SVRF operations. Hyperscaling can only be used in an MT or Calibre MTflex environment and must be specified with -turbo, -turbo_litho, or both options.

Note

 The -hyper switch should not be specified for a single-threaded run. If you specify -hyper and do not specify -turbo, Calibre exits with an error message. If you specify -hyper and -turbo 1, Calibre issues a warning and continues:

```
WARNING: HYPERSCALING requested but only 1 cpu available, continuing without it.
```

Examples

Example 1

This example invokes Calibre nmDRC-H and specifies multithreaded parallel processing. Pseudo HDBs (PHDBs) are distributed across the remote hosts and the recovery option is disabled.

```
calibre -drc -hier -turbo -hyper remote -remotedata -recoveroff  
-remotefile remote.config my_rules
```

Example 2

The following transcript identifies the number of Remote Pseudo HDBs that are created based on the remote hosts configuration. Each multi-CPU remote host is issued one Pseudo HDB. In this example, ten remote hosts are used, each with four CPUs per host, thus a total of ten Pseudo HDBs are created.

```
// Determining Remote Pseudo HDBs.  
// REMOTE HOST node55 used for Remote Pseudo HDB  
// REMOTE HOST node40 used for Remote Pseudo HDB  
// REMOTE HOST node58 used for Remote Pseudo HDB  
// REMOTE HOST node42 used for Remote Pseudo HDB  
// REMOTE HOST node56 used for Remote Pseudo HDB  
// REMOTE HOST node41 used for Remote Pseudo HDB  
// REMOTE HOST node44 used for Remote Pseudo HDB  
// REMOTE HOST node43 used for Remote Pseudo HDB  
// REMOTE HOST node57 used for Remote Pseudo HDB  
// REMOTE HOST node59 used for Remote Pseudo HDB  
// Connected to Remote Pseudo HDB 1 on remote host node55  
// Connected to Remote Pseudo HDB 2 on remote host node40  
// Connected to Remote Pseudo HDB 3 on remote host node58  
// Connected to Remote Pseudo HDB 4 on remote host node42  
// Connected to Remote Pseudo HDB 5 on remote host node56  
// Connected to Remote Pseudo HDB 6 on remote host node41  
// Connected to Remote Pseudo HDB 7 on remote host node44  
// Connected to Remote Pseudo HDB 8 on remote host node43  
// Connected to Remote Pseudo HDB 9 on remote host node57  
// Connected to Remote Pseudo HDB 10 on remote host node59  
  
// HYPERSCALING ENABLED with 10 pseudo HDBs.
```

A cumulative summary is expanded for each HDB to specify the CPU and Real time for each category.

```
HDB 0: CPU TIME = 1211 + 10151 MAX LVHEAP = 6664  
HDB 1: CPU TIME = 0 + 21786 MAX LVHEAP = 836  
HDB 2: CPU TIME = 0 + 24803 MAX LVHEAP = 1093  
HDB 3: CPU TIME = 0 + 16712 MAX LVHEAP = 749  
HDB 4: CPU TIME = 0 + 27160 MAX LVHEAP = 960  
HDB 5: CPU TIME = 0 + 23112 MAX LVHEAP = 1168  
HDB 6: CPU TIME = 0 + 23354 MAX LVHEAP = 866  
HDB 7: CPU TIME = 0 + 21426 MAX LVHEAP = 1228  
HDB 8: CPU TIME = 0 + 20366 MAX LVHEAP = 830  
HDB 9: CPU TIME = 0 + 23549 MAX LVHEAP = 1098  
HDB 10: CPU TIME = 0 + 19061 MAX LVHEAP = 916  
TOTAL MAX LVHEAP = 6664 REMOTE = 9744
```

The HDB summary section appears at the end of the transcript and specifies the peak memory consumed per HDB in addition to the peak memory consumed by the primary host. The TOTAL MAX LVHEAP can be distinguished into two parts:

- The first value (LVHEAP = 6664) is the memory consumed by HDB 0 on the primary host.
- The second (REMOTE = 9744) is the sum of memory consumed by the PHDBs across the remote hosts.

Related Topics

[Hyperscaling](#)

[Hyper Remote](#)

-mtflex

Starts a server process on a remote host and connects the remote host to the primary host.

Usage

-mtflex *host_connection* [*options*]

Arguments

- ***host_connection***

Specifies the connection information (IP address and port number) for the primary host. This information is specified using the format:

host_ip_address:host_port_number

- ***options***

A set of optional arguments that can be specified with -mtflex. Options include:

- f — Option that specifies to keep the server process in the foreground. By default, the server process is placed in the background.
- v — Option that specifies verbose mode allowing the server to print more informational messages as it runs.
- maxmem *limit* —Option that enables the MAXMEM capabilities provided by the REMOTE HOST statement.

The *limit* you specify is per process. For example, if you are using two remote CPUs, then you should set the *limit* to half of the total RAM of the remote hosts.

When using hyperscaling, this is the recommended setting:

limit = (Total RAM of remote hosts / Total CPU count of remote hosts) / 5

This is a conservative number and prevents the remote hosts from going into swap memory.

Description

The -mtflex option is used in LAUNCH MANUAL mode to start a server process on a remote host and to connect the remote host to the primary host manually. If your remote host is a multi-processor machine, you must execute calibre -mtflex once for each processor. Once a server is started on the remote host, the server uses the information specified by *host_ip_address:host_port_number* to connect to the primary host and, if successful, become a daemon process.

Examples

In this example, Calibre is invoked on the primary host using the -remotefile argument, and the configuration file contains a LAUNCH MANUAL command. This displays the remote connection information, as follows:

```
// Calibre Remote Connection 172.16.11.22:12345
```

You can then rlogin to the remote host to set arguments (like the CALIBRE_HOME variable and other environment variables), and then manually connect the remote host to the primary host using the -mtflex argument, as follows:

```
calibre -mtflex 172.16.11.22:12345
```

Related Topics

[LAUNCH MANUAL](#)

[REMOTE HOST](#)

[Running Calibre MTflex in a Heterogeneous Environment](#)

-remote

Specifies the names of remote hosts for Calibre MTflex.

Usage

-remote host[,host ...]

Arguments

- **host**

Specifies the hostname of one or more remote hosts. You must specify a minimum of one host with this option. To specify multiple hosts, use a comma to separate the host names.

Description

The -remote option is used in a homogeneous network environment to specify the names of the remote hosts to use for the Calibre MTflex run. This option must be specified with the -turbo or -turbo_litho option. You use this option to enable multithreaded, parallel processing operations on the remote hosts in a distributed network. The required number of licenses must be available to run the job.

This -remote option supports the CALIBRE_MTFLEX_LOCAL_HOST_DIR environment variable definition.

Examples

Example 1

The following example invokes Calibre nmDRC-H and specifies the remote hosts for the Calibre MTflex run as machine1, machine2, and machine3. The -turbo_litho argument instructs Calibre to use multithreaded (MT) parallel processing when performing Litho operations. Because no value is specified for the -turbo and -turbo_litho options, the Calibre nmDRC-H and Litho operations run on the maximum number of available CPUs for which there are licenses. Hyperscaling is also enabled.

```
calibre -drc -hier -turbo -turbo_litho \
        -remote machine1,machine2,machine3 -hyper rules
```

Example 2

The following example invokes Calibre xACT 3D and specifies the remote hosts for the Calibre MTflex run as machine1 and machine2. In this example, the -turbo argument instructs Calibre xACT 3D to use 24 CPUs on the remote hosts.

```
calibre -xact -3d -rcc -xcell xcell_list -turbo 24 \
        -remote machine1,machine2 rules
```

Related Topics

[Calibre MTflex Processing](#)

[CALIBRE_MTFLEX_MONITOR_LOCAL](#)

CALIBRE_MTFLEX_LOCAL_HOST_DIR

-remotecommand

Used to auto-generate a Calibre MTflex configuration file.

Usage

-remotecommand *filename count*

Arguments

- ***filename***

Specifies the path to the file on the primary host containing the commands for invoking a remote server on the remote host.

- ***count***

Specifies the total number of CPUs that Calibre will attempt to connect to for the run.

This behavior is not enabled if -hyper remote or -remotedata is used, because these two modes require static initialization at startup time so all remote CPUs must be connected at startup.

Description

The -remotecommand argument can be used in place of -remotefile to effectively auto-generate a Calibre MTflex configuration file. The purpose of the -remoteoption option is to simplify the execution of Calibre jobs when using IBM Spectrum LSF or Univa Grid Engine by eliminating the need for a configuration file.

Examples

In the following example, the -remoteoption argument specifies the path to a script containing the commands for launching remote servers on remote hosts. The -remoteoption argument also specifies that the primary host connects to four CPUs on the remote hosts.

```
calibre -turbo -drc -hier -hyper -remoteoption /home/jsmith/remote.cmd
        4 rules
```

Using -remotedata, the Calibre MTflex configuration file produced is:

```
LAUNCH CLUSTER MINCOUNT 2 COUNT 4
REMOTE COMMAND /home/jsmith/remote.cmd ARGUMENTS [%H:%P %C]
```

The *remote.cmd* script gets two arguments:

- Argument 1 (\$1) is the “%H:%P” or hostname:port number value for the primary host; for example, hostx:48851.
- Argument 2 (\$2) is the “%C” or total number of CPUs on the remote host to which the primary host will connect; for example, 4.

The *remote.cmd* script contains the following IBM Spectrum LSF command to start remote processes. The “\${1%*:}” string is used to strip off the port number value (48851 in this

example) in order to obtain the hostname of the remote host for use in the remote command file script.

```
bsub -q priority -o <log_file_path> -J "calibreDRC[1-4]"  
-R "select [hostname!=${1%%:*} && type==X86_64]"  
"/<path_to_script>/run_cluster"
```

The *run_cluster* script contains information for setting the CALIBRE_HOME environment variable and establishing the remote connection:

```
#!/bin/csh  
setenv CALIBRE_HOME /server1/tools/cal_2015.3  
$CALIBRE_HOME/bin/rcalibre "/tmp" 1 -mtflex $CALIBRE_REMOTE_CONNECTION  
$CALIBRE_HOME/bin/rcalibre "/tmp" 1 -mtflex $CALIBRE_REMOTE_CONNECTION -f
```

Related Topics

[rcalibre](#)
[-hyper](#)
[-mtflex](#)
[-remotedata](#)
[-remotefile](#)
[CALIBRE_REMOTE_CONNECTION](#)

-remotedata

Reduces primary host memory requirements and improves Calibre MTflex performance by distributing data across the network.

Usage

-remotedata [-recoveroff | -recoverremote]

Arguments

- -recoveroff

Specifies that recovery is disabled in the event of a catastrophic remote host failure (default). This mode consumes the least amount of remote memory and achieves the best performance. It is primarily useful for DRC runs on reliable hardware. Loss of a remote host (for example, from a crash or reboot) will result in the loss of the run data.

- -recoverremote

Specifies that recovery is via backup data on the remote hosts. If you use -remotedata for a job running on a large cluster, this argument is recommended. In the event of a complete failure to recover from multiple remote host failures, the entire Calibre run is restarted once. This recovery mode is primarily useful for standard DRC runs and for runs longer than 12 hours or anytime a remote failure (for example, from a crash or reboot) is likely.

By default, one RDS process can store up to 2GB of data. The -recoverremote argument launches one backup RDS process for every RDS process.

Specifying -hyper remote launches one RDS process for every remote CPU, regardless of the remote host configuration size. For example, if you launch a job with 100 remote CPUs, then Calibre launches 100 RDS processes that can store up to 200GB of data.

Without -hyper remote, Calibre launches:

- One RDS process for every full group of four CPUs on each remote host for large configurations (>=24 CPUs), with a minimum of one RDS process per remote host.
- One RDS process for every full group of two CPUs on each remote host for small configurations (<24 CPUs), with a minimum of one RDS process per remote host

When working with a job scheduler (such as IBM Spectrum LSF or Univa Grid Engine), you may have an uneven distribution of CPUs per remote host, causing the RDS space to be hard to predict. In this case, you can use one of the following options to add one RDS for every remote CPU (as with -hyper remote mode):

- Add the “HYPER RDSTMULTIPLE 1” statement to your Calibre MTflex configuration file.
- Set the CALIBRE_HYPER_RDSTMULTIPLE environment variable to 1 before running the job.

Description

The **-remotedata** option triggers the Calibre RDS (Remote Data Server) technology. This technology is designed to reduce primary host memory requirements and improve Calibre MTflex performance by distributing data across remote hosts. Specifying “[-hyper remote](#)” enables the **-remotedata** argument by default.

Due to the distributed nature of the data, every remote host used in a Calibre MTflex run must be able to communicate with every other remote host in the run. This means that every remote must be able to “ping” every other remote. Subnets should not be used with RDS unless all remotes are in the same subnet and connected to a single Network Interface Card (NIC) on the primary host.

Recovery from catastrophic remote host failures is governed by the use of the **-recoveroff** and **-recoverremote** command line options. If a recovery option is not specified, it defaults to the **-recoveroff** option. RDS memory consumption varies depending on the recovery mode.

The RDS technology is triggered by appending the **-remotedata** argument to the **-remote**, **-remotefile**, or **-remotecmd** argument. Calibre generates an error and fails when you specify **-remotedata** without the **-remote**, **-remotefile**, or **-remotecmd** argument. The **-remotedata** argument is required when using the hyper remote ([-hyper remote](#)) functionality.

Calibre generates an error if you specify the following command line options and only one remote host is available:

```
calibre tool-specific_options -remote host -remotedata
```

Specifying the following does not generate an error:

```
calibre tool-specific_options -remote host -remotedata -recoveroff
```

Examples

Example 1

The following example specifies that, in the event of a catastrophic failure on a remote host, recovery is disabled.

```
calibre -drc -hier -remote machine1,machine2,machine3 -remotedata \
        -recoveroff -hyper -turbo rules
```

Example 2

The following example specifies that, in the event of a catastrophic failure on a remote host, recovery is performed via backup data on the remote hosts.

```
calibre -drc -pdb -c -xcell xcell_list -turbo 24 \
        -remotefile mtflex.config rules -remotedata -recoverremote
```

Example 3

The following example specifies that recovery is via the backup data on the remote hosts. In the event of a complete failure to recover from multiple remote host failures, the entire Calibre run is restarted once. At most, 2 GB of data is stored by the RDS technology for each CPU on each remote host. This recovery mode is the default recovery mode when no option is specified.

```
calibre -drc -hier -turbo -hyper -remotedata -recoverremote \
-remotefile mtflex.config rules
```

Example 4

The following example shows the transcript from an RDS job. This message first validates distributed memory is enabled and the primary host can effectively communicate with each data server. The connection to each remote host memory (RAM) signifies a maximum 1 GB block of memory accessible to Calibre for data storage. Under this assumption, 10 GB of remote memory is available in this example.

```
// Connecting to Remote Data Servers
// Connected to RAM on remote host node40
// Connected to RAM on remote host node42
// Connected to RAM on remote host node55
// Connected to RAM on remote host node41
// Connected to RAM on remote host node58
// Connected to RAM on remote host node59
// Connected to RAM on remote host node44
// Connected to RAM on remote host node56
// Connected to RAM on remote host node57
// Connected to RAM on remote host node43
```

During the Executive Module the current peak distributed memory is reported at the completion of each operation.

```
VIA8i = OR VIA8i

VIA8i (HIER TYP=1 CFG=1 HGC=0 FGC=0 HEC=0 FEC=0 VHC=F VPC=F)

CPU TIME = 0 + 0 REAL TIME = 0 LVHEAP = 387/3724/3724 REMOTE = 2712/2720
OPS COMPLETE = 37 OF 3911

Operation COMPLETED on HDB 0 LVHEAP = 387/3724/3724
```

Finally, the peak distributed memory for the entire run is reported at the completion of the last operation. This is the maximum amount of memory used for data storage across the remote hosts throughout the job. If at any point all of the distributed memory (1 GB for every four CPUs on each remote host) is filled then shared memory on the primary host is used and a “SHARED” memory statistic appears in the completed summary.

```
LUP.3::<1> = PMOSi NOT TMP3
LUP.3::<1> (HIER TYP=1 CFG=1 HGC=0 FGC=0 HEC=0 FEC=0 VHC=F VPC=F)
CPU TIME = 3 + 0 REAL TIME = 2 LVHEAP = 612/5503/5754 SHARED = 0/704
REMOTE = 0/10240 OPS COMPLETE = 3911 OF 3911
Operation COMPLETED on HDB 4 LVHEAP = 334/1250/1252
```

The HDB summary section specifies the peak memory consumed per HDB in addition to the peak primary host memory. This information is reported at the end of the transcript.

```
HDB 0 : CPU TIME = 1220 + 10132 MAX LVHEAP = 4451
HDB 1 : CPU TIME = 3452 + 53173 MAX LVHEAP = 1174
HDB 2 : CPU TIME = 3359 + 52606 MAX LVHEAP = 1106
HDB 3 : CPU TIME = 4497 + 51672 MAX LVHEAP = 1257
HDB 4 : CPU TIME = 4490 + 50670 MAX LVHEAP = 1174
TOTAL MAX LVHEAP = 9162
```

Related Topics

[Remote Data Server](#)

[-hyper](#)

[-remote](#)

[-remotecmd](#)

[-remotefile](#)

[HYPER](#)

[CALIBRE_HYPER_RDSMULTIPLE](#)

-remotefile

Used to specify a configuration file that contains remote host information.

Usage

-remotefile *filename*

Arguments

- *filename*

Specifies the path to a configuration file containing remote host information.

Description

The -remotefile argument can be used in a distributed network environment to specify the pathname to a configuration file that contains remote hosts information. This argument must be specified with the -turbo or -turbo_litho arguments. You must have the required number of licenses for your job.

Refer to “[Distributed Calibre: Creating a Configuration File](#)” for information about Calibre MTflex configuration files.

Using -remotefile with xCalibrate

You can use the -remotefile argument to specify the path to a configuration file for xCalibrate jobs. The -turbo_litho argument is not applicable to running xCalibrate jobs. Refer to [xcalibrate -exec](#) in the *xCalibrate Batch User’s Manual* for usage information.

The REMOTE COMMAND in the xCalibrate configuration file must specify an executable script that calls the [rxcalibrate](#) (remote xcalibrate) utility. This utility initiates a Calibre MTflex run for xCalibrate on remote hosts. Refer to “[Executing xCalibrate With Calibre MTflex Using IBM Spectrum LSF](#)” for information on creating and using a configuration file with xCalibrate.

Examples

Example 1

The following example invokes Calibre nmDRC-H and uses the -remotefile argument to specify the path to the configuration file. The -turbo option instructs Calibre nmDRC-H to use multithreaded parallel processing for all stages except Litho operations, while the -turbo_litho argument instructs Calibre to use multithreaded parallel processing for Litho operations. Because no value is specified for the -turbo and -turbo_litho options, the Calibre nmDRC-H and Litho operations run on the maximum number of available CPUs for which there are licenses.

```
calibre -drc -hier -turbo -turbo_litho -remotefile mtflex.config -hyper \
rules
```

Example 2

The following example invokes Calibre xACT 3D and uses the -remotefile argument to specify the path to the configuration file. In this example, the -turbo argument instructs Calibre xACT 3D to use 24 CPUs on the remote hosts.

```
calibre -xact -3d -rcc -xcell xcell_list -turbo 24 \
        -remotefile mtflex.config rules
```

Related Topics

[REMOTE COMMAND](#)

[REMOTE HOST](#)

[Running Calibre MTflex in a Heterogeneous Environment](#)

[Running Calibre MTflex in a Homogeneous Environment](#)

[Calibre MTflex Processing](#)

[CALIBRE_MTFLEX_LOCAL_HOST_DIR](#)

-turbo

Instructs Calibre to use multithreaded parallel processing.

Usage

-turbo [#CPUs]

Arguments

- #CPUs

An optional argument specifying the number of CPUs to use for multithreaded processing. The specified number must be a positive integer. If this value is not specified, Calibre runs on the maximum number of CPUs available for which you have licenses. For best performance, it is recommended that you avoid specifying this value.

Description

The -turbo argument instructs Calibre to use multithreaded parallel processing for all stages except LITHO operations. This argument must be specified with -hier. If you specify a number that is greater than the maximum available, Calibre runs on the maximum number of CPUs available.

You can specify the -turbo and the -turbo_litho arguments concurrently in a single command line and the respective #CPUs strings can vary between the two arguments.

Examples

Example 1

In this example, the Calibre job is executed in MT (multithreaded) mode and since the number of CPUs is not specified, the job runs on the maximum number of CPUs available.

```
calibre -drc -hier -turbo -hyper rules
```

Example 2

In this example, a value of 3 is specified for the number of CPUs. However, the Calibre job is executed on a 2-CPU machine so it only runs on two processors.

```
calibre -drc -hier ... -turbo 3 ...
```

Related Topics

[Multithreaded \(MT\) Processing](#)

[-turbo_litho](#)

[Licensing for Multithreaded Operations](#)

-turbo_all

Used to halt Calibre tool invocation when the tool cannot obtain the exact number of CPUs specified by -turbo or -turbo_litho.

Usage

-turbo_all

Arguments

None.

Description

The -turbo_all argument is used with the -turbo, -turbo_litho, or both arguments. When enough licenses cannot be secured for the requested number of threads, this argument causes Calibre to exit with an error. The -turbo_all argument is ignored if loop licensing is defined in the licensing configuration file or if loop licensing is specified when the job is executed. Refer to “[Calibre Loop Licensing](#)” on page 61 for more information on loop licensing.

Examples

In the following example, Calibre is executed on an eight CPU machine. Calibre exits with an error if sufficient licenses for Calibre nmDRC and Calibre nmDRC-H to run on eight CPUs cannot be secured.

```
calibre -drc -hier -turbo 8 -turbo_all rule_file
```

Related Topics

[-turbo](#)

[-turbo_litho](#)

[Calibre Licensing Models](#)

[Retry Mode Classic](#)

[Multithreaded \(MT\) Processing](#)

-turbo_litho

Instructs Calibre to use multithreaded parallel processing for LITHO (RET and MDP) operations. May only be specified with -turbo.

Usage

-turbo_litho [#CPUs]

Arguments

- **#CPUs**

An optional positive integer specifying the number of CPUs to use for RET and MDP processes. If this value is not specified, Calibre runs on the maximum number of CPUs available for which you have licenses, regardless of the -turbo setting.

You can specify the -turbo and the -turbo_litho options concurrently in a single command line and the respective #CPUs arguments can vary between the two options.

Description

The -turbo_litho option instructs Calibre to use multithreaded parallel processing when performing LITHO (RET and MDP) operations. The optional #CPUs argument is a positive integer that specifies the number of CPUs to use for processing. This option must be specified with the -turbo and -hier options.

If you specify a number that is greater than the maximum available, Calibre runs on the maximum number of CPUs available.

You can specify the -turbo and -turbo_litho arguments concurrently in a single command line and the respective #CPUs strings can vary between the two arguments.

Prior to the 2008.2 release, when -turbo_litho is omitted from the command line, the #CPUs defaults to 1. With the 2008.2 and subsequent releases, when -turbo_litho is omitted from the command line, it automatically defaults to the value specified for -turbo. For example, specifying: 4

calibre -turbo

is equivalent to:

calibre -turbo 4 -turbo_litho 4

Note

 RET operations in the Calibre Multi-Patterning tools ignore the -turbo_litho argument and use the value specified for -turbo. Refer to “[Multi-Patterning SVRF Commands](#)” in the *Calibre Multi-Patterning User’s and Reference Manual* for a list of commands that ignore the -turbo_litho argument.

Refer to “[Licensing for Multithreaded Operations](#)” for additional information about license scaling with CPU count.

Examples

```
calibre -drc -hier -turbo 4 -turbo_litho 4 rulefile
```

Related Topics

[Multithreaded \(MT\) Processing](#)

[-turbo](#)

Utilities

You can use the rcalibre utility to initiate a Calibre MTflex run on remote hosts.

rcalibre	398
rxcalibrate	400

rcalibre

Used to initiate a Calibre MTflex run on remote hosts.

Usage

```
$CALIBRE_HOME/bin/rcalibre [rundir] env_var_count {[env_var_1] [env_var_value] } ...  
-mtflex host_inet_address:host_port_number [-v] [-f]
```

Arguments

- ***rundir***
An optional argument used to specify an alternate directory for the CalibreRemoteLog file. The default location for the log files on the remote hosts is */tmp*.
- ***env_var_count***
A required argument used to specify the total number of environment variables that are defined during the execution of rcalibre. For example, you should specify “2” if you define values for two environment variables during the execution of rcalibre. You must specify “0” when no environment variables are defined.
- ***env_var_name***
An optional argument used to specify the name of the environment variable; for example: CALIBRE_HOME. You can define more than one environment variable using the *env_var_value* argument but it must have an associated value defined for the *env_var_value* argument.
- ***env_var_value***
An optional argument used to specify a value for the environment variable identified by the preceding *env_var_name*.
- **-mtflex *host_inet_address:host_port_number***
A required argument set that specifies the primary host IP address and port number for the connection. The CALIBRE_REMOTE_CONNECTION environment variable may be used to set the address and port.
- **-v**
An optional argument that specifies verbose mode allowing the server to print more messages as it runs.
- **-f**
An optional argument used for running Calibre in the foreground on the remote machine. This argument is commonly used when running IBM Spectrum LSF. By default, when running Calibre on a remote machine, the remote process runs in the background causing IBM Spectrum LSF to assume the Calibre job is finished. This argument keeps the remote process running in the foreground until the process completes, allowing IBM Spectrum LSF to monitor the process until completion.

Description

The rcalibre (remote Calibre) utility is used in a REMOTE COMMAND script to initiate a Calibre MTflex run on remote hosts. Refer to the [REMOTE COMMAND](#) reference page for a detailed example using rcalibre. *\$CALIBRE_HOME/bin/rcalibre* must be readable on a remote host.

Every invocation of rcalibre results in the creation of two Calibre processes. One process performs the operations requested as part of the rcalibre operation, while the other process idles in the background and is ready to perform “clean up” operations in the event the first process fails. The second process consumes a trivial amount of memory and no CPU time.

The launch of the rcalibre utility is successful if the following output is seen on the remote stdout:

```
// Starting Calibre Remote Protocol
```

The parent process then echoes remote stdout to primary stdout.

If this output is not generated on the remote stdout within the wait period, the launch fails and the parent process closes the pipe to the remote.

Examples

Assume \$aoi_home and \$CALIBRE_REMOTE_CONNECTION are defined in the environment.

Example 1

The following example specifies the MAXMEM limit for the remote hosts.

```
rcalibre "/scratch1" 2 CALIBRE_HOME $aoi_home \
CALIBRE_MTFLEX_REMOTE_MAXMEM 1000 -mtflex $CALIBRE_REMOTE_CONNECTION
```

Example 2

The following example illustrates the use of both a “layer directory” option and MAXMEM limit for the remote host. In general, the use of layer directories is generally discouraged for performance reasons.

```
rcalibre "/scratch1" 3 CALIBRE_HOME $aoi_home \
CALIBRE_MTFLEX_REMOTE_MAXMEM 1000 \
-mtflex $CALIBRE_REMOTE_CONNECTION
```

Related Topics

[REMOTE COMMAND](#)

[CALIBRE_REMOTE_CONNECTION](#)

rxcalibrate

Used to initiate a Calibre MTflex run for xCalibrate on remote hosts.

Usage

```
$CALIBRE_HOME/bin/rxcalibrate [rundir] {env_var_count {[env_var_name]  
[env_var_value] ...}} -mtflex host_inet_address:host_port_number [-v] [-f]
```

Arguments

- *rundir*

An optional argument that specifies the directory in which to store the log file (*CalibreRemoteLog*) on a remote host. The default location is */tmp*.

- *env_var_count* {[*env_var_name*] [*env_var_value*] ...}

A required argument specifying a value that identifies the total number of environment variables that are used during the execution of rxcalibre. The value is specified as a positive integer. You must specify “0” when no environment variables are defined. If you specify a value greater than 0, you must also specify an equivalent set of values for *env_var_name* *env_var_value*.

env_var_name — Specifies the name of the environment variable that is used during the execution of rxcalibrate.

env_var_value — Specifies the path to associate with the environment variable name.

- **-mtflex host_inet_address:host_port_number**

A required argument set that specifies the primary host IP address and port number for the remote host to use to connect with the primary host. The

CALIBRE_REMOTE_CONNECTION environment variable may be used to set the address and port.

- **-v**

An optional argument that specifies verbose mode allowing the server to print more messages as it runs.

- **-f**

An optional argument used for running Calibre in the foreground on the remote machine.

This argument is commonly used when running IBM Spectrum LSF. By default, when running Calibre on a remote machine, the remote process runs in the background causing IBM Spectrum LSF to assume the Calibre job is finished. This argument keeps the remote process running in the foreground until the process completes, allowing IBM Spectrum LSF to monitor the process until completion.

Description

The rxcalibrate (remote xCalibrate) utility is used by the [REMOTE COMMAND](#) statement to create Calibre MTflex connections on remote hosts for running xCalibrate. This utility must be executable on the remote hosts.

An invocation of rxcalibrate creates two Calibre processes: one process performs operations that are requested as part of the rxcalibrate operation, while the second process idles in the background and is ready to perform “clean up” operations in the event the first process fails. The second process consumes a trivial amount of memory and no CPU time.

The launch of the rxcalibrate utility is successful if the following output is seen on the remote stdout:

```
// Starting Calibre Remote Protocol
```

The parent process then echoes remote stdout to primary stdout.

If this output is not generated on the remote stdout within the wait period, the launch fails and the parent process closes the pipe to the remote.

Examples

This example illustrates the use of the rxcalibrate command defined in a script that is executed by the REMOTE COMMAND. The command specifies there are two environment variables with associated environment variable names (MGC_HOME and MY DESIGN) and paths.

```
rxcalibrate "/tmp" 2 MGC_HOME $MGC_HOME MY_DESIGN $HOME/layout -mtflex
$CALIBRE_REMOTE_CONNECTION -f &
```

Related Topics

[REMOTE COMMAND](#)

[CALIBRE_REMOTE_CONNECTION](#)

[Executing xCalibrate With Calibre MTflex Using IBM Spectrum LSF](#)

[xcalibrate -exec \[xCalibrate Batch User's Manual\]](#)

Errors and Warnings

Calibre MTflex is designed to recover from failures on remote hosts, whether they be problems with the remote host itself or communications with the remote host. When a problem is detected on a remote host, the operation in progress is restarted and executed on the primary host, and the remote host is no longer used. A warning message is issued on the primary host to indicate the remote failure. A failure on the primary host results in failure of the run.

Table 17-1 provides a list of common Calibre MTflex error messages and associated descriptions.

Table 17-1. Calibre MTflex Error Messages

Message	Description
MTFLEX RUNTIME ERROR: -hyper specified, but turbo number (2) is less than number of physical cpus (4).	If the turbo number is required, execute the command again without specifying -hyper. Or, if -hyper is required, re-execute the command and specify -turbo 4.
MTFLEX RUNTIME ERROR: Cell extent conformance for cell ICV_xxx done on master, count = yyy	Remote was 32-bit and could not handle run, thus was sent back to the 64-bit primary host for completion. Message indicates significant flattening has occurred and the amount of data is too large for 32-bit remote processing. Recommend using 64-bit to improve runtime.
MTFLEX RUNTIME ERROR: Remote shell failed in remote connection, status=1	Unable to connect to any of the specified remote machines. Verify network accessibility.
MTFLEX RUNTIME ERROR: No remote hosts were connected.	The MGC software tree on the primary host must be visible from the remote hosts. Additionally, the rcalibre utility must be visible to the primary and remote hosts.
MTFLEX RUNTIME ERROR: Socket read failed. MTFLEX RECV FAILED:code = 110	The connection to a remote host timed out and is no longer responding. The remote host will be dropped from the Calibre MTflex run. This may be due to a hardware failure or excessive disk swapping on the remote host.
MTFLEX RUNTIME ERROR: OPEXECL failed rc = 102	The remote is being dropped from the run. The error is not fatal. Whatever task that the remote was performing is completed by the primary host and subsequent tasks are sent to other remote hosts.
MTFLEX RUNTIME ERROR: Socket read failed, received EOF	This sort of behavior can be seen when there is a latency in the network connection between the primary and remote host.
MTFLEX RUNTIME ERROR: Remote process invocation failed	A process failed to invoke on a remote host, causing the remote host to be dropped from the Calibre MTflex run.

Table 17-1. Calibre MTflex Error Messages (cont.)

Message	Description
MTFLEX RUNTIME ERROR: Remote shell did not invoke properly in remote	Unable to start a remote shell on a remote host. Verify network accessibility and login information.

[Table 17-2](#) provides a list of common Calibre MTflex warning messages and associated descriptions.

Table 17-2. Calibre MTflex Warning Messages

Message	Description
MTFLEX RUNTIME WARNING: Problem with CPU on remote host:< <i>machine_name</i> >, it will no longer be used.	A remote process, which has already connected to the host, is lost or killed.
MTFLEX RUNTIME WARNING: Connect failed to remote host: < <i>machine_name</i> >	Unable to access a remote machine specified through the use of -remote or -remotefile. Calibre issues the following errors and stops the run if it cannot connect to any of the remote machines you specify: ERROR: Remote shell failed in remote connection, status=1 WARNING: Connect failed to remote host: < <i>machine_name</i> > ERROR: No remote hosts were connected.

Appendix A

Calibre Environment Variables

Environment variables allow you to define some aspect of the Calibre environment that can vary.

For environment variables used with the following products, see the referenced section in the product manual:

Product	Refer to...
Calibre Interactive	“Environment Variables for Calibre Interactive” in the <i>Calibre Interactive User’s Manual</i> .
Calibre RVE	“Environment Variables for Calibre RVE” in the <i>Calibre RVE User’s Manual</i> .
Calibre RealTime	“Calibre RealTime Environment Variables” in the <i>Calibre RealTime User’s Manual</i> .
FastXOR	“FastXOR Environment Variables” in the <i>Calibre Layout Comparison and Translation Guide</i> .
FDI	“FDI Environment Variables” in the <i>Calibre Layout Comparison and Translation Guide</i> .

General and Administrative Environment Variables	406
MT and Calibre MTflex Environment Variables.....	407
Licensing Environment Variables	409
Documentation Environment Variables	410
Calibre DESIGNrev Application Environment Variables.....	411
Calibre nmDRC Application Environment Variables.....	412
Calibre nmLVS Application Environment Variables	413
Calibre OPCpro Environment Variables	414
Calibre PERC Environment Variables	415
Third Party Database Environment Variables.....	416

General and Administrative Environment Variables

Some environment variables are available for use by all Calibre tools.

Table A-1. General and Administrative Environment Variables

Variables listed in the Admin Guide	Definition
CALIBRE_DISPLAY_ALL_ENV_VARS	Reports all environment variables in the transcript prior to invoking Calibre. Set this variable to 1 to enable this feature. The environment variables are bracketed at the top of the transcript by BEGIN and END markers. For example: // BEGIN:: Environment variable listing ... // END:: Environment variable listing
CALIBRE_HOME	Sets the path to the Calibre software tree.
CALIBRE_PRINT_TIME_STAMPS	Controls output of date and time information to the transcript.
MGC_HOME	Sets the path to the MGC software tree.

MT and Calibre MTflex Environment Variables

There are some environment variables you can use to control MT (multithreaded), Calibre MTflex, hyperscaling, and simultaneous multithreaded (SMT) operations.

Table A-2. MT and Calibre MTflex Environment Variables

Variables listed in the Admin Guide	Definition
<code>CALIBRE_HDBFLEX_MIN</code>	Specifies the behavior for the MINCOUNT argument (in LAUNCH AUTOMATIC, LAUNCH CLUSTER, or LAUNCH MANUAL) when HDBFLEX is enabled.
<code>CALIBRE_HYPER_SMTFACTOR</code>	Controls the “-hyper” capability for OPC operations on HDB0 when run on hyperscaled systems.
<code>CALIBRE_HYPERFLEX_SAVE_LOGS</code>	Saves PHDB log files at the end of a Calibre run.
<code>CALIBRE_MT_MONITOR_LOAD</code>	Prints the LOAD associated MON statements to the transcript at a specified interval.
<code>CALIBRE_MT_MONITOR_MEMORY</code>	Prints the MEMORY associated MON statements to the transcript at a specified interval.
<code>CALIBRE_MT_SMT_FACTOR</code>	Overrides the BIOS setting for simultaneous multithreading.
<code>CALIBRE_MTFLEX_HYPER</code>	Specifies the same information as that defined by the <code>HYPER</code> configuration statement.
<code>CALIBRE_MTFLEX_LAUNCH</code>	Sets the parameters to the set of LAUNCH configuration file commands.
<code>CALIBRE_MTFLEX_LOCAL_HOST_DIR</code>	Performs the function of the <code>LOCAL HOST DIR</code> configuration file command when running without a configuration file.
<code>CALIBRE_MTFLEX_MONITOR_LOCAL</code>	Provides the same information as the <code>MONITOR LOCAL</code> command when running calibre -remote.
<code>CALIBRE_MTFLEX_MONITOR_REMOTE</code>	Provides the same information as the <code>MONITOR REMOTE</code> command.
<code>CALIBRE_MTFLEX_REMOTE_DIR</code>	Specifies a location on the remote host for writing log files.

Table A-2. MT and Calibre MTflex Environment Variables (cont.)

Variables listed in the Admin Guide	Definition
CALIBRE_MTFLEX_REMOTE_MAXMEM	Sets the MAXMEM parameter for the REMOTE HOST command.
CALIBRE_MTFLEX_SAVE_LOGS	Saves Remote Compute Server (RCS) log files at the end of a Calibre run.
CALIBRE_REMOTE_CONNECTION	Specifies the host and port number for each connection to a remote host when using the realibre or rxcalibre utility. Execution of the REMOTE COMMAND automatically sets the <i><host>:<port></i> values for this environment variable in the child process.
CALIBRE_REMOTE_COUNT	An environment variable set by the tool in a child process from REMOTE COMMAND , and equal to the COUNT parameter for the LAUNCH CLUSTER command.
CALIBRE_REMOTE_MINCOUNT	An environment variable set by the tool in a child process from REMOTE COMMAND , and equal to the MINCOUNT parameter for the LAUNCH CLUSTER command.
LITHO_SOCS KERNELS_SHARED	Enables shared memory in post-tapeout runs.

Licensing Environment Variables

Some environment variables control operations related to licensing.

Table A-3. Licensing Environment Variables

Variable	Definition
CALIBRE_DRC_WAIVER_SUB_DRC_DISABLE	Setting this variable to 308635 disables the substitution of the Calibre DRC and Calibre DRC-H licenses for the Calibre Auto-Waivers license.
LM_LICENSE_FILE	Sets the FLEXnet licensing environment to point to either a license file or server.
MGLS_LICENSE_FILE	Sets the path to the MGC license file or daemon. If set, MGC products refer to this location for obtaining licenses.
CALIBRE_LM_CONFIG	Specifies the path to the licensing configuration file.
CALIBRE_LM_LOG_LEVEL	Controls the log information produced by Calibre classic and Calibre loop licensing.

Documentation Environment Variables

Some environment variables control operations related to viewing documentation.

Table A-4. Documentation Environment Variables

Variable	Definition
MGC_PDF_READER	Sets the name of the executable for displaying PDF documentation. The default is acroread.

Calibre DESIGNrev Application Environment Variables

Some environment variables control Calibre DESIGNrev operations.

Table A-5. General and Administrative Environment Variables

Variables listed in the Admin Guide	Definition
MGC_CWB_CONFIG_DIRS	Specifies the path to the Calibre DESIGNrev configuration files.
MGC_CWB_RELOAD_ORIGINAL_LAYER_NAMES	Prevents the reloading of layer names when using File > Reload Layout or Alt+r in Calibre DESIGNrev.
MGC_CWB_PCR_PATH	Specifies a directory for saving Peek Cache Repository (PCR) files.
MGC_CWB_TMP_DIR	Specifies the path to a <i>tmp</i> directory.
MGC_DRV_RELEASE_LICENSE_TIME	Specifies a license timeout interval (in hours) for Calibre DESIGNrev.
CWB_DUMP_CONFIGURATION_ACTIVITIES	Enables or disables the writing of status messages to the transcript.

Calibre nmDRC Application Environment Variables

Some environment variables control Calibre nmDRC operations.

Table A-6. Calibre nmDRC Application Environment Variables

Variable	Definition
CALIBRE_ECHO_RULE_FILE	Setting this to a non-null value causes the main rule file and the text of all included rule files to be echoed to the transcript. The actual include statements and code that are not executed due to preprocessor directives are commented out in the transcript.
CALIBRE_EXIT_MAXIMUM_RESULTS	Setting this to a non-null value causes Calibre to exit if a rule check is output to a GDS or OASIS database and the maximum results limit specified by DRC Maximum Results or DRC Check Map is less than ALL.

Calibre nmLVS Application Environment Variables

Some environment variables control Calibre nmLVS operations.

Table A-7. Calibre nmLVS Application Environment Variables

Variable	Definition
CALIBRE_LVS_REMOTE_CONNECTION	Specifies local host connection information that is used by the LVS compare process for the remote host to connect with the local host.
MGC_TMPDIR	Sets the path where the flat LVS SPICE parser stores temporary files. The default is <i>\$MGC_HOME/tmp</i> .

Calibre OPCpro Environment Variables

Some environment variables control Calibre OPCpro operations.

Table A-8. Calibre OPCpro Application Environment Variables

Variable	Definition
LITHO_REPORT_REMOTE_CPU_TIME	Used for Calibre OPCpro running in Calibre MTflex mode, this variable enables the printing of status messages to the transcript for remote CPUs during the cell processing loop.

Calibre PERC Environment Variables

Some environment variables control Calibre PERC operations.

Refer to the [Calibre PERC User's Manual](#) for more information on using these environment variables.

Table A-9. Calibre PERC Application Environment Variables

Variable	Definition
CALIBRE_ENABLE_PERC_SIMULATION_REMOTEFILE_MTFLEX	Enables the resistance simulation module in LDL runs to support Calibre MTflex processing through the -remote and -remotefile command line options.
CALIBRE_ENABLE_PERC_XRC_MTFLEX	Enables the LDL parasitic extraction module to support Calibre MTflex processing through the -remote and -remotefile command line options.
CALIBRE_PERC_MTFLEX_MEMORY_SUGGESTION	Specifies integral values (in MB) that override the default memory limit.
CALIBRE_PERC_SIMULATION_MTFLEX_EXPORT_VARS	Specifies a list of environment variables that are passed to remote hosts.

Third Party Database Environment Variables

Some environment variables configure third-party database tools and options.

Table A-10. Calibre Third Party Database Environment Variables

Variable	Definition
ICC_PATH	Specifies the directory that includes the <code>icc_shell</code> executable. This environment variable is not required if <code>icc_shell</code> is defined in the current PATH variable.
MGC_CALIBRE_CELLMAP_FILE	Sets the path of a cell mapping file for LEF/DEF and OpenAccess databases. See Layout System .
MGC_CALIBRE_DB_READ_OPTIONS	Sets options for reading of LEF/DEF and OpenAccess databases. See Layout System .
MGC_CALIBRE_LAYERMAP_FILE	Sets the path of a layer mapping file for LEF/DEF and OpenAccess databases. See Layout System .
OA_HOME	Calibre uses the OpenAccess library at <code><calibre_install_dir>/shared/pkgs/icv_oa</code> . If additional plug-ins are needed, use the <code>OA_HOME</code> environment variable to specify the path to an OA library that contains the plug-ins. Note, that the library specified by the <code>OA_HOME</code> variable is not used. Calibre always uses the default OpenAccess library included with the Calibre installation. This variable applies only to <code>dbdiff</code> , <code>fdi2gds</code> , and <code>fdi2oasis</code> .

Appendix B

Configuration: Operating Systems and Virtual Memory

If you run Calibre on a 32-bit platform, you should be aware of potential 32-bit addressing limitations. There are also best practices you may want to follow to conserve memory when running Calibre, in addition to information for handling virtual memory allocation failures.

Memory Usage Best Practices.....	417
Virtual Memory Allocation Failures	417

Memory Usage Best Practices

There are some best practices you can follow in order to conserve memory with Calibre tools on Linux platforms.

To conserve memory with Calibre tools on Linux platforms:

- Ensure you put the most repeated cells in your hcell file if you run the hierarchical version of the Calibre nmLVS-H tool.
- Monitor the size of the Calibre tool's processes using Linux OS-native memory monitoring utilities or using the **MONITOR SYSTEM** statement.
- Ensure no other processes are using memory extensively when Calibre is running on the same host.

Virtual Memory Allocation Failures

The most common memory allocation error you might encounter when using Calibre tools is **LVHEAP MEMORY ALLOCATION FAILURE**. In most cases, the Calibre tool issues this error because the platform's swap is insufficient. Generally, you should set the platform's stacksize no less than 8192 kilobytes.

If the stacksize is insufficient, the Calibre transcript contains this warning:

```
WARNING: Please increase stacksize for best performance
```

Failure to mitigate the warning can cause Calibre to abort.

You can use the **limit** (C shell) or **ulimit -a** (Bourne shells) command to verify the stacksize.
For example:

```
% limit
cputime      unlimited
filesize     unlimited
datasize     unlimited
stacksize    8192 kbytes
coredumpsize 0 kbytes
memoryuse   unlimited
vmmemoryuse unlimited
descriptors 1024
memorylocked 32 kbytes
maxproc     16384
```

To temporarily modify the stacksize, enter the following commands:

```
% limit stacksize 8192; # csh
$ ulimit -s 8192; # Bourne shells
```

The following files can be configured with the appropriate setting:

RHEL7: */etc/security/limits.d/20-nproc.conf*

RHEL6 and 5: */etc/security/limits.d/90-nproc.conf*

Any: */etc/security/limits.conf*

Virtual Memory Allocation Failure on Linux Platforms

On Linux platforms, the **numactl** command is used to output additional information about the memory nodes when a virtual memory allocation failure occurs. Following is an example of the transcript output from a virtual memory allocation failure on Linux:

```
malloc() failed: return value = 0, errno = 12

available: 4 nodes (0-3)
node 0 size: 16127 MB
node 0 free: 76 MB
node 1 size: 16160 MB
node 1 free: 3892 MB
node 2 size: 16160 MB
node 2 free: 12667 MB
node 3 size: 16160 MB
node 3 free: 9565 MB

node distances:
node 0 1 2 3
 0: 10 20 20 20
 1: 20 10 20 20
 2: 20 20 10 20
 3: 20 20 20 10

MALLINFO:
arena = 1188474880
ordblk = 20
smbblk = 0
hblk = 1361
hblkhd = -1439100928
usmblk = 0
fsmblk = 0
uordblk = 1188340496
fordblk = 134384
keepcost = 133976

LVHEAP MEMORY ALLOCATION FAILURE, requested 2097152 bytes, current
allocation is 4034199836
DRC_ABORT: Success
```


Glossary

Calibre MTflex

Calibre MTflex functionality provides a parallel processing architecture for running Multithreaded (MT) Calibre tools on distributed networked computers. When you run a Calibre tool with Calibre MTflex functionality, Calibre creates hierarchical threads and runs these threads across the network to remote computers.

Cluster

A cluster is a group of computers (hosts) that work together as a single unit, combining compute power and sharing workload and resources. A cluster provides a single-system image for disparate computing resources.

Configuration file

An ASCII file that specifies the local and remote host information for a Calibre MTflex run. You add configuration file statements that:

- control how Calibre MTflex is launched on the primary and remote hosts
- define the local host configuration
- enable monitoring operations
- enable the execution of remote operations

CPU core

A physical execution unit which processes a unique instruction stream. The operating system assigns processes for execution on cores. Only physical cores, not virtual cores, are included in this definition. Refer to “[Simultaneous Multithreading With Virtual CPUs](#)” for information on virtual cores.

Dropdown

Refers to behavior that occurs when Calibre cannot secure enough license sets for the requested number of concurrent threads causing Calibre to automatically reduce the number of threads to match the available licenses.

Heterogeneous environment

A network environment in which the local and remote hosts are different platforms or different address space. Refer to “[Using the LAUNCH CLUSTER and REMOTE COMMAND Statements](#)” for information on running Calibre MTflex in a heterogeneous environment. When Calibre MTflex functionality is used in this environment, threads are run on different operating systems or address spaces.

Homogeneous environment

A network environment consisting of local and remote hosts that are all the same platform and address space. When Calibre MTflex is used in this environment, threads are run on identical operating systems. Refer to “[Running Calibre MTflex in a Homogeneous Environment](#)” for information on running Calibre MTflex in a homogeneous environment.

Hyperscaling

Hyperscaling refers to how the tasks are divided and can be deployed in either the MT or Calibre MTflex environments. SVRF operations, such as those that perform geometric layer operations, can be hyperscaled while others cannot. SVRF operations requiring connectivity information, such as those in nmLVS and PEX can not be hyperscaled because connectivity of nets spans many cells. Hyperscaling is not a competitor to Calibre MTflex, as it uses a different processing engine that enables more efficient use of Calibre MTflex multithreading.

License queueing

Places license requests in a queue to wait for an available license when the appropriate license is not available.

License substitution

Enables selected Calibre tools to check the license file for the available required licenses. If the required license is not defined and a substitute license is defined, the tool acquires or queues for the substitute license.

Local CPU

CPUs which share memory. The host on which the local CPUs reside is referred to as the Primary (Server) Host.

Local Host

See [Primary Host](#).

LM_LICENSE_FILE

The FLEXnet Licensing environment variable used by the license server and application to determine the location of license data files.

Master Host

See [Primary Host](#).

MGLS_LICENSE_FILE

Used when multiple products have FLEXnet Licensing to allow other vendor products to use the LM_LICENSE_FILE variable, while allowing Calibre products to get their licenses from the value of MGLS_LICENSE_FILE. Only Siemens EDA software recognizes this variable; software from other vendors that use FLEXnet Licensing ignore it.

Primary Host

In MT mode, the primary host is a machine with multiple CPUs. Multithreading occurs across the CPUs on the MT host only. In Calibre MTflex mode, the primary host has multiple CPUs, as in MT mode, and it connects to remote hosts, which may or may not have multiple CPUs. The primary host coordinates the scheduling of Calibre MTflex operations among all of the hosts,

performs operations that cannot be run on remote hosts, and collects the results from all hosts. The primary host is sometimes referred to in commands, environment variables, and transcripts as a “local” or “master” host. For information on primary host requirements, refer to “[Primary Host](#)” on page 238.

Remote (Client) Hosts

The platforms that run in a Calibre MTflex distributed network mode. The remote hosts must have access to an MGC_HOME tree that is the same numerical version as that running on the local (server) host. For environments using multiple platforms, the remote hosts may need to access different MGC_HOME trees for their respective platforms. For information on remote host requirements, refer to “[Remote Host](#)” on page 239.

Scalability

The percentage of scaling on N cores. Scalability is always represented as a percentage value (0-100 percent). A scalable system demonstrates performance improvement proportional to the hardware capacity. For more information, refer to the “[Calibre Scaling and Scalability](#)” on page 214 section.

Scaling

The number of CPU cores (N) that are used for the Calibre run. Scaling is generally calculated by dividing the transcribed CPU time by the REAL time (CPU/REAL). For more information, refer to the “[Calibre Scaling and Scalability](#)” on page 214 section.

Shared-memory processor (SMP)

Shared-memory processors and multiprocessors are systems that:

- are based on a parallel computing architecture using a single address space
- allow processor communication through variables stored in a shared address space

SMP

See Shared-memory processor (SMP).

VCO

An acronym that represents the platform vendor, CPU, and operating system. Refer to “[Using the calibre_vco Utility](#)” on page 47 for information on determining the VCO for a platform.

Index

— Symbols —

[]¹⁹
{}¹⁹
|¹⁹

— B —

Bold words, [19](#)

— C —

calblankclassify, [201](#)
calcellarrayopc, [157](#), [170](#)
calclnmbias, [166](#)
calclnmopc, [167](#)
calcmpmb, [131](#)
calcmpsim, [130](#)
calcmpsimconx, [130](#)
cald2dbdataprep, [185](#)
caldefectreview, [187](#)
caldefectsim, [186](#)
caldesignrev, [91](#), [99](#), [134](#), [182](#)
caldrclvseve, [96](#), [100](#), [120](#), [125](#), [127](#)
caldsasynth, [158](#)
caleuv, [160](#), [163](#)
caleuvda, [202](#)
calfractureall, [188](#), [189](#), [191](#), [192](#), [195](#), [196](#),
[204](#), [205](#)
calfracturec, [188](#), [204](#), [205](#)
calfractureh, [189](#), [204](#), [205](#)
calfracturei, [190](#)
calfracturej, [191](#), [204](#), [205](#)
calfracturem, [192](#), [204](#), [205](#)
calfracturen, [193](#)
calfracturep, [194](#)
calfracturet, [195](#), [204](#), [205](#)
calfracturev, [196](#), [204](#), [205](#)
Calibre 3DSTACK licensing, [91](#)
Calibre ADP (Advanced Device Properties)
licensing, [92](#)
Calibre AutoFix licensing, [94](#)
Calibre Auto-Waiver licensing, [95](#)

Calibre Auto-Waivers licensing, [95](#)
Calibre CB (Cell/Block) licensing, [96](#)
Calibre CD (Cell/Block) licensing, [96](#)
Calibre CI (Connectivity Interface) licensing,
[98](#)
Calibre classic licensing
queueing, [78](#)
substituting, [79](#)
Calibre Cluster Manager (CalCM) licensing,
[82](#)
Calibre CMP Model Builder licensing, [131](#)
Calibre CMPAnalyzer licensing, [130](#)
Calibre DESIGNrev licensing, [99](#), [114](#), [115](#)
Calibre DSA Mask Synthesis licensing, [158](#)
Calibre DSA Verification licensing, [159](#)
Calibre Dynamic Resource Allocator (DRA)
licensing, [84](#)
Calibre e2lvs licensing, [100](#)
Calibre EUV licensing, [158](#), [159](#), [160](#)
Calibre FRACTUREc
distributed operations, [269](#)
licensing, [188](#)
Calibre FRACTUREh
distributed operations, [269](#)
licensing, [190](#)
Calibre FRACTUREj
distributed operations, [269](#)
licensing, [191](#)
Calibre FRACTUREm
distributed operations, [269](#)
licensing, [192](#)
Calibre FRACTUREt
distributed operations, [269](#)
licensing, [195](#)
Calibre FRACTUREv
distributed operations, [269](#)
licensing, [196](#)
Calibre Interactive licensing, [101](#)
Calibre Job Deck Editor licensing, [197](#)
Calibre LFD licensing, [132](#)

-
- Calibre License Broker Daemon, 138
 - Calibre licensing models, 56
 - Calibre classic licensing, 56
 - Calibre loop licensing, 56
 - Calibre LITHOview licensing, 114, 115, 161
 - Calibre loop licensing
 - configuring, 80
 - definition of, 56
 - Calibre LPE licensing, 162
 - Calibre MAPI (Metrology API) licensing, 207
 - Calibre MASKOPT licensing, 198
 - Calibre MDP EMBED licensing, 199
 - Calibre MDP Embedded SVRF licensing, 200
 - Calibre MDPAutoClassify, 201
 - Calibre MDPDefectAvoidance, 202
 - licensing, 202
 - Calibre MDPDefectReview, 187
 - Calibre MDPmerge
 - distributed operations, 269
 - licensing, 203
 - Calibre MDPPatternClassify, 186
 - Calibre MDPstat licensing, 204
 - Calibre MDPverify licensing, 205
 - Calibre MDPview licensing, 114, 115, 206
 - Calibre mlOPC licensing, 163
 - Calibre MPCpro licensing, 208
 - Calibre MPCverify licensing, 209
 - Calibre MTflex
 - definition of, 219, 421
 - heterogeneous environment, 281
 - homogeneous environment, 279
 - license requirements, 249
 - output differences, 257
 - running in a heterogeneous environment, 281
 - running in a homogeneous environment, 279
 - Calibre Multi-Patterning licensing, 102, 103, 164, 165
 - Calibre nmDRC licensing, 105, 114, 115, 116, 136
 - Calibre nmDRCgold licensing, 106
 - Calibre nmDRC-H licensing, 107, 114, 115, 116, 136
 - Calibre nmLVS licensing, 109
 - Calibre nmLVS Reconnaissance, 113
 - Calibre nmLVS-H Advanced licensing, 111
 - Calibre nmLVS-H licensing, 110
 - Calibre nmModelflow licensing, 168
 - Calibre nmMPC licensing, 209, 211
 - Calibre nmOPC licensing, 170
 - Calibre nmSRAF licensing, 171
 - Calibre OPCpro
 - distributed operations, 268
 - licensing, 172
 - Calibre OPCsbar licensing, 173
 - Calibre OPCverify Classic Plus licensing, 175
 - Calibre OPCverify licensing, 174
 - Calibre ORC licensing, 176
 - Calibre Pattern Match licensing, 114
 - Calibre Pattern Matching licensing, 114
 - Calibre Pattern Matching Manufacturing licensing, 115
 - Calibre PERC Advanced licensing, 118
 - Calibre PERC licensing, 116
 - Calibre Physical Verification Suite (PVS) licensing, 119, 128
 - Calibre PRINTimage licensing, 177
 - Calibre pxOPC licensing, 178
 - Calibre Query Server licensing, 98, 120
 - Calibre RealTime licensing, 121
 - Calibre Rule File Analyzer licensing, 124
 - Calibre RVE licensing, 125
 - Calibre SMO licensing, 179
 - Calibre SVRFencrypt licensing, 126
 - Calibre system tests
 - running, 47
 - test.CalibreRelease, 47
 - Calibre TDopc licensing, 181
 - Calibre v2lvs licensing, 127
 - Calibre WORKbench licensing, 114, 115, 182
 - Calibre xACT 3D licensing, 142, 144
 - Calibre xACTView licensing, 145
 - Calibre xL licensing, 146
 - Calibre xRC
 - distributed operations, 268
 - licensing, 148, 151
 - Calibre xRC CB licensing, 149
 - Calibre xRC to ADVance MS licensing, 150
 - Calibre YieldAnalyzer licensing, 133

Calibre YieldEnhancer licensing, 134
Calibre YieldServer licensing, 136
CALIBRE_DRC_WAIVER_SUB_DRC_DIS_ABLE, 95
CALIBRE_FLATTEN_LIKE_HYPER, 355
CALIBRE_HYPERFLEX_SAVE_LOGS, 360
CALIBRE_MT_MONITOR_LOAD, 362
CALIBRE_MT_MONITOR_MEMORY, 363
CALIBRE_MTFLEX_LAUNCH, 314, 317, 321, 366
CALIBRE_MTFLEX_LOCAL_HOST, 367
CALIBRE_MTFLEX_LOCAL_HOST_DIR, 368
CALIBRE_MTFLEX_LOCAL_HOST_DIR environment variable, 323
CALIBRE_MTFLEX_MONITOR_LOCAL, 370
CALIBRE_MTFLEX_MONITOR_REMOTE, 330, 371
CALIBRE_MTFLEX_REMOTE_MAXMEM, 372
CALIBRE_PRINT_TIME_STAMPS, 31
CALIBRE_REMOTE_CONNECTION, 374
CALIBRE_REMOTE_COUNT, 375
CALIBRE_REMOTE_MINCOUNT, 376
CALIBRE_REMOTE_SAVE_LOGS, 373
calibre_vco utility, 47
calibreadp, 92
calibreautofix, 94
calibreci, 98
calibreclmpc, 210
calibredrc, 91, 95, 96, 105, 107, 109, 110, 124, 130, 134, 136, 199
calibrehdrc, 91, 95, 107, 110, 124, 130, 134, 136, 199
calibrehlvs, 91, 110, 116, 124, 136
calibrehlvsadv, 102, 111
calibreinroute, 94
calibreldf, 96, 105, 107, 114, 132, 170, 172, 173, 174
calibrelv, 99, 161, 206
calibrelv, 91, 96, 100, 109, 110, 116, 124, 127, 136
calibremt2cpu, 107
calibrenmdrcgold, 95, 105, 106, 107, 114
calibreorc, 176
calibreperc, 95, 100, 116, 118, 127, 136
calibrepercadv, 118
calibrepvs_s, 96, 105, 107, 109, 110, 119, 120, 125, 128
calibreqdb, 98, 120, 124, 125, 130
calibreexact, 92, 98, 142, 144, 148
calibreexact3d, 92, 98, 142, 144, 145
calibreexactsoc, 92, 98, 142, 148
calibrexl, 142, 146
calibrexrc, 92, 96, 98, 109, 142, 144, 148
calibrexcadms, 150
calibrexccb, 92, 96, 98, 149
calibrexrctoact, 142
calinteractive, 101, 119, 128, 130
caljobdeckedit, 197
callpe, 162
calmaskopt, 198
calmdf, 168
calmdpesvrf, 200
calmdpmapi, 207
calmdpmerge, 203
calmdpview, 99, 206
calmfgmpadv, 103, 165
calmfgmpgold, 103, 164, 165
calmgfmpgold, 102
calmlopc, 163
calmpadv, 103
calmpcpro, 166, 181, 200, 208
calmpcverify, 209
calmpgold, 102, 103
calmtopcpro, 166, 170, 172, 176, 181
calmultipattern, 89, 102, 103
calnmdpc, 89, 102, 103, 154, 164, 165
calnmdrcgold, 102, 154
calnmmpc, 166, 181, 198, 200, 208, 209, 211
calnmopc, 163, 166, 170, 172, 174, 177, 181
calnmsraf, 160, 171, 173
calopcpro, 166, 170, 172, 176, 181
calopcsbar, 173
calopcvclassifyp, 175
calopcvdsa, 159
calopcverify, 170, 174, 176, 177
calpatnclassify, 186
calpmatch, 114

-
- calpmatchmfg, 115
 - calprintimage, 177
 - calpxopc, 160, 170, 178
 - calrealtime, 121
 - calrealtimedig, 123
 - calrealtimeelite, 122
 - calreconlvs, 113
 - calresourceman, 82, 83, 84, 85
 - calsmo, 179
 - calsonr, 180
 - calsvrfencrypt, 126
 - caltannerpvs, 128
 - caltannerxrc, 151
 - caltdopc, 181
 - calwaiver, 95
 - calworkbench, 99, 131, 161, 182, 206
 - calyieldanalyze, 133, 134
 - calyieldenhance, 134
 - calyieldserver, 130, 136
 - Cluster, definition of, 421
 - Configuration file
 - comment characters, 272
 - definition, 272
 - definition of, 421
 - HYPER statement, 311
 - LAUNCH AUTOMATIC statement, 314
 - LAUNCH CLUSTER statement, 317
 - LAUNCH MANUAL statement, 321
 - LOCAL HOST DIR statement, 323
 - MONITOR LOCAL statement, 327
 - MONITOR REMOTE statement, 330
 - MONITOR SYSTEM statement, 335
 - REMOTE COMMAND statement, 341
 - REMOTE HOST statement, 346
 - statement reference, 274
 - syntax conventions, 272
 - Configuration file statements
 - HYPER, 311
 - LAUNCH AUTOMATIC, 313
 - LAUNCH CLUSTER, 316
 - LAUNCH MANUAL, 320
 - LOCAL HOST DIR, 323
 - MONITOR LOCAL, 325
 - MONITOR REMOTE, 329
 - MONITOR SYSTEM, 332
 - REMOTE COMMAND, 341
 - REMOTE HOST, 345
 - Courier font, 19
 - CPU Core, definition of, 421
- D —**
- DFM Analyze, 133
 - DFM CAF, 133
 - DFM Circle Analyze, 107
 - DFM Classify, 107
 - DFM Contour, 134
 - DFM Copy, 107
 - DFM Critical Area, 133
 - DFM Density, 134
 - DFM DV, 107
 - DFM DV Analyze, 107
 - DFM Expand Edge, 134
 - DFM Expand Enclosure, 134
 - DFM Fill, 134
 - DFM Fill Wrap, 134
 - DFM GCA, 133
 - DFM Grow, 134
 - DFM Histogram, 133
 - DFM Jogclean, 134
 - DFM Measure, 133
 - DFM Merge, 107
 - DFM NARAC, 107
 - DFM Optimize, 134
 - DFM OR, 107
 - DFM OR Edge, 135
 - DFM Partition, 135
 - DFM Property, 108
 - DFM Property Merge, 108
 - DFM Property Net, 108
 - DFM Property Select Secondary, 108
 - DFM RDB, 108
 - DFM RDB (nodal only), 135
 - DFM RDB ASCII, 135
 - DFM Read, 108
 - DFM Rectangles, 135
 - DFM Redundant Vias, 133
 - DFM Reshape, 135
 - DFM Segment, 135
 - DFM Select By Net, 108
 - DFM Shift, 135
 - DFM Shift Edge, 135

-
- DFM Size, 135
 - DFM Space, 108
 - DFM Stamp, 108
 - DFM Stripe, 135
 - DFM Text, 108
 - DFM Transform, 135
 - DFM Transition, 135
 - DFM Via Shift, 135
 - Disk swap space
 - primary hosts, 249
 - remote hosts, 249
 - Documentation
 - alternate documentation location, 51
 - directory structure, 49
 - document handle, 51
 - downloading, 22
 - environment variables, 51
 - installation, 49
 - legal directory, 51
 - MGC_PDF_READER, 51
 - Double pipes, 19
 - downloading
 - documentation, 22
 - software, 22
 - Dropdown, 421
- E —
- Environment variables
 - CALIBRE_DRC_WAIVER_SUB_DRC_DISABLE, 95
 - Environment variables
 - CALIBRE_DISPLAY_ALL_ENV_VARS, 406
 - CALIBRE_FLATTEN_LIKE_HYPER, 355
 - CALIBRE_HDBFLEX_MIN, 356
 - CALIBRE_HOME, 46
 - CALIBRE_HYPER_RDSMULTIPLE, 359
 - CALIBRE_HYPER_SMTFACTOR, 357
 - CALIBRE_HYPERFLEX_SAVE_LOGS, 360
 - CALIBRE_LVS_REMOTE_CONNECTION, 361
 - CALIBRE_MT_MONITOR_LOAD, 362
- F —
- FLEXnet licensing, 53
 - FullScale, 233
- H —
- Hardware
 - recommended primary host configuration, 236
 - recommended remote host configuration, 236
 - Heavy font, 19
 - Heterogeneous environment
 - definition of, 421
 - running Calibre MTflex, 281
 - Homogeneous environment
 - definition of, 422
 - running Calibre MTflex, 279
 - HYPER statement, 311
 - Hyperscaling
 - definition of, 422

-
- definition of, 220
- I —
- ictrace, 96, 100, 109, 127
- Internet Address Resolution Protocol (ARP) table, 248
- Italic font, 19
- L —
- LAUNCH AUTOMATIC statement, 279, 313, 314
- LAUNCH CLUSTER example, 283
- LAUNCH CLUSTER statement, 316, 317
- LAUNCH MANUAL example, 284
- LAUNCH MANUAL statement, 320, 321
- LAYOUT TURBO FLEX, 234
- LAYOUT ULTRA FLEX, 235
- LBD
- see* License Broker Daemon
- License Broker Daemon
- definition of, 138
- License file location, 54
- License queueing, 56
- License reference page
- also enables definition, 74
- considerations and exceptions definition, 74
- required license definition, 74
- substitute license definition, 74
- License substitution, 57
- Licenses
- leveling, 59
- lost connections, 55
- multithreaded queueing, 57
- queueing, 56
- releasing, 55
- securing, 55
- single threaded queueing, 57
- Licensing
- command line options, 68
- dropdown, 57
- environment, 77
- Exact Access methodology, 54
- FLEXnet, 53
- models, 56
- MSL, 53
- reference page description, 74
- RET batch tools, 155
- two-CPU machines, 72
- Licensing configuration file
- Retry Mode Classic, 65
- Retry Mode Loop, 66
- Linux
- distribution compatibility statement, 25
- LM_LICENSE_FILE, 54, 422
- lmconfig command line option, 69
- lmretry command line option, 68
- LOCAL HOST DIR statement, 323
- Log file output, configuring, 30
- M —
- Mentor Graphics Licensing System (MGLS), 54
- MGC_PDF_READER, 51
- MGLS
- see also* Mentor Graphics Licensing System
- MGLS_LICENSE_FILE, 54, 422
- Minimum keyword, 19
- MONITOR LOCAL statement, 288, 325
- IO parameter, 291
- MEMORY parameter, 290
- MONITOR REMOTE statement, 288, 329
- MONITOR SYSTEM statement, 332
- Monitoring remote processes, 288
- Multithreaded Calibre, monitoring, 30
- Multithreaded 1 licenses, example, 71
- Multithreaded processing, output, 213
- Multithreading, warnings, 59
- N —
- Name service cache, 249
- Network
- private network settings, 243
- recommended configuration, 240
- Network driver settings, 248
- Network performance, measuring, 242
- Network TCP settings, 248
- NIC settings, 248
- nowait command line option, 68
- O —
- olympusstation, 94

OS file limits, 29

OS patches, minimum OS patch level, 23

— P —

Parentheses, 19

PDF

naming conventions, 50

package contents, 50

Pipes, 19

Primary Host, definition of, 422

pto switch, 233

— Q —

Quotation marks, 19

— R —

rcalibre Utility, 398

Releasing licenses, 55

REMOTE COMMAND example, 283

REMOTE COMMAND statement, 341

REMOTE HOST statement, 279, 345, 346

Remote Hosts

definition of, 423

RET batch tools, licensing, 155

Retry Mode Classic statement, 65

Retry Mode Loop statement, 66

RHEL, variable CPU speed option, 30

routerautofix, 94

RSH limits, 248

rxcalibrate Utility, 400

— S —

Scalability

definition of, 214, 423

for mula, 214

Scaling, 214, 423

Securing licenses, 55

Shared memory, guidelines, 250

Shared-memory processor (SMP), definition of, 423

Simultaneous Multithreading (SMT)

Calibre support, 73

Slanted words, 19

SLES, variable CPU speed option, 30

Square parentheses, 19

Sticky bit setting, 29

Supported platforms, 22

— T —

-turbo command line option, 218

-turbo_litho command line option, 218

TURBOflex

LAYOUT TURBO FLEX, 234

TURBOflex, definition of, 234

Two-CPU licensing, 72

— U —

ulimit settings, 246

coredumpsize (core file size), 247

descriptors (open files), 247

maxproc (max user processes), 247

ULTRAflex

definition of, 235

LAYOUT ULTRA FLEX, 235

Underlined words, 19

Utilities

calibre_vco, 47

rcalibre, 398

rxcalibrate, 400

— V —

Virtual machines (VMs), 28

Virtual memory zone reclaim mode, 249

— W —

-wait command line option, 68

— X —

xcalibrate, 152

xCalibrate Rule File Generator licensing, 152

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.

