

# Centrify Zero Trust Privilege Services: Authentication Service, Privilege Elevation Service, and Audit and Monitoring Service

*Configuration and Tuning Reference Guide*

December 2019 (release 19.9)

Centrify Corporation





## Legal Notice

This document and the software described in this document are furnished under and are subject to the terms of a license agreement or a non-disclosure agreement. Except as expressly set forth in such license agreement or non-disclosure agreement, Centrifly Corporation provides this document and the software described in this document “as is” without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose. Some states do not allow disclaimers of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This document and the software described in this document may not be lent, sold, or given away without the prior written permission of Centrifly Corporation, except as otherwise permitted by law. Except as expressly set forth in such license agreement or non-disclosure agreement, no part of this document or the software described in this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, or otherwise, without the prior written consent of Centrifly Corporation. Some companies, names, and data in this document are used for illustration purposes and may not represent real companies, individuals, or data.

This document could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein. These changes may be incorporated in new editions of this document. Centrifly Corporation may make improvements in or changes to the software described in this document at any time.

© 2004-2019 Centrifly Corporation. All rights reserved. Portions of Centrifly software are derived from third party or open source software. Copyright and legal notices for these sources are listed separately in the Acknowledgements.txt file included with the software.

U.S. Government Restricted Rights: If the software and documentation are being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), in accordance with 48 C.F.R. 227.7202-4 (for Department of Defense (DOD) acquisitions) and 48 C.F.R. 2.101 and 12.212 (for non-DOD acquisitions), the government’s rights in the software and documentation, including its rights to use, modify, reproduce, release, perform, display or disclose the software or documentation, will be subject in all respects to the commercial license rights and restrictions provided in the license agreement.

Centrifly, DirectControl, DirectAuthorize, DirectAudit, DirectSecure, DirectControl Express, Centrifly for Mobile, Centrifly for SaaS, DirectManage, Centrifly Express, DirectManage Express, Centrifly Suite, Centrifly User Suite, Centrifly Identity Service, Centrifly Privilege Service and Centrifly Server Suite are registered trademarks of Centrifly Corporation in the United States and other countries. Microsoft, Active Directory, Windows, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.

Centrifly software is protected by U.S. Patents 7,591,005; 8,024,360; 8,321,523; 9,015,103; 9,112,846; 9,197,670; 9,442,962 and 9,378,391.

The names of any other companies and products mentioned in this document may be the trademarks or registered trademarks of their respective owners. Unless otherwise noted, all of the names used as examples of companies, organizations, domain names, people and events herein are fictitious. No association with any real company, organization, domain name, person, or event is intended or should be inferred.



# Contents

<b>About this guide .....</b>	<b>22</b>
Intended audience .....	22
Limitations of this guide .....	22
Documentation conventions .....	23
Finding more information about Centrify products .....	23
Product names .....	24
Contacting Centrify .....	26
Getting additional support .....	26
<b>Working with parameters and agent configuration files</b>	<b>27</b>
Controlling agent operations .....	27
Basic syntax used in configuration files .....	28
Setting configuration parameter names .....	28
Setting configuration parameter values .....	29
Rereading parameter values after making changes .....	30
Securing parameter settings .....	31
Using group policies to configure settings .....	31
Parameters and values are subject to change .....	32
<b>Customizing adclient configuration parameters .....</b>	<b>34</b>
adclient.altupns .....	40
adclient.autoedit .....	41
adclient.binding.dc.failover.delay .....	46
adclient.binding.idle.time .....	47
adclient.binding.refresh.force .....	47
adclient.binding.refresh.interval .....	47
adclient.get.builtin.membership .....	48
adclient.cache.cleanup.interval .....	48



adclient.cache.encrypt .....	49
adclient.cache.encryption.type .....	49
adclient.cache.expires .....	50
adclient.cache.expires.computer .....	51
adclient.cache.expires.extension .....	51
adclient.cache.expires.gc .....	52
adclient.cache.expires.group .....	53
adclient.cache.expires.group.membership .....	54
adclient.cache.expires.search .....	55
adclient.cache.expires.user .....	55
adclient.cache.expires.user.membership .....	56
adclient.cache.flush.interval .....	57
adclient.cache.negative.lifetime .....	57
adclient.cache.object.lifetime .....	58
adclient.cache.refresh .....	58
adclient.cache.refresh.computer .....	59
adclient.cache.refresh.extension .....	60
adclient.cache.refresh.gc .....	61
adclient.cache.refresh.group .....	61
adclient.cache.refresh.search .....	62
adclient.cache.refresh.user .....	63
adclient.cache.upn.index .....	64
adclient.client.idle.timeout .....	64
adclient.clients.listen.backlog .....	65
adclient.clients.socket .....	65
adclient.clients.threads .....	65
adclient.clients.threads.max .....	66
adclient.clients.threads.poll .....	66
adclient.cloud.auth.token.max .....	67



adclient.cloud.cert.store .....	67
adclient.cloud.connector .....	68
adclient.cloud.connector.refresh.interval .....	68
adclient.cloud.skip.cert.verification .....	68
adclient.cloud.connector.subnet.preference.enabled .....	69
adclient.custom.attributes .....	69
adclient.disk.check.free .....	70
adclient.disk.check.interval .....	70
adclient.dns.cache.timeout .....	71
adclient.dns.cachingserver .....	72
adclient.dumpcore .....	73
adclient.dynamic.dns.command .....	73
adclient.dynamic.dns.enabled .....	74
adclient.dynamic.dns.refresh.interval .....	74
adclient.excluded.domains .....	74
adclient.exit.on.incomplete.zone.hierarchy .....	75
adclient.fetch.object.count .....	75
adclient.force.salt.lookup .....	76
adclient.get.primarygroup.membership .....	76
adclient.hash.allow .....	77
adclient.hash.deny .....	77
adclient.hash.expires .....	78
adclient.ignore.setgrpsrc .....	78
adclient.included.domains .....	79
adclient.iterate.private.groups .....	80
adclient.krb5.principal.lower .....	80
adclient.krb5.conf.domain_realm.anysite .....	81
adclient.ldap.packet.encrypt .....	81
adclient.ldap.socket.timeout .....	82



adclient.ldap.timeout .....	82
adclient.ldap.timeout.search .....	83
adclient.ldap.trust.enabled .....	83
adclient.ldap.trust.timeout .....	84
adclient.legacyzone.mfa.background.fetch.interval .....	84
adclient.legacyzone.mfa.cloudurl .....	85
adclient.legacyzone.mfa.enabled .....	86
adclient.legacyzone.mfa.required.groups .....	86
adclient.legacyzone.mfa.required.users .....	87
adclient.legacyzone.mfa.rescue.users .....	89
adclient.legacyzone.mfa.tenantid .....	90
adclient.local.account.manage .....	90
adclient.local.account.manage.strict .....	91
adclient.local.account.notification.cli .....	91
adclient.local.account.notification.cli.arg.length.max .....	92
adclient.local.forest.altupn.lookup .....	92
adclient.local.group.merge .....	92
adclient.logonhours.local.enforcement .....	93
adclient.lookup.sites .....	94
adclient.lrpc2.receive.timeout .....	95
adclient.lrpc2.send.timeout .....	95
adclient.ntlm.domains .....	95
adclient.ntlm.separators .....	96
adclient.one-way.x-forest.trust.force .....	97
adclient.os.name .....	98
adclient.os.version .....	98
adclient.os.version.use.win7prefix .....	98
adclient.paged.search.max .....	99
adclient.prefer.cache.validation .....	99



adclient.preferred.login.domains .....	99
adclient.preferred.site .....	100
adclient.prevalidate.allow.groups .....	101
adclient.prevalidate.allow.users .....	104
adclient.prevalidate.deny.groups .....	106
adclient.prevalidate.deny.users .....	107
adclient.prevalidate.interval .....	107
adclient.prevalidate.service .....	108
adclient.random.password.generate.try .....	108
adclient.random.password.complexity.pattern .....	108
adclient.random.password.length.min .....	109
adclient.random.password.length.max .....	109
adclient.samba.sync .....	109
adclient.server.try.max .....	110
adclient.skip.inbound.trusts .....	111
adclient.skip.unused.outbound.trusts .....	111
adclient.sntp.enabled .....	112
adclient.sntp.poll .....	112
adclient.tcp.connect.timeout .....	113
adclient.udp.timeout .....	113
adclient.update.os.interval .....	113
adclient.use.all.cpus .....	114
adclient.use.tokengroups .....	114
adclient.user.computers .....	115
adclient.user.lookup.cn .....	115
adclient.user.lookup.display .....	115
adclient.user.name.max.exceed.disallow .....	116
adclient.version2.compatible .....	116
adclient.zone.group.count .....	117



addns.tcp.timeout .....	118
addns.wait.time .....	118
adjust.offset .....	118
audittrail.audited.command.with.args .....	119
audittrail.Centrify_Suite.Trusted_Path.machinecred.skipda .....	119
audittrail.targets .....	119
audittrail.<product>.<component>.overrides .....	120
audittrail.<product>.<component>.targets .....	121
adsec.gpfas .....	122
capi.cache.enabled .....	122
capi.cache.hash.table.size .....	123
capi.cache.log.interval .....	123
capi.cache.max.objects .....	124
capi.cache.negative.ttl .....	124
capi.cache.ttl .....	124
db2.implement.pam.ignore.users .....	125
db2.user.zone_enabled .....	125
db2.userpass.username.lower .....	126
dc.dead.cache.refresh .....	126
dc.live.cache.refresh .....	126
dc.penalty.time .....	126
dns.alive.resweep.interval .....	127
dns.block .....	127
dns.cache.negative .....	128
dns.cache.timeout .....	128
dns.dc.domain_name .....	128
dns.dead.resweep.interval .....	129
dns.gc.domain_name .....	130
dns.query.all.servers .....	130



dns.servers .....	131
dns.sort .....	131
dns.sweep.pattern .....	132
dns.tcp.timeout .....	133
dns.udp.timeouts .....	133
domain.dead.cache.refresh .....	134
domain.live.cache.refresh .....	134
fips.mode.enable .....	134
log .....	136
logger.facility.adclient .....	136
logger.facility.adclient.audit .....	137
logger.facility.diag .....	137
logger.memory.bufsize .....	138
logger.memory.enabled .....	138
logger.memory.log .....	138
logger.queue.size .....	139
lrpc.connect.timeout .....	139
lrpc.session.timeout .....	140
lrpc.timeout .....	140
secdit.system.access.lockout.allowofflinelogin .....	141
queueable.random.delay.interval .....	141

## Customizing Kerberos-related configuration parameters ..... 142

adclient.dc.switch.update.krb5.conf .....	144
adclient.krb5.allow_weak_crypto .....	144
adclient.krb5.autoedit .....	145
adclient.krb5.cache.renewal.service.accounts .....	145
adclient.krb5.ccache.dir .....	146



adclient.krb5.conf.file.custom .....	147
adclient.krb5.conf.domain_realm.anysite .....	149
adclient.krb5.extra_addresses .....	149
adclient.krb5.keytab.clean.nonfips.enctypes .....	150
adclient.krb5.keytab.entries .....	150
adclient.krb5.password.change.interval .....	150
adclient.krb5.password.change.verify.interval .....	151
adclient.krb5.password.change.verify.retries .....	151
adclient.krb5.passwd_check_s_address .....	152
adclient.krb5.permitted.encryption.types .....	152
adclient.krb5.permitted.encryption.types.strict .....	153
adclient.krb5.principal .....	154
adclient.krb5.send.netbios.name .....	154
adclient.krb5.service.principals .....	155
adclient.krb5.tkt.encryption.types .....	155
adclient.krb5.tkt.encryption.type.strict .....	155
adclient.krb5.use_addresses .....	157
fips.mode.enable .....	157
krb5.cache.clean .....	158
krb5.cache.clean.exclusion .....	158
krb5.cache.clean.force.max .....	159
krb5.cache.clean.interval .....	159
krb5.cache.infinite.renewal .....	160
krb5.cache.infinite.renewal.batch.groups .....	160
krb5.cache.infinite.renewal.batch.users .....	161
krb5.cache.renew.exclusion .....	161
krb5.cache.renew.interval .....	162
krb5.conf.plugins.ccselect.disable .....	162
krb5.cache.type .....	163



krb5.conf.k5login.directory .....	164
krb5.conf.kcm.socket.path .....	164
krb5.config.update .....	165
krb5.forcetcp .....	166
krb5.forwardable.user.tickets .....	166
krb5.pac.validation .....	167
krb5.permit.dns.spn.lookups .....	167
krb5.sso.block.local_user .....	168
krb5.sso.ignore.k5login .....	168
krb5.support.alt.identities .....	168
krb5.unique.cache.files .....	169
krb5.use.kdc.timesync .....	169
krb5.verify.credentials .....	170
krb5.udp.preference.limit .....	170

## Customizing PAM-related configuration parameters ....172

Configuring PAM-related parameters on IBM AIX computers .....	174
pam.account.conflict.both.mesg .....	175
pam.account.conflict.name.mesg .....	176
pam.account.conflict.uid.mesg .....	176
pam.account.disabled.mesg .....	177
pam.account.expired.mesg .....	177
pam.account.locked.mesg .....	177
pam.adclient.down.mesg .....	178
pam.allow.groups .....	178
pam.allow.override .....	180
pam.allow.password.change .....	181
pam.allow.password.change.mesg .....	182
pam.allow.password.expired.access .....	182



pam.allow.password.expired.access.mesg .....	182
pam.allow.users .....	183
pam.auth.create.krb5.cache .....	184
pam.auth.failure.mesg .....	185
pam.config.program.check .....	185
pam.create.k5login .....	185
pam.deny.change.shell .....	186
pam.deny.groups .....	186
pam.deny.users .....	188
pam.homedir.create .....	189
pam.homedir.create.mesg .....	190
pam.homedir.perms .....	190
pam.homedir.perms.recursive .....	190
pam.homeskel.dir .....	190
pam.ignore.users .....	191
pam.mapuser.username .....	192
pam.mfa.program.ignore .....	193
pam.ntlm.auth.domains .....	194
pam.password.change.mesg .....	195
pam.password.change.required.mesg .....	195
pam.password.confirm.mesg .....	196
pam.password.empty.mesg .....	196
pam.password.enter.mesg .....	196
pam.password.expiry.warn .....	197
pam.password.expiry.warn.mesg .....	197
pam.password.new.mesg .....	197
pam.password.new.mismatch.mesg .....	198
pam.password.old.mesg .....	198
pam.policy.violation.mesg .....	198



pam.setcred.respect.sufficient .....	199
pam.setcred.support.refresh .....	199
pam.setcred.support.reinitialize .....	199
pam.sync.mapuser .....	200
pam.uid.conflict .....	201
pam.workstation.denied.mesg .....	201
microsoft.pam.privilege.escalation.enabled .....	202

## Customizing group policy configuration parameters .....203

gp.disable.all .....	204
gp.disable.machine .....	204
gp.disable.user .....	204
gp.disk.space.check.folders .....	205
gp.disk.space.min .....	205
gp.mappers.certgp.pl.additional.cafiles .....	205
gp.mappers.certgp.pl.exclude.cacerts .....	206
gp.mappers.directory.machine .....	206
gp.mappers.directory.user .....	206
gp.mappers.error_file .....	207
gp.mappers.machine .....	207
gp.mappers.runmappers .....	208
gp.mappers.timeout .....	208
gp.mappers.timeout.all .....	209
gp.mappers.umask .....	209
gp.mappers.user .....	210
gp.refresh.disable .....	210
gp.reg.directory.machine .....	211
gp.reg.directory.user .....	211
gp.use.user.credential.for.user.policy .....	211



gp.user.login.run .....	212
-------------------------	-----

## Customizing NSS-related configuration parameters .....213

nss.alias.source .....	214
nss.gecos.attribute .....	214
nss.gid.ignore .....	215
nss.group.ignore .....	216
nss.group.override .....	217
nss.group.skip.members .....	218
nss.nobody.gid .....	218
nss.nobody.group .....	219
nss.nobody.uid .....	219
nss.nobody.user .....	219
nss.passwd.hash .....	219
nss.passwd.info.hide .....	220
nss.passwd.override .....	220
nss.program.ignore .....	222
nss.shell.emergency.enabled .....	223
nss.shell.nologin .....	223
nss.split.group.membership .....	224
nss.squash.root .....	225
nss.uid.ignore .....	225
nss.user.ignore .....	226
nss.user.ignore.all .....	227
lam.attributes.group.ignore .....	227
lam.attributes.user.ignore .....	228
lam.max.group.count .....	228
lam.max.user.count .....	228



## Customizing NIS configuration parameters .....230

log.adnisd .....	231
log.adnisd.netgroup .....	231
logger.facility.adnisd .....	232
nisd.domain.name .....	232
nisd.exclude.maps .....	232
nisd.largegroup.name.length .....	233
nisd.largegroup.suffix .....	233
nisd.maps .....	234
nisd.maps.max .....	235
nisd.net_addr .....	235
nisd.passwd.expired.allow .....	236
nisd.port.tcp .....	236
nisd.port.udp .....	237
nisd.securenets .....	237
nisd.server.switch.delay .....	238
nisd.startup.delay .....	238
nisd.threads .....	239
nisd.update.interval .....	239

## Customizing AIX configuration parameters .....240

Setting extended attributes .....	241
aix.cache.extended.attr.enable .....	243
aix.user.attr.admgroups .....	244
aix.user.attr.admin .....	244
aix.user.attr.auditclasses .....	244
aix.user.attr.core .....	245
aix.user.attr.cpu .....	245
aix.user.attr.data .....	245



aix.user.attr.daemon .....	246
aix.user.attr.fsize .....	246
aix.user.attr.nofiles .....	246
aix.user.attr.nprocs .....	247
aix.user.attr.rlogin .....	247
aix.user.attr.rss .....	247
aix.user.attr.stack .....	248
aix.user.attr.su .....	248
aix.user.attr.sugroups .....	248
aix.user.attr.threads .....	249
aix.user.attr.tpath .....	249
aix.user.attr.ttys .....	250
aix.user.attr.umask .....	250

## Customizing Centrifry UNIX programs configuration parameters ..... 251

adjoin.adclient.wait.seconds .....	251
adjoin.krb5.conf.file .....	252
adjoin.samaccountname.length .....	252
adpasswd.account.disabled.mesg .....	253
adpasswd.account.invalid.mesg .....	253
adpasswd.password.change.disabled.mesg .....	253
adpasswd.password.change.perm.mesg .....	253

## Customizing smart card configuration parameters ..... 255

rhel.smartcard.pkcs11.module .....	255
smartcard.allow.noeku .....	256
smartcard.name.mapping .....	256



## Customizing authorization configuration parameters ... 257

adclient.azman.refresh.interval .....	258
adclient.cache.flush.interval.dz .....	259
adclient.dzdo.clear.passwd.timestamp .....	259
adclient.refresh.interval.dz .....	260
adclient.sudo.clear.passwd.timestamp .....	260
adclient.sudo.timestampdir .....	262
audittrail.dz.command.with.args .....	262
dz.auto.anchors .....	262
dz.enabled .....	263
dz.system.path .....	263
dz.user.path .....	264
dzdo.always_set_home .....	264
dzdo.badpass_message .....	264
dzdo.command_alias .....	265
dzdo.edit.checkdir .....	265
dzdo.edit.follow .....	266
dzdo.env_check .....	266
dzdo.env_delete .....	267
dzdo.env_keep .....	268
dzdo.lecture .....	268
dzdo.lecture_file .....	269
dzdo.legacyzone.mfa.enabled .....	269
dzdo.log_good .....	269
dzdo.passprompt .....	270
dzdo.passwd_timeout .....	270
dzdo.path_info .....	271
dzdo.search_path .....	271
dzdo.requiretty .....	272



dzdo.secure_path .....	272
dzdo.set_home .....	273
dzdo.set.runas.explicit .....	273
dzdo.timestampdir .....	274
dzdo.timestamp_timeout .....	275
dzdo.tty_tickets .....	275
dzdo.use.realpath .....	275
dzdo.user.command.timeout .....	276
dzdo.validator .....	276
dzdo.validator.required .....	277
dzsh.roleswitch.silent .....	277

## Customizing Auto Zone configuration parameters .....279

auto.schema.allow.groups .....	280
auto.schema.allow.users .....	282
auto.schema.apple_scheme .....	284
auto.schema.domain.prefix .....	284
auto.schema.groups .....	285
auto.schema.homedir .....	287
auto.schema.primary.gid .....	287
auto.schema.private.group .....	288
auto.schema.shell .....	289
auto.schema.use.adhomedir .....	289
auto.schema.remote.file.service .....	289
auto.schema.name.format .....	290
auto.schema.separator .....	290
auto.schema.search.return.max .....	291
auto.schema.name.lower .....	291
auto.schema.iterate.cache .....	291



auto.schema.override.uid .....	291
auto.schema.uid.conflict .....	292
auto.schema.homedir.illegal_chars .....	292
auto.schema.unix.name.disallow.chars .....	293
auto.schema.substitute.chars .....	293
auto.schema.max.unix.name.length .....	294

## Customizing auditing configuration parameters ..... 295

agent.max.missed.update.tolerance .....	297
agent.send.hostname .....	298
agent.video.capture .....	298
autofix.nss.conf .....	299
cache.enable .....	299
cache.max.size .....	300
cache.time.to.live .....	300
dad.client.idle.timeout .....	300
dad.collector.connect.timeout .....	300
dad.dumpcore .....	301
dad.gssapi.seal .....	301
dad.gssapi.sign .....	301
dad.process.fdlimit .....	301
dad.resource.cpulimit .....	302
dad.resource.cpulimit.tolerance .....	302
dad.resource.fdlimit .....	303
dad.resource.memlimit .....	303
dad.resource.restart .....	304
dad.resource.timer .....	305
dad.timer.diskspace .....	305
dad.timer.monitor.nss.conf .....	305



dash.allinvoked .....	306
dash.auditstdin .....	306
dash.auditstdin.except .....	306
dash.cmd.audit.blacklist .....	307
dash.cmd.audit.show.actual.user .....	307
dash.cont.without.dad .....	307
dash.force.audit .....	307
dash.loginrecord .....	308
dash.obfuscate.pattern .....	308
dash.obfuscate.regex .....	310
dash.parent.skiplist .....	310
dash.reconnect.dad.retry.count .....	311
dash.reconnect.dad.wait.time .....	311
dash.shell.env.var.set .....	311
dash.ssh.command.skiplist .....	311
dash.user.alwaysallowed.list .....	312
dash.user.skiplist .....	312
event.execution.monitor .....	312
event.execution.monitor.user.skiplist .....	313
event.file.monitor .....	313
event.file.monitor.process.skiplist .....	314
event.file.monitor.user.skiplist .....	314
event.monitor.commands .....	315
event.monitor.commands.user.skiplist .....	315
lang_setting .....	315
lrpc2.message.signing .....	316
lrpc2.timeout .....	316
lrpc2.rebind.timeout .....	316
nss.alt.zone.auditlevel .....	316



nss.nologin.shell .....	317
nss.user.conflict.auditlevel .....	318
nss.user.override.auditlevel .....	318
nss.user.override.userlist .....	318
preferred.audit.store .....	320
spool.diskspace.logstate.reset.threshold .....	320
spool.diskspace.min .....	320
spool.diskspace.softlimit .....	321
spool.maxdbsize .....	322
uid.ignore .....	322
user.ignore .....	323
user.ignore.audit.level .....	323

## Customizing LDAP Proxy configuration parameters ..... 325

ldaproxy.cache.credential.expire .....	325
ldaproxy.netgroup.use.rfc2307nisnetgroup .....	326
ldaproxy.performance.log.interval .....	326



# About this guide

The *Configuration and Tuning Reference Guide* provides reference information for Centrify configuration parameters. Configuration parameters can be set locally on Linux, UNIX, and Mac OS X computers to fine tune the operation of Centrify Authentication Service, Privilege Elevation Service, and Audit & Monitoring Service components and subsystems. Centrify Authentication Service, Privilege Elevation Service, and Audit & Monitoring Service is an integrated software solution that delivers secure access control and centralized identity management through Microsoft Active Directory. With Centrify software, your organization can improve IT efficiency, regulatory compliance, and security for on-premise, mobile, and hosted resources.

## Intended audience

The *Configuration and Tuning Reference Guide* is intended for administrators who want to customize the operation of Centrify components and subsystems by modifying locally-defined configuration parameters. Many of these operations can also be configured remotely using group policies.

This guide is intended as a supplement to the main Centrify Authentication Service, Privilege Elevation Service, and Audit & Monitoring Service documentation set. It assumes that you have a working knowledge of Centrify components and administration. For information about planning a deployment and installing components, see the *Planning and Deployment Guide*. For information about performing administrative tasks using Access Manager, see the *Administrator's Guide for Linux and UNIX*.

## Limitations of this guide

This guide is updated with every major release of Centrify Authentication Service, Privilege Elevation Service, and Audit & Monitoring Service. Because the supported configuration parameters can change from one release to



another, have different default values between releases, or be designed to address very specific conditions, you should consider the configuration files (`centrifdc.conf` and `centrifda.conf` for example) included with the software to be the definitive source of information for the parameters in the version of the software you are using. If there are differences between the information in the configuration files and this guide, you should consider the comments in the configuration file itself to be the most current or accurate for your environment.

## Documentation conventions

The following conventions are used in Centrifly documentation:

- Fixed-width font is used for sample code, program names, program output, file names, and commands that you type at the command line. When *italicized*, this font indicates variables. Square brackets (`[ ]`) indicate optional command-line arguments.
- **Bold** text is used to emphasize commands or key command results; buttons or user interface text; and new terms.
- *Italics* are used for book titles and to emphasize specific words or terms. In fixed-width font, italics indicate variable values.
- Standalone software packages include version and architecture information in the file name. Full file names are not documented in this guide. For complete file names for the software packages you want to install, see the distribution media.
- For simplicity, UNIX is used to refer to all supported versions of the UNIX and Linux operating systems. Some parameters can also be used on Mac OS X computers.

## Finding more information about Centrifly products

Centrifly provides extensive documentation targeted for specific audiences, functional roles, or topics of interest. If you want to learn more about Centrifly and Centrifly products and features, start by visiting the [Centrifly website](#). From the Centrifly website, you can download data sheets and evaluation software, view video demonstrations and technical presentations about Centrifly products, and get the latest news about upcoming events and webinars.



For access to documentation for all Centrify products and services, visit the [Centrify documentation portal](https://docs.centrify.com) at [docs.centrify.com](https://docs.centrify.com). From the Centrify documentation portal, you can always view or download the most up-to-date version of this guide and all other product documentation.

For details about supported platforms, please consult the release notes.

For the most up to date list of known issues, please login to the Customer Support Portal at <http://www.centrify.com/support> and refer to Knowledge Base articles for any known issues with the release.

## Product names

Over the years we've made some changes to some of our product offerings and features and some of these previous product names still exist in some areas. Our current product offerings include the following services:

Current Overall Product Name	Current Services Available
Centrify Zero Trust Privilege Services	Privileged Access Service
	Gateway Session Audit and Monitoring
	Authentication Service
	Privilege Elevation Service
	Audit and Monitoring Service
	Privilege Threat Analytics Service

Whether you're a long-time or new customer, here are some quick summaries of which features belong to which current product offerings:

Previous Product Offering	Previous Product Offering	Description	Current Product Offering
	Centrify Privileged Service (CPS)		Privileged Access Service
DirectControl (DC)			Authentication Service
DirectAuthorize (DZ or DZwin)			Privilege Elevation Service
DirectAudit (DA)			Audit and Monitoring Service



Previous Product Offering	Previous Product Offering	Description	Current Product Offering
	Infrastructure Services		Privileged Access Service, Authentication Service, Privilege Elevation Service, Audit and Monitoring Service, and Privilege Threat Analytics Service
DirectManage (DM)	Management Services	Consoles that are used by all 3 services: Authentication Service, Privilege Elevation Service, and Audit and Monitoring Service	
DirectSecure (DS)	Isolation and Encryption Service		Still supported but no longer being developed or updated
	User Analytics Service		Privilege Threat Analytics Service

Depending on when you purchased a Centrify product offering, you may have purchased one of the following product bundles:

Previous Product Bundle	Previous Product Bundle	Current Product Bundle	Services Included	Description
		Centrify Zero Trust Privilege Services Core Edition	Privileged Access Service and Gateway Session Audit and Monitoring	
Centrify Server Suite Standard Edition	Centrify Infrastructure Services Standard Edition	Centrify Zero Trust Privilege Services Standard Edition	Privileged Access Service, Authentication Service, and Privilege Elevation Service	
Centrify Server Suite Enterprise Edition	Centrify Infrastructure Services Enterprise Edition	Centrify Zero Trust Privilege Services	Privileged Access Service, Authentication Service, Privilege Elevation Service, Audit and Monitoring Service (includes Gateway Session	



Previous Product Bundle	Previous Product Bundle	Current Product Bundle	Services Included	Description
		Enterprise Edition	Audit and Monitoring)	
Centrify Server Suite Platinum Edition				Discontinued bundle that included DirectControl, DirectAuthorize, DirectManage, DirectAudit, and DirectSecure

## Contacting Centrify

You can contact Centrify by visiting our website, [www.centrify.com](http://www.centrify.com). On the website, you can find information about Centrify office locations worldwide, email and phone numbers for contacting Centrify sales, and links for following Centrify on social media. If you have questions or comments, we look forward to hearing from you.

## Getting additional support

If you have a Centrify account, click Support on the Centrify website to log on and access the [Centrify Technical Support Portal](#). From the support portal, you can search knowledge base articles, open and view support cases, download software, and access other resources.

To connect with other Centrify users, ask questions, or share information, visit the [Centrify Community](#) website to check in on customer forums, read the latest blog posts, view how-to videos, or exchange ideas with members of the community.



# Working with parameters and agent configuration files

The Centrify agent configuration files, `centrifydc.conf` and `centrifyda.conf`, can be used to customize and control the operation of Centrify Authentication Service, Privilege Elevation Service, and Audit & Monitoring Service components and subsystems on a local host computer. This chapter provides an introduction to using the configuration file and setting values for the configuration parameters defined in the file.

## Controlling agent operations

The Centrify configuration file for access control and privilege management is `/etc/centrifydc/centrifydc.conf`. The Centrify configuration file for auditing is `/etc/centrifyda/centrifyda.conf`. Depending on the deployment options selected when you install the agent, one or both of these files might be available on each Centrify-managed computer. The configuration files contain parameters that specify how Centrify components and subsystems operate on the local computer. They can be used to tune operations to suit your environment, for example to address bandwidth or latency constraints or address specific requirements, for example, to prevent the storage of a password hash. Many of the operations controlled locally by configuration parameters can also be controlled remotely using group policies. For information about customizing operations using group policies, see the *Group Policies Guide*.

You only have to edit the `/etc/centrifydc/centrifydc.conf` or `/etc/centrifyda/centrifyda.conf` file if you want to set custom values for one or more configuration parameters. For most organizations, the default



values are appropriate. However, if you decide that you want to use a custom value for any parameter, you can uncomment the parameter name in the appropriate configuration file, then set an appropriate parameter value in place of the default value.

**Note:** In most cases, you only modify settings in the configuration files if you want to customize specific behavior locally on an individual computer. For most parameters, you can make changes, then run the `adreload` command to have the changes take effect immediately. Some parameters, however, require you to restart the agent (`adclient`). Similarly, if you make changes to the configuration parameters used by the Centrify Network Information Service (`adnisd`), you may need to run the `adreload` command or restart that service.

## Basic syntax used in configuration files

Configuration parameters are defined using a key/value pair that identifies the configuration parameter name and the value assigned to that parameter. If a configuration parameter is not explicitly set in the configuration file, the Centrify agent assumes a default value for that parameter.

A key/value pair in the configuration file typically takes the following form:

```
parameter_name: value
```

where `parameter_name` is the name of the configuration parameter that describes the component the setting applies to or the purpose of the parameter, and `value` is the value assigned to that parameter. Variations in the formatting of the key/value pair are allowed. For example, the parameter name can be followed by a colon (:), equal sign (=), or a space:

```
parameter_name=value
```

```
parameter_name value
```

## Setting configuration parameter names

In most cases, parameter names are fixed strings that are defined in the default `centrifydc.conf` file and commented out to illustrate the default value or how to configure a setting. In some cases, however, the parameter name itself must be customized to enable a setting. For example, the configuration



parameter `pam.mapuser.localuser` must include the specific local user name you are mapping to an Active Directory account. For example, to map the local user `joan7` to the Active Directory user `joan.adams`, you must set the parameter name to `pam.mapuser.joan7` to specify that the mapping is for the local user `joan7`:

```
pam.mapuser.joan7: joan.adams
```

## Setting configuration parameter values

Depending on the configuration parameter you are setting, the parameter value can be a string, a numeric value, or a Boolean value. For example, user names and group names defined in Active Directory are specified as strings using a valid Active Directory form, such as `user[@domain]`. In some cases, string parameter values can include environment variables.

In general, you can specify user names in the configuration file with any of the following valid formats:

- Standard Windows format: `domain\user_name`
- Universal Principal Name (UPN): `user_name@domain`
- Alternate UPN: `alt_user_name@alt_domain`
- UNIX user name: `user`

However, you must include the domain name in the format if the user account is not in the local computer's current Active Directory domain. In addition, if you are specifying an Active Directory logon name that contains spaces, you should use quotes around the string. For example:

```
adclient.hash.allow: 'marco sanchez@arcade.com'
```

## Using special characters

Configuration parameter values can include the following special characters that are often used in UNIX scripts:

- The dollar sign (\$) signifies an environment variable that can be resolved to an appropriate value if recognized by the agent. Valid environment variable names can consist of alphanumeric characters and underscores.



- A backslash (\) signifies that the next character is a literal, and is used, among other things, to specify a trailing space (\ ) or a single backslash (\\).

Boolean values are case-insensitive. The permissible values are `true`, `yes`, `false`, and `no`.

If a parameter can take multiple values, those values are separated from each other by a comma or a space. Spaces preceding or trailing each value are ignored.

## Using environment variables

The values in key/value pairs can include standard shell environment variables. The variables are resolved to their current value when the Centrify agent reads the configuration file. For example, you can use the environment variable `$HOSTNAME` to include the local computer's host name in any parameter value setting:

```
example_parameter: test_${HOSTNAME}
```

If the name of the current managed computer is `host1`, the configuration parameter `example_parameter` takes the value `test_host1`.

In addition to standard environment variables, you can use the following Centrify-specific environment variables in the configuration file:

- `$ZONE` is the name of the host computer's Centrify zone.
- `$JOINNAME` is the name of the host computer's account name in Active Directory.
- `$DOMAIN` is the name of the Active Directory domain to which the host computer is joined.
- `$SITE` is the name of the Active Directory site for the host computer.

## Rereading parameter values after making changes

In most cases, you can either run the `adreload` command or restart the agent (`adclient`) to have changes to any configuration parameter take effect. Running the `adreload` command or restarting the `adclient` process forces the Centrify agent to reread the configuration parameters that have been defined,



including any values that have changed since the last time the configuration file was read.

For most configuration parameters, you can run the `adreload` command to have changes take effect without restarting the `adcli` process. There are a few configuration parameters, however, that cannot be reloaded by running the `adreload` command. If you want to ensure that the agent rereads all configuration parameters, you should restart the `adcli` process. For example, to ensure all changes to `adcli`-related configuration parameters take effect, you can restart the `adcli` process.

Similarly, if you make changes to the configuration parameters used by the Centrify Network Information Service (`adnisd`), you can either run the `adreload` command or restart the `adnisd` service to ensure those changes take effect. If you change LRPC- or NSS-related parameters, you should restart both the `adcli` and `adnisd` processes if both are running when you make the change.

## Securing parameter settings

By default, the configuration files—`centrifydc.conf` and `centrifyda.conf`—are owned by `root`. In most cases, therefore, the parameter settings you specify are secure because they can only be set or modified by the `root` user and access to the `root` account is tightly controlled. However, there are many parameters that allow you to specify settings in an external file. For example, the `pam.allow.groups` parameter allows you to specify a list of groups in an external file, then set the parameter value to use the `file:` keyword and the file path and file name of that external file.

If you are using an external file to configure parameter settings, you should ensure that the external file meets the following security requirements:

- The external file is owned by `root` or an equivalently-protected account.
- The external file is not group or world writable.
- The path you specify to the external file is not a symbolic link.

## Using group policies to configure settings

Many configuration parameter values can be controlled by enabling and applying corresponding Centrify group policies through the Group Policy



Management Editor. When you use group policies to set configuration parameters, the group policy setting overrides any local configuration setting and the group policy setting is reapplied if the computer is rebooted and periodically when the group policy is automatically refreshed. Therefore, if a group policy exists for configuring a specific setting, in most cases, you should use the group policy rather than edit the local configuration file.

If no group policy exists for a configuration parameter you want to change or if no group policy is applied to the local computer, you can customize the local configuration file to set configuration parameter values as needed.

To determine whether a group policy exists to configure a specific setting, and which group policies affect which settings, see the *Group Policy Guide* ([centrify-grouppolicy-guide.pdf](#)). When you open the Group Policy Guide PDF file, use the PDF reader search function to search for a setting (for example, `adclient.cache.expires.gc`). If the setting can be configured with a group policy, the setting is referred to in the group policy description.

**Note:** It is possible for an Active Directory administrator to override virtually any setting in the local configuration file using group policies applied to a local computer. This effectively gives administrators with permission to enable or disable group policies root-level access to computers in the zones they manage. There is no way to effectively prevent settings from being changed, except by disabling user, computer, or all group policies in the local `centrifydc.conf` or `centrifyda.conf` file or by strictly controlling who has permission to enable and apply group policies to computers that join an Active Directory domain.

For information about disabling the application of group policies using settings in the local `centrifydc.conf` file, see [Customizing group policy configuration parameters](#). For more information about enabling and applying group policies rather than setting configuration parameters locally on a computer, see the *Group Policy Guide*.

## Parameters and values are subject to change

Configuration parameters are added, updated, and retired with each release of the Centrify agent. In addition, some parameters are intended only for specific circumstances and are intentionally not documented in this guide. If a configuration parameter setting is recommended by Centrify Support, but not documented in this guide, you should consider the recommendation made by



Support to be authoritative. You should also consider the comments in the configuration file to be the most authoritative reference for the release of the software you are using. Because parameters are often created to address specific issues in specific environments, the default values and recommendations for changes to the default values are also subject to change from one release to another.



# Customizing adclient configuration parameters

This section describes the configuration parameters that affect the operation of the core agent (`adclient`) process on the local host computer.

<code>adclient.altupns</code> .....	40
<code>adclient.autoedit</code> .....	41
<code>adclient.binding.dc.failover.delay</code> .....	46
<code>adclient.binding.idle.time</code> .....	47
<code>adclient.binding.refresh.force</code> .....	47
<code>adclient.binding.refresh.interval</code> .....	47
<code>adclient.get.builtin.membership</code> .....	48
<code>adclient.cache.cleanup.interval</code> .....	48
<code>adclient.cache.encrypt</code> .....	49
<code>adclient.cache.encryption.type</code> .....	49
<code>adclient.cache.expires</code> .....	50
<code>adclient.cache.expires.computer</code> .....	51
<code>adclient.cache.expires.extension</code> .....	51
<code>adclient.cache.expires.gc</code> .....	52
<code>adclient.cache.expires.group</code> .....	53
<code>adclient.cache.expires.group.membership</code> .....	54
<code>adclient.cache.expires.search</code> .....	55
<code>adclient.cache.expires.user</code> .....	55



adclient.cache.expires.user.membership .....	56
adclient.cache.flush.interval .....	57
adclient.cache.negative.lifetime .....	57
adclient.cache.object.lifetime .....	58
adclient.cache.refresh .....	58
adclient.cache.refresh.computer .....	59
adclient.cache.refresh.extension .....	60
adclient.cache.refresh.gc .....	61
adclient.cache.refresh.group .....	61
adclient.cache.refresh.search .....	62
adclient.cache.refresh.user .....	63
adclient.cache.upn.index .....	64
adclient.client.idle.timeout .....	64
adclient.clients.listen.backlog .....	65
adclient.clients.socket .....	65
adclient.clients.threads .....	65
adclient.clients.threads.max .....	66
adclient.clients.threads.poll .....	66
adclient.cloud.auth.token.max .....	67
adclient.cloud.cert.store .....	67
adclient.cloud.connector .....	68
adclient.cloud.connector.refresh.interval .....	68
adclient.cloud.skip.cert.verification .....	68
adclient.cloud.connector.subnet.preference.enabled .....	69
adclient.custom.attributes .....	69
adclient.disk.check.free .....	70
adclient.disk.check.interval .....	70



adclient.dns.cache.timeout .....	71
adclient.dns.cachingserver .....	72
adclient.dumpcore .....	73
adclient.dynamic.dns.command .....	73
adclient.dynamic.dns.enabled .....	74
adclient.dynamic.dns.refresh.interval .....	74
adclient.excluded.domains .....	74
adclient.exit.on.incomplete.zone.hierarchy .....	75
adclient.fetch.object.count .....	75
adclient.force.salt.lookup .....	76
adclient.get.primarygroup.membership .....	76
adclient.hash.allow .....	77
adclient.hash.deny .....	77
adclient.hash.expires .....	78
adclient.ignore.setgrpsrc .....	78
adclient.included.domains .....	79
adclient.iterate.private.groups .....	80
adclient.krb5.principal.lower .....	80
adclient.krb5.conf.domain_realm.any site .....	81
adclient.ldap.packet.encrypt .....	81
adclient.ldap.socket.timeout .....	82
adclient.ldap.timeout .....	82
adclient.ldap.timeout.search .....	83
adclient.ldap.trust.enabled .....	83
adclient.ldap.trust.timeout .....	84
adclient.legacyzone.mfa.background.fetch.interval .....	84
adclient.legacyzone.mfa.cloudurl .....	85



adclient.legacyzone.mfa.enabled .....	86
adclient.legacyzone.mfa.required.groups .....	86
adclient.legacyzone.mfa.required.users .....	87
adclient.legacyzone.mfa.rescue.users .....	89
adclient.legacyzone.mfa.tenantid .....	90
adclient.local.account.manage .....	90
adclient.local.account.manage.strict .....	91
adclient.local.account.notification.cli .....	91
adclient.local.account.notification.cli.arg.length.max .....	92
adclient.local.forest.altupn.lookup .....	92
adclient.local.group.merge .....	92
adclient.logonhours.local.enforcement .....	93
adclient.lookup.sites .....	94
adclient.lrpc2.receive.timeout .....	95
adclient.lrpc2.send.timeout .....	95
adclient.ntlm.domains .....	95
adclient.ntlm.separators .....	96
adclient.one-way.x-forest.trust.force .....	97
adclient.os.name .....	98
adclient.os.version .....	98
adclient.os.version.use.win7prefix .....	98
adclient.paged.search.max .....	99
adclient.prefer.cache.validation .....	99
adclient.preferred.login.domains .....	99
adclient.preferred.site .....	100
adclient.prevalidate.allow.groups .....	101
adclient.prevalidate.allow.users .....	104



adclient.prevalidate.deny.groups .....	106
adclient.prevalidate.deny.users .....	107
adclient.prevalidate.interval .....	107
adclient.prevalidate.service .....	108
adclient.random.password.generate.try .....	108
adclient.random.password.complexity.pattern .....	108
adclient.random.password.length.min .....	109
adclient.random.password.length.max .....	109
adclient.samba.sync .....	109
adclient.server.try.max .....	110
adclient.skip.inbound.trusts .....	111
adclient.skip.unused.outbound.trusts .....	111
adclient.sntp.enabled .....	112
adclient.sntp.poll .....	112
adclient.tcp.connect.timeout .....	113
adclient.udp.timeout .....	113
adclient.update.os.interval .....	113
adclient.use.all.cpus .....	114
adclient.use.tokengroups .....	114
adclient.user.computers .....	115
adclient.user.lookup.cn .....	115
adclient.user.lookup.display .....	115
adclient.user.name.max.exceed.disallow .....	116
adclient.version2.compatible .....	116
adclient.zone.group.count .....	117
addns.tcp.timeout .....	118
addns.wait.time .....	118



adjust.offset .....	118
audittrail.audited.command.with.args .....	119
audittrail.Centrify_Suite.Trusted_Path.machinecred.skipda .....	119
audittrail.targets .....	119
audittrail.<product>.<component>.overrides .....	120
audittrail.<product>.<component>.targets .....	121
adsec.gpfas .....	122
capi.cache.enabled .....	122
capi.cache.hash.table.size .....	123
capi.cache.log.interval .....	123
capi.cache.max.objects .....	124
capi.cache.negative.ttl .....	124
capi.cache.ttl .....	124
db2.implement.pam.ignore.users .....	125
db2.user.zone_enabled .....	125
db2.userpass.username.lower .....	126
dc.dead.cache.refresh .....	126
dc.live.cache.refresh .....	126
dc.penalty.time .....	126
dns.alive.resweep.interval .....	127
dns.block .....	127
dns.cache.negative .....	128
dns.cache.timeout .....	128
dns.dc.domain_name .....	128
dns.dead.resweep.interval .....	129
dns.gc.domain_name .....	130
dns.query.all.servers .....	130



dns.servers .....	131
dns.sort .....	131
dns.sweep.pattern .....	132
dns.tcp.timeout .....	133
dns.udp.timeouts .....	133
domain.dead.cache.refresh .....	134
domain.live.cache.refresh .....	134
fips.mode.enable .....	134
log .....	136
logger.facility.adclient .....	136
logger.facility.adclient.audit .....	137
logger.facility.diag .....	137
logger.memory.bufsize .....	138
logger.memory.enabled .....	138
logger.memory.log .....	138
logger.queue.size .....	139
lrpc.connect.timeout .....	139
lrpc.session.timeout .....	140
lrpc.timeout .....	140
secdit.system.access.lockout.allowofflinelogin .....	141
queueable.random.delay.interval .....	141

## adclient.altupns

This configuration parameter specifies a UPN suffix that `adclient` will recognize as a valid UPN suffix even if it is a realm unknown by Kerberos.

The default value is “mil”.

For example, to specify “biz” as a suffix to recognize:



```
adclient.altupns: biz
```

You can also use multiple UPN suffixes separated by a space. For example, to specify “biz” and “mil” as suffixes to recognize:

```
adclient.altupns: biz mil
```

This parameter does not support wildcards (\*.acme.com) or preceding dots (.acme.com).

## adclient.autoedit

This configuration parameter specifies whether the agent is allowed to automatically edit the NSS and PAM configuration files on the local computer.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

The parameter value is set to `true` to allow the files to be edited or `false` to prevent the files from being edited. The following example allows both the NSS and PAM configuration files to be edited automatically:

```
adclient.autoedit: true
```

In most cases, this parameter should be set to `true` to allow the agent to maintain configuration files automatically. When this parameter is set to `true`, you can further control the specific individual files to be automatically edited in different operating environments through additional configuration parameters. For example, you can use the `adclient.autoedit.nss` to enable or disable automatic editing of the `nsswitch.conf` file or the `adclient.autoedit.pam` to enable or disable automatic editing of the PAM configuration file. These additional configuration parameters are ignored if the `adclient.autoedit` parameter is set to `false`. For more information about the configuration parameters to control the editing of specific files on different platforms, see [Enabling automatic editing for specific files](#).

If you set the `adclient.autoedit` parameter to `false`, you must manually edit the appropriate configuration files to enable agent operation. For example, if you set this parameter to `false`, you should manually edit the `nsswitch.conf` and `/etc/pam.d/system-auth` or `/etc/pam.d` files to include Centrify information or authentication through Active Directory will fail and you may disable login access entirely.



If you want to manually edit the configuration files, you should first make a backup copy of the existing files. After you make a backup copy of the files, you can use the following examples to manually update the files with the configuration information for the agent.

**Note:** If the `adclient.autoedit` parameter is not defined in the configuration file, its default value is `true`.

## Enabling automatic editing for specific files

If you set the `adclient.autoedit` parameter to `true`, you can use the following parameters to identify the specific files to be automatically edited in different operating environments:

Use this parameter	To do this
<code>adclient.autoedit.nss</code>	<p>Specify whether you want to automatically edit the Name Service Switch configuration (<code>nsswitch.conf</code>) file on <b>HP-UX</b>, <b>Solaris</b>, and <b>Linux</b> computers. For example:</p> <pre>adclient.autoedit.nss: true</pre> <p>You can also use group policy to set this parameter.</p>
<code>adclient.autoedit.pam</code>	<p>Specify whether you want to automatically edit the PAM configuration (<code>pam.conf</code> file or <code>pam.d</code> directory) on <b>AIX</b>, <b>HP-UX</b>, <b>Solaris</b>, <b>Mac OS X</b>, and <b>Linux</b> computers. For example:</p> <pre>adclient.autoedit.pam: true</pre> <p>You can also use group policy to set this parameter.</p>
<code>adclient.autoedit.centrifypam</code>	<p>Specify whether to activate the Centrify authorization plug-in and add it to the authorization mechanism every time <code>adclient</code> starts. The default value is <code>true</code>. For example:</p> <pre>adclient.autoedit.centrifypam: true</pre>
<code>adclient.autoedit.centrifypam.restart.securityagent</code>	<p>Specify whether to restart SecurityAgent after the authorization database is edited. The default value is <code>true</code>. For example:</p> <pre>adclient.autoedit.centrifypam.restart.securityagent: true</pre> <p>If this parameter is set to <code>false</code>, you must</p>



Use this parameter	To do this
	<p>restart the SecurityAgent process or reboot the computer manually after the authorization database is edited. If you do not restart SecurityAgent or reboot, users might not be able to log in.</p>
<code>adclient.autoedit.nscd</code>	<p>Specify whether you want to disable automatic editing of the nscd passwd and group cache (<code>nscd.conf</code>) on <b>Solaris</b> and <b>Linux</b> computers.</p> <p>By default, this parameter is set to <code>false</code>, which means automatic editing is disabled. Setting this parameter to <code>true</code> enables automatic editing.</p> <p>It is recommended that you change the default setting and enable automatic editing of the nscd caches; doing so reduces NSS response time substantially for large volumes of repeated queries.</p> <p><b>Note:</b> Some operating systems do not install nscd by default; be sure that nscd is installed before configuring this setting.</p> <p>For example:</p> <pre>adclient.autoedit.nscd: false</pre> <p>You can also use group policy to set this parameter.</p>
<code>adclient.autoedit.methods</code>	<p>Specify whether you want to automatically edit the Loadable Authentication Module (LAM) <code>methods.cfg</code> configuration file on <b>AIX</b> computers. For example:</p> <pre>adclient.autoedit.methods: true</pre> <p>You can also use group policy to set this parameter.</p>



Use this parameter	To do this
<code>adclient.autoedit.user</code>	<p>Specify whether you want to automatically edit the <code>/etc/security/user</code> file. The default value is <code>true</code>. For example:</p> <pre>adclient.autoedit.user: true</pre> <p>You can also use group policy to set this parameter.</p>
<code>adclient.autoedit.user.root</code>	<p>Specify whether root login is controlled by the Centrify authentication mechanism. If this parameter is set to <code>true</code>, the root stanza <code>'SYSTEM = "compat"'</code> in <code>/etc/security/user</code> will be commented out and root login must go through the Centrify authentication mechanism.</p> <p>The default value is <code>false</code> (so that by default, root login does not go through the Centrify authentication mechanism). For example:</p> <pre>adclient.autoedit.user.root: false</pre>
<code>adclient.autoedit.pwgrd</code>	<p>Specify whether you want to automatically edit the password and group hashing and caching daemon (<code>pwgrd</code>) on <b>HP-UX</b> computers. For example:</p> <pre>adclient.autoedit.pwgrd: true</pre> <p>You can also use group policy to set this parameter.</p>

Note that if you make any changes to any `adclient.autoedit.*` parameter, you must restart the `adclient` process for the change to take effect. Restarting `adclient` is required whether you set the parameters manually in the configuration file or by enabling a group policy.

## Editing the NSS configuration manually

To manually edit the NSS configuration, modify the `/etc/nsswitch.conf` file to include `centrifydc` as the first entry for the password and group lines as appropriate for your environment. For example:

```
passwd: centrifydc files
shadow: centrifydc files
group: centrifydc files
```



By placing `centrifydc` at the beginning of each line, you ensure that Active Directory authentication takes precedence over other forms of authentication.

## Editing the PAM configuration manually

In most cases, you should not manually edit the PAM configuration on a computer unless absolutely necessary because changes can produce unexpected and undesirable results. If you choose to edit the file manually, you should use caution and limit the changes you make.

To manually edit the PAM configuration to use Centrify and Active Directory, you need to add several lines to the top of the appropriate PAM configuration file for the local operating environment.

For example, on Linux you need to add the following lines to the top of the `/etc/pam.d/system-auth` file:

```
auth    sufficient pam_centrifydc.so debug
auth    requisite  pam_centrifydc.so deny debug
account sufficient pam_centrifydc.so debug
session sufficient pam_centrifydc.so homedir
password sufficient pam_centrifydc.so try_first_pass
password requisite pam_centrifydc.so deny
```

On Solaris and other platforms, you need to add the following lines to the top of the `/etc/pam.conf` file:

```
rlogin auth    sufficient pam_centrifydc.so debug
rlogin auth    requisite  pam_centrifydc.so deny debug
login  auth    sufficient pam_centrifydc.so debug
login  auth    requisite  pam_centrifydc.so deny debug
passwd auth    sufficient pam_centrifydc.so try_first_pass
debug
passwd auth    requisite  pam_centrifydc.so deny debug
other  auth    sufficient pam_centrifydc.so debug
other  auth    requisite  pam_centrifydc.so deny debug
cron   account  sufficient pam_centrifydc.so debug
other  account  sufficient pam_centrifydc.so debug
other  password sufficient pam_centrifydc.so debug
other  session  sufficient pam_centrifydc.so debug
```

**Note:** In most operating environments, when new users log on successfully, the Centrify agent automatically attempts to create the user's home directory. In Solaris environments, however, the home directory is often automounted over NFS, so the attempt to automatically create a new home directory for new users typically fails. If you use NFS to automount home directories, you can turn off the automatic creation of the home directory by



setting the `pam.homedir.create` parameter in the `centrifydc.conf` file to `false`. For more information about setting this parameter, see [pam.homedir.create](#).

By adding the appropriate lines to the beginning of the PAM configuration file, you ensure that Active Directory authentication takes precedence over other forms of authentication.

## Editing the LAM configuration manually

To manually edit the LAM configuration for AIX computers, you need to add Centrify-specific information to the `/usr/lib/methods.cfg` and `/etc/security/user` files.

In the `/usr/lib/methods.cfg` file, add the following lines to enable authentication through the Centrify agent and Active Directory:

```
CENTRIFYDC:  
    program = /usr/lib/security/CENTRIFYDC  
    program_64 = /usr/lib/security/CENTRIFYDC64  
    options = noprompt
```

In the `/etc/security/user` file, you need to change the `SYSTEM` attribute for your users. The easiest way to do this is to change the `SYSTEM` attribute in the “default” stanza. For example:

```
...  
SYSTEM = “CENTRIFYDC OR CENTRIFYDC[NOTFOUND] AND compat”  
...
```

In addition, if any user has an explicit setting for the `SYSTEM` attribute, you should remove the setting. For example, by default, the root account has an explicit `SYSTEM` setting, so you should delete this line or comment it out.

## adclient.binding.dc.failover.delay

This configuration parameter specifies the time, in minutes, to wait before `adclient` fail over to the next domain controller when the currently-connected domain controller is either down or not responding. The default is fail over immediately.

**Note:** This configuration parameter only takes effect when `adclient` is running. If the domain controller stored in `kset` is down when



adclient is not running, starting up adclient will force adclient to lookup a healthy domain controller.

## adclient.binding.idle.time

This configuration parameter specifies the maximum number of minutes to allow as idle time when binding to Active Directory.

For example, to allow a maximum idle time of 5 minutes during a bind operation:

```
adclient.binding.idle.time: 5
```

## adclient.binding.refresh.force

This configuration parameter specifies whether to force LDAP bindings to be refreshed even if the current binding is to a local (preferred) Active Directory site. Under some conditions, binding to a different site can help facilitate load balancing between servers. However, in environments with many machines joined to a large domain, binding to a new domain controller can cause serious performance problems because the agent must entirely rebuild the cache.

If you set this parameter to `true`, the agent will attempt to connect to another local domain controller when the period specified in `adclient.binding.refresh.interval` expires.

By default, this configuration parameter is set to `false`. For example:

```
adclient.binding.refresh.force: false
```

## adclient.binding.refresh.interval

This configuration parameter specifies how often to refresh the LDAP bindings to the preferred Active Directory site under these conditions:

- If the computer is currently bound to a local domain controller, bindings are refreshed only if `adclient.binding.refresh.force` is set to `true`.
- If the computer is currently bound to a domain controller in another site, bindings are refreshed regardless the `adclient.binding.refresh.force` setting.



If the agent is unable to communicate with a local domain controller, it automatically connects to an available domain controller in another site until a domain controller in its preferred site becomes available. To determine when a domain controller in the preferred site is available, the agent periodically attempts to re-connect to domain controllers in its preferred site whenever it is connected to a backup domain controller in another site. This parameter controls how frequently the agent performs the attempt to re-connect to the preferred site.

The parameter value specifies the number of minutes between refresh attempts. It must be an integer greater than zero. The following example sets the interval time to 60 minutes:

```
adclient.binding.refresh.interval: 60
```

If this parameter is not defined in the configuration file, its default value is 30 minutes.

In changing this parameter, you should consider your network and site topology and the reliability of your servers. If you have highly reliable network links and very good connections between sites, you may find it safe to increase this value, but if communication between sites is slow you should keep this interval short to ensure the agent communicates with domain controllers in its preferred site as soon as possible.

## adclient.get.builtin.membership

This configuration parameter determines whether the agent checks for valid users in built-in Active Directory groups, such as Administrators. By default, this parameter's value is `false`, in which case, the `adclient` process ignores members of built-in groups.

To include members of built-in groups, set this parameter to `true` in the configuration file:

```
adclient.get.builtin.membership: true
```

## adclient.cache.cleanup.interval

This configuration parameter specifies how often the agent should clean up the local cache. At each cleanup interval, the agent checks the cache for objects to be removed or expired, and at every 10th interval, the agent rebuilds local



indexes. This parameter's value should be less than the values specified for the `adclient.cache.negative.lifetime`, `adclient.cache.flush.interval`, and `adclient.cache.object.lifetime` parameters.

The default cleanup interval is 10 minutes.

For example:

```
adclient.cache.cleanup.interval: 10
```

## adclient.cache.encrypt

This configuration parameter specifies whether you want to encrypt the local cache of Active Directory data. If you set this parameter to `true`, all of the Active Directory data stored in the cache is encrypted and the cache is flushed each time the agent starts up. If you set this parameter to `false`, the cache is not encrypted and is not flushed when the agent starts up.

For example, to encrypt all data in the cache:

```
adclient.cache.encrypt: true
```

If this parameter is not defined in the configuration file, its default value is `false`.

## adclient.cache.encryption.type

This configuration parameter specifies the type of encryption to use when encrypting the local cache. The encryption type you specify must be a type supported in the Kerberos environment. For example, Windows Server 2003 Kerberos supports the following cryptographic algorithms: RC4-HMAC, DES-CBC-CRC and DES-CBC-MD5.

For example:

```
adclient.cache.encryption.type: des-cbc-md5
```

This configuration parameter is only used if `adclient.cache.encrypt` is set to `true`. If the `adclient.cache.encrypt` parameter is set to `false`, this configuration parameter is ignored.



## adclient.cache.expires

This configuration parameter specifies the number of seconds before an object in the domain controller cache expires. This parameter controls how frequently the agent checks Active Directory to see if an object in the cache has been updated.

Every object retrieved from Active Directory is stamped with the system time when it enters the domain controller cache. Once an object expires, if it is needed again, the agent contacts Active Directory to determine whether to retrieve an updated object (because the object has changed) or renew the expired object (because no changes have been made). To make this determination, the agent checks the `highestUSN` for the expired object. If the value has changed, the agent retrieves the updated object. If the `highestUSN` has not changed, the agent resets the object's timestamp to the new system time and retrieves the object from the cache.

If the agent is unable to contact Active Directory to check for updates to an expired object—for example because the computer is disconnected from the network—the agent returns the currently cached object until it can successfully contact Active Directory.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you aren't using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be a positive integer. The following example sets the cache expiration time to 600 seconds (10 minutes):

```
adclient.cache.expires: 600
```

If this parameter is not defined in the configuration file, its default value is 3600 seconds (1 hour).

**Note:** The `adclient.cache.expires` parameter defines the default cache expiration time for all objects types. You can override this default value for specific object types by appending the object type to the parameter name. For example, if you want to explicitly override the default expiration time for computer objects, you can define a different value for the `adclient.cache.expires.computer` parameter. The valid object types you can append to the parameter name to override the default value are: `computer`, `extension`, `gc`, `group`, `search`, `user`,



user.membership and zone. Note that `adclient.cache.expires.gc`, if not set, does not default to the value of `adclient.cache.expires`, but has its own default value.

## `adclient.cache.expires.computer`

This configuration parameter specifies the number of seconds before a computer object in the domain controller cache expires. If this parameter is not specified, the generic object cache expiration value is used.

Every computer object retrieved from Active Directory is stamped with the system time when it enters the domain controller cache. Once an object expires, if it is needed again, the agent contacts Active Directory to determine whether to retrieve an updated object (because the object has changed) or renew the expired object (because no changes have been made). To make this determination, the agent checks the `highestUSN` for the expired object. If the value has changed, the agent retrieves the updated object. If the `highestUSN` has not changed, the agent resets the object's timestamp to the new system time and retrieves the object from the cache.

If the agent is unable to contact Active Directory to check for updates to an expired object—for example because the computer is disconnected from the network—the agent returns the currently cached object until it can successfully contact Active Directory.

If you are manually setting this parameter, the parameter value must be a positive integer. The following example sets the cache expiration time for computer objects to 600 seconds (10 minutes):

```
adclient.cache.expires.computer: 600
```

**Note:** The default cache expiration time for all objects types is defined with the `adclient.cache.expires` parameter. If you explicitly set the `adclient.cache.expires.computer` parameter, its value overrides the default value for cached objects.

## `adclient.cache.expires.extension`

This configuration parameter specifies the number of seconds before an extension object in the domain controller cache expires. If this parameter is not specified, the generic object cache expiration value is used.



Every object retrieved from Active Directory is stamped with the system time when it enters the domain controller cache. Once an object expires, if it is needed again, the agent contacts Active Directory to determine whether to retrieve an updated object (because the object has changed) or renew the expired object (because no changes have been made). To make this determination, the agent checks the `highestUSN` for the expired object. If the value has changed, the agent retrieves the updated object. If the `highestUSN` has not changed, the agent resets the object's timestamp to the new system time and retrieves the object from the cache.

If the agent is unable to contact Active Directory to check for updates to an expired object—for example because the computer is disconnected from the network—the agent returns the currently cached object until it can successfully contact Active Directory.

If you are manually setting this parameter, the parameter value must be a positive integer. The following example sets the cache expiration time for extension objects to 1800 seconds (30 minutes):

```
adclient.cache.expires.extension: 1800
```

**Note:** The default cache expiration time for all objects types is defined with the `adclient.cache.expires` parameter. If you explicitly set the `adclient.cache.expires.extension` parameter, its value overrides the default value for cached objects.

## `adclient.cache.expires.gc`

This configuration parameter specifies the number of seconds before information in the global catalog cache expires. The global catalog cache contains the distinguished name (DN) for each object that has been looked up in Active Directory. The primary purpose of the global catalog cache is to store the results from paged object searches. Object attributes are stored in the domain controller cache.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be a positive integer. The following example sets the cache expiration time for global catalog objects to 3600 seconds (60 minutes), which is the default value:



`adclient.cache.expires.gc: 3600`

**Note:** If you do not define the `adclient.cache.expires.gc` parameter in the configuration file, it has a default value of 3600 seconds (1 hour). Unlike the default value for other object types, the default value for `adclient.cache.expires.gc` is not dependent on the value of `adclient.cache.expires`.

## `adclient.cache.expires.group`

This configuration parameter specifies the number of seconds before a group object in the domain controller cache expires. The domain controller cache contains object attributes including the object's Active Directory properties, memberships, indexes and other parameters. If this parameter is not specified, the generic object cache expiration value is used.

Every group object retrieved from Active Directory is stamped with the system time when it enters the domain controller cache. Once an object expires, if it is needed again, the agent contacts Active Directory to determine whether to retrieve an updated object (because the object has changed) or renew the expired object (because no changes have been made). To make this determination, the agent checks the `highestUSN` for the expired object. If the value has changed, the agent retrieves the updated object. If the `highestUSN` has not changed, the agent resets the object's timestamp to the new system time and retrieves the object from the cache.

If the agent is unable to contact Active Directory to check for updates to an expired object—for example because the computer is disconnected from the network—the agent returns the currently cached object until it can successfully contact Active Directory.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be a positive integer. The following example sets the cache expiration time for group objects to 1800 seconds (30 minutes):

`adclient.cache.expires.group: 1800`



**Note:** The default cache expiration time for all objects types is defined with the `adclient.cache.expires` parameter. If you explicitly set the `adclient.cache.expires.group` parameter, its value overrides the default value for cached objects.

## `adclient.cache.expires.group.membership`

This configuration parameter specifies the number of seconds before a group membership object in the domain controller cache expires. The domain controller cache contains object attributes including the object's Active Directory properties, memberships, indexes and other parameters. If this parameter is not specified, the generic object cache expiration value is used.

Every group membership object retrieved from Active Directory is stamped with the system time when it enters the domain controller cache. Once an object expires, if it is needed again, the agent contacts Active Directory to determine whether to retrieve an updated object (because the object has changed) or renew the expired object (because no changes have been made). To make this determination, the agent checks the `highestUSN` for the expired object. If the value has changed, the agent retrieves the updated object. If the `highestUSN` has not changed, the agent resets the object's timestamp to the new system time and retrieves the object from the cache.

If the agent is unable to contact Active Directory to check for updates to an expired object—for example because the computer is disconnected from the network—the agent returns the currently cached object until it can successfully contact Active Directory.

If you are manually setting this parameter, the parameter value must be a positive integer. The following example sets the cache expiration time for group objects to 1800 seconds (30 minutes):

```
adclient.cache.expires.group: 1800
```

**Note:** The default cache expiration time for all objects types is defined with the `adclient.cache.expires` parameter. If you explicitly set the `adclient.cache.expires.group.membership` parameter, its value overrides the default value for cached objects.



## adclient.cache.expires.search

This configuration parameter specifies the number of seconds before the results of an Active Directory search expire.

Search expiration is handled separately from object expiration because a search result may include objects that have been deleted or be missing objects that have been added that meet the search criteria.

You can set this configuration parameter by manually adding it to the `centrifdc.conf` configuration file and specifying the maximum number of seconds for a search result to be kept in the local cache.

If you are manually setting this parameter, the parameter value must be a positive integer. The following example sets the cache expiration time for search to 1800 seconds (30 minutes):

```
adclient.cache.expires.search: 1800
```

**Note:** The default cache expiration time for all objects types is defined with the `adclient.cache.expires` parameter. If you explicitly set the `adclient.cache.expires.search` parameter, its value overrides the default value for cached objects.

## adclient.cache.expires.user

This configuration parameter specifies the number of seconds before a user object in the domain controller cache expires. If this parameter is not specified, the generic object cache expiration value is used.

Every user object retrieved from Active Directory is stamped with the system time when it enters the domain controller cache. Once an object expires, if it is needed again, the agent contacts Active Directory to determine whether to retrieve an updated object (because the object has changed) or renew the expired object (because no changes have been made). To make this determination, the agent checks the `highestUSN` for the expired object. If the value has changed, the agent retrieves the updated object. If the `highestUSN` has not changed, the agent resets the object's timestamp to the new system time and retrieves the object from the cache.

If the agent is unable to contact Active Directory to check for updates to an expired object—for example because the computer is disconnected from the



network—the agent returns the currently cached object until it can successfully contact Active Directory.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be a positive integer. The following example sets the cache expiration time for user objects to 1800 seconds (30 minutes):

```
adclient.cache.expires.user: 1800
```

**Note:** The default cache expiration time for all objects types is defined with the `adclient.cache.expires` parameter. If you explicitly set the `adclient.cache.expires.user` parameter, its value overrides the default value for cached objects.

## `adclient.cache.expires.user.membership`

This configuration parameter specifies the number of seconds before a user's group membership information in the domain controller cache expires. If this parameter is not specified, the user object cache expiration value (`adclient.cache.expires.user`) is used.

Every user object retrieved from Active Directory is stamped with the system time when it enters the domain controller cache. Once an object expires, if it is needed again, the agent contacts Active Directory to determine whether to retrieve an updated object (because the object has changed) or renew the expired object (because no changes have been made). To make this determination, the agent checks the `highestUSN` for the expired object. If the value has changed, the agent retrieves the updated object. If the `highestUSN` has not changed, the agent resets the object's timestamp to the new system time and retrieves the object from the cache.

If the agent is unable to contact Active Directory to check for updates to an expired object—for example because the computer is disconnected from the network—the agent returns the currently cached object until it can successfully contact Active Directory.

If you are manually setting this parameter, the parameter value must be a positive integer. The following example sets the cache expiration time for user objects to 1800 seconds (30 minutes):

```
adclient.cache.expires.user.membership: 1800
```



**Note:** The default cache expiration time for all objects types is defined with the `adclient.cache.expires` parameter. If you explicitly set the `adclient.cache.expires.user.membership` parameter, its value overrides the default value for cache objects.

## `adclient.cache.flush.interval`

This configuration parameter specifies how frequently, in hours, to flush all objects from the domain controller cache. The domain controller cache contains object attributes including the object's Active Directory properties, memberships, indexes and other parameters. The parameter value must be a positive integer. Unlike the other cache management parameters, which flush objects selectively, this parameter removes all objects in the cache at the interval you specify.

Specify the interval, in hours, using an integer value. The default value is 0, which disables the complete flushing of the cache.

For example, the following setting flushes all values in the cache every 12 hours:

```
adclient.cache.flush.interval: 12
```

## `adclient.cache.negative.lifetime`

This configuration parameter specifies how long, in minutes, a negative object should remain in the domain controller cache. The domain controller cache contains object attributes including the object's Active Directory properties, memberships, indexes and other parameters. A negative object is returned when an object is not found in a search result. This configuration parameter determines how long that negative result should remain in the cache, regardless of the object type or object expiration time. By storing this negative result in the cache, the agent does not need to connect to Active Directory to look for an object that was previously not found.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value should be a positive integer. The default period of time for keeping negative results is 5



minutes. Setting the parameter value to 0 keeps negative objects in the cache indefinitely.

The following example sets the lifetime for negative objects to 10 minutes:

```
adclient.cache.negative.lifetime: 10
```

## adclient.cache.object.lifetime

This configuration parameter specifies how long, in hours, an Active Directory object should remain in the domain controller cache. Setting the parameter value to 0 keeps objects in the cache indefinitely. When you set this parameter to 0, objects remain in the cache until they are deleted from Active Directory or the cache is manually flushed with the `adflush` command. If you don't want objects to remain in the cache indefinitely, you can use this parameter to set the maximum amount of time an object should be available in the cache.

For example, if you want to set the maximum time for an object to be held in the cache to 12 hours, you can set this configuration parameters as follows:

```
adclient.cache.object.lifetime: 12
```

With this setting, object values can be retrieved from the local domain controller cache for 12 hours. At the end of the 12 hour period, however, the object is removed from the local cache and must be retrieved from Active Directory if it is needed again.

If this parameter is not defined in the configuration file, its default value is 0.

## adclient.cache.refresh

This configuration parameter specifies the maximum number of seconds an object can be read from the domain controller cache before it needs to be refreshed. This parameter allows an object to be read from the cache if the age of the object in the cache is less than the parameter value.

This parameter is useful in cases where reading objects from Active Directory may result in duplicate object requests. For example, the PAM-enabled login process is designed to always retrieve the user object from Active Directory first to ensure that the most recent version of the user object is available for logging on. It only retrieves the user object from the cache if Active Directory is



unavailable. Logging on, however, may require this same information to be requested from Active Directory more than once.

To prevent sending the duplicate object requests during the login process, the Centrify agent checks this parameter. If the age of the object in the cache is less than the refresh time specified by this configuration parameter, the object is allowed to be read from cache. If the object in the cache is older than the refresh interval, the login process retrieves the information from Active Directory.

The parameter value must be a positive integer. The default value is 5 seconds. For example:

```
adclient.cache.refresh: 5
```

**Note:** This configuration parameter applies to generic objects in the domain controller cache and becomes the default refresh period for all object types. You can set separate refresh periods for specific objects types using the object-specific configuration parameters. For example, you can set different refresh times for computer objects and user objects using the `adclient.cache.refresh.computer`, and `adclient.cache.refresh.user` configuration parameters. This generic object refresh setting applies to any object for which you do not set an object-specific refresh period.

## adclient.cache.refresh.computer

This configuration parameter specifies the maximum number of seconds a computer object can be read from the domain controller cache before it needs to be refreshed. This parameter allows a computer object to be read from the cache if the age of the object in the cache is less than the parameter value.

This parameter is useful in cases where reading objects from Active Directory may result in duplicate object requests. For example, the PAM-enabled login process is designed to always retrieve the user object from Active Directory first to ensure that the most recent version of the user object is available for logging on. It only retrieves the user object from the cache if Active Directory is unavailable. Logging on, however, may require this same information to be requested from Active Directory more than once.

To prevent sending the duplicate object requests during the login process, the Centrify agent checks this parameter. If the age of the object in the cache is less



than the refresh time specified by this configuration parameter, the object is allowed to be read from cache. If the object in the cache is older than the refresh interval, the login process retrieves the information from Active Directory.

The parameter value must be a positive integer. The default value is 5 seconds. For example:

```
adclient.cache.refresh.computer: 5
```

**Note:** The default refresh time for all objects types is defined with the `adclient.cache.refresh` parameter. If you set the `adclient.cache.refresh.computer` parameter, its value overrides the default value for objects.

## adclient.cache.refresh.extension

This configuration parameter specifies the maximum number of seconds an extension object can be read from the domain controller cache before it needs to be refreshed. The domain controller cache contains object attributes including the object's Active Directory properties, memberships, indexes and other parameters. This parameter allows an extension object to be read from the cache if the age of the object in the cache is less than the parameter value.

This parameter is useful in cases where reading objects from Active Directory may result in duplicate object requests. For example, the PAM-enabled login process is designed to always retrieve the user object from Active Directory first to ensure that the most recent version of the user object is available for logging on. It only retrieves the user object from the cache if Active Directory is unavailable. Logging on, however, may require this same information to be requested from Active Directory more than once.

To prevent sending the duplicate object requests during the login process, the Centrify agent checks this parameter. If the age of the object in the cache is less than the refresh time specified by this configuration parameter, the object is allowed to be read from cache. If the object in the cache is older than the refresh interval, the login process retrieves the information from Active Directory.

The parameter value must be a positive integer. The default value is 5 seconds. For example:

```
adclient.cache.refresh.extension: 5
```



**Note:** The default refresh time for all objects types is defined with the `adclient.cache.refresh` parameter. If you set the `adclient.cache.refresh.extension` parameter, its value overrides the default value for objects.

## `adclient.cache.refresh.gc`

This configuration parameter specifies the maximum number of seconds an entry can be read from the global catalog cache before it needs to be refreshed. This parameter allows an object to be read from the cache if the age of the object in the cache is less than the parameter value.

This parameter is useful in cases where reading objects from Active Directory may result in duplicate object requests. For example, the PAM-enabled login process is designed to always retrieve the user object from Active Directory first to ensure that the most recent version of the user object is available for logging on. It only retrieves the user object from the cache if Active Directory is unavailable. Logging on, however, may require this same information to be requested from Active Directory more than once.

To prevent sending the duplicate object requests during the login process, the Centrify agent checks this parameter. If the age of the object in the cache is less than the refresh time specified by this configuration parameter, the object is allowed to be read from cache. If the object in the cache is older than the refresh interval, the login process retrieves the information from Active Directory.

The parameter value must be a positive integer. The default value is 5 seconds. For example:

```
adclient.cache.refresh.gc: 5
```

**Note:** The default refresh time for all objects types is defined with the `adclient.cache.refresh` parameter. If you set the `adclient.cache.refresh.gc` parameter, its value overrides the default value for refreshing objects.

## `adclient.cache.refresh.group`

This configuration parameter specifies the maximum number of seconds a group object can be read from the domain controller cache before it needs to



be refreshed. This parameter allows a group object to be read from the cache if the age of the object in the cache is less than the parameter value.

This parameter is useful in cases where reading objects from Active Directory may result in duplicate object requests. For example, the PAM-enabled login process is designed to always retrieve the user object from Active Directory first to ensure that the most recent version of the user object is available for logging on. It only retrieves the user object from the cache if Active Directory is unavailable. Logging on, however, may require this same information to be requested from Active Directory more than once.

To prevent sending the duplicate object requests during the login process, the Centrify agent checks this parameter. If the age of the object in the cache is less than the refresh time specified by this configuration parameter, the object is allowed to be read from cache. If the object in the cache is older than the refresh interval, the login process retrieves the information from Active Directory.

The parameter value must be a positive integer. The default value is 5 seconds. For example:

```
adclient.cache.refresh.group: 5
```

**Note:** The default refresh time for all objects types is defined with the `adclient.cache.refresh` parameter. If you set the `adclient.cache.refresh.group` parameter, its value overrides the default value for refreshing objects.

## adclient.cache.refresh.search

This configuration parameter specifies the maximum number of seconds search results can be read from the domain controller cache before it needs to be refreshed. This parameter allows the search results to be read from the cache if the age of the object in the cache is less than the parameter value.

This parameter is useful in cases where reading objects from Active Directory may result in duplicate object requests. For example, the PAM-enabled login process is designed to always retrieve the user object from Active Directory first to ensure that the most recent version of the user object is available for logging on. It only retrieves the user object from the cache if Active Directory is unavailable. Logging on, however, may require this same information to be requested from Active Directory more than once.



To prevent sending the duplicate object requests during the login process, the Centrify agent checks this parameter. If the age of the object in the cache is less than the refresh time specified by this configuration parameter, the object is allowed to be read from cache. If the object in the cache is older than the refresh interval, the login process retrieves the information from Active Directory.

The parameter value must be a positive integer. The default value is 5 seconds. For example:

```
adclient.cache.refresh.search: 5
```

**Note:** The default refresh time for all objects types is defined with the `adclient.cache.refresh` parameter. If you set the `adclient.cache.refresh.search` parameter, its value overrides the default value for refreshing objects.

## adclient.cache.refresh.user

This configuration parameter specifies the maximum number of seconds a user object can be read from the domain controller cache before it needs to be refreshed. This parameter allows a user object to be read from the cache if the age of the object in the cache is less than the parameter value.

This parameter is useful in cases where reading objects from Active Directory may result in duplicate object requests. For example, the PAM-enabled login process is designed to always retrieve the user object from Active Directory first to ensure that the most recent version of the user object is available for logging on. It only retrieves the user object from the cache if Active Directory is unavailable. Logging on, however, may require this same information to be requested from Active Directory more than once.

To prevent sending the duplicate object requests during the login process, the Centrify agent checks this parameter. If the age of the object in the cache is less than the refresh time specified by this configuration parameter, the object is allowed to be read from cache. If the object in the cache is older than the refresh interval, the login process retrieves the information from Active Directory.

The parameter value must be a positive integer. The default value is 5 seconds. For example:

```
adclient.cache.refresh.user: 5
```



**Note:** The default refresh time for all objects types is defined with the `adclient.cache.refresh` parameter. If you set the `adclient.cache.refresh.user` parameter, its value overrides the default value for refreshing objects.

## `adclient.cache.upn.index`

This configuration parameter specifies whether to index user principle names (UPNs) that are stored in the Centrify user cache. You can use this parameter to differentiate between two users when the UPN of one user is equal to the SAM@domain\_name of another user, and both user objects are stored in the user cache.

To enable UPN indexing, set this parameter to `true`. For example:

```
adclient.cache.upn.index: true
```

By default, this parameter is set to `false`, and UPNs are not indexed.

## `adclient.client.idle.timeout`

This configuration parameter specifies the number of seconds before the agent will drop a socket connection to an inactive client.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be an integer greater than zero. The following example sets the inactive client timeout to 900 seconds:

```
adclient.client.idle.timeout: 900
```

If you set this parameter to zero, the agent will never drop the socket connection. Therefore, you should always specify a value greater than zero.

If this parameter is not defined in the configuration file, its default value is 5 seconds.

**Note:** You must restart `adclient` for changes to this parameter to take effect. There is a Group Policy setting for this property but



changing it has no effect until `adclient` is restarted on the affected computers. (Ref: CS-18792c)

## `adclient.clients.listen.backlog`

This configuration parameter specifies the number of backlog connections to maintain when all threads are busy. Through operating system services, the agent maintains a queue of pending connection requests that are received from the processes that need the services of the agent. This configuration parameter controls the maximum number of pending requests to hold in the queue.

Decreasing the value of this parameter may prevent processes from performing tasks that require `adclient` services, for example, a login request may be unable to authenticate a user. Increasing the value of this parameter may reduce the chance of service request failure, but may waste system memory resources and impact system performance.

For example:

```
adclient.clients.listen.backlog: 50
```

If you change this parameter, you must restart the `adclient` process for the change take effect.

## `adclient.clients.socket`

This configuration parameter specifies the named socket through which clients communicate with the agent.

The parameter value must be the name of the socket. For example:

```
adclient.clients.socket: /var/centrifydc/daemon
```

If this parameter is not defined in the configuration file, its default value is `daemon`.

## `adclient.clients.threads`

This configuration parameter specifies the number of threads the agent pre-allocates for processing client requests.



The parameter value must be an integer zero or greater. If you set this parameter to zero, the agent processes requests sequentially. For example:

```
adclient.clients.threads: 4
```

If this parameter is not defined in the configuration file, its default value is 4 threads.

If you change this parameter, you must restart the `adclient` process for the change take effect.

## adclient.clients.threads.max

This configuration parameter specifies the maximum number of threads the agent will allocate for processing client requests. This parameter value should be greater than or equal to the number of pre-allocated threads specified by the `adclient.clients.threads` parameter. The default value is 20 threads. For example:

```
adclient.clients.threads.max: 20
```

If you change this parameter, you must restart the `adclient` process for the change take effect.

## adclient.clients.threads.poll

This configuration parameter specifies the number of milliseconds the agent waits between checks to see if a client's request has been completed.

Request completion polling is necessary to eliminate race conditions in operating environments such as Linux that don't have `pselect` implemented in the kernel. This polling mechanism should be disabled if the operating system has an atomic `pselect`.

The parameter value must be an integer zero or greater. A value of zero turns off polling. For example:

```
adclient.clients.threads.poll: 100
```

If this parameter is not defined in the configuration file, its default value is 100 milliseconds.



## adclient.cloud.auth.token.max

This configuration parameter specifies the maximum number of cloud authentication requests that can be processed simultaneously. The default is 10 requests.

If you change this parameter, you must restart the `adclient` process. When the number of simultaneous connection requests exceeds this setting, the next authentication request will fail. If an authentication request times out waiting for a response, the connection is closed and the token is cleared to allow a new request.

The default value of this parameter is 10 simultaneous requests. For example:

```
adclient.cloud.auth.token.max: 10
```

## adclient.cloud.cert.store

Use this configuration parameter to specify the file in which to store the certificate that verifies cloud server connections.

By default, if there is no set file location, `adclient` will automatically locate the certificate.

**Note:** The agent searches the following locations by default:

- `/etc/ssl/certs/ca-certificates.crt`
- `/etc/pki/tls/certs/ca-bundle.crt`
- `/user/share/ssl/certs/ca-bundle.crt`
- `/usr/local/share/certs/ca-root-nss.crt`
- `/etc/ssl/cert.pem`

This configuration parameter should only be used if `adclient.cloud.skip.cert.verification` is set to `false` (which is the default).

For example:

```
adclient.cloud.cert.store: /etc/ssl/ca-bundle.crt
```



## adclient.cloud.connector

This configuration parameter specifies a cloud connector to use in the current Active Directory forest. This parameter enables you to explicitly designate the cloud connector to use for connections between Centrify-managed Linux and UNIX computers and the cloud instance providing cloud authentication services.

By default, `adclient` will automatically select the most appropriate cloud connector to use based on network topology. You can use this parameter, however, to manually specify the IP address or fully-qualified domain name of the cloud connector you want connections to go through. By designating a cloud connector, `adclient` will use that cloud connector and won't do automatic discovery to other connectors.

For example, to specify the cloud connector by IP address:

```
adclient.cloud.connector: 192.168.1.61:8080
```

To specify the cloud connector using the fully-qualified domain name:

```
adclient.cloud.connector: connector.mydomain.com:8080
```

Note that port 8080 is the default port for cloud connectors to use.

## adclient.cloud.connector.refresh.interval

This configuration parameter specifies how frequently `adclient` contacts its cloud connector. The refresh task is a background process that searches for and selects the nearest available cloud connector to use for connectivity between the Active Directory forest and the cloud service. By default, the process runs every 8 hours. You can use this group policy to modify that interval.

For example, to set the cloud connector refresh interval to 12 hours:

```
adclient.cloud.connector.refresh.interval: 12
```

## adclient.cloud.skip.cert.verification

Use this configuration parameter to skip verification of the security certificate for cloud connections.



By default, this parameter is set to `false`, and certificate verification is required.

For example:

```
adclient.cloud.skip.cert.verification: false
```

## adclient.cloud.connector.subnet.preference.enabled

This configuration parameter specifies whether or not to enable the ability to select subnet preferences when the agent connects to a cloud connector.

By default, this option is not enabled (`false`), which means that the agent selects the cloud connector based on the closest Active Directory site.

If you enable this option, the agent selects the cloud connector located in the same subnet as the client within the current Active Directory site, then in different subnets within the current Active Directory site, and then in an Active Directory site that's different than the current one.

## adclient.custom.attributes

This configuration parameter enables you to add Active Directory attributes to the Centrify authentication cache that are not retrieved by default. You can specify computer, user, or group attributes by using the appropriate form of the parameter:

```
adclient.custom.attributes.computer:attributeName  
adclient.custom.attributes.user:attributeName  
adclient.custom.attributes.group:attributeName
```

Separate multiple attributes by a space. For example, to specify the user attributes `comment` and `company` and the group attributes `info` and `telephoneNumber`:

```
adclient.custom.attributes.user: comment company  
adclient.custom.attributes.group: info telephoneNumber
```

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.



**Note:** You can use the adquery `--dump` command to see a list of the attributes that `adclient` caches for users or groups.

## adclient.disk.check.free

This configuration parameter specifies the size, in KB, of disk space available for the local cache that should generate a warning message. The agent will check the availability of free disk space at the interval specified with the `adclient.disk.check.interval` parameter. If the disk space available at any interval is less than the value you set for the `adclient.disk.check.free` parameter, the agent stops saving data in the local cache and displays a warning message to indicate that you should free up disk space. At the next interval when the available disk space exceeds the size you set for this parameter, the agent resumes normal operation and saving data to its cache.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

The parameter value must be an integer of zero or greater. A value of zero disables the display of a warning message about the available disk space. The default minimum of available disk space that triggers a warning message is 51200 KB. For example:

```
adclient.disk.check.free: 51200
```

**Note:** Keep in mind that the value you set for this parameter can affect the recovery of a system. The agent will only resume saving data in its local cache if there is more disk space available than what you have specified to generate the warning. For example, if you have specified that the agent issue a warning when the available disk space falls to 51200 KB, there must be more than 51200 KB of disk space available for the agent to return to normal operation and write to the cache.

## adclient.disk.check.interval

This configuration parameter specifies how frequently the agent should check the disk space available for the local cache. The default interval checks the available disk space every 5 minutes. If the disk space available at any interval is less than the value you set for the `adclient.disk.check.free` parameter,



the agent will stop saving data in the local cache and will discard any new data until you free up enough disk space for it to resume saving data in the local cache.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

The parameter value must be an integer zero or greater. A value of zero disables checking for available disk space. For example:

```
adclient.disk.check.interval: 5
```

**Note:** Keep in mind that the value you set for this parameter can affect the recovery of a system after the agent stops writing data to the local cache. If you set this parameter to 0, the agent will not check for available disk space so it will not return to normal operation when disk space is freed up. In addition, setting this parameter to 0 or to a long interval may cause the agent to consume too much of the disk for its local cache and make the computer unstable or unusable. Therefore, you should keep the interval for checking the available disk space relatively short. Keeping the interval short will also help to ensure that the agent resumes normal operation and saving data to its cache at the earliest opportunity.

## adclient.dns.cache.timeout

**Note:** This parameter has been deprecated in favor of [dns.cache.timeout](#).

This configuration parameter specifies the amount of time, in seconds, before a cached DNS response expires.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value should be a positive integer. For example, the default values expires a cached DNS response after 300 seconds:

```
adclient.dns.cache.timeout: 300
```



## adclient.dns.cachingserver

Cache-only DNS servers cannot provide sufficient authoritative responses to DNS requests directly. They refer to authoritative servers, such as a Windows server and then relay the answer to the DNS request. This means, for some cache-only DNS servers, DNS requests, sent to cache-only DNS server, need to have recursive flag. For example dnscache. Other cache-only DNS servers do not require setting the recursive flag. See your DNS server specifications.

Examples of cache-only DNS servers, include:

- dnsmasq
- dnscache
- tinyDNS
- pdnsd
- unbound
- dnrd

The `adclient.dns.cachingserver` configuration parameter determines whether to send recursive DNS requests or not. When set to `true`, this parameter sends recursive DNS requests, as apposed to the standard non-recursive requests. Default is `false`.

To use a cache-only DNS server, in the `centrifdc.conf` file, set in the `adclient.dns.cachingserver` parameter to `true`. There might be some DNS functionality loss in `adclient`, when this parameter is set to `true`.

Parameter syntax:

```
adclient.dns.cachingserver: false
```

The default setting is `false`.

When set to `true`, recursive DNS requests are allowed.

Optionally, the `install.sh` script also provides an option for handling cache-only DNS servers with `adcheck`.

```
install.sh [--dns_cache]
```

This invokes `adcheck` with option `-r` and allows DNS recursion with cache-only DNS servers.



## adclient.dumpcore

This configuration parameter specifies whether the agent should be allowed to dump core.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

The value you set for this parameter overrides the default `u l i m i t` setting. The parameter value must be one of the following valid options:

- `never` to specify that the agent never dump core.
- `once` to specify that the agent should dump core only when there is no existing core dump file.
- `always` to specify that the agent dump core on every crash.

For example:

```
adclient.dumpcore: never
```

## adclient.dynamic.dns.command

This configuration parameter specifies the parameters to use for the `addns` command if it is launched by `adclient` (see [adclient.dynamic.dns.enabled](#)).

For example, the default setting is:

```
adclient.dynamic.dns.command: /usr/sbin/addns -U -m
```

The `-u` option creates or updates the IP address and domain name pointer (PTR) records in the DNS server for the local computer.

The `-m` option uses the local computer account's Active Directory credentials to establish a security context with the DNS server.

**Note:** UNIX computers that act as a gateway between networks may require you to specify the network adapter IP address in the `addns` command line. To ensure that you register the correct network address with the Active Directory DNS server, set `adclient.dynamic.dns.command` with a command line that uses the correct IP address for the network interface you want to use. (Ref: CS-20319c)



## adclient.dynamic.dns.enabled

This configuration parameter specifies whether `adclient` will automatically launch the `addns` command. The `addns` command dynamically updates DNS records on an Active Directory-based DNS server in environments where the DHCP server cannot update DNS records automatically.

**Note:** In most cases, you do not need to use the `addns` command if a host's IP address is managed by a Windows-based DNS server and the host obtains its IP address from a Windows-based DHCP server because the DHCP server updates the DNS record for the host automatically. If you are not using a Windows-based DNS server, you should use `nsupdate` or a similar command appropriate to the operating environment of the DNS server to update DNS records.

The `addns` command is launched with the parameters specified by the `adclient.dynamic.dns.command` configuration parameter.

The default value for Mac OS X computers is `True`.

The default value for all other platforms is `False`.

## adclient.dynamic.dns.refresh.interval

This configuration parameter specifies whether or not dynamic DNS records are periodically updated for this host and, if there are updates, the interval between updates. The parameter takes an integer of 0 or greater. If set to 0, it turns the DNS update feature off. If set to 1 or greater, it specifies the number of seconds between DNS update attempts.

The default value for this parameter is 0 (off).

## adclient.excluded.domains

This configuration parameter specifies a list of domains to exclude from the list of trusted domains.



For example, you might want to exclude specific domains that are contained within a trusted forest. To specify domains to exclude, enter one or more domain names in dotted-name format, separated by spaces. For example:

```
adclient.excluded.domains: eng.acme.com qa.acme.com
```

The Centrify agent does not probe any excluded domains and consequently ignores users from these domains.

The default value for this parameter is the empty list, which does not exclude any domains.

## adclient.exit.on.incomplete.zone.hierarchy

This configuration parameter specifies whether or not the agent stops if it can't load the complete zone hierarchy.

Adclient loads the zone hierarchy during startup only. There might be some situations where not all zones can be loaded: for example, if this is the first time loading zones since running `adjoin`. Normally, adclient will run with an incomplete or incorrect zone hierarchy.

The parameter value must be a boolean value of `true` or `false`. For example:

```
adclient.exit.on.incomplete.zone.hierarchy: true
```

If this parameter is not defined in the configuration file, its default value is `false`.

## adclient.fetch.object.count

This configuration parameter specifies the number of objects to obtain in a single LDAP request. You can use this parameter to optimize the number of objects to suit your environment.

The parameter value must be a positive integer. For example:

```
adclient.fetch.object.count: 5
```

With this parameter, there is a trade-off here between speed and memory usage as well as bandwidth and latency. As you increase the number of objects included in an LDAP request, you may improve the overall performance by decreasing the number of connections to Active Directory and reducing the overall demand on the server, but you increase the RAM used by the agent. If you decrease the number of objects included in an LDAP request, you may



reduce overall performance because of the additional network traffic, but decrease the memory used by the agent.

On faster networks, you can safely retrieve a small number of objects. On slower networks or when retrieving information for large groups (for example, groups with more than 1000 users), you may want to increase the value for this parameter.

## adclient.force.salt.lookup

This configuration parameter specifies that you want to force the Centrify agent to look up the complete principal name, including the Kerberos realm used as the key salt, from the KDC. Setting this parameter to `true` is required if you remove `arcfour-hmac-md5` from the list of encryption types specified for the `adclient.krb5.tkt.encryption.types` parameter and if you change a `userPrincipalName` attribute in Active Directory without changing the user's password.

The parameter value can be `true` or `false`. The default value is `true`. For example:

```
adclient.force.salt.lookup: false
```

**Note:** When this parameter value is set to `true` it may cause “pre-auth required” warning messages to appear in the Active Directory event log.

## adclient.get.primarygroup.membership

This configuration parameter specifies whether zone users are added as members of a primary group.

By default, Active Directory users are not members of their primary Active Directory group. This parameter is used to control whether zone users are added as members of this primary group.

The parameter value can be `true` or `false`. If you set this parameter to `true`, zone users are added to the primary group. If you set this parameter to `false`, zone users are not added to the primary group.

Setting this parameter to `true` can have performance implications when you query the group (for example, by using the `adquery group` command) because



`adclient` has to search for all Active Directory users who have has this group as their primary group.

The default value is `false`. For example:

```
adclient.get.primarygroup.membership: false
```

## `adclient.hash.allow`

This configuration parameter specifies the list of users you want to allow to have their password hash stored. By default, the Centrify agent stores a UNIX-style MD5 hash of each user's password in the cache when the user is authenticated during login. Storing the password hash allows previously authenticated users to log on when the computer is disconnected from the network or Active Directory is unavailable.

Although the default behavior is to store the password hash for all users, you can use this parameter to explicitly list the users whose hashed passwords are stored in the cache. If you use this parameter, only the users you specify can log on when the computer is disconnected from the network or Active Directory is unavailable.

The parameter value can be one or more user names. If more than one name, the names can be separated by commas or spaces. For example:

```
adclient.hash.allow: jdoe bsmith
```

If no user names are specified or the parameter is not defined in the configuration file, the password hash is stored for all users.

## `adclient.hash.deny`

This configuration parameter specifies the list of users you want to prevent from having their password hash stored. By default, the Centrify agent stores a UNIX-style MD5 hash of each user's password in the cache when the user is authenticated during login. Storing the password hash allows previously authenticated users to log on when the computer is disconnected from the network or Active Directory is unavailable.

Although the default behavior is to store the password hash for all users, you can use this parameter to explicitly list the users whose hashed passwords must not be stored in the cache. If you use this parameter, the users you specify



cannot log on when the computer is disconnected from the network or Active Directory is unavailable. All other users are permitted to have their password hash stored and allowed to log on when the computer is disconnected from the network or Active Directory is unavailable.

The parameter value can be one or more user names. If more than one name, the names can be separated by commas or spaces. For example:

```
adclient.hash.deny: jdoe bsmith
```

If no user names are specified or the parameter is not defined in the configuration file, the password hash is stored for all users.

## adclient.hash.expires

This configuration parameter specifies the number of days the password hash for any user can be stored in the cache before it expires.

The parameter value must be a positive integer. A value of zero (0) specifies that the password hash should never expire. For example, to set this parameter so that the password hash expires after 7 days:

```
adclient.hash.expires: 7
```

If this parameter is not defined in the configuration file, its default value is 0.

## adclient.ignore.setgrpsrc

This configuration parameter specifies whether `adclient` accesses the `~/home/.setgrpsrc` file when a user issues commands such as `groups` and `id` that return information about users' group membership.

Whenever it accesses the `~/home/.setgrpsrc` file, `adclient` mounts the specified user's home directory. This behavior can be problematic because it makes it difficult for administrators to control mounts to file servers. For example, moving a file server requires removing all mounts, so before doing so, an administrator must scan for mounts that may have been created by running `groups` or `id`. In addition, the extra NFS mounts generated by this behavior reduce the number of reserved ports available on Red Hat systems and can slow system performance.

You can set this parameter's value to either `true` or `false`. When `true`, `adclient` ignores the `~/home/.setgrpsrc` file when a user issues commands



such as `groups` and `id`, and does not mount the specified user's home directory. When `false`, `adclient` does check the `~/home/.setgrpsrc` file when a user issues commands such as `groups` and `id`, and does mount the specified user's home directory.

The default value is `false`.

To change the value to `true`, such that `adclient` does not check the `~/home/.setgrpsrc` file and mount a user's home directory when checking for group membership, set this parameter's value to `true` in the configuration file:

```
adclient.ignore.setgrpsrc: true
```

After setting this parameter, you must run `adreload` to reload the configuration file.

**Note:** Setting this parameter does not affect the use of the `adsetgroups` command by the current user to view or change group membership. For a logged on user, the home directory is already mounted.

## adclient.included.domains

This configuration parameter specifies a list of domains to include as trusted domains.

If this parameter specifies any domains (that is, is not empty), only the specified domains and the joined domain will be trusted. For example, you might want to specify specific domains to trust in a trusted forest, rather than trust all domains in the forest.

**Note:** Alternately, you may use the `adclient.excluded.domains` parameter to exclude from the trusted list specific domains that are contained within a trusted forest.

To specify domains to include, enter one or more domain names in dotted-name format, separated by spaces. For example:

```
adclient.included.domains: eng.acme.com qa.acme.com
```

In this example, the only trusted domains are `eng.acme.com`, `qa.acme.com`, and the domain to which the computer is joined.

The Centrify agent does not probe any domains that are not on the list (except the joined domain) and consequently ignores users from other domains.



The default value for this parameter is the empty list, which has no effect on determining which domains to trust.

## adclient.iterate.private.groups

This configuration parameter specifies whether `adclient` iterates through users to look for private groups when searching for groups.

The `adclient` process may receive periodic requests from processes such as `adnisd` for all zone-enabled users and groups. `adclient` queries Active Directory for those users and groups. By default, `adclient` queries only for group objects when searching for groups. When dynamic private groups are turned on (using the configuration parameter `auto.schema.private.group`), it creates private groups with a single user where the primary GID of the private group is set to the user's UID. When dynamic private groups are present, `adclient` must search through user objects as well as group objects when looking for groups.

This parameter's value must be either `true` or `false`. When `true`, `adclient` iterates through user objects in Active Directory when searching for groups. When `false`, `adclient` does not iterate through user objects when searching for groups.

Note that iterating through users isn't noticeably slower than iterating only through groups until the numbers of users get into tens or hundreds of thousands. In these numbers, iteration may take more time.

If this parameter is not defined in the configuration file, its default value is initially `false`. Once `adclient` encounters a private group, it sets this parameter's value to `true` for the rest of `adclient`'s process lifetime or until a user sets this parameter in the configuration file.

## adclient.krb5.principal.lower

This configuration parameter converts the principal in Kerberos tickets to lowercase for compatibility with some third-party applications.

Set to `true` to change the principal in Kerberos tickets to lowercase.

Set to `false` to leave the case unchanged for the principal in Kerberos tickets.

The default value is `false`.



**Note:** Use this parameter when a machine is joined to classic zones or hierarchical zones. When a machine is in Auto Zone, please use `auto.schema.name.lower` instead.

## adclient.krb5.conf.domain\_realm.any\_site

This configuration parameter specifies whether or not to search for all domain controllers in a kerberized realm or just the domain controllers within the current, preferred site.

If this parameter is set to true, then the system will list all reachable domain controllers in a kerberized realm, regardless of which site they're located in.

If this parameter is set to false, then only the domain controller in the current, preferred site is listed.

For example:

```
adclient.krb5.conf.domain_realm.any_site: true
```

If this parameter is not defined in the configuration file, its default value is false.

## adclient.ldap.packet.encrypt

This configuration parameter specifies the LDAP encryption policy you use. For example, if your organization has a security policy that does not allow unencrypted LDAP traffic, you can use this parameter to specify that all connections to Active Directory are encrypted. If your organization isn't concerned with the encryption of LDAP data and you want better performance, you can force all connections to be unencrypted.

The parameter value must be one of the following valid options:

- `Allowed` to allow both encrypted and unencrypted LDAP traffic.
- `Disabled` to prevent encrypted LDAP traffic.
- `SignOnly` to require all LDAP traffic to be signed to ensure packet integrity, but not encrypted.
- `Required` to require all LDAP traffic to be signed and encrypted. If you select this setting and a server doesn't support encryption, the connection will be refused.



For example:

```
adclient.ldap.packet.encrypt: Allowed
```

If this parameter is not defined in the configuration file, its default value is Allowed.

## adclient.ldap.socket.timeout

This configuration parameter specifies the time, in seconds, the agent will wait for a socket connection timeout during LDAP binding.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be a positive integer greater than zero. For example:

```
adclient.ldap.socket.timeout: 30
```

If this parameter is not defined in the configuration file, its default value is 5 seconds.

## adclient.ldap.timeout

This configuration parameter specifies the time, in seconds, the agent will wait for a response from Active Directory before it gives up on the LDAP connection during fetch, update, or delete requests.

If a request is made to Active Directory and a response is not received within the number of seconds specified by this parameter, that request is retried once. For the second request, the agent will wait up to twice as long for a response. If the second request is not answered within that amount of time, the connection to that specific domain controller is considered disconnected. For example, if you set this parameter value to 7, the agent waits 7 seconds for a response from Active Directory to a fetch, update, or delete request. If the request isn't answered within 7 seconds, the agent retries the request once more and waits up to 14 seconds for a response before switching to disconnected mode. This results in a maximum elapsed time of 21 seconds for the agent to determine that Active Directory is unavailable.



In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be a positive integer. For example:

```
adclient.ldap.timeout: 10
```

If this parameter is not defined in the configuration file, its default value is 7 seconds.

## adclient.ldap.timeout.search

This configuration parameter specifies the time, in seconds, the agent will wait for a response from Active Directory before it gives up on the LDAP connection during search requests.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be a positive integer. For example:

```
adclient.ldap.timeout: 10
```

If this parameter is not defined in the configuration file, its default value is double the value specified for the `adclient.ldap.timeout` parameter.

## adclient.ldap.trust.enabled

This configuration parameter specifies whether you want to allow the agent to query trusted domains and forests for transitive trust information. The parameter's value can be `true` or `false`. If you set this parameter to `true`, the `adclient` process generates a `krb5.conf` that includes information from all trusted forests and can be used to authenticate cross-forest users to Kerberos applications. If you set this parameter to `false`, the agent does not query external trusted domains or forests for information.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.



If you are manually setting this parameter, the parameter value should be `true` or `false`. The default value is `true`. For example:

```
adclient.ldap.trust.enabled: true
```

**Note:** Querying external trusted forests can take a significant amount of time if the other forests are blocked by firewalls. You may want to set this parameter to `false` if your trust relationships, network topology, or firewalls are not configured properly for access.

## adclient.ldap.trust.timeout

This configuration parameter specifies the maximum number of seconds to wait for responses from external forests and trusted domains when attempting to determine trust relationships. If your trusted domains and forests are widely distributed, have slow or unreliable network connections, or are protected by firewalls, you may want to increase the value for this parameter to allow time for the agent to collect information from external domains and forests.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value should be a positive integer. For example, to time out if a response is not received in within two minutes, you can set this parameter value to 120:

```
adclient.ldap.trust.timeout: 120
```

The default value is 5 seconds. Before changing this setting, you should consider your network topology, the reliability of network connections, and the network bandwidth, speed, and latency for connecting to external forests and domains. If the value is set too low to consistently receive a response, you may be unable to search trusted external domains.

## adclient.legacyzone.mfa.background.fetch.interval

This configuration parameter specifies how often the Centrify agent updates the cache with the list of groups in classic zones and Auto Zones specified in the following parameters:

```
adclient.legacyzone.mfa.required.groups
```



This is a background process that updates the cache periodically according to the interval specified. Enabling this configuration parameter will improve multi-factor authentication performance.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy, or to temporarily override group policy.

For example, to set the parameter interval to 45 minutes:

```
adclient.legacyzone.mfa.background.fetch.interval: 45
```

To disable this process, set the parameter value to 0.

The default parameter value is 30 minutes.

## adclient.legacyzone.mfa.cloudurl

This configuration parameter specifies which cloud instance URL the agent will access in order to implement multi-factor authentication for users in classic zones and Auto Zones.

If all of the cloud connectors in the Active Directory forest use a single cloud instance URL, the agent will use this instance for multi-factor authentication by default, and you do not have to specify the URL using this parameter. If you have access to more than one cloud instance URL, you must specify the URL you would like use for multi-factor authentication for the zone using this parameter or the group policy that modifies this parameter.

If you have access to more than one cloud instance URL, but do not specify which one should be used for multi-factor authentication, you will not be able to configure the zone to use multi-factor authentication.

In most cases, you set this configuration parameter using group policy. If you are manually setting this parameter, the parameter value must be a URL in the following format:

```
https://tenantid.domainfqdn:port/
```

For example:

```
adclient.legacyzone.mfa.cloudurl:  
https://abc0123.mydomain.com:8080/
```



## adclient.legacyzone.mfa.enabled

This configuration parameter specifies whether multi-factor authentication is enabled for a classic zone or an Auto Zone. If you enable multi-factor authentication, users must be authenticated using more than one method. For example, users might be required to provide a password and respond to a text message or a phone call, or answer a security question. To enable multi-factor authentication, set this parameter to `true`. Set the parameter to `false` if multi-factor authentication is not required for any users.

In most cases, you set this configuration parameter using group policy. If you are manually setting this parameter, the parameter value must `true` or `false`. For example:

```
adclient.legacyzone.mfa.enable: true
```

If this parameter is not defined in the configuration file, its default value is `false`.

## adclient.legacyzone.mfa.required.groups

This configuration parameter specifies a list of Active Directory groups in a classic zone or an Auto Zone that are required to use multi-factor authentication when logging on or using privileged commands. For example, if you want to require all members of the Qualtrak Admin group to use multi-factor authentication when they log on to computers that host sensitive information, you can add that group to this parameter.

Groups specified in this parameter must be security groups; distribution groups are not supported.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy, or to temporarily override group policy.

By default, multi-factor authentication is not enabled for groups in classic or Auto Zones.

You can separate each group by a space or a comma and you can use double quotes or escape characters to included spaces or special characters in group names. For example:

```
adclient.legacyzone.mfa.required.groups: centrifys_users,  
"Qualtrak Admins", Domain\Users
```



## Supported group name formats

You can specify groups by name or you can list the group names in a file in the following formats:

- SAM account name: sAMAccountName
- SAM account name of a group in a different domain: sAMAccountName@domain
- canonicalName: domain/container/cn

## Specifying the parameter value in a separate file

To specify a file that contains a list of Active Directory group names, you can set the parameter value using the `file:` keyword and a file location. For example:

```
adclient.legacyzone.mfa.required.groups:  
file:/etc/centrifydc/legacy_user_groups_mfa.require
```

In the `etc/centrifydc/legacy_user_groups_mfa.require` file, you would type each group name on its own line using any of the supported name formats. For example:

```
centrify_users  
"Qualtrak Admins"  
Domain\ Users  
group4@domain.com
```

## adclient.legacyzone.mfa.required.users

This configuration parameter specifies a list of Active Directory users in a classic zone or an Auto Zone that are required to use multi-factor authentication when logging on or using privileged commands. For example, if you want to require Bill Hill to use multi-factor authentication to log on to a computer that hosts sensitive information, you can add her to this parameter.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy, or to temporarily override group policy.

By default, multi-factor authentication is not enabled for users in classic or Auto zones.



You can separate each user name by a space or a comma and you can use double quotes or escape characters to include spaces or special characters in user names.

For example, to specify that multi-factor authentication is required for users `bill hill` and `tetsu.xu` to log on to computers in an Auto Zone you would define the parameter value in the following way:

```
adclient.legacyzone.mfa.required.users: "bill.hill",  
tetsu.xu@ajax.org
```

## Supported user name formats

You can specify users by name or you can list the user names in a file in the following formats:

- SAM account name: `sAMAccountName`
- SAM account name of a user in a different domain:  
`sAMAccountName@domain`
- User Principal Name: `name@domain`
- Canonical Name: `domain/container/cn`
- Full DN: `CN=commonName, . . . ,DC=domain_component,DC=domain_`  
`component`
- An asterisk (\*), which includes all Active Directory users.

## Specifying the parameter value in a separate file

To specify a file that contains a list of Active Directory user names, you can set the parameter value using the `file:` keyword and a file location. For example:

```
adclient.legacyzone.mfa.required.users:  
file:/etc/centrifydc/legacy_user_users_mfa.require
```

In the `etc/centrifydc/legacy_user_users_mfa.require` file, you would type each user name on its own line using any of the supported name formats. For example:

```
tetsu.xu  
jane/ doe@ajax.org  
Domain Users
```



## adclient.legacyzone.mfa.rescue.users

This configuration parameter specifies a list of Active Directory users who can log on to computers in a classic zone or an Auto Zone when multi-factor authentication is required, but the agent cannot connect to the Centrify cloud service. You should specify at least one user account for this parameter to ensure that someone can access the computers in the event that multi-factor authentication is required but not available.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy, or to temporarily override group policy.

You can separate each user by a space or a comma and you can use double quotes or escape characters to include spaces or special characters in user names.

For example, to specify that user `amy adams` has the ability to log on to a computer in an Auto Zone if she is required but unable to authenticate using multi-factor authentication, you would define the parameter value in the following way:

```
adclient.legacyzone.mfa.rescue.users: amy.adams
```

### Supported user name formats

You can specify users by name or you can list the user names in a file in the following formats:

- SAM account name: `SAMAccountName`
- SAM account name of a user in a different domain:  
`SAMAccountName@domain`
- User Principal Name: `name@domain`
- Canonical Name: `domain/container/cn`
- Full DN: `CN=commonName, . . . ,DC=domain_component,DC=domain_component`
- An asterisk (\*), which includes all Active Directory users.



## Specifying the parameter value in a separate file

To specify a file that contains a list of Active Directory user names, you can set the parameter value using the `file:` keyword and a file location. For example:

```
adclient.legacyzone.mfa.rescue.users:  
file:/etc/centrifydc/legacy_user_users_mfa.rescue
```

In the `etc/centrifydc/legacy_user_users_mfa.rescue` file, you would type each user name on its own line using any of the supported name formats. For example:

```
tetsu.xu  
amyadams  
jane/ doe@ajax.org  
user1@domain.com
```

## adclient.legacyzone.mfa.tenantid

This configuration parameter specifies which Centrify Identity Platform instance ID (also known as the tenant ID) the agent will access in order to implement multi-factor authentication for users in classic zones and Auto Zones.

By default, this parameter is empty and the agent uses the `adclient.legacyzone.mfa.cloudurl` parameter to locate Centrify connectors.

For example:

```
adclient.legacyzone.mfa.tenantid: ABC1234
```

## adclient.local.account.manage

This configuration parameter specifies whether the agent manages local user and local group accounts on computers where the agent is installed.

When this parameter is set to `true`:

- The agent gets the local user and local group profiles from the zone, and updates the local `/etc/passwd` and `/etc/group` files using the information defined in the zone.
- You can view and manage local users and groups in Access Manager as described in the *System Administrator's Guide for Linux and UNIX*.



The default value of this parameter is `false`, unless you upgraded from an authentication, privilege elevation, and audit and monitoring services release in which local account management was enabled (in which case, it is set to `true`).

You can also set this configuration parameter using group policy.

## `adclient.local.account.manage.strict`

This configuration parameter applies enforcement mode for local account management. The default is `false` and it is defined as not strict.

The following are sub-parameters for this configuration parameter:

- `adclient.local.account.manage.strict.passwd`: `false`
- `adclient.local.account.manage.strict.group`: `false`

When enabled in strict mode for user (except user with UID 0) any unmanaged local user's password entry is removed from `/etc/passwd`. If `/etc/shadow` file exist, shadow entry is removed as well. If user's extended attributes exist, those are removed.

When enabled in strict mode for group (except user with GID 0), any unmanaged local group entry is removed from `/etc/group`. If group's extended attributes exist, those are removed as well.

After switching to strict enforcement of local account management, switching back to non strict local account management does not restore the unmanaged local user or group.

## `adclient.local.account.notification.cli`

This configuration parameter lets you define a command to process changes to local account profiles after the agent synchronizes local user and group profiles with profiles defined in a zone.

For example, if new local users are added, removed, or have their enabled/disabled status changed locally, the command that you define in this parameter is executed. Typical activities that this command might perform include setting the password for new or updated local accounts, or notifying password vault about changes to local accounts and defining actions to take regarding those accounts.



When this parameter is enabled, the agent invokes the defined command in another process and passes a comma separated UNIX name list to the command for further processing.

By default, this parameter's value is empty (that is, no command is defined). This parameter takes effect only when local account management is enable through group policy, or when the `adclient.local.account.manage` parameter is set to `true`.

You can also set this configuration parameter using group policy.

## adclient.local.account.notification.cli.arg.length.max

This configuration parameter specifies the maximum argument length for the command that you define in the `adclient.local.account.notification.cli` parameter.

To determine the default argument length for your environment, execute `getconf ARG_MAX`.

After determining the default argument length for your environment, you can set this parameter to the same value to ensure that the agent's setting is consistent with the environment setting.

The default value of this parameter is 131072, which is 128KB. This parameter takes effect only when the `adclient.local.account.manage` parameter is set to `true`.

## adclient.local.forest.altupn.lookup

This configuration parameter specifies whether or not to perform the local forest altupn lookup. The default is `true`. If you set this parameter to `false`, the local forest altupn lookup is skipped.

## adclient.local.group.merge

This configuration parameter determines whether to merge local group membership from the `/etc/group` file into the zone group membership for



groups that have the same name and GID. For example, if the Centrify agent retrieves the membership list of kwan, emily, and sam for the group profile with the group name performx1 and GID 92531 from Active Directory and there is also a local group named performx1 with the GID 92531 with users wilson and jae, the merged group would include all five members (kwan,emily,sam,wilson,jae).

By default, this parameter value is set to `false` to prevent unexpected results. For example:

```
adclient.local.group.merge: false
```

Setting this parameter to `true` violates normal NSS behavior and, therefore, may have unexpected side effects. You should analyze your environment carefully before changing this parameter to `true`. If you determine you can safely merge local and Active Directory group profiles, you can uncomment this parameter and change its value.

**Note:** If you set this parameter to `true`, you must run `adreload` to detect changes in the local group file.

## adclient.logonhours.local.enforcement

This configuration parameter determines whether the agent and Active Directory both check for user logon hour restrictions, or whether only Active Directory checks for logon hour restrictions. This parameter is useful in cases where users are in time zones that are different from the time zone that the agent is in.

When this parameter is set to `true`, the agent and Active Directory both check for local logon hour restrictions. If the agent and user are in different time zones, and one time zone recognizes Daylight Savings Time while the other does not, the user may not be able to log on during permissible hours. (Ref: CS-33553 a)

When this parameter is set to `false`, only Active Directory checks for local hour restrictions, so there is no Daylight Savings Time conflict with the agent.

The default value for this parameter is `true`.

You should set this parameter to `false` if you have users that are not in the same time zone as the agent.

For example:

```
adclient.logonhours.local.enforcement: false
```



## adclient.lookup.sites

This configuration parameter specifies a list of sites, and optionally a domain, to search for domain controllers and the global catalog if they are not found in the preferred site.

**Note:** You can specify the preferred site in the `adclient.preferred.site` configuration parameter, and the preferred site is displayed when you execute the `adinfo` command.

The format for this parameter is:

```
adclient.lookup.sites: site1 [site2] [site3]...
```

The agent performs the following steps whenever it attempts to connect to a DC or GC:

1. Discover the preferred site.
2. From DNS, get a list of DCs or GCs in the preferred site and attempt to connect to each one until a connection is successful or the list is exhausted.
3. If unable to connect to a DC or GC in the preferred site, try to connect to a DC or GC in any site.

By using this configuration parameter, you can restrict step 3 to a specific set of alternate sites to search for DCs or GCs. Run Active Directory Sites and Services to see a list of sites for your environment. Sites are searched in the list order that you specify.

You can use any legal Active Directory site name when you set this parameter. For example:

```
adclient.lookup.sites USTEXAS USCALIFORNIA
```

You can optionally specify a domain suffix in this parameter, so that the site list is searched only in the domain that you specify. Use the following format to specify a domain:

```
adclient.lookup.sites.domainsuffix: site1 [site2] [site3]...
```

For example:

```
adclient.lookup.sites.example.com: USTEXAS USCALIFORNIA
```



If this configuration parameter is not configured, the agent tries to connect to a DC or GC in any site, as described in step 3 above. By default, this configuration parameter is not configured.

**Note:** Do not add the preferred site to this list, as the preferred site will be searched anyway.

## adclient.lrpc2.receive.timeout

This configuration parameter specifies how long, in seconds, the agent should wait to receive data coming from a client request.

In most cases, you set this configuration parameter using group policy.

If you are manually setting this parameter, the parameter value must be a positive integer. For example:

```
adclient.lrpc2.receive.timeout: 30
```

If this parameter is not defined in the configuration file, its default value is 30 seconds.

## adclient.lrpc2.send.timeout

This configuration parameter specifies the maximum number of seconds the agent should wait for reply data to be sent in response to a client request.

In most cases, you set this configuration parameter using group policy.

If you are manually setting this parameter, the parameter value must be a positive integer. For example:

```
adclient.lrpc2.send.timeout: 30
```

If this parameter is not defined in the configuration file, its default value is 30 seconds.

## adclient.ntlm.domains

This configuration parameter allows you to manually map Active Directory domain names to NTLM domains. This parameter is useful in cases where you need to use NTLM authentication because firewalls prevent Kerberos



authentication and when firewall constraints prevent the automatic discovery of Active Directory to NTLM domain mapping.

You can specify the parameter's value as one or more domain name pairs, separated by a colon (:), or using the `file:` keyword and a file location. For example, you can set the parameter value using the format `ActiveDirectory_DomainName:NTLM_DomainName` to specify a list of domain name pairs:

```
adclient.ntlm.domains: AJAX.ORG:AJAX FIREFLY.COM:FIREFLY
```

To specify a file that contains a list of colon-separated values in the form of `ActiveDirectory_DomainName:NTLM_DomainName`, you can set the parameter value using the `file:` keyword and a file location:

```
adclient.ntlm.domains: file:/etc/centrifydc/domains.conf
```

Keep in mind that you must manually define how Active Directory domains map to NTLM domains. If you define this information in a separate file, such as `domains.conf`, the file should consist of entries similar to the following:

```
AJAX.ORG:AJAX
```

```
FIREFLY.COM:FIREFLY
```

```
HR1.FIREFLY.COM:HR1
```

After you have manually defined the mapping of Active Directory domains to NTLM domains, you can use the `pam.ntlm.auth.domains` parameter to specify the list of domains that should use NTLM authentication instead of Kerberos authentication. For more information about defining the domains that should use Kerberos authentication, see `pam.ntlm.auth.domains`.

Alternatively, you can set the group policy, **Computer Configuration > Centrify Settings > DirectControl Settings > Network and Cache Settings > Specify AD to NTLM domain mappings**.

## `adclient.ntlm.separators`

This configuration parameter specifies the separators that may be used between the domain name and the user name when NTLM format is used. For example, the following setting:

```
adclient.ntlm.separators: +/\
```

allows any of the following formats (assuming a user `joe` in the `acme.com` domain):



acme.com+joe  
acme.com/joe  
acme.com\joe

**Note:** The backslash character (\) can be problematic on some UNIX shells, in which case you may need to specify domain\\user.

The first character in the list is the one that `adclient` uses when generating NTLM names.

The default values are `+/\`, with `+` being the `adclient` default.

## adclient.one-way.x-forest.trust.force

Use this configuration parameter, `adclient.one-way.x-forest.trust.force`, to specify a list of two-way trusted domains that need to be treated as one-way trusted domains. This is useful when two-way trusted domains are not accessible from currently joining machine, for example, they are behind a firewall. Configuring this parameter allows x-forest users to authenticate onto the trusting machines.

The options are:

- An empty list (default)
- A list of forests or domains to be treated as one-way trusted.

Specify a list of two-way trusted forests, and domains that have two-way external trust relationship with the local domain, to be treated by DirectControl Agent as one-way trusted forests or domains.

This parameter is likely to be used together with the configuration parameters, `pam.ntlm.auth.domains` and `adclient.ntlm.domains`, if these forests and domains are not accessible from the currently joining machine.

- Use the `pam.ntlm.auth.domains` parameter to specify the list of domains that use NTLM authentication instead of Kerberos authentication.
- Use the `adclient.ntlm.domains` parameter to map AD domains to NTLM domains.

Alternatively, you can set the group policy **Computer Configuration > Centrify Settings > DirectControl Settings > Adclient Settings > Force domains and forests to be one-way trusted**.



## adclient.os.name

This configuration parameter specifies the name of the operating system for the local computer. This information is dynamically determined by the `uname` command and stored in the Active Directory computer object. The configuration parameter value can be manually overridden by defining a different value, if desired. If you change the value, however, you must restart `adclient` for the change to take effect.

For example, to set the parameter value manually to Linux:

```
adclient.os.name: Linux
```

## adclient.os.version

This configuration parameter specifies the version of the operating system for the local computer. This information is dynamically determined by the `uname` command and stored in the Active Directory computer object. The configuration parameter value can be manually overridden by defining a different value, if desired. If you change the value, however, you must restart `adclient` for the change to take effect.

For example, to set the parameter value manually to 3.0-125:

```
adclient.os.version: 3.0-125
```

## adclient.os.version.use.win7prefix

This configuration parameter specifies whether the operating system version prefix (6.1:) should be added automatically to the computer object's `operatingSystemVersion` attribute when a computer joins the domain. This prefix is used to indicate whether a computer supports FIPS encryption. The valid values are:

- 1 to add the prefix only when FIPS encryption is enabled.
- 2 to add the prefix regardless of FIPS encryption.

Depending on the version of the agent you have installed, the default value might be either of these values. The recommended setting for this parameter is 2. For example, to always add the prefix:

```
adclient.os.version.use.win7prefix: 2
```



## adclient.paged.search.max

This configuration parameter specifies the maximum number of items included in each page of a paged LDAP search.

The parameter value must be a positive integer. For example:

```
adclient.paged.search.max: 100
```

If this parameter is not defined in the configuration file, its default value is 100 items.

Before changing this parameter setting, you should consider its impact on your environment. As you decrease the number of items included in each LDAP page, you increase the number of connections made to Active Directory and the added demand that increased traffic places on the server, but you decrease the RAM used by the agent. If you increase the number of items included in each LDAP page, you decrease the number of connections to Active Directory and reduce the overall demand on the server, but you increase the RAM used by the agent.

## adclient.prefer.cache.validation

This configuration parameter instructs adclient to authenticate the user using the cached credentials first regardless of the current connectivity state with the Active Directory domain controller.

The parameter value is either `true` or `false`. The default is `false`; for example

```
adclient.prefer.cache.validation: false
```

Set this option to `true` to reduce traffic on slow networks. However, if the Active Directory credentials are not synchronized with the cached credentials, you run the risk of undesired side effects when the computer is online.

You can also set this configuration parameter using group policy.

## adclient.preferred.login.domains

This configuration parameter enables you to specify the domain names against which to authenticate SAM account names. Use this parameter if your



environment contains identical SAM account names on multiple domains, and you want to authenticate against a specific domain.

If you use this parameter, the `adclient.cache.upn.index` parameter must be set to `true`.

To use this parameter, type a space-separated list of domains as the parameter value. For example:

```
adclient.preferred.login.domains: demo1.centriify.com  
demo2.centriify.com
```

## adclient.preferred.site

This configuration parameter enables you to identify a specific site to use to locate available domain controllers. By default, the `adclient` process uses CLDAP `NETLOGON` requests to automatically discover its site based on how sites are configured using Active Directory Sites and Services. This default behavior enables `adclient` to select domain controllers in the same site as preferred domain controllers because they are likely to provide the best performance and least replication delays. This configuration parameters enables you to override the site returned by Active Directory and use a specific site.

If you don't define a value for the parameter, `adclient` continues to discover sites based on how sites are configured using Active Directory Sites and Services.

If you want to define a specific site to use, you can use the following override options:

- You can specify a “universal” site override that does not include an Active Directory forest in the parameter name. The override applies to all Active Directory domains that do not have a forest-specific override.
- You can specify one or more “forest-specific” site overrides that includes the name of an Active Directory forest in the parameter names. This type of override limits the domain controllers to the domain controllers in the specified forest-specific site.

Forest-specific site overrides take precedence over universal site overrides. Depending on your requirements, you can use the site override options to override sites for all forests, specific forests, or a combination of the two.

The following is an example a “universal” site override that applies to all forests that do not have a forest specific override:



```
adclient.preferred.site: my-preferred-site
```

To specify a forest-specific site override, you specify the configuration parameter using the following format:

```
adclient.preferred.site.forest_name: my-forest-site
```

The following example illustrates how you would define the configuration parameters to use the USNORTH Active Directory site for all forests except the ocean-site forest-specific site.

```
adclient.preferred.site: USNORTH
adclient.preferred.site.ocean.net: ocean-site
```

## adclient.prevalidate.allow.groups

This configuration parameter specifies the groups that are prevalidated to access the local UNIX computer using Active Directory credentials when the computer is offline even if users in the group have not previously logged onto the computer.

Under normal circumstances, only users who have previously logged on to a computer can be authenticated when the computer is disconnected from the network. For those users, authentication is based on the password hashes stored during the previous log-on. In some cases, however, you may require users who have never logged on to a particular computer to be authenticated when the computer is disconnected from the network. For example, you may have an administrative group that requires access to computers that are disconnected from the network but on which they have never previously logged in. To accommodate the users in that group, you can configure the group for prevalidation.

In most cases, you set this configuration parameter using group policy.

If you are manually setting this parameter, the parameter value must be a comma-separated list of UNIX group names. Enclose group names with spaces in double quotes, for example:

```
adclient.prevalidate.allow.groups: performx,qualtrak,"domain admins"
```



## Using this parameter with other prevalidation parameters

If you do not specify any groups for this parameter, then no group accounts are prevalidated to access the local computer. If you specify either the `adclient.prevalidate.allow.users` or `adclient.prevalidate.allow.groups` parameters, only those users and groups are prevalidated, with the exception of any users or groups specified by `adclient.prevalidate.deny.users` and `adclient.prevalidate.deny.groups` parameters. For example, to allow all users in the `admins` group to be prevalidated, except the users who are also members of the `outsource` group, you could set the `adclient.prevalidate.allow.groups` and `adclient.prevalidate.deny.groups` parameters like this:

```
adclient.prevalidate.allow.groups: admins
adclient.prevalidate.deny.groups: outsource
```

To allow prevalidation for all users in the zone without any exceptions, you can set the `adclient.prevalidate.allow.groups` parameter to `all@zone`. For example:

```
adclient.prevalidate.allow.groups: all@zone
```

For users or groups of users to be prevalidated, their accounts must be active accounts with permission to log on to the local computer and have a service principal name (SPN) set in the form of:

```
preval/username
```

Where `preval` is the service name specified by the `adclient.prevalidate.service` parameter and `username` is the user logon name, which can be either of the following:

- the name part of the user's UPN, if the domain part matches the user's domain
- `samAccountName`, if the UPN is empty or the UPN's domain part is different from the user's domain

## Registering service principal names

To enable prevalidation for a user, you can use the Windows `setspn.exe` utility to add a service principal name for the user. For example, to register the service principal name for the user `kai@arcade.com` using `preval` as the



service name, you could type a command similar to the following in a Windows Command Prompt window:

```
setspn -A preva1/kai kai
```

This `setspn` command registers the SPN in Active Directory for the `preva1` service for the specified user account, the Active Directory user `kai`. On the computers where this user is allowed to be prevalidated, the user can be authenticated without having logged on previously.

If you are allowing prevalidation for an administrative group, you must register a service principal name (SPN) for each member of the group. For example, if you are allowing prevalidation for the `admins` group and this group has five members, you would use the `setspn.exe` utility to register a Service Principal Name for each of those members.

## Specifying the supported encryption types

All prevalidated users must have their Active Directory `msDS-SupportedEncryptionTypes` attribute set to `0x18` (for just AES128 and AES256 support) or above to be able to login when disconnected. The parameter value represents the sum of the encryption types supported. Use the sum of the following encryption type values to determine the parameter value:

```
DES_CBC_CRC = 0x01  
DES_CBC_MD5 = 0x02  
RC4_HMAC_MD5 = 0x4  
AES128_CTS_HMAC_SHA1_96 = 0x08  
AES256_CTS_HMAC_SHA1_96 = 0x10
```

For example, `0x1c` indicates support for `RC4_HMAC-MD5`, `AES128_CTS_HMAC_SHA1_96`, and `AES256_CTS_HMAC_SHA1_96`.

## Refreshing prevalidated credentials

To ensure their validity, the credentials for prevalidated users and groups are periodically retrieved from Active Directory. For example, the credentials are refreshed whenever you do the following:

- Reboot the local computer.
- Start or restart the `adcli`ent process.



- Run the `adflush` command to clear the cache.
- Changes a password from the local system.

The credentials are also periodically refreshed at the interval defined by the `adclient.prevalidate.interval` parameter to ensure that prevalidation will continue working after password changes.

## **adclient.prevalidate.allow.users**

This configuration parameter specifies the users that are prevalidated to access the local UNIX computer using Active Directory credentials when the computer is offline even if they have not previously logged onto the computer.

Under normal circumstances, only users who have previously logged on to a computer can be authenticated when the computer is disconnected from the network. For those users, authentication is based on the password hashes stored during a previous log on. In some cases, however, you may require users who have never logged on to a particular computer to be authenticated when the computer is disconnected from the network. For example, you may have administrative users who require access to computers that are disconnected from the network but on which they have never previously logged in. To accommodate those users, you can configure them for prevalidation.

In most cases, you set this configuration parameter using group policy.

If you are manually setting this parameter, the parameter value must be a comma-separated list of UNIX user names. Enclose user names with spaces in double quotes, for example:

```
adclient.prevalidate.allow.users: jesse,rae,tai,"sp1 user"
```

### **Using this parameter with other prevalidation parameters**

If you do not specify any users for this parameter, then no specific user accounts are prevalidated to access the local computer. If you specify either the `adclient.prevalidate.allow.users` or `adclient.prevalidate.allow.groups` parameters, only those users and groups are prevalidated, with the exception of any users or groups specified by `adclient.prevalidate.deny.users` and `adclient.prevalidate.deny.groups` parameters. For example, to allow all users in the `admins` group and the users `ali`, `kai`, and `tanya` who are not



members of the `admins` group to be prevalidated, but prevent the users `jorge` and `maurice` from being prevalidated, you could set the `allow` and `deny` parameters like this:

```
adclient.prevalidate.allow.groups: admins
adclient.prevalidate.allow.users: ali,kai,tanya
adclient.prevalidate.deny.users: jorge,maurice
```

For users or groups to be prevalidated, their accounts must be active accounts with permission to log on to the local computer and have a Service Principal Name (SPN) set in the form of:

```
preval/username
```

Where `preval` is the service name specified by the `adclient.prevalidate.service` parameter and `username` is the user logon name, which can be either of the following:

- the name part of the user's UPN, if the domain part matches the user's domain
- `samAccountName`, if the UPN is empty or the UPN's domain part is different from the user's domain

## Registering service principal names

To enable prevalidation for a user, you can use the Windows `setspn.exe` utility to add a service principal name for the user. For example, to register the Service Principal Name for the user `kai@arcade.com` using `preval` as the service name, you could type a command similar to the following in a Windows Command Prompt window:

```
setspn -A preval/kai kai
```

This `setspn` command registers the SPN in Active Directory for the `preval` service for the specified user account, the Active Directory user `kai`. On the computers where this user is allowed to be prevalidated, the user can be authenticated without having logged on previously.

## Specifying the supported encryption types

All prevalidated users must have their Active Directory `msDS-SupportedEncryptionTypes` attribute set to `0x18` (for just AES128 and AES256 support) or above to be able to login when disconnected. The



parameter value represents the sum of the encryption types supported. Use the sum of the following encryption type values to determine the parameter value:

```
DES_CBC_CRC = 0x01
DES_CBC_MD5= 0x02
RC4_HMAC_MD5 = 0x4
AES128_CTS_HMAC_SHA1_96 = 0x08
AES256_CTS_HMAC_SHA1_96 = 0x10
```

For example, 0x1c indicates support for RC4\_HMAC-MD5, AES128\_CTS\_HMAC\_SHA1\_96, and AES256\_CTS\_HMAC\_SHA1\_96.

## Refreshing prevalidated credentials

To ensure their validity, the credentials for prevalidated users and groups are periodically retrieved from Active Directory. For example, the credentials are refreshed whenever you do the following:

- Reboot the local computer.
- Start or restart the `adclient` process.
- Run the `adflush` command to clear the cache.
- Changes a password from the local system.

The credentials are also periodically refreshed at the interval defined by the `adclient.prevalidate.interval` parameter to ensure that prevalidation will continue working after password changes.

## `adclient.prevalidate.deny.groups`

This configuration parameter specifies the groups that cannot be prevalidated to access the local UNIX computer. If you allow any groups or users to be prevalidated, you can use this parameter to define exceptions for any groups that should be prevented from prevalidation. In most cases, you would use this parameter to exclude a subset of users that are in a member group of an allowed group. For example, to allow all users in the `admins` group to be prevalidated, except the users who are members of the `outsource` subgroup, you could set the `adclient.prevalidate.allow.groups` and `adclient.prevalidate.deny.groups` parameters like this:

```
adclient.prevalidate.allow.groups: admins
adclient.prevalidate.deny.groups: outsource
```

In most cases, you set this configuration parameter using group policy.



If you are manually setting this parameter, the parameter value must be a comma-separated list of UNIX group names. Enclose group names with spaces in double quotes, for example:

```
adclient.prevalidate.deny.groups: performx,qualtrak,"domain  
admins"
```

## adclient.prevalidate.deny.users

This configuration parameter specifies the users that cannot be prevalidated to access the local UNIX computer. If you allow any groups or users to be prevalidated, you can use this parameter to define exceptions for any users who should be prevented from prevalidation. In most cases, you would use this parameter to exclude a subset of users that are members of an allowed group. For example, to allow all users in the admins group except the users jorge and maurice who are members of the admins group to be prevalidated, you could set the allow and deny parameters like this:

```
adclient.prevalidate.allow.groups: admins  
adclient.prevalidate.deny.users: jorge,maurice
```

In most cases, you set this configuration parameter using group policy.

If you are manually setting this parameter, the parameter value must be a comma-separated list of UNIX user names. Enclose user names with spaces in double quotes, for example:

```
adclient.prevalidate.deny.users: jesse,rae,tai,"sp1 user"
```

## adclient.prevalidate.interval

This configuration parameter specifies the interval, in hours, for refreshing the credentials for prevalidated user and group accounts. The credentials for prevalidated users must be periodically refreshed to ensure they are in sync with Active Directory and that prevalidation will continue working after password changes.

The parameter value should be a positive integer. A value of 0 disables all prevalidation of users. For example, to refresh the credentials for prevalidated users every 8 hours:

```
adclient.prevalidate.interval: 8
```

In most cases, you set this configuration parameter using group policy.



## adclient.prevalidate.service

This configuration parameter specifies the service name to use for prevalidated users and groups. You must use the name you specify in this parameter when you register the Service Principal Name (SPN) for a user or group with the `setspn.exe` utility.

For example, to set the service name to `preval`:

```
adclient.prevalidate.service: preval
```

In most cases, you set this configuration parameter using group policy.

## adclient.random.password.generate.try

This configuration parameter specifies the maximum number of times that the agent attempts to generate a random Active Directory password. Depending on the complexity requirements of your environment, you may need to set this value higher than the default to ensure an appropriately complex password is generated.

The default value is 10.

For example:

```
adclient.random.password.generate.try: 10
```

## adclient.random.password.complexity.pattern

This configuration parameter specifies the complexity requirements for the generation of a random Active Directory password. Each complexity requirement is assigned a numeric value:

English uppercase characters (A through Z) = **1**

English lowercase characters (a through z) = **2**

Base 10 digits (0 through 9) = **4**

Special, non-alphanumeric characters (!, \$, #, %, etc...) = **8**

The parameter value is the additive value assigned to the different complexity requirements you require of the password.



For example, if you wanted to require the generated password to include at least one uppercase letter, at least one lower case letter, and at least one digit, you would set the value at  $1 + 2 + 4 = 7$ ; or:

```
adclient.random.password.complexity.pattern: 7
```

The default value for this configuration parameter is 7.

## adclient.random.password.length.min

This configuration parameter specifies the minimum character length of a randomly generated Active Directory password.

The default value is 15 characters. For example:

```
adclient.random.password.length.min: 15
```

## adclient.random.password.length.max

This configuration parameter specifies the maximum character length of a randomly generated Active Directory password.

The default value is 21 characters. For example:

```
adclient.random.password.length.max: 21
```

## adclient.samba.sync

This configuration parameter specifies whether you want to have the Centrify agent work in conjunction with Samba. The parameter value can be either `true` or `false`. You should set this parameter to `false` if you do not want any interaction between Centrify and Samba.

If you want the agent to work with Samba, you may need to make changes to your environment or configure additional settings. For Centrify and Samba to operate in the same environment, you need to do the following:

- Check that the `samba.base.path` configuration parameter specifies the correct path to the Samba binaries.
- Check that the `samba.winbind.listen.path` configuration parameter specifies the correct path to the Samba winbind listen path.



- Check that Samba is configured for ADS security.
- Check that Samba belongs to the same REALM as the Centrify agent.
- Verify that Samba and the Centrify agent share an Active Directory computer object.
- Set the `adclient.samba.sync` configuration parameter to `true`.

For example:

```
adclient.samba.sync: true
samba.base.path: /usr
samba.windbind.listen.path: /run/samba/windbindd
```

For more information about installing and configuring Samba to work with Centrify software, see the *Samba Integration Guide* available on the Centrify web site.

## adclient.server.try.max

This configuration parameter specifies the maximum number of servers per domain the agent should attempt to connect to before going into disconnected mode. This parameter is used if the agent is unable to connect to its primary domain controller to enable it to query DNS for a list of other domain controllers and try each server in the list up to the maximum number of servers you specify. For example, if you have a large number of replica domain controllers for a given domain, you may want to use this parameter to limit the number of servers for the agent to try in order to limit network traffic and improve performance.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be a positive integer or 0. Setting the parameter value to 0 means that the agent attempts to connect to every server in the list until successful.

The default value is 0.

For example, to allow the agent to attempt to contact up to five domain controllers before going into disconnected mode:

```
adclient.server.try.max: 5
```



This parameter is ignored if you have defined a master domain controller for the zone to which the computer belongs because the computer only connects to that domain controller.

**Note:** This parameter is deprecated for versions of `adclient` from 4.4.3 to 5.0.x. It is available in version 5.1.0 and later.

## `adclient.skip.inbound.trusts`

This configuration parameter specifies whether you want `adclient` to skip probing inbound trusts for the `domaininfomap`.

Options are:

- `false`: If set to `false`, when building `domaininfomap`, both two-way and incoming trusts are probed. (Default)
- `true`: If set to `true`, when building `domaininfomap`, only two-way trusts are probed.

Set `adclient.skip.inbound.trusts` in the `centrifdc.conf` file. For example:

```
adclient.skip.inbound.trusts: true
```

To apply this configuration parameter while `adclient` is running, follow the recommended sequence:

1. Perform `adreload`.
2. Rebuild the `domaininfomap`. Choose a method:
  - Run `adflush -t` to rebuild the `domaininfomap` manually.
  - Wait for the next rebuild cycle from `adclient`.

## `adclient.skip.unused.outbound.trusts`

This configuration parameter specifies whether you want to prevent the agent from sending network queries to outbound trust domains that do not have users in Centrify zones.

If you set this parameter to `true`, the agent will only send network queries to outbound trust domains that have users in Centrify zones.



If you are manually setting this parameter, the parameter value must be `true` or `false`. For example:

```
adclient.skip.unused.outbound.trusts: true
```

If the parameter is not explicitly defined in the configuration file or by group policy, its default value is `false`.

## adclient.snmp.enabled

This configuration parameter specifies whether you want to use the Windows Time Service to keep the local system clock in sync with the domain the computer has joined.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be `true` or `false`. For example:

```
adclient.snmp.enabled: true
```

If the parameter is not explicitly defined in the configuration file or by group policy, its default value is `true`.

## adclient.snmp.poll

This configuration parameter specifies the interval between SNMP clock updates when you are using the Windows Time Service to keep the local system clock in sync with the domain the computer has joined.

In most cases, you set the polling interval using group policy.

If you are manually setting this parameter, the value is the base 2 logarithm of the time in seconds. For example, setting this parameter value to 6 sets the update interval to 64 seconds ( $2^6$ ), and a value of 15 sets the update interval to 32768 seconds, or 9.1 hours. For example, to set the update interval to 256 seconds:

```
adclient.snmp.poll: 8
```

If the parameter is not explicitly defined in the configuration file or by group policy, its default value is 15 (9.1 hours).



## adclient.tcp.connect.timeout

This parameter specifies the timeout of all TCP port probing used in adclient. This parameter default is ten seconds.

## adclient.udp.timeout

This configuration parameter specifies the maximum number of seconds to allow to complete UDP binding. The agent will attempt to bind twice. If the first bind request is not complete within the period specified by this parameter, the agent sends a second request with a timeout period that is double the setting of this parameter. If both bind requests fail to complete within the allotted time, the agent sets its status to disconnected.

For example, if you set this parameter to 10 seconds and the bind request is not complete within 10 seconds, the agent sends a second bind request and waits a maximum of 20 seconds for the bind to complete before assuming the computer is disconnected from the network or Active Directory is unavailable.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value should be a positive integer. The default value for this parameter is 15 seconds. For example:

```
adclient.udp.timeout: 15
```

## adclient.update.os.interval

This configuration parameter specifies the number of seconds to wait before updating operating system information after adclient starts in disconnected mode.

If you are manually setting this parameter, the parameter value should be a positive integer. The default value for this parameter is 30 seconds. For example:

```
adclient.update.os.interval: 30
```



## adclient.use.all.cpus

This configuration parameter specifies whether to use all processors on a multi-processor system. The parameter value can be `true` or `false`. Setting this parameter to `true` allows the `adclient` process to use additional CPUs on a computer to process background tasks in parallel when logging on and can significantly decrease the startup time in sites with a large number of domain controllers.

For example:

```
adclient.use.all.cpus: true
```

If the parameter is not explicitly defined in the configuration file, its default value is `true`. If you change this parameter, you must restart the `adclient` process for the change take effect.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

## adclient.use.tokengroups

This configuration parameter specifies whether the agent should attempt to use the Active Directory `tokenGroups` attribute on the user object to determine a user's group membership when the Kerberos Privilege Attribute Certificate (PAC) is not available.

In most cases, allowing the agent to use this attribute when necessary is desirable and the default setting for this attribute is `true`. For example:

```
adclient.use.tokengroups: true
```

In mixed-mode domains with both Windows 2000 and Windows 2003 computers, however, the `tokenGroups` attribute can include Universal groups in the user's group membership list. If you have Universal groups in mixed-mode domains and want to prevent those Universal groups from being included in the user's group membership list, you can set this parameter value to `false`.

Setting this value to `false` will force the agent to use a slower mechanism for finding group membership instead of the `tokenGroups` attribute and can result in a slower user login experience, but the results will be consistent with what would be retrieved using the Kerberos PAC.



## adclient.user.computers

This configuration parameter specifies whether to allow computer principals to be treated as users with login capabilities when added to the zone. The parameter value can be `true` or `false`. The configuration parameter must be set to `true` to allow Distributed File System support for Samba. Setting this to `true` may impact performance, however, in domains with heavily-loaded domain controllers or large user and computer populations.

For example:

```
adclient.user.computers: true
```

## adclient.user.lookup.cn

This configuration parameter specifies whether you want to allow users to be found by their common name (`cn`) attribute. The parameter value can be `true` or `false`.

By default, you can allow users to login using their UNIX profile name, Active Directory `displayName`, or Active Directory `cn` attribute. However, allowing users to log on using these additional attributes can require the agent to perform multiple searches to locate a user account in Active Directory. In environments with domain controllers under heavy load or with large user populations, searching Active Directory multiple times might negatively impact performance.

If you want to prevent the agent from attempting to access to user information by the common name, you can set this configuration parameter to `false`. For example:

```
adclient.user.lookup.cn: false
```

The default value for Mac OS X computers is `false`.

The default parameter value for all other platforms is `true`.

## adclient.user.lookup.display

This configuration parameter specifies whether you want to allow users to be found by their display name (`displayName`) attribute. The parameter value can be `true` or `false`.



By default, you can allow users to login using their UNIX profile name, Active Directory `displayName`, or Active Directory `cn` attribute. However, allowing users to log on using these additional attributes can require the agent to perform multiple searches to locate a user account in Active Directory. In environments with domain controllers under heavy load or with large user populations, searching Active Directory multiple times might negatively impact performance.

If you want to prevent the agent from attempting to access to user information by the `displayName` attribute, you can set this configuration parameter to `false`. For example:

```
adclient.user.lookup.display: false
```

The default value for Mac OS X computers is `false`.

The default parameter value for all other platforms is `true`.

## `adclient.user.name.max.exceed.disallow`

When this parameter is set to `false`, users with a login name longer than eight characters are permitted to log in. When set to `true`, users with a login name longer than eight characters are not permitted to log in. This configuration parameter applies to local account management in that a local user with rights in addition to the platform-specified limit is not be added to the system.

## `adclient.version2.compatible`

This configuration parameter is used to maintain compatibility with zones created using version 2.0 or 3.0 of Access Manager. The default is `true` for zones created using the 2.0 or 3.0 console. The default is `false` for zones created with the 4.x or later console.

If you do not have users or groups that were given access to UNIX computers using an older console, having this parameter set to `false` results in a performance improvement on Windows 2000 domain controllers. Setting the value to `true` decreases login performance on Windows 2000 domain controllers.

For example:

```
adclient.version2.compatible: false
```



If you have users or groups that were given access to UNIX computers using an older console, you may want to upgrade those users and groups to take advantage of the performance improvements.

To determine whether you have zones and users from an older version of centrify software , open the console and click **Analyze**. You can then review the Analysis Results and attempt to update user properties, if needed.

## adclient.zone.group.count

This configuration parameter provides a calculated value that controls the method used to determine group membership for users. If the calculated value for this parameter is larger than the number of groups a user is a member of, the Centrify agent iterates over the user's group list to determine group membership. For example, if there are more group profiles defined for the zone than the number of groups the user is a member of, the agent uses the user's group list to determine group membership.

If the calculated value for this parameter is smaller than the typical number of groups a user is a member of, the agent iterates over all of the group profiles enabled for the zone to determine group membership. For example, if there are fewer group profiles defined for the zone than the number of groups the user is a member of, the agent uses the zone's group profile list to determine group membership.

Switching between the two methods for determining group membership may improve the log-in time for some users. You can use this configuration parameter to override the calculated value. For example, if you always want to use the user's group membership list rather than iterate through the list of group profiles defined for the zone, you can set this parameter to an artificially high value. If you always want to use the zone's group profile list rather iterate through the user's group membership list, you can set this parameter to an artificially low value.

For example:

```
adclient.zone.group.count: 6
```



## addns.tcp.timeout

This configuration parameter controls the amount of time, in seconds, that the addns process waits for responses to its requests for updates.

The parameter value can be any positive integer. The default value of this parameter is 7 seconds:

```
addns.tcp.timeout: 7
```

## addns.wait.time

This configuration parameter controls the amount of time, in seconds (default 60), that the addns process should wait for another addns process to exit before proceeding.

Because the addns process enables dynamic updates to DNS records on Active Directory-based DNS servers, it includes a mechanism to prevent two addns processes from running at the same time. This configuration parameter value controls how long a addns command request will wait for another addns process to complete its execution before proceeding.

The parameter value can be any positive integer. For example, to set the wait time to 45 seconds:

```
addns.wait.time: 45
```

## adjust.offset

This configuration parameter specifies the time difference between the local host and the domain that should trigger an adjustment to the local computer's time-of-day setting.

The default parameter value is 5 minutes. With this setting, if the time difference between the local host and the domain controller is less than 5 minutes, the `adcli` process calls the `adjtime` function to update the local host time to match the Active Directory domain. If the offset between the local computer and the domain controller is more than 5 minutes, `adcli` process calls the `settimeofday` function to update local computer's time.

The parameter value can be any positive integer. For example:

```
adjust.offset.time: 5
```



## audittrail.audited.command.with.args

This configuration parameter controls whether audit trails for audited command include command parameters. If set to `true`, the command name and parameters are displayed in the audit trail. If set to `false`, just the command name is displayed in the audit trail.

The default value is `false`.

## audittrail.Centrify\_Suite.Trusted\_Path.machinecred.skipda

This configuration parameter specifies whether trusted path audit trail events are sent to the audit installation database in situations where the user is using a computer credential. The default value is `true` (that is, events are not sent to the audit database). For example:

```
audittrail.Centrify_Suite.Trusted_Path.machinecred.skipda: true
```

Events are sent to the system log even if this parameter is set to `true`.

## audittrail.targets

This configuration parameter specifies the target for audit trail information. Possible settings are:

0. Audit information is not sent.
1. Audit information is sent to DirectAudit. This capability is supported by DirectAudit version 3.2 and later.
2. Audit information is sent to the local logging facility (syslog on UNIX systems, Windows event log on Windows systems).
3. Audit information is sent to both DirectAudit and the local logging facility.

If DirectAudit 3.2 or later is installed, the default value is 3 (local logging facility and DirectAudit). Otherwise, the default value is 2 (local logging facility only). For example:

```
audittrail.targets: 3
```

In most cases, you set this configuration parameter using group policy.



## audittrail.<product>.<component>.overrides

This parameter specifies whether to override the global audit trail targets. If this parameter is set, the system uses the targets value in the current component; otherwise, the system uses the global configured value.

There are two target settings that can be overridden:

- Whether the system sends the audit trail information to DirectAudit or not
- Whether the system sends the audit trail information to the local logging system or not. On UNIX systems, the local logging system is syslog and on Windows systems it's the Windows event log.

For this setting, you specify a single numeric value to represent where the system will send the audit trail information. (Setting one value to signify two settings is called a bit mask.) The possible settings are as follows:

Value	Override whether the audit trail information is sent to DirectAudit?	Override whether the audit trail information is sent to the local logging system?	Description
0	No	No	There is no override to the audit trail target of the current component. The system uses the global audit trail target value.
1	Yes	No	The system overrides just the audit trail target for DirectAudit. This capability is supported by DirectAuditversion 3.2 and later.
2	No	Yes	The system overrides just the audit trail target for the local logging system. If you're using a DirectAuditversion prior to version 3.2, this is the default setting.
3	Yes	Yes	The system overrides both the audit trail targets for DirectAuditand the local logging system. If you're using DirectAuditversion 3.2 or later, this is the default setting.

In most cases, you set this configuration by way of group policy.



## audittrail.<product>.<component>.targets

This parameter specifies how to calculate where the system sends the audit trail information for a particular component if you have also set the corresponding `audittrail.<product>.<component>.overrides` parameter.

There are two kinds of audit trail targets that can be overridden:

- Whether to enable the DirectAudit audit trail target for the component or not
- Whether to enable the local logging system audit trail target or not. On UNIX systems, the local logging system is syslog and on Windows systems it's the Windows event log.

For this setting, you specify a single numeric value to represent which audit trail targets are enabled for the component. The possible settings are as follows:

Value	Enable the DirectAudit audit trail target for the component?	Enable the local logging audit trail target for the component?	Description
-1	No	No	Use the global audit trail target value. This is the default setting.
0	No	No	Neither the DirectAudit nor the local logging target are enabled for the component.
1	Yes	No	Enable only the DirectAudit audit trail target for the component. This capability is supported by DirectAudit version 3.2 and later.
2	No	Yes	Enable only the local logging audit trail target for the component.
3	Yes	Yes	Enable the audit trail targets for both DirectAudit and the local logging system.

In most cases, you set this configuration parameter using the **Computer Configuration > Policies > Administrative Templates Policy definitions (ADMX files) > Centrify Audit Trail Settings** group policy.



The system calculates the final audit trail targets for a component based on the following information:

- If `audittrail.<product>.<component>.overrides` is not specified, the system uses the global audit trail target value
- If `audittrail.<product>.<component>.overrides` is specified, for each target (DirectAudit and local logging), whether the audit trail information will be sent to this target is determined by the following:
  - If `audittrail.<product>.<component>.targets` is set to `-1`, or the setting is not overridden in `audittrail.<product>.<component>.overrides`, the system uses the global audit trail target value
  - If the target is overridden by `audittrail.<product>.<component>.overrides` and enabled by `audittrail.<product>.<component>.targets`, the system sends the audit trail information to this target

## adsec.gpfas

This configuration parameter specifies whether to use Windows Firewall with Advanced Security (GPFAS) settings or IP Security Policies on Active Directory (GPIPsec) settings for all DirectSecure policies.

By default, this parameter is set to `false`, meaning that IP Security Policies on Active Directory (GPIPsec) settings are used. To specify that Windows Firewall with Advanced Security (GPFAS) settings be used, set this parameter to `true`.

You can also set this parameter using group policy.

## capi.cache.enabled

This configuration parameter specifies whether the in-process memory CAPI cache is enabled. If the cache is enabled, lookups are sent to the cache before being sent to `adclient`.

- If the object is found in the cache and has a valid TTL (as configured in the `capi.cache.negative.ttl` and `capi.cache.ttl` parameters), the object is returned.



- If the TTL has expired, the lookup is sent to adclient.
- If the object is not found in the cache, the lookup is sent to adclient.

If the object is found in adclient, the cache entry (that is, the key-value and acquisition time stamp) is updated.

If you set this parameter to `true`, the CAPI cache is enabled.

The following attributes are supported:

- `sid`
- `_UnixName`
- `SAMAccountName`
- `userPrincipalName`
- `guid`
- `unixid`

The default value of this parameter is `false`. For example:

```
capicache.enabled: false
```

## capicache.hash.table.size

This configuration parameter specifies the number of hash map buckets that are allocated if the in-memory CAPI SID cache is enabled through the `capicache.enabled` parameter.

The default value of this parameter is 769. For example:

```
capicache.hash.table.size: 769
```

## capicache.log.interval

This configuration parameter specifies the number of seconds between log events that dump information about the performance of the in-memory CAPI SID cache. This parameter takes effect only if the in-memory CAPI SID cache is enabled through the `capicache.enabled` parameter.

- Summary information such as hits, misses, and so on are DEBUG level events.
- Details about the bucket distributions are TRACE level events.



Setting this parameter to 0 disables all hash map log dumps pertaining to the in-memory CAPI SID cache.

The default value of this parameter is 0. For example:

```
capl.cache.log.interval: 0
```

## capl.cache.max.objects

This configuration parameter specifies the maximum number of objects that are kept in the in-memory CAPI SID cache if the cache is enabled through the `capl.cache.enabled` parameter. If the number is exceeded, cached objects that are the oldest are replaced with new objects.

The default value of this parameter is ten thousand objects. For example:

```
capl.cache.max.objects: 10000
```

## capl.cache.negative.ttl

This configuration parameter specifies the number of seconds that a negative cached SID object remains in the in-memory CAPI SID cache before it is refreshed. This parameter takes effect only if the in-memory CAPI SID cache is enabled through the `capl.cache.enabled` parameter.

The default value of this parameter is 3,600 seconds. For example:

```
capl.cache.negative.ttl: 3600
```

## capl.cache.ttl

This configuration parameter specifies the number of seconds that a positive cached SID object remains in the in-memory CAPI SID cache before it is refreshed. This parameter takes effect only if the in-memory CAPI SID cache is enabled through the `capl.cache.enabled` parameter.

The default value of this parameter is 3,600 seconds. For example:

```
capl.cache.ttl: 3600
```



## db2.implement.pam.ignore.users

Starting with Centrify DB2 agent 5.2.3, this configuration parameter specifies whether the Centrify DB2 agent checks `pam.ignore.users` for a list of users to authenticate locally, without first attempting to authenticate those users in Active Directory.

By default, the Centrify DB2 agent authenticates users in Active Directory first. If users do not exist in Active Directory, the Centrify DB2 agent then authenticates users locally.

If you set this parameter to `true`, users defined in the `pam.ignore.users` list are authenticated locally only (that is, no attempt is made to authenticate them in Active Directory first). For example:

```
db2.implement.pam.ignore.users: true
```

To specify that an Active Directory authentication attempt should be made for all users, and that local authentication be attempted only for users not in Active Directory, set this parameter to `false`:

```
db2.implement.pam.ignore.users: false
```

If you change the setting of this parameter, restart the DB2 instance to activate the new setting.

## db2.user.zone\_enabled

This configuration parameter specifies whether to constrain Centrify DB2 agent authentication to zone enabled Active Directory users only. By default, the Centrify DB2 agent authenticates all Active Directory users even if the Active Directory user is not in the zone. To constrain the authentication to zone enabled Active Directory users only, add the following parameter to the `/etc/centrifydc/centrifydc.conf` file:

```
db2.user.zone_enabled.db2_instance_name: true
```

In this parameter, `db2_instance_name` is the name of the DB2 instance (for example, `db2inst1`).

After you add this parameter, restart the DB2 instance to pick up the new setting.



## db2.userpass.username.lower

This configuration parameter specifies whether the DB2 userpass plugin is used to convert the user name to lowercase before attempting authentication (`true`) or not make the conversion (`false`, the default).

## dc.dead.cache.refresh

This configuration parameter specifies how long, in seconds, to keep in cache the fact that a domain controller is dead.

The default value is 60 seconds. For example:

```
dc.dead.cache.refresh: 60
```

## dc.live.cache.refresh

This configuration parameter specifies how long, in seconds, to keep in cache the fact that a domain controller is alive.

The default value is 3600 seconds (one hour). For example:

```
dc.live.cache.refresh: 3600
```

## dc.penalty.time

This configuration parameter controls how long a domain controller that has failed is considered less preferable to the other domain controllers in the forest that either have not failed or have failed farther back in time.

The default setting is 3600 seconds (one hour).

This parameter helps you avoid domain controllers that appear to be alive, but when they are selected exhibit higher level failures such as crashed, tombstoned, or dead netlogon service.

The value specifies the number of seconds. For example, the following specifies two hours:

```
dc.penalty.time: 7200
```



## dns.alive.resweep.interval

This configuration parameter specifies the amount of time to wait, when DNS is active, before triggering a DNS server sweep to see if any DNS servers are responding faster than the current one.

The `adclient` process periodically checks in the background to see if any DNS servers are available with faster response times than the currently active DNS server. This parameter, `dns.alive.resweep.interval`, determines how often this check, or sweep, occurs. The default is one hour (3600 seconds).

For the sweep, the `dns.sweep.pattern` parameter determines the probe pattern that is used to find a live DNS server; that is, it sets:

- The protocol to use (TCP or UDP)
- The amount of time to wait for a response.

The DNS server that responds fastest is selected, is cached in memory, and is used for all DNS requests until one of the following occurs:

- It stops responding.
- A new server sweep discovers a faster DNS server and replaces it.
- `adclient` is stopped and restarted.

If the newly selected server is different than the previous server, the `kset.dns.server` file is updated with the address of the newly selected server.

The default value for this parameter is 3600 seconds.

The parameter value must be a positive integer. For example:

```
dns.alive.resweep.interval: 3600
```

## dns.block

This configuration parameter specifies the list of domain controllers that should be filtered out when resolving the domain controller to contact through DNS.

This configuration parameter enables you to prevent the `adclient` process from attempting to contact domain controllers that are known to be inaccessible, for example, because they reside behind a firewall, or domain controllers that shouldn't be contacted, for example, because of their physical location or because they are no longer valid domain controllers for the site.



The parameter value can be one or more fully-qualified domain controller server names. If you are specifying more than one domain controller name, the names can be separated by commas or spaces. For example:

```
dns.block: ginger.ajax.org,salt.ajax.org,nc1.sea.ajax.org
```

In most cases, you set this configuration parameter using group policy.

If you don't specify a value for this parameter, access is not blocked for any domain controllers.

## dns.cache.negative

This configuration parameter specifies whether to cache negative DNS responses. A negative response is returned when a DNS server is not found. By storing a negative result in the cache, the agent does not look for a server that was previously not found.

Set this parameter to true to cache negative DNS responses or false to not cache negative responses. When this parameter is false, the system attempts to respond to all requests. A cached response expires after the amount of time specified by the `dns.cache.timeout` parameter (default value is 300 seconds).

The default is true; for example:

```
dns.cache.negative:true
```

## dns.cache.timeout

This configuration parameter specifies the amount of time, in seconds, before a cached DNS response expires.

The default value is 300 seconds.

Specify a positive integer; for example:

```
dns.cache.timeout:300
```

## dns.dc.domain\_name

This configuration parameter can be used to specify the domain controller host names if your DNS is not configured to use Active Directory. In most cases, you



should not use this configuration parameter in a production environment because Active Directory automatically updates DNS with fail-over and replica servers optimized for the Active Directory site configuration. This configuration parameter is used primarily for configuring an evaluation environment when the DNS server is on a UNIX computer and can't provide the `_ldap` service records.

To set this parameter, the Active Directory domain name must be specified as the last portion of the configuration parameter name, and the parameter value is the host name of the domain controller. For example, if the Active Directory domain is `acme.com` and the domain controller for that domain is `coyote.acme.com`:

```
dns.dc.acme.com: coyote.acme.com
```

**Note:** You must specify the name of the domain controller, not its IP address. In addition, the domain controller name must be resolvable using either DNS or in the local `/etc/hosts` file. Therefore, you must add entries to the local `/etc/hosts` for each domain controller you want to use if you are not using DNS or if the DNS server cannot locate your domain controllers.

To specify multiple servers for a domain, use a space to separate the domain controller server names. For example:

```
dns.dc.lab.test: dc1.lab.test dc2.lab.test
```

## dns.dead.resweep.interval

This configuration parameter specifies the amount of time to wait, in seconds, when DNS is down, before triggering a DNS server sweep to see if any DNS servers are alive.

If the current DNS server times out on a request (does not respond within the interval and number of retries specified by `dns.tcp.timeout` or `dns.udptimeouts`), the agent attempts to acquire another DNS server. If it fails to find a live server, DNS is considered down and the agent waits for the interval specified by this parameter, `dns.dead.resweep.interval`, before attempting to acquire another DNS server.

The default is 60 seconds.

The parameter value must be a positive integer. For example:

```
dns.dead.resweep.interval: 60
```



## dns.gc.domain\_name

This configuration parameter can be used to specify the domain controller used as the global catalog if your DNS is not configured to use Active Directory. In most cases, you do not use this configuration parameter in a production environment. This configuration parameter is used primarily for configuring an evaluation environment when the DNS server is on a UNIX computer and can't provide the `_gc` service records.

To set this parameter, the Active Directory domain name must be specified as the last portion of the configuration parameter name, and the parameter value is the host name of the domain controller. For example, if the Active Directory domain is `arcade.com` and the domain controller for that domain is `fire.arcade.com`:

```
dns.gc.arcade.com: fire.arcade.com
```

**Note:** You must specify the name of the domain controller, not its IP address. In addition, the domain controller name must be resolvable using either DNS or in the local `/etc/hosts` file. Therefore, you must add entries to the local `/etc/hosts` for each domain controller you want to use if you are not using DNS or if the DNS server cannot locate your domain controllers.

To specify multiple servers for a domain, use a space to separate the domain controller server names. For example:

```
dns.dc.lab.test: dc1.lab.test dc2.lab.test
```

## dns.query.all.servers

This configuration parameter specifies whether the DNS subsystem should try all live DNS servers until either the lookup succeeds or the list is exhausted.

When this parameter is set to `true` (the default), DNS tries each server on the list of all DNS servers in `/etc/resolv.conf` (or `dns.servers`) one-by-one until either the list is exhausted or the object is resolved. By default, this configuration parameter is configured as `true`:

```
dns.query.all.servers: true
```

When this parameter is set to `false`, the DNS subsystem stops querying after the first "record not found" response.



This feature is useful in environments that contain multiple DNS servers that do not all hold the same records (and are therefore not all aware of the same AD domains).

## dns.servers

This configuration parameter specifies a space separated list of IP addresses of DNS servers that are used to resolve domain controller names. Set this parameter if a computer running Mac OS X 10.7 or later cannot connect to a domain controller through a VPN connection.

Starting with Mac OS X 10.7, `/etc/resolv.conf` is no longer used for domain controller name resolution. Therefore, some VPN programs no longer update DNS server information in `/etc/resolv.conf` when signing on. On computers running Mac OS X 10.7 and later, this can result in the computer not being able to connect to a domain controller through a VPN if the DNS server locations are not specified as described here.

The following example shows the setting of two IP addresses for DNS servers:

```
dns.servers: 111.22.333.4 555.66.777.8
```

## dns.sort

This configuration parameter determines whether to sort by speed during background sweeps or to pick the first DNS server that responds.

**Note:** This parameter only applies to *background* sweeps. During *initial* sweeps, the first server to respond is always chosen, regardless of how `dns.sort` is set.

Generally, the first server in the list (as specified in `/etc/resolv.conf` or by the `dns.servers` parameter) responds first. However, if a server was previously chosen, and is still configured in `/var/centrify/kset.dns.server`, it is always tried first regardless of how `dns.sort` is set.

This parameter is useful if you have multiple DNS servers specified in `/etc/resolv.conf`, some of which are not compatible with DirectControl. If you list the DirectControl-compatible first, and set this parameter to `false`, an incompatible server will never be chosen unless the compatible servers are unavailable.



Set the value of this parameter to `true` to sort by speed. Set the value to `false` to select the first server that responds.

The default is to sort by speed (`true`); for example:

```
dns.sort: true
```

## dns.sweep.pattern

This configuration parameter specifies a comma separated list to use when scanning for live DNS servers. For each item in the list, specify the type of scan (`t` for TCP; `u` for UDP) and the number of seconds to wait for a response.

For example, the following pattern:

```
dns.sweep.pattern: t1, u1,u2
```

specifies:

- A TCP scan with a one second wait for a response
- A UDP scan with a one second wait for a response
- Another UDP scan with a two second wait for a response

For each value, all known DNS servers are queried. If the `kset.dns.server` file exists, the server it defines is queried first.

For initial DNS server acquisition, the first DNS server to respond is chosen, at which point the sweep is terminated. Since the `kset.dns.server` file is queried first, the server it defines is most likely to be selected. Otherwise, the first server specified in `/etc/resolv.conf` responds first

For background DNS sweeps, the entire sweep pattern is completed, at which point the fastest server to respond is chosen and the sweep is terminated.

If a new DNS server is selected, the `kset.dns.server` file is updated with its address.

If the end of the list is reached and no DNS servers respond, DNS is considered down. A new sweep begins after the period of time specified by the `dns.dead.resweep.interval`.

The default pattern for Linux and Unix is:

```
dns.sweep.pattern: t1,u1,u1,t2,u2,u2
```

The default pattern for OS X is:

```
dns.sweep.pattern: u1,u1
```



## dns.tcp.timeout

This configuration parameter specifies the amount of time, in seconds, to wait before re-sending a TCP request, when there is no response from the current DNS server. If the current server does not respond to this request, it is considered down, which triggers a sweep to acquire a new server as specified by the `dns.sweep.pattern` parameter. The new server becomes the selected server (it is cached in memory and its address is put in `kset.dns.server`), and it attempts to handle the DNS request.

The default value is 1 second. You may specify only one TCP retry.

Specify a positive integer; for example:

```
dns.tcp.timeout: 1
```

This parameter specifies the timeout values for TCP requests. Use `dns.udp.timeouts` to specify timeout values for UDP requests.

## dns.udp.timeouts

This configuration parameter specifies the number of times to re-send a UDP request, and the number of seconds to wait for each, when there is no response from the current DNS server to a UDP request. Specify a comma separated list of values, up to three entries. If the current server does not respond to any of the requests, it is considered down, which triggers a sweep to acquire a new server as specified by the `dns.sweep.pattern` parameter. The new server becomes the selected server (it is cached in memory and its address is put in `kset.dns.server`), and it attempts to handle the DNS request.

The default value on Linux and Unix is three retries of 1, 2, and 4 seconds, respectively.

The default value on OS X is 1 second.

Specify a positive integer; for example:

```
dns.udp.timeouts: 1, 2, 4
```

This parameter specifies the timeout values for UDP requests. Use `dns.tcp.timeout` to specify timeout values for TCP requests.



## domain.dead.cache.refresh

This configuration parameter specifies how long, in seconds, to keep in cache the fact that a domain is dead (that is, the domain does not contain any live domain controllers).

The default value is 60 seconds. For example:

```
domain.dead.cache.refresh: 60
```

## domain.live.cache.refresh

This configuration parameter specifies how long, in seconds, to keep in cache the fact that a domain is alive (that is, the domain contains at least one live domain controller).

The default value is 3600 seconds (one hour). For example:

```
domain.live.cache.refresh: 3600
```

## fips.mode.enable

This configuration parameter indicates whether FIPS 140-2 compliant algorithms are used in the authentication protocols. FIPS 140-2 compliance is available for authentication using Kerberos and NTLM with the following caveats and requirements:

- FIPS mode is available on Centrify agents version 5.0.2 or later but only on specific UNIX platforms. See the NIST validation entry for the Centrify FIPS mode for the current list of supported platforms.
- Domain controllers must be at Windows 2008 domain functional level or greater. If the domain controller domain functional level does not meet the required level, `adcli` does not start and returns an error message.
- FIPS 140-2 compliance uses only the following algorithms: AES128-CTS or AES256-CTS encryption types, RSA for public key generation, DSA for digital signature generation and SHA1, SHA256, SHA384 or SHA512 for hashing.
- Inter-realm keys for the AES128-CTS or AES256-CTS encryption types must be established between any trusted domains to enable Active



Directory users to log on to a joined computer (see the `ksetup` utility to set up inter-realm keys).

- FIPS mode only allows NTLM pass-through authentication over SChannel; FIPS mode is not available for 'NTLM authentication over SMB or SMB2.

In most cases, you set this configuration parameter using group policy. As long as the UNIX computer is running a supported platform, this policy sets the `fips.mode.enable` configuration parameter to `true` and restarts `adcli`ent.

**Note:** The administrator must explicitly add the `centrifydc_fips.xml` or `centrifydc_fips.adm` group policy template on the domain controller to set `fips.mode.enable`. The template needs to be imported to just one domain controller in a forest.

If you are manually setting this parameter, the parameter value must be `true` or `false`. For example, to enable FIPS 140-2 compliant algorithms, set the following:

```
fips.mode.enable: true
```

The default is `false`.

After manually setting this parameter, you must restart `adcli`ent to enable FIPS mode.

There are several restrictions and rules governing the use of FIPS mode. For example:

- Prevalidated groups and users that use FIPS mode to log in when disconnected must have their Active Directory `msDS-SupportedEncryptionTypes` attribute set to at least `0x18` (prevalidated login for users in FIPS mode requires Kerberos AES 128- or 256-bit encryption). See [adcli.entropy.validate.allow.groups](#) and [adcli.entropy.validate.allow.users](#) for the full explanation of the Active Directory `msDS-SupportedEncryptionTypes` options.
- The value of the corresponding Windows policy (**Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Option > System Cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing**) has no effect on the Windows, Linux, UNIX, or Mac OS X computers managed through the Centrify agent. You must use the configuration parameter or the Centrify policy to enable FIPS mode.

The following configuration parameters affect `adcli`ent operation when FIPS mode is enabled:



- `adclient.krb5.keytab.clean.nonfips.enctypes`: Set this configuration parameter to `true` to have `adclient` scan the computer's keytab file and remove all non-AES encryption keys for service principal names (SPNs) during startup. (The default is `false`.)
- `adclient.krb5.permitted.encryption.types`: If you include the `arcfour-hmac-md5` encryption type in this configuration parameter AND `adclient.krb5.extra_addresses` is `true`, `adclient` generates the MD4 hash for the computer password and saves it in the keytab file.

For more information about using FIPS encryption, see the *Administrator's Guide for Linux and UNIX*.

## log

This configuration parameter defines the level of detail written to the agent log file. The log level works as a filter to define the type of information you are interested in and ensure that only the messages that meet the criteria are written to the log. For example, if you want to see warning and error messages but not informational messages, you can change the log level from `INFO` to `WARN`.

The parameter value can be `FATAL`, `ERROR`, `WARN`, `INFO`, `DEBUG`, or `TRACE`. For example:

```
log: WARN
```

You can also modify this configuration parameter to define a different logging level for specific library messages. For example:

```
log: info
log.pam: debug
```

## logger.facility.adclient

This configuration parameter defines the `syslog` facility to use for logging general `adclient` activity. You can specify separate `syslog` facilities for logging general `adclient` messages, `adclient` auditing messages, and `adnisd` messages. This parameter's value can be any valid `syslog` facility. For example, you can set this parameter to log messages to `auth`, `authpriv`, `daemon`, `security`, or `localn` facilities.

The default facility is `auth`. For example:



`logger.facility.adclient: auth`

**Note:** You can specify other process names for logging, or use an asterisk (\*) to specify the default facility to use for all agent processes. For example, you can specify `logger.facility.*: auth` in the configuration file to direct all agent processes send messages to the `auth` facility of `syslog`.

## logger.facility.adclient.audit

This configuration parameter defines the `syslog` facility to use for logging `adclient` auditing messages. You can specify separate `syslog` facilities for logging general `adclient` messages, `adclient` auditing messages, and `adnisd` messages. This parameter's value can be any valid `syslog` facility. For example, you can set this parameter to log messages to `auth`, `authpriv`, `daemon`, `security`, or `localn` facilities.

The default facility is `auth`. For example:

`logger.facility.adclient.audit: auth`

If this parameter is not defined in the configuration file, the audit messages are logged in the facility defined for the `logger.facility.adclient` parameter.

## logger.facility.diag

This configuration parameter defines the `syslog` facility to use for logging diagnostic messages. Diagnostic messages are intended to help you troubleshoot operations and trace all of the LDAP, Kerberos, NTLM and RPC messages that are generated for the following tasks:

- adjoin operations
- adleave operations
- lookup object operations
- authentication operations
- log on operations
- password change

This parameter enables you to specify a separate `syslog` facilities for logging diagnostic from the facility used to log general `adclient` messages, `adclient`



auditing messages, and `adnisd` messages. This parameter's value can be any valid `syslog` facility. For example, you can set this parameter to log messages to `auth`, `authpriv`, `daemon`, `security`, or `localn` facilities.

The default facility is `auth`. For example:

```
logger.facility.diag: auth
```

You should note that diagnostic messages are only logged if you enable logging with the `addebug` command. If the parameter is not defined in the configuration file, the messages are logged in the default facility or the facility defined for the `logger.facility.adclient` parameter.

## logger.memory.bufsize

This configuration parameter specifies the default size for the in-memory circular log buffer. The in-memory circular log buffer is only enabled if the `adclient` watchdog process is forced to restart the `adclient` process. The default parameter value is 128K. You should not manually set this parameter value in the configuration file unless you are instructed to make the setting by Centrifify Support.

## logger.memory.enabled

This configuration parameter specifies whether the in-memory circular log buffer is enabled. The in-memory log buffer should only be enabled automatically if the `adclient` watchdog process is forced to restart the `adclient` process. Therefore, the default value for this parameter is `false`. You should not manually set this parameter value in the configuration file unless you are instructed to make the setting by Centrifify Support.

## logger.memory.log

This configuration parameter specifies the default log level for the in-memory circular log buffer. The in-memory circular log buffer is only enabled if the `adclient` watchdog process is forced to restart the `adclient` process. The default value for this parameter is `DEBUG`. You should not manually set this parameter value in the configuration file unless you are instructed to make the setting by Centrifify Support.



## logger.queue.size

This configuration parameter controls the maximum number of messages that may be queued before they are sent to `syslog`. The messages in the queue are sent to `syslog` asynchronously. During normal operation, if the number of messages in the queue reaches the value set for this parameter, no new messages are added until the number of messages in the queue decreases below the maximum number you have specified.

Each message consumes about 100 bytes of storage in the message queue.

If the logging level is set to `DEBUG`, this parameter's value is automatically multiplied by a factor of 4 to allow additional messages to be logged.

The parameter value must be a positive integer. For example:

```
log.queue.size: 256
```

**Note:** Setting this parameter to zero (0) disables the message queue, and causes all log messages to be written to the `syslog` facility synchronously. In most cases, disabling the message queue degrades system performance, and in extreme cases, may cause a dead lock with the `syslog` daemon during log rotations. Therefore, Centrify recommends that you never set this parameter value to 0.

If this parameter is not defined in the configuration file, its default value is 256 KB.

If you change this parameter, you must restart the agent, `adclient`, for the change take effect.

## lrpc.connect.timeout

This configuration parameter specifies the number of seconds the NSS or PAM service should wait for a response from the agent during an initial connection attempt. If the initial connection to `adclient` takes longer than specified by this parameter, the service will time out and terminate the attempt to connect. In most cases, there's no need to modify this parameter.

The parameter value must be a positive integer. For example:

```
lrpc.connect.timeout: 5
```



If this parameter is not defined in the configuration file, its default value is 5 seconds.

## lrpc.session.timeout

This configuration parameter specifies the maximum number of seconds to keep the `adcli`ent connection open to respond to context-dependent requests, such as `pwgetent` or `lsgroup` requests. Lowering this value reduces the chance of a multi-threaded program being affected by an `adcli`ent restart, but may cause slow context-dependent commands to fail to return results because the session times out before the command completes its operation. Increasing the value of this parameter reduces the overhead of re-establishing a connection for multiple requests.

For example:

```
ldap.session.timeout: 30
```

## lrpc.timeout

This configuration parameter specifies the number of seconds the local client should wait for a response from the agent before ending a requested operation.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be an integer greater than zero. The following example sets the inactive client timeout to 5 minutes:

```
lrpc.timeout: 300
```

If this parameter is not defined in the configuration file, its default value is 5 minutes.

Although in some environments increasing or decreasing the value of this parameter may be beneficial to optimize agent operations and Active Directory for your network topology, you should take care in changing this setting. For example, in most cases, you should not decrease this value because of the potential problems it may cause when transferring data. If you set this value



too low and have a slow connection or a large amount of data to be transferred, the local client may end the operation prematurely and prevent the data transfer from completing successfully.

## secdit.system.access.lockout.allowofflinelogin

This configuration parameter specifies whether to allow users to log in when the user account is locked out and the computer is not connected to Active Directory. The default value is `false` (that is, users cannot log in). For example:

```
secdit.system.access.lockout.allowofflinelogin: false
```

You can also set this parameter using group policy.

## queueable.random.delay.interval

This configuration parameter specifies a delay, in minutes, for activities to stagger background tasks. Once defined, scheduling of those background tasks calculates a random period of time within the interval and adds the same time to the delay of those tasks. If you change the interval setting, however, the period of time is recalculated. The default setting is 0 and no delay.



# Customizing Kerberos-related configuration parameters

This chapter describes the configuration parameters that affect the operation of Kerberos-related activity on the local host computer.

adclient.dc.switch.update.krb5.conf .....	144
adclient.krb5.allow_weak_crypto .....	144
adclient.krb5.autoedit .....	145
adclient.krb5.cache.renewal.service.accounts .....	145
adclient.krb5.ccache.dir .....	146
adclient.krb5.conf.file.custom .....	147
adclient.krb5.conf.domain_realm.anysite .....	149
adclient.krb5.extra_addresses .....	149
adclient.krb5.keytab.clean.nonfips.enctypes .....	150
adclient.krb5.keytab.entries .....	150
adclient.krb5.password.change.interval .....	150
adclient.krb5.password.change.verify.interval .....	151
adclient.krb5.password.change.verify.retries .....	151
adclient.krb5.passwd_check_s_address .....	152
adclient.krb5.permitted.encryption.types .....	152
adclient.krb5.permitted.encryption.types.strict .....	153



adclient.krb5.principal .....	154
adclient.krb5.send.netbios.name .....	154
adclient.krb5.service.principals .....	155
adclient.krb5.tkt.encryption.types .....	155
adclient.krb5.tkt.encryption.type.strict .....	155
adclient.krb5.use.addresses .....	157
fips.mode.enable .....	157
krb5.cache.clean .....	158
krb5.cache.clean.exclusion .....	158
krb5.cache.clean.force.max .....	159
krb5.cache.clean.interval .....	159
krb5.cache.infinite.renewal .....	160
krb5.cache.infinite.renewal.batch.groups .....	160
krb5.cache.infinite.renewal.batch.users .....	161
krb5.cache.renew.exclusion .....	161
krb5.cache.renew.interval .....	162
krb5.conf.plugins.ccselect.disable .....	162
krb5.cache.type .....	163
krb5.conf.k5login.directory .....	164
krb5.conf.kcm.socket.path .....	164
krb5.config.update .....	165
krb5.forcetcp .....	166
krb5.forwardable.user.tickets .....	166
krb5.pac.validation .....	167
krb5.permit.dns.spn.lookups .....	167
krb5.sso.block.local_user .....	168
krb5.sso.ignore.k5login .....	168



<code>krb5.support.alt.identities</code> .....	168
<code>krb5.unique.cache.files</code> .....	169
<code>krb5.use.kdc.timesync</code> .....	169
<code>krb5.verify.credentials</code> .....	170
<code>krb5.udp.preference.limit</code> .....	170

## `adclient.dc.switch.update.krb5.conf`

The `adclient.dc.switch.update.krb5.conf` configuration parameter specifies that `adclient` updates the `krb5.conf` file immediately with the current domain controller when `adclient` switches the domain controller, such as in failover situations. This can be helpful in situations where third party applications use the `krb5.conf` file to locate the domain controllers that `adclient` uses.

By default, this parameter is true.

If this parameter is set to false, then `adclient` does not update the `krb5.conf` file with the updated domain controller information immediately but at the next update interval that's specified by the `krb5.config.update` parameter.

## `adclient.krb5.allow_weak_crypto`

This configuration parameter specifies whether to allow weak encryption types for Kerberos authentication.

By default (true), this parameter allows the weak encryption types specified in the parameters `adclient.krb5.permitted.encryption.types` and `adclient.krb5.tkt.encryption.types`.

These encryption types include: `des-cdc-crc`, `des-cbc-md4`, `des-cbc-md5`, `des-cbc-raw`, `des3-cbc-raw`, `des-hmac-sha1`, `arcfour-hmac-exp`, `rc4-hmac-exp`, and `arcfour-hmac-md5-exp`.

If you disable this parameter, the above encryption types are not supported. Note that setting this parameter to false may cause authentication failures in existing Kerberos environments that do not support strong cryptography. Users



in these environments should leave this parameter value set to “true” until their environment adopts stronger cyphers.

By default, this parameter is set to `true`.

```
adclient.krb5.allow_weak_crypto: true
```

## adclient.krb5.autoedit

This configuration parameter specifies whether the agent should automatically update the Kerberos configuration file with new information, such as domains and IP addresses, as the agent discovers this information.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be `true` or `false`. In most cases, this parameter should be set to `true` to allow the agent to maintain the configuration files automatically. For example:

```
adclient.krb5.autoedit: true
```

If this parameter is not defined in the configuration file, its default value is `true`.

## adclient.krb5.cache.renewal.service.accounts

This configuration parameter specifies which service accounts are renewed automatically.

The parameter value can be a comma-separated list of service accounts, or the name of a file that contains the list of service accounts.

For example, if you specify a file that contains the service accounts using the `file:` keyword and a file location:

```
adclient.krb5.cache.renewal.service.accounts:  
file:/etc/centrifydc/service_accts.lst
```

The default value of this parameter is `file:/etc/centrifydc/service_accts.lst` as shown in the example.



## adclient.krb5.ccache.dir

The `adclient.krb5.ccache.dir` parameter specifies the directory where Kerberos ccache files are stored when `krb5.cache.type` is `FILE`.

This is useful when kerberos applications in docker containers use the kerberos cache files. This parameter, in conjunction with `adclient.krb5.ccache.dir.secure.usable.check` enables volume bind mapping so that kerberos cache files in the host OS are available to the docker containers.

Default is empty string.

- If `adclient.krb5.ccache.dir` is not configured or set to default empty string, then:

The system default ccache directory is used. If a `default_ccache_name` exists in the `[libdefaults]` stanza of `krb5.conf`, it is removed.

- If `adclient.krb5.ccache.dir` is specified, AND `adclient.krb5.ccache.dir.secure.usable.check` is `false`, then:

The specified directory is used for the `default_ccache_name` in the `[libdefaults]` stanza of `krb5.conf`.

- If `adclient.krb5.ccache.dir` is specified, AND `adclient.krb5.ccache.dir.secure.usable.check` is `true`, BUT the kerberos cache directory is neither secure nor usable, then:

The system default ccache directory is used. If a `default_ccache_name` exists in the `[libdefaults]` stanza of `krb5.conf`, it is removed.

- If `adclient.krb5.ccache.dir` is specified, AND `adclient.krb5.ccache.dir.secure.usable.check` is `true`, AND the kerberos cache directory is secure and usable then:

The specified directory is used for the `default_ccache_name` in the `[libdefaults]` stanza of `krb5.conf`.

Note: When ccache type is `KCM`, the `klist` lists `KCM` caches and file ccaches under the system default ccache directory. If the ccache directory is changed when ccache type is `FILE`, the newly created file ccaches might not be listed when ccache type is switched to `KCM`.



## **adclient.krb5.ccache.dir.secure.usable.check**

The `adclient.krb5.ccache.dir.secure.usable.check` parameter specifies whether to perform a secure and usability check on a configured Kerberos ccache directory. Only used when `adclient.krb5.ccache.dir` set. Options are:

- `false` — Default. No action taken.
- `true` — If `adclient.krb5.ccache.dir` is configured, then `adclient.krb5.ccache.dir.secure.usable.check` checks the specified directory.

For the kerberos cache directory to be secure and usable it must meet the following criteria:

- the directory exists
- the directory is not a symlink
- the directory is root owned
- the directory is world writable and has sticky bit set

## **adclient.krb5.conf.file.custom**

This configuration parameter enables the merging of custom `krb5.conf` entries into the original `krb5.conf` file. To use this parameter, you specify the keyword `file:` and the absolute path to a syntactically valid custom `krb5.conf` file.

For example:

```
adclient.krb5.conf.file.custom: file:/etc/custom.conf
```

By default, this parameter is not enabled, and the default value is an empty string.

After you enable this parameter, when `krb5.conf` is regenerated the additional directives in the custom `krb5.conf` file are merged into the original `krb5.conf` file, and conflicting lines are discarded.

The required format of the custom `krb5.conf` file is as follows:

```
[libdefaults]
keyword1 = value1
keyword2 = value2
[domain_realm]
domain = realm
hostname = realm
```



```
[realms]
REALM1 = {
tag1 = value1
tag2 = value2
}
REALM2 = {
tag1 = value1
}
[appdefaults]
to-be-copied-as-is
[capaths]
to-be-copied-as-is
[dbdefaults]
to-be-copied-as-is
[dbmodules]
to-be-copied-as-is
[kadmin]
to-be-copied-as-is
[kdc]
to-be-copied-as-is
[kdcdefaults]
to-be-copied-as-is
[logging]
to-be-copied-as-is
[login]
to-be-copied-as-is
[otp]
to-be-copied-as-is
[password_quality]
to-be-copied-as-is
[plugins]
to-be-copied-as-is
```

When you use this parameter, the following actions take place when the `krb5.conf` file is regenerated:

- For the directives `[libdefaults]`, `[domain_realm]`, and `[realms]`, the new keyword = value pairs from the custom `krb5.conf` file are added to the corresponding directive in the original `krb5.conf` file.
- New realms from the custom `krb5.conf` file are added under `[realms]` in the original `krb5.conf` file.
- If a keyword already exists in the original `krb5.conf` file, the keyword entry from the custom file is discarded.
- For the additional sections `[appdefaults]`, `[capaths]`, `[dbdefaults]`, `[dbmodules]`, `[kaadmin]`, `[kdc]`, `[kdcdefaults]`, `[logging]`, `[login]`, `[otp]`, and `[plugins]`, the entire section from the custom file is added directly into the original `krb5.conf` file, and any existing entries in those sections in the original `krb5.conf` file are overwritten.
- Warning messages are displayed in the log for every conflict.



**Note:** The specified custom `krb5.conf` file must be owned by root.

**Note:** To use this parameter in a Mac environment, the configuration parameter `adclient.krb5.autoedit` must be set to `true`.

## `adclient.krb5.conf.domain_realm.anysite`

This configuration parameter specifies whether or not to search for all domain controllers in a kerberized realm or just the domain controllers within the current, preferred site.

If this parameter is set to `true`, then the system will list all reachable domain controllers in a kerberized realm, regardless of which site they're located in.

If this parameter is set to `false`, then only the domain controller in the current, preferred site is listed.

For example:

```
adclient.krb5.conf.domain_realm.anysite: true
```

If this parameter is not defined in the configuration file, its default value is `false`.

## `adclient.krb5.extra_addresses`

This configuration parameter specifies 0, 1, or more IP addresses. The Centrify agent adds these IP addresses to the host computer's own IP address when it makes a Kerberos authentication request that includes IP addresses. Multiple addresses accommodate authentication in a network that uses NAT.

The IP addresses in this parameter should be in dotted quad form, each address separated from the next by a comma. As an example:

```
adclient.krb5.extra_addresses: 192.68.21.189,192.68.35.2
```

adds two IP addresses to the host machine's own IP address.

Note that this configuration parameter sets the Kerberos configuration parameter `extra_addresses` in `krb5.conf`.

This parameter has no effect unless `adclient.krb5.use_addresses` is set to `true`.



If this parameter is not defined in the configuration file, its default value is empty, which defines no extra IP addresses.

## adclient.krb5.keytab.clean.nonfips.etypes

This configuration parameter specifies whether adclient scans the computer's keytab file and removes any non-AES encryption keys for service principal names during startup. The default is false.

Use this configuration parameter to remove the keys for encryption types that are not supported when you enable FIPS mode (see [fips.mode.enable](#)). To remove the non-AES keys, enter the following

```
adclient.krb5.keytab.clean.nonfips.etypes: true
```

**Note:** If you specify `arcfour-hmac-md5` in the `adclient.krb5.permitted.encryption.types` configuration parameter, the MD4 hash of the computer password is generated and saved in the keytab file.

## adclient.krb5.keytab.entries

This configuration parameter specifies the number of entries that the agent maintains in the Kerberos key table for a service principal.

This value determines the number of key versions that are kept per service principal. Its value must be a positive integer. For example:

```
adclient.krb5.keytab.entries: 3
```

If this parameter is not defined in the configuration file, its default value is 3 entries.

## adclient.krb5.password.change.interval

This configuration parameter specifies the number of days in the interval between the last Active Directory password change for the computer account and the next password change for the account. At the interval, Active Directory prompts for a new account password. The agent then automatically generates



a new password for the computer account and issues the new password to Active Directory.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be an integer equal to or greater than zero. If the value is zero, then the change interval is turned off and the account is not prompted for password change. For example:

```
adclient.krb5.password.change.interval: 28
```

If this parameter is not defined in the configuration file, its default value is 28 days.

## adclient.krb5.password.change.verify.interval

This configuration parameter controls how long adkeytab waits between attempts to verify password changes. For example, to set the interval between verification attempts to 600 seconds (10 minutes), enter the following:

```
adclient.krb5.password.change.verify.interval: 600
```

The default setting for this parameter is 300 seconds (5 minutes).

You can specify the number of password change verifications that adkeytab attempts by using the `adclient.krb5.password.change.verify.retries` configuration parameter.

## adclient.krb5.password.change.verify.retries

This configuration parameter controls how many times adkeytab tries to verify password changes running in the background.

In some Active Directory environments, such as those employing a read-only domain controller (RODC), Kerberos password changes may not be verified through `adclient` due to a replication delay. As a result of this delay, the new password is not saved to the keytab file. When this parameter is set to a value other than 0, `adclient` will retry verification of the new password a corresponding number of times.



If your RODC has latency problems, you may want to address this by setting `adkeytab` to attempt to verify password changes multiple times. For example, to direct `adkeytab` to attempt a total of 4 password change verifications, you would set this parameter to 3 as follows:

```
adclient.krb5.password.change.verify.retries: 3
```

The time between verification attempts can be set using the `adclient.krb5.password.change.verify.interval` configuration parameter.

The default setting for this parameter is 0, meaning that `adkeytab` will not try to verify password changes after the initial attempt.

## adclient.krb5.passwd\_check\_s\_address

This configuration parameter specifies whether Kerberos should ignore the source address on private messages. This setting is useful when Active Directory uses NAT.

The parameter value can be `true` or `false`. The default value for this parameter is `true`. For example:

```
adclient.krb5.passwd_check_s_address: false
```

## adclient.krb5.permitted.encryption.types

This configuration parameter specifies the types of encryption that can be used in Kerberos client credentials.

The parameter value must be one or more encryption types, separated by a space. For example:

```
adclient.krb5.permitted.encryption.types: arcfour-hmac-md5 des-cbc-md5
```

If this parameter is not defined in the configuration file, the default encryption types permitted are:

- Windows 2000 server and Windows Server 2003: `arcfour-hmac-md5`, `des-cbc-md5`, and `des-cbc-crc`.
- Windows Server 2008 domain functional level supports these additional types:  
`aes128-cts` and `aes256-cts`. Although you can specify these types in an



environment other than 2008 domain functional level, they are not useful and may cause extra network round trips during the authentication process.

## adclient.krb5.permitted.encryption.types.strict

The `adclient.krb5.permitted.encryption.types.strict` parameter controls whether to add to or replace the encryption types specified in the setting, `permitted_enctypes`, in `krb5.conf` with the encryption types specified in the setting, `adclient.krb5.permitted.encryption.types`, in `centrifdc.conf`.

- When `adclient.krb5.permitted.encryption.types.strict` is `false` (default), then:

The encryption types listed in `adclient.krb5.permitted.encryption.types` in `centrifdc.conf` are added to the list of encryption types in `permitted_enctypes` in `krb5.conf`.

This only ensures that what is specified in `centrifdc.conf` is present in `krb5.conf`. It does not remove unknown items.

- When `adclient.krb5.permitted.encryption.types.strict` is set to `true`, then:

The encryption types listed in `adclient.krb5.permitted.encryption.types` in `centrifdc.conf` replace the encryption types specified in the setting, `permitted_enctypes`, in `krb5.conf`.

The permitted encryption types in `krb5.conf` exactly match the permitted encryption types in `centrifdc.conf`. Extra or unknown encryption types are removed.

Example:

```
adclient.krb5.permitted.encryption.types.strict: false
```

- `false` — Default is `false`. No change in behavior. `permitted_enctypes` are updated from the `centrifdc.conf` file.

Items from `centrifdc.conf` are added, if they were not already listed. Other items that were already in `permitted_enctypes` are left alone and not removed.

- `true` — replace the targeted `krb5.conf` parameters so they match



exactly what is specified in `centrifdc.conf`.

Items from `centrifdc.conf` are added, if they were not already listed. Other items that were already in `permitted_encTypes`, and not in `centrifdc.conf`, are removed.

To apply changes to this parameter, either restart `adclient` or ensure the group policy is set as follows: **Computer Configuration > Centrify Settings > DirectControl Settings > Kerberos Settings > Control if strictly enforce the permitted\_encTypes**.

## `adclient.krb5.principal`

This configuration parameter specifies whether SAM account names or user principal names (UPNs) are used as the principal in Kerberos tickets. Supported values are `sam` and `upn`.

For example:

```
adclient.krb5.principal: sam
```

The default value is `sam`.

If you set this parameter to `upn` and no UPN is available, the `sAMAccountName` attribute with the format `SAMAccountName@DomainName` is used.

In MIT Kerberos environments, however, the UPN is used even if this parameter is set to `sam`.

## `adclient.krb5.send.netbios.name`

This configuration parameter specifies whether the Centrify agent sends the host computer's NetBIOS name (the computer's pre-Windows 2000 name) together with the host computer's IP address (or addresses) when the agent makes a Kerberos authentication request that includes IP addresses. The NetBIOS name appears in the domain controller log on the host Windows server and helps identify the computer making the request.

If this parameter is set to `true`, the agent sends the NetBIOS name. If set to `false`, the agent does not send the NetBIOS name.

This parameter has no effect unless `adclient.krb5.use.addresses` is set to `true`.

If this parameter is not defined in the configuration file, its default value is `true`.



## adclient.krb5.service.principals

This configuration parameter specifies additional service principals for entries in the Kerberos key table. The key table is populated by default with the service principals `host` and `ftp cifs`.

This parameter's value must be one or more principal service names, separated by a space or by a comma. For example:

```
adclient.krb5.service.principals: ldap nfs
```

If this parameter is not defined in the configuration file, no additional principal names are added to the Kerberos key table.

## adclient.krb5.tkt.encryption.types

This configuration parameter specifies the types of encryption that can be presented to the server in the TGT when the computer is requesting service tickets.

The parameter value must be one or more encryption types, separated by a space. For example:

```
adclient.krb5.tkt.encryption.types: arcfour-hmac-md5 des-cbc-md5
```

If this parameter is not defined in the configuration file, the default encryption types permitted are:

- Windows 2000 server and Windows Server 2003: `arcfour-hmac-md5`, `des-cbc-md5`, and `des-cbc-crc`.
- Windows Server 2008 domain functional level supports these additional types:  
`aes128-cts` and `aes256-cts`.  
Although you can specify these types in an environment other than 2008 domain functional level, they are not useful and may cause extra network round trips during the authentication process.

## adclient.krb5.tkt.encryption.type.strict

The `adclient.krb5.tkt.encryption.type.strict` parameter controls whether to replace the encryption types set in `default_tgs_enctypes` and



`default_tkt_encTypes` in `krb5.conf` with the encryption types specified in `adclient.krb5.tkt.encryption.types` in `centrifydc.conf`.

- When `adclient.krb5.tkt.encryption.type.strict` is `false` (default), then:

The encryption types listed `adclient.krb5.tkt.encryption.types` in `centrifydc.conf` are added to the list of encryption types in `default_tgs_encTypes` and `default_tkt_encTypes` in `krb5.conf`.

This only ensures that what is specified in `centrifydc.conf` is present in `krb5.conf`. It does not remove unknown items.

- When `adclient.krb5.tkt.encryption.type.strict` is set to `true`, then:

The encryption types listed in `adclient.krb5.tkt.encryption.types` in `centrifydc.conf` replace the encryption types specified in the settings, `default_tgs_encTypes` and `default_tkt_encTypes`, in `krb5.conf`.

The permitted encryption types in `krb5.conf` exactly match the permitted encryption types in `centrifydc.conf`. Extra or unknown encryption types are removed.

Example:

`adclient.krb5.tkt.encryption.type.strict: false`

- `false` — Default is `false`. No change in behavior. `default_tgs_encTypes` and `default_tkt_encTypes` are updated from the `centrifydc.conf` file.

Items from `centrifydc.conf` are added, if they were not already listed. Other items that were already in `default_tgs_encTypes` and `default_tkt_encTypes` are left alone and not removed.

- `true` — Replace the targeted `krb5.conf` parameters so they match **exactly** what is specified in `centrifydc.conf`.

Items from `centrifydc.conf` are added, if they were not already listed. Other items that were already in `default_tgs_encTypes` and `default_tkt_encTypes`, and not in `centrifydc.conf`, are removed.

To apply changes to this parameter, either restart `adclient` or ensure the group policy is set as follows: **Computer Configuration > Centrify Settings > DirectControl Settings > Kerberos Settings > Control if strictly enforce the encTypes**.



## adclient.krb5.use.addresses

This configuration parameter controls whether the Centrify agent should send the host computer's local IP address (or addresses) to the Windows domain controller as part of a Kerberos authentication request. When set to `true`, the agent sends the IP addresses; when set to `false`, the agent does not send the IP addresses.

When the agent sends the host computer's IP address with a Kerberos request, the IP address appears in the Windows event logs associated with the request.

This configuration parameter works with the parameters `adclient.krb5.extra_addresses` and `adclient.krb5.send.netbios.name`. Use the first of these two parameters to add additional IP addresses to the host computer's IP address (useful in networks using NAT). Use the second to add the host computer's NetBIOS name to the IP address (or addresses) (useful for identifying the requesting computer in event logs).

If `adclient.krb5.use.addresses` is set to `false`, neither of these two parameters has any effect because the agent does not send addresses with an authentication request.

**Note:** This configuration parameter sets the Kerberos configuration parameter `noaddresses` in `krb5.conf`. Setting `adclient.krb5.use.addresses` to `true` sets `noaddresses` to `false`; setting `adclient.krb5.use.addresses` to `false` sets `noaddresses` to `true`.

If `adclient.krb5.use.addresses` is not defined in the configuration file, its default value is `false`.

## fips.mode.enable

This configuration parameter specifies whether Kerberos uses the algorithms in the FIPS 140-2 compliant library to sign and seal messages. See `fips.mode.enable` for the description.



## krb5.cache.clean

This configuration parameter specifies whether Kerberos credentials in the cache should be deleted when a user logs out. By default, credentials stored in the Kerberos cache that belong to users who are not logged in are periodically deleted.

To keep the credentials available in the cache use this parameter to turn off the cache clean process entirely. Alternatively, use the `krb5.cache.clean.exclusion` to turn off cache cleaning for specific users.

This configuration parameter allows you to control this operation specifically for zone users or for all users.

The parameter value must be one of the following valid settings:

- `off` to turn off the deletion of the credentials cache for all users.
- `cdc` to remove all of the `/tmp/krb5cc*` files created by the agent (`adcli`) that belong to any user not found in the `utmp` database (that is, the user has logged out).
- `all` to remove all of the `/tmp/krb5cc*` files that belong to any user not found in the `utmp` database. This setting removes files created by the agent (`adcli`), `telnet`, and `openssh`.

For example, to remove the credentials cache for all users when they log out:

```
krb5.cache.clean: all
```

The default value for this parameter is `cdc`.

## krb5.cache.clean.exclusion

This configuration parameter specifies a list of users whose credentials in the Kerberos cache will not be deleted during a periodic Kerberos cache clean-out of unlogged-in users.

Each user is specified by the user's UNIX name. Separate the names in the list using a comma.

For example, to specify that three users be excluded from periodic credential clean-up:

```
krb5.cache.clean.exclusion: admin,paula,jeffrey
```



This parameter is useful in a batch processing environment where a logged-out user may leave behind running processes that require Kerberos credentials. It allows some users' credentials to remain for processes while cleaning out all other users' credentials.

The default value for this parameter is empty.

## krb5.cache.clean.force.max

This configuration parameter controls whether adclient deletes credentials from the Kerberos cache if they are the specified number of days old.

If you activate this parameter, the credentials will be cleared for all users whether or not they are logged on, have active processes running, or are specified in the following lists:

```
krb5.cache.clean.exclusion  
krb5.cache.infinite.renewal.batch.users  
krb5.cache.infinite.renewal.batch.groups
```

For example, to force adclient to clear the cache of credentials that were authenticated 6 days previously:

```
krb5.cache.clean.force.max: 6
```

The default value for this parameter is 0, which means that this configuration parameter will not clear the credential cache for any users.

## krb5.cache.clean.interval

This configuration parameter specifies how frequently in minutes to check the Kerberos cache for credentials that belong to users who are not logged on. If the user is not logged on, the credentials are deleted.

The parameter value should be a positive integer. Setting this parameter to zero disables periodic clean-up of the cache.

For example, to set the clean-up interval to 5 minutes:

```
krb5.cache.clean.interval: 5
```

The default value for this parameter deletes the credential cache for users who have logged off every one minute.



## krb5.cache.infinite.renewal

This configuration parameter specifies whether you want user credentials to be automatically reissued when they expire. The parameter value can be set to `true` or `false`. If you set this parameter to `true`, the agent keeps a hash of the user's password in memory indefinitely. If you set this parameter to `false`, a user's credentials periodically expire and the user must be re-authenticated by re-entering a valid password.

If you set this parameter to `true`, user credentials are automatically reissued, as needed, as long as the `adcli` process continues to run even if the computer is disconnected from Active Directory. If you stop or restart `adcli`, however, the user's password hash is removed from memory. After stopping or restarting `adcli`, users must be re-authenticated by logging on with a valid user name and password.

The default parameter value is `false`. For example:

```
krb5.cache.infinite.renewal: false
```

## krb5.cache.infinite.renewal.batch.groups

This configuration parameter specifies a list of Active Directory groups whose members' Kerberos credentials require infinite renewal even after the users have logged out.

Requirements to use this parameter:

- Specified groups must be Active Directory groups.
- Groups do not need to be zone enabled.
- To have their credentials automatically renewed, users in the group must:
  - Be zone enabled (that is, mapped users are not supported).
  - Log into the desired system once using the Account Password.

You must use the following format to specify group names:

```
SamAccountName@domain
```

For example:

```
krb5.cache.infinite.renewal.batch.groups: test_group_
sam@example.com
```

By default, this parameter does not list any groups.



You can also use group policy to set this parameter.

## krb5.cache.infinite.renewal.batch.users

This configuration parameter specifies a list of users whose Kerberos credentials require infinite renewal even after the users have logged out.

Requirements to use this parameter:

- The users must be zone enabled (that is, mapped users are not supported).
- The users must log into the desired system once using the Account Password.

You can use any of the following formats to specify user names:

```
unixName  
userPrincipleName  
SamAccountName  
SamAccountName@domain
```

For example:

```
krb5.cache.infinite.renewal.batch.users: test_user, test_  
user@example.com, test_user_sam, test_user_sam@example.com
```

By default, this parameter does not list any users.

You can also use group policy to set this parameter.

## krb5.cache.renew.exclusion

This configuration parameter specifies a list of UNIX users for whom you don't want adclient to automatically renew their Kerberos credential caches. This parameter is useful in situations where you need to directly manage certain users' Kerberos caches.

Specify each user by the user's UNIX name. Separate the names in the list using a comma.

For example, to specify that adclient doesn't renew these three users' Kerberos credential caches:

```
krb5.cache.renew.exclusion: admin,paula,jeffrey
```



Alternatively, you can use the `file:` keyword to specify a separate file that contains UNIX user names.

For example:

```
krb5.cache.renew.exclude: file:/etc/centrifydc/renew.exclude
```

You can put a UNIX user name in each single line, and be sure to run the `adreload` command after modifying the file to have the changes take effect.

The default value for this parameter is empty.

## krb5.cache.renew.interval

This configuration parameter specifies, in hours, how often to renew the Kerberos credentials stored in the cache for users who have logged on successfully. Because Kerberos tickets expire after a set period of time, you can use this configuration parameter to periodically renew the existing Kerberos ticket to keep existing credentials valid.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be a positive integer. A value of zero disables renewal. For example, to set the renewal interval to 8 hours:

```
krb5.cache.renew.interval = 8
```

If this parameter is not defined in the configuration file, its default value is 4 hours. The default value of 4 hours allows two attempts at renewal over a typical Kerberos ticket lifespan of 10 hours. If possible, you should allow enough time for at least two renewal attempts if you reset the value to something other than the 4-hour default.

## krb5.conf.plugins.ccselect.disable

This configuration parameter controls whether `adclient` disables the Kerberos built-in `ccselect` plugins.

If you set this parameter to `false`, the plugins will not be disabled.

For example,



`krb5.conf.plugins.ccselect.disable: false`

By default, this parameter is set to `true`, and the built-in `ccselect` plugins are disabled.

You can also set this parameter using group policy.

## krb5.cache.type

This configuration parameter specifies the type of Kerberos credential cache that the agent (`adcli`) creates when an Active Directory user logs in. The parameter value can be set to `FILE` or `KCM`.

**Note:** The use of in-memory credential caches such as `KCM` is not supported on Mac OS X computers. In Mac OS X environments, credential caches are file-based, and setting this parameter has no effect.

If you set this parameter to `FILE`, the agent creates a file-based credential cache for each Active Directory user in `/tmp` when the user logs in. A file-based credential cache persists until the file is deleted.

If you set this parameter to `KCM`, the agent creates an in-memory credential cache for each Active Directory user when the user logs in. The `Centrify-KCM` service, run as root, manages in-memory credential caches. When the agent, `adcli`, starts up, if the parameter is set to `KCM`, `adcli` starts the `KCM` service. If you change the parameter from `FILE` to `KCM` while `adcli` is running, `adcli` starts the `KCM` service the next time it is forced to reload configuration parameters, for example, if you run the `adreload` command or if a user opens a new session.

Setting this parameter affects new users only — not users who have already logged in. For example, if you change from a file-based, to an in-memory credential cache, the agent will continue to use the file-based credential cache for any user who was logged in at the time of the change. If a logged in user opens a new session, or a new user logs in, the agent will use an in-memory cache for them.

An in-memory credential cache ends as soon as the `Centrify-KCM` service is stopped.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.



The default parameter value is `FILE`, which specifies a file-based credential cache. To specify an in-memory credential cache, set the value to `KCM`. For example:

```
krb5.cache.type: KCM
```

## krb5.conf.k5login.directory

Use this policy to specify an alternative location for user `.k5login` files.

If specified, this string value will be used for the `k5login_directory` in the `[libdefaults]` stanza in `krb5.conf` and the user's `.k5login` file will be named as `<k5login_directory>/<unix_name>`.

For security reasons the specified directory should be owned by `root` and writeable by `root` only. If the directory does not exist, `adcli` will create it.

## krb5.conf.kcm.socket.path

The `krb5.conf.kcm.socket.path` parameter specifies an alternate socket path for the `KCM` server. It applies when `krb5.cache.type` is `KCM`.

This is useful, as it allows you to configure an alternative `kcm` socket path, for example, `/var/centrifydc`. Using an alternative socket path then allows the socket to be shared between `docker` hosts and `docker` containers. It requires `adreload` after a change in value.

- When the parameter is an empty string (default), the default path `/var/run/.centrify-kcm-socket` is used.
- When the parameter is set to a non-empty string AND `krb5.conf.kcm.socket.path.secure.usable.check` is `false`, then this socket path is used without secure and usable check.
- When the parameter is set to a non-empty string AND `krb5.conf.kcm.socket.path.secure.usable.check` is `true`, then the configured socket path is checked to see if it is valid:
  - If the socket path is valid, this configured socket path is used.
  - If the socket path is not valid, the default socket path, `/var/run/.centrify-kcm-socket`, is used.



To change the socket path:

1. In `centrifydc.conf`, set `krb5.conf.kcm.socket.path` to a valid path.
2. If the configured kcm socket path is not secure, but you still want to use it, ensure the parameter, `krb5.conf.kcm.socket.path.secure.usable.check`, is `false`.
3. Run `adreload`.

## **krb5.conf.kcm.socket.path.secure.usable.check**

The `krb5.conf.kcm.socket.path.secure.usable.check` parameter specifies whether to perform a secure and usable check on the alternate socket path for the KCM server. This parameter works in conjunction with `krb5_conf_kcm_socket_path`. Options are:

- `false` — Default. No action taken.
- `true` — If `krb5.conf.kcm.socket.path` is configured, then `krb5.conf.kcm.socket.path.secure.usable.check` checks the specified directory.

A socket path is valid when it meets the following criteria:

- the parent directory exists
- the parent directory is not a symlink
- the parent directory is writable by root only
- the socket path does not exist, or it exists but it is not directory

## **krb5.config.update**

This configuration parameter specifies, in hours, how frequently the agent updates the Kerberos configuration file.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If `adclient.krb5.autoedit` is set to `false`, this parameter has no effect. If `adclient.krb5.autoedit` is set to `true`, this parameter value must be a positive integer. For example, to set the update interval to 8 hours:



`krb5.config.update: 8`

If this parameter is not defined in the configuration file, its default value is 8 hours.

## `krb5.forcetcp`

This configuration parameter specifies whether to allow Kerberos requests to use UDP or to force all Kerberos requests to use TCP.

If `krb5.forcetcp` is set to `false`, Kerberos requests may use UDP. If `krb5.forcetcp` is set to `true`, all Kerberos requests use TCP only.

In most cases, you set this configuration parameter using group policy.

You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If this parameter is not defined in the configuration file, its default value is `true`.

## `krb5.forwardable.user.tickets`

This configuration parameter specifies whether you want the agent to create forwardable Kerberos user tickets. Creating a forwardable ticket allows a user's logon ticket to be sent to another computer and used to access to additional systems and resources. For example, if a user logs on and is authenticated on one computer, then uses a Kerberized telnet session to connect to a second computer, a forwarded ticket allows the user to access to additional Kerberized resources from that second computer without separate authentication.

In most environments, forwarding user tickets is a safe practice. However, if you do not want tickets to be forwarded, you can use this parameter to prevent the agent from creating forwardable tickets.

The parameter value should be 1 if you want to allow ticket forwarding or 0 if you want to prevent ticket forwarding. For example, if you want the agent to create forwardable user tickets:

`krb5.forwardable.user.tickets: 1`

If this parameter is not defined in the configuration file, its default value is 1 (yes).



## krb5.pac.validation

This configuration parameter specifies whether or not to verify that the user's PAC (Privilege Authorization Certificate) information is from a trusted KDC (Key Distribution Center) so as to prevent what's referred to as a "silver ticket" attack.

When performing credential verification, a service ticket is fetched for the local system. After the credential is verified, the local system uses the PAC information in the service ticket.

This setting take effect when `krb5.verify.credentials` is enabled or when DirectControl is using the user's PAC from a service ticket. This setting does not apply to retrieving the PAC by way of the S4U2Self protocol.

There are 3 possible values for `krb.pac.validation`:

- `disabled` (default): NO PAC validation will be done at all.
- `enabled`: If PAC Validation fails, the PAC information is used and the user login is allowed.
- `enforced`: If PAC Validation fails, the PAC information is discarded and the user login is denied.

Setting this parameter to `enabled` or `enforced` will have significant impact on the user login and user's group fetch performance.

For example:

```
krb5.pac.validation: disabled
```

If this parameter is not defined in the configuration file, its default value is `disabled`.

## krb5.permit.dns.spn.lookups

This configuration parameter specifies whether you want to permit the agent to look up service principal names (SPN) using DNS. In most cases, you should set this parameter to `false` to ensure the security of the system. You should only set this configuration parameter to `true` if you can safely rely on DNS for security and want to use programs that use the Centrify Kerberos libraries to access a computer using an IP address or localhost.

For example:



```
krb5.permit.dns.spn.lookups: false
```

If this parameter is not defined in the configuration file, its default value is `false`.

## krb5.sso.block.local\_user

This configuration parameter specifies whether single sign-on (SSO) is permitted for local users, or if only zone-enabled Active Directory users are allowed to log in through SSO.

By default, this parameter is set to `true`, and the user UNIX name is checked against the `nss.ignore.user` list. If the UNIX name is in the list, the user is considered a local user, and SSO is not allowed. In this situation, the user must enter the local user password to log in.

If this parameter is set to `false`, local users are allowed to log in through SSO.

For example:

```
krb5.sso.block.local_user: true
```

## krb5.sso.ignore.k5login

This configuration parameter specifies whether the `k5login` module should ignore `.k5login` for SSO.

The default value is `false`.

## krb5.support.alt.identities

This configuration parameter specifies whether the agent uses the Kerberos `altSecurityIdentities` name for user authentication (`true`) or not (`false`) instead of the Windows user name, regardless of which names are supplied.

Using `altSecurityIdentities` for authentication works as long as the alternate name is always used or the passwords are synchronized, and if the third-party key distribution center (KDC) is reachable. If these two conditions aren't met, you can disable the feature by setting this parameter to `false`. In that case, the agent uses only Windows to authenticate the user and ignores any Kerberos `altSecurityIdentities`.



For example:

```
krb5.support.alt.identities: false
```

If this parameter is not defined in the configuration file, its default value is `true`.

## krb5.unique.cache.files

This configuration parameter specifies whether to generate a unique ticket cache file name for each Kerberos authentication for a given user (except the first). The unique ticket cache file name takes the following form:

```
krb5cc_cdc<uid>_XXXXXX
```

The `<uid>` is the users Unix ID, and the `XXXXXX` is a unique set of characters (i.e. `krb5cc_cdc512_u0PSdt`). This allows a given user to log on more than once, without subsequent logoffs interfering with other logon instances.

If this parameter is set to `false`, the ticket cache filename takes the following form:

```
krb5cc_<UID>
```

With this parameter set to `false`, old versions of the ticket cache file are overwritten. If a user logs in twice, the first logout causes the file to be deleted, leaving the other logon instance without a credential cache.

The environment variable `KRB5CCNAME` is populated with the generated name.

The default value is `true`, except on macOS where it is `false`.

## krb5.use.kdc.timesync

This configuration parameter enables Kerberos to automatically correct for a time difference between the system clock and the clock used by the KDC. You only need to set this parameter if your system clock is drifting and the system is not using NTP and adclient SNTP settings.

In most cases, you set this configuration parameter using group policy.

You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

For example:

```
krb5.use.kdc.timesync: true
```



If this parameter is not defined in the configuration file, its default value is `false`.

## krb5.verify.credentials

This configuration parameter specifies whether to perform a spoofing check to verify a TGT for the local system.

By default, the agent verifies a user's TGT by retrieving and verifying a service ticket for the local system. This check is done to prevent a well-known attack (the Zandarotti or screen-saver attack) whereby a rogue KDC could respond to the agent's request for the user's TGT.

However, the spoofing check can be time consuming, so you can set this parameter to `false` to disable the spoofing check and significantly improve authentication performance.

For example, to disable the check:

```
krb5.verify.credentials: false
```

If this parameter is not defined in the configuration file, the default value is `true`.

## krb5.udp.preference.limit

This configuration parameter sets the maximum size packet that the Kerberos libraries will attempt to send over a UDP connection before retrying with TCP. If the packet size is larger than this value, only TCP will be tried. If the value is set to 1, TCP will always be used. The hard UDP limit is 32700. Values larger than this are ignored and the UDP hard limit is enforced.

This key only takes effect if `krb5.forcetcp` is set to `false`.

If `krb5.forcetcp` is `true`, and the agent is managing the `krb5.conf` file, it will set `udp_preference_limit = 1`, so that the Kerberos libraries will always use TCP.

In most cases, you set this configuration parameter using group policy to set a specific value.

You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.



If this parameter is not defined in the configuration file, the default value is 1465; for example:

```
krb5.udp.preference.limit:1465
```



# Customizing PAM-related configuration parameters

This section describes the configuration parameters that affect the operation of PAM-related activity on the local host computer.

Configuring PAM-related parameters on IBM AIX computers .....	174
pam.account.conflict.both.mesg .....	175
pam.account.conflict.name.mesg .....	176
pam.account.conflict.uid.mesg .....	176
pam.account.disabled.mesg .....	177
pam.account.expired.mesg .....	177
pam.account.locked.mesg .....	177
pam.adclient.down.mesg .....	178
pam.allow.groups .....	178
pam.allow.override .....	180
pam.allow.password.change .....	181
pam.allow.password.change.mesg .....	182
pam.allow.password.expired.access .....	182
pam.allow.password.expired.access.mesg .....	182
pam.allow.users .....	183
pam.auth.create.krb5.cache .....	184
pam.auth.failure.mesg .....	185
pam.config.program.check .....	185



pam.create.k5login .....	185
pam.deny.change.shell .....	186
pam.deny.groups .....	186
pam.deny.users .....	188
pam.homedir.create .....	189
pam.homedir.create.mesg .....	190
pam.homedir.perms .....	190
pam.homedir.perms.recursive .....	190
pam.homeskel.dir .....	190
pam.ignore.users .....	191
pam.mapuser.username .....	192
pam.mfa.program.ignore .....	193
pam.ntlm.auth.domains .....	194
pam.password.change.mesg .....	195
pam.password.change.required.mesg .....	195
pam.password.confirm.mesg .....	196
pam.password.empty.mesg .....	196
pam.password.enter.mesg .....	196
pam.password.expiry.warn .....	197
pam.password.expiry.warn.mesg .....	197
pam.password.new.mesg .....	197
pam.password.new.mismatch.mesg .....	198
pam.password.old.mesg .....	198
pam.policy.violation.mesg .....	198
pam.setcred.respect.sufficient .....	199
pam.setcred.support.refresh .....	199
pam.setcred.support.reinitialize .....	199



pam.sync.mapuser .....	200
pam.uid.conflict .....	201
pam.workstation.denied.mesg .....	201
microsoft.pam.privilege.escalation.enabled .....	202

## Configuring PAM-related parameters on IBM AIX computers

On IBM AIX computers, the PAM configuration parameters described in this chapter apply to the AIX Loadable Authentication Module (LAM) or to the PAM interface. If you have configured the AIX computer to use the PAM interface, the configuration parameters apply to the PAM settings. If the AIX computer is configured to use the LAM interface, the parameters configure LAM settings, as applicable. For more information about AIX-specific configuration parameters, see the [Customizing AIX configuration parameters](#).

### Controlling access to AIX computers

On most computers, the predefined `login-all` PAM access right is required to allow users who are assigned the UNIX Login role to log on and use PAM-enabled applications in the zones they have permission to access. However, if you have AIX computers that are configured to use the Loadable Authentication Module (LAM) instead of PAM in a zone, users will be able to log on even if they have not been assigned the UNIX Login role. In addition, if you define your own custom PAM access rights, those rights will not be applicable on AIX computers that use the LAM interface.

To prevent users from logging on to or using unauthorized applications on AIX computers in a zone, you can explicitly allow or deny access to specific users and groups through configuration parameters or group policies or change the configuration of your AIX computers to use the more commonly supported Pluggable Authentication Module (PAM) interface. For more information about controlling access, see [Enforcing access rights on AIX computers](#).



## Explicitly allowing and denying access

If you have AIX computers that use the Loadable Authentication Module (LAM) interface, you cannot use the predefined `login-all` PAM access right or custom PAM access rights to authorize who can log on and who can use specific applications. Therefore, the default UNIX Login role does not apply on AIX computers that use the LAM interface. If you are primarily concerned with who can log on to those computers, you can use the `pam.allow.groups`, `pam.allow.users`, or both parameters to explicitly specify the groups and users who can log on to AIX computers that use the LAM interface. All other groups and users—including those assigned the UNIX Login role—will be denied access. Alternatively, you can use the `pam.deny.users`, `pam.deny.groups`, or both parameters to explicitly specify the users and groups who are not allowed to log on.

## Changing the configuration of AIX computers

By default, AIX computers are configured to use the Loadable Authentication Module (LAM) instead of the Pluggable Authentication Module (PAM) subsystem. If you want to be able to use the default or custom PAM access rights to authorize access to specific applications, you might want to reconfigure your AIX computers to use the PAM interface instead of the LAM interface. If you choose to reconfigure AIX computers, you should also be sure to replace the `openssh` package for LAM with the `openssh` for PAM and thoroughly test your applications.

## `pam.account.conflict.both.mesg`

This configuration parameter specifies the message displayed if both user name and user ID conflicts are detected during login; that is, there are two local account conflicts. For example, a local user (`user2`) and the Active Directory user (`user1`) have the same UID (10001) but different user names, and another local account has the same user name (`user1`) as the Active Directory user but has a different UID value (10002):

```
user1 10001 #AD User
user1 10002 #local user
user2 10001 #local user
```



When the message is displayed, the %s token in the message string is replaced with the name of the first conflicting local account, and the %d token is replaced with the UID of the second conflicting local account. The message string you define must contain exactly one %s token and exactly one %d token, in that order, and no other string replacement (%) characters.

For example:

```
pam.account.conflict.both.mesg: \  
Accounts with conflicting name (%s) and UID (%d) exist locally
```

For more information about displaying a warning when local conflicts are detected, see [pam.uid.conflict](#).

## pam.account.conflict.name.mesg

This configuration parameter specifies the message displayed if a user name conflict is detected during login; that is, if there is a local user with the same name but a different UID than the Active Directory user logging on; for example,

```
user1 10001 #local user  
user1 10002 #AD user
```

When the message is displayed, the %s token in the message string is replaced with the name of the conflicting local account. The message string you define must contain exactly one %s token, and no other string replacement (%) characters.

For example:

```
pam.account.conflict.name.mesg: \  
Accounts with conflicting name (%s) exist locally
```

For more information about displaying a warning when local conflicts are detected, see [pam.uid.conflict](#).

## pam.account.conflict.uid.mesg

This configuration parameter specifies the message displayed if a user identifier (UID) conflict is detected during login; that is, if there is a local user with a different user name but the same UID as the Active Directory user logging on. For example:



```
user1 10001 #local user
user2 10001 #AD user
```

When the message is displayed, the %d token is replaced with the UID of the conflicting local account. The message string you define must contain exactly one %d token, and no other string replacement (%) characters.

For example:

```
pam.account.conflict.uid.mesg: \
Account with conflicting UID (%d) exists locally
```

For more information about displaying a warning when local conflicts are detected, see [pam.uid.conflict](#).

## pam.account.disabled.mesg

This configuration parameter specifies the message displayed if a user attempting to log on is denied access because the user's account has been disabled in Access Manager or Active Directory Users and Computers.

For example:

```
pam.account.disabled.mesg: Account cannot be accessed at this
time. Please contact your system administrator.
```

## pam.account.expired.mesg

This configuration parameter specifies the message displayed if a user attempting to log on is denied access because the user's account has expired.

For example:

```
pam.account.expired.mesg:
Account cannot be accessed at this time. Please contact your
system administrator.
```

## pam.account.locked.mesg

This configuration parameter specifies the message displayed if a user account is locked because of too many failed login attempts.

For example:

```
pam.account.locked.mesg: Account locked
```



**Note:** These messages may not be displayed depending on the login method, the daemon version, or the version of the operating system. (Ref: CS-16710c )

## pam.adclient.down.mesg

This configuration parameter specifies the message displayed during password change if user is a local UNIX user that's mapped to an Active Directory account, and the Centrify agent (`adclient`) is not accessible.

For example:

```
pam.adclient.down.mesg: (Unable to reach Active Directory - using local account)
```

In most cases, you set this configuration parameter by selecting **Enabled** and specifying the message to be displayed.

## pam.allow.groups

This configuration parameter specifies the groups allowed to access PAM-enabled applications. When this parameter is defined, only the listed groups are allowed access. All other groups are denied access.

**Note:** This parameter does not support cross-forest groups. (Ref: CS-18659a)

If you want to use this parameter to control which users can log in based on group membership, the groups you specify should be valid Active Directory groups, but the groups you specify do not have to be enabled for UNIX. Local group membership and invalid Active Directory group names are ignored.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you use this parameter to control access by group name, the agent checks the Active Directory group membership for every user who attempts to use PAM-enabled applications on the host computer.

When a user attempts to log on or access a PAM-enabled service, the `pam_centri_fydc` module checks with Active Directory to see what groups the user



belongs to. If the user is a member of any Active Directory group specified by this parameter, the user is accepted and authentication proceeds. If the user is not a member of any group specified by this parameter, authentication fails and the user is rejected.

The parameter's value can be one or more group names, separated by commas, or the `file:` keyword and a file location. For example, to allow only members of the `administrators`, `sales`, and `engineering` groups in Active Directory to log in:

```
pam.allow.groups: administrators,sales,engineering
```

You can use the short format of the group name or the full canonical name of the group.

To enter group names with spaces, enclose them in double quotes; for example:

```
pam.allow.groups: "domain admins",sales,"domain users"
```

To specify a file that contains a list of the groups allowed access, type the path to the file:

```
pam.allow.groups: file:/etc/centrifydc/groups.allow
```

If no group names are specified, no group filtering is performed.

If you make changes to this parameter, you should run `adflush` to clear the cache to ensure your changes take effect.

## Specifying group names for computers joined to Auto Zone

If a computer is configured to use the Auto Zone instead of a specific zone, you should specify group names using the format defined by the `auto.schema.name.format` parameter. For example the `auto.schema.name.format` parameter can be set to the following:

- SAM (default) uses the `samAccountName` attribute for the group—`web_qa`
- `SAM@domainName` uses the `samAccountName@domain_name` format—`web_qa@acme.com`
- NTLM uses the NTLM format and separator defined for `adclient.ntlm.separators`—`acme.com+web_qa`

You can look in the `centrifydc.conf` configuration file for the value of `auto.schema.name.format`, or run `adedit` or `adquery` commands to see the UNIX name for any group. For example, to see the UNIX name for the `web_qa`



Active Directory group when the `auto.schema.name.format` parameter is set to SAM, you can execute a command similar to this to return the UNIX group profile name:

```
adquery group -n web_qa
webqa.us
```

## pam.allow.override

This configuration parameter is used to override authentication through Active Directory to ensure the root user or another local account has permission to log on when authentication through Active Directory is not possible, when there are problems running the `adcli` process, or when there are network communication issues.

When you specify a user account for this parameter, authentication is passed on to a legacy authentication mechanism, such as `/etc/passwd`. You can use this parameter to specify an account that you want to ensure always has access, even if communication with Active Directory or the `adcli` process fails. For example, to ensure the local root user always has access to a system even in an environment where you have enabled root mapping, you can specify:

```
pam.allow.override: root
```

To log in locally with the override account, you must specify the local user name and password. However, because the account is mapped to an Active Directory account, you must append `@localhost` to the user name. For example, if you have specified `root` as the override account and are using root mapping, you would type `root@localhost` when prompted for the user name. You can then type the local password for the root account and log in without being authenticated through Active Directory.

**Note:** If you are mapping the root user to an Active Directory account and password, you should set this parameter to `root` or to a local user account with root-level permissions (UID 0), so that you always have at least one local account with permission to access system files and perform privileged tasks on the computer even if there are problems with the network connection, Active Directory, or the `adcli` process.



**Note:** If you are using a Solaris machine with the Name Switch Cache Daemon (NSCD) running, you will not be able to log in as an override user using `<username>@localhost`. (Ref: CS-29816c)

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

**Note:** The `pam.allow.override` configuration parameter is not supported on AIX computers. This is because using the user name with the suffix `@localhost` is not supported on AIX. The `LAMGetEntry` call that is used to get user information and extended attribute information does not support login name changes. So, the login fails as there is no way to find the user or authenticate the user. There is no equivalent setting for AIX computers. (Ref: CS-33506a )

## pam.allow.password.change

This configuration parameter specifies whether users who log in with an expired password should be allowed to change their password. You can set this parameter to `true` or `false` and use it in conjunction with the `pam.allow.password.expired.access` parameter to control access for users who attempt to log on with an expired password.

If both this parameter and `pam.allow.password.expired.access` are set to `true`, users logging on with an expired password are allowed to log on and are prompted to change their password.

If the `pam.allow.password.expired.access` parameter is set to `true`, but this parameter is set to `false`, users logging on with an expired password are allowed to log on but are not prompted to change their password and the message defined for the `pam.allow.password.change.msg` parameter is displayed.

If both this parameter and `pam.allow.password.expired.access` are set to `false`, users who attempt to log on with an expired password are not allowed to log on or change their password and the message defined for the `pam.allow.password.change.msg` parameter is displayed.

For example, to allow users with expired passwords to change their password:

```
pam.allow.password.change: true
```



## pam.allow.password.change.msg

This configuration parameter specifies the message displayed when users are not permitted to change their expired password because the `pam.allow.password.change` parameter is set to `false`.

For example:

```
pam.allow.password.change.msg: Password change not permitted
```

## pam.allow.password.expired.access

This configuration parameter specifies whether users who log in with an expired password should be allowed access. You can set this parameter to `true` or `false` and use it in conjunction with the `pam.allow.password.change` parameter to control access for users who attempt to log on with an expired password.

If this parameter is set to `true`, users logging on with an expired password are allowed to log on, and either prompted to change their password if the `pam.allow.password.change` parameter is set to `true`, or notified that they are not allowed to change their expired password if the `pam.allow.password.change` parameter is set to `false`.

If this parameter is set to `false`, users logging on with an expired password are not allowed to log on and the message defined for the `pam.allow.password.expired.access.msg` parameter is displayed.

For example, to allow users with expired passwords to log on:

```
pam.allow.password.expired.access: true
```

## pam.allow.password.expired.access.msg

This configuration parameter specifies the message displayed when users are not permitted to log on with an expired password because the `pam.allow.password.expired.access` parameter is set to `false`.

For example:

```
pam.allow.password.expired.access.msg: Password expired - access denied
```



## pam.allow.users

This configuration parameter specifies the users who are allowed to access PAM-enabled applications. When this parameter is defined, only the listed users are allowed access. All other users are denied access.

If you want to use this parameter to control which users can log in, the users you specify should be valid Active Directory users that have a valid UNIX profile for the local computer's zone. If you specify local user accounts or invalid Active Directory user names, these entries are ignored.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you specify one or more users with this parameter, user filtering is performed for all PAM-enabled applications on the host computer.

When a user attempts to log on or access a PAM-enabled service, the `pam_centri_fydc` module checks the users specified by this parameter to see if the user is listed there. If the user is included in the list, the user is accepted and authentication proceeds. If the user is not listed, the user is rejected.

The parameter value can be one or more user names, separated by commas, or the `file:` keyword and a file location. For example:

```
pam.allow.users: root,joan7,bbenton  
pam.allow.groups: administrators,sales,engineering
```

You can use the short format of the user name or the full canonical name of the user.

To enter user names with spaces, enclose them in double quotes; for example:

```
pam.allow.users: "sp1 user@acme.com",joan@acme.com,"sp2  
user@acme.com"
```

To specify a file that contains a list of the users allowed access, type the path to the file:

```
pam.allow.users: file:/etc/centri_fydc/users.allow
```

If no user names are specified, then no user filtering is performed.

If you make changes to this parameter, you should run `adf1ush` to clear the cache to ensure



## Specifying user names for computers joined to Auto Zone

If a computer is configured to use the Auto Zone instead of a specific zone, you should specify user names using the format defined by the `auto.schema.name.format` parameter. For example the `auto.schema.name.format` parameter can be set to the following:

- SAM (default) uses the `samAccountName` attribute for the user—`jcool`
- `SAM@domainName` uses the `samAccountName@domain_name` format—`jcool@acme.com`
- NTLM uses the NTLM format and separator defined for `adclient.ntlm.separators`—`acme.com+jcool`

You can look in the `centrifdc.conf` configuration file for the value of `auto.schema.name.format` parameter or run `adedit` or `adquery` commands to see the UNIX name for any user. For example, to see the UNIX name for the `jcool` Active Directory user when the `auto.schema.name.format` parameter is set to `SAM`, you can execute a command similar to this to return the UNIX user profile name:

```
adquery user -n jcool
```

## `pam.auth.create.krb5.cache`

This configuration parameter specifies whether PAM creates the Kerberos user credential cache. A value of `true` specifies that the Kerberos user credential cache is created. A value of `false` specifies that the Kerberos user credential cache is not created. The default value is `true`. For example:

```
pam.auth.create.krb5.cache: true
```

### Note:

- When this parameter is set to `false`, the Kerberos user credential cache is not created, and any attempt to perform an SSO operation will fail.
- The Kerberos user credential cache can be file-based or it can be a KCM in-memory cache, depending on the `krb5.cache.type` setting (see `krb5.cache.type`).
- This parameter is also controlled by group policy.



## pam.auth.failure.mesg

This configuration parameter specifies the message displayed during a password change if the user enters an incorrect old password.

For example:

```
pam.auth.failure.mesg: Password authentication failed
```

## pam.config.program.check

This configuration parameter specifies a list of extra PAM configuration files that are in the `pam.d` directory and that the authentication service updates (in addition to the standard PAM configuration files, such as `/pam.d/system-auth`, `/pam.d/common-auth` and so forth).

The default list is as follows:

```
pam.config.program.check: ftp,pure-ftpd,vsftpd,wu-ftpd,dzdo,sasauth
```

## pam.create.k5login

This configuration parameter specifies whether the `.k5login` file should be created automatically in the user's home directory. This file is used to enable Kerberos authentication and single sign-on in PAM-aware applications.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

The parameter value can be `true` or `false`. If set to `true`, the agent will create the `.k5login` file in the user's home directory.

For example:

```
pam.create.k5login: true
```



## pam.deny.change.shell

This configuration parameter specifies whether a user who is denied access, for example, because they are listed as a user in the `pam.deny.user` or are not listed in the `pam.allow.user` parameter, should have their shell set to the shell defined by the `nss.shell.nologin` parameter. The parameter value can be set to `true` or `false`.

If set to `true`, this parameter adds an extra level of security by ensuring that the zone user who is denied access cannot obtain any shell access, even if authenticated through Kerberos, SSH, or some other non-PAM related method. If this parameter is set to `false`, the denied user's shell is not changed and so may be able to access the system.

Because of the potential security issue, the default value for this parameter is `true`. However, since group lookups can be time-consuming for simple NSS queries, you can set this parameter to `false` to prevent the agent from changing the user's shell when denied access.

For example, to leave the user's shell unchanged when denied access, set this parameter to `false`.

```
pam.deny.change.shell: false
```

## pam.deny.groups

This configuration parameter specifies the groups that should be denied access to PAM-enabled applications. When this parameter is defined, only the listed groups are denied access. All other groups are allowed access.

**Note:** This parameter does not support cross-forest groups. (Ref: CS-18659a)

If you want to use this parameter to control which users can log in based on group membership, the groups you specify should be valid Active Directory groups, but the groups you specify do not need to be enabled for UNIX. Local group membership and invalid Active Directory group names are ignored.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.



When a user attempts to log on or access a PAM-enabled service, the `pam_centri_fydc` module checks with Active Directory to see which groups the user belongs to. If the user is a member of any Active Directory group specified by this parameter, the user is denied access and authentication fails. If the user is not a member of any group specified by this parameter, authentication succeeds and the user is logged on.

The parameter's value can be one or more group names, separated by commas or spaces, or the `file:` keyword and a file location. For example, to prevent all members of the `vendors` and `azu1` groups in Active Directory from logging on:

```
pam.deny.groups: vendors,azu1
```

You can use the short format of the group name or the full canonical name of the group.

To enter group names with spaces, enclose them in double quotes; for example:

```
pam.deny.groups: "domain admins",sales,"domain users"
```

To specify a file that contains a list of the groups that should be denied access:

```
pam.deny.groups: file:/etc/centri_fydc/groups.deny
```

**Note:** If a computer is configured to use Auto Zone without a zone, enter group names in the format specified by the `auto.schema.name.format` parameter:

- SAM (`samAccountName` — this is the default); for example: `finance_admins`
- `samAccountName@domain_name`; for example: `finance_admins@acme.com`
- NTLM; for example: `acme.com+finance_admins`

**Note:** You can look in the `centri_fydc.conf` configuration file for the value of `auto.schema.name.format`, or run `adquery group -n` to see the UNIX name for any group. For example, to see the UNIX name for the `Finance_Admins` group (and SAM, the default, is set for `auto.schema.name.format`), execute the following command, which returns the UNIX name as shown:

```
[root]#adquery group -n Finance_Admins
finance_admins
```

If this parameter is not defined in the configuration file, no group filtering is performed.



**Note:** If you make changes to this parameter, you should run `adflush` to clear the cache to ensure your changes take effect.

## pam.deny.users

This configuration parameter specifies the users that should be denied access to PAM-enabled applications. When this parameter is defined, only the listed users are denied access. All other users are allowed access.

If you want to use this parameter to control which users can log in, the users you specify should be valid Active Directory users that have been enabled for UNIX. If you specify local user accounts or invalid Active Directory user names, these entries are ignored.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

When a user attempts to log on or access a PAM-enabled service, the `pam_centrikydc` module checks the users specified by this parameter to see if the user is listed there. If the user is included in the list, the user is rejected and authentication fails. If the user is not listed, the user is accepted and authentication proceeds.

The parameter value can be one or more user names, separated by commas or spaces, or the `file:` keyword and a file location. For example, to prevent the user accounts `starr` and `guestuser` from logging on:

```
pam.deny.users: starr, guestuser
```

You can use the short format of the user name or the full canonical name of the user.

To enter user names with spaces, enclose them in double quotes; for example:

```
pam.deny.users: "sp1 user@acme.com", joan@acme.com, "sp2  
user@acme.com"
```

To specify a file that contains a list of the users that should be denied access:

```
pam.deny.users: file:/etc/centrikydc/users.deny
```

**Note:** If a computer is configured to use Auto Zone without a zone, enter user names in the format specified by the `auto.schema.name.format` parameter:



- SAM (`samAccountName` — this is the default); for example: `jcool`
- `samAccountName@domain_name`; for example: `jcool@acme.com`
- NTLM; for example: `acme.com+jcool`

**Note:** You can look in the `centrifdc.conf` configuration file for the value of `auto.schema.name.format`, or run `adquery user -n` to see the UNIX name for any user. For example, to see the UNIX name for `jcool` (and SAM, the default, is set for `auto.schema.name.format`), execute the following command, which returns the UNIX name as shown:

```
[root]#adquery user -n jcool
jcool
```

If this parameter is not defined in the configuration file, no user filtering is performed.

**Note:** If you make changes to this parameter, you should run `adflush` to clear the cache to ensure your changes take effect.

## pam.homedir.create

This configuration parameter specifies whether a new home directory should be created automatically when a new Active Directory user logs on to a system for the first time.

For example, to specify that home directories be created automatically when new Active Directory users log on to a system for the first time:

```
pam.homedir.create: true
```

In most cases, you set this configuration parameter using group policy.

**Note:** For computers that use NFS to mount home directories, you should set this parameter to `false`. If you have a Solaris environment and set this parameter to `true`, you should make sure the default location for creating a home directory is not `/home/${user}` since this path is not allowed in a typical Solaris environment. In addition, some platforms may require you to manually create a skeleton directory that contains default initial profiles to use when creating new home directories. You can use the `pam.homeskel.dir` parameter to specify the location of this skeleton directory if it exists in your environment.



## pam.homedir.create.msg

This configuration parameter specifies the message displayed when a user's home directory is created.

For example:

```
pam.homedir.create.msg: Created home directory
```

## pam.homedir.perms

This configuration parameter specifies the permissions for a user's home directory if a new home directory is created for the user on the local computer.

For example, to give read, write, and execute permissions on the directory to the user and no other permissions:

```
pam.homedir.perms: 0700
```

In most cases, you set this configuration parameter using group policy.

## pam.homedir.perms.recursive

This configuration parameter specifies whether to use the permissions defined in the PAM skeleton directory or the permissions defined in `pam.homedir.perms` when a new home directory is created for a user.

This parameter can have a value of `true` or `false`. When set to `true`, a user's new home directory is created with the contents of the skeleton directory and the permissions defined in `pam.homedir.perms`. When set to `false`, a user's new home directory is created using the contents and permissions of the skeleton directory.

This parameter has a default value of `false`. For example:

```
pam.homedir.perms.recursive: false
```

## pam.homeskel.dir

This configuration parameter specifies where the PAM skeleton directory is located. The skeleton directory to used to automatically create a new home



directory and UNIX profile for a new user, if needed.

The parameter value must be a path name. For example:

```
pam.homeskel.dir: /etc/skel
```

If this parameter is not defined in the configuration file, no files are copied when a new user directory is created.

## pam.ignore.users

This configuration parameter specifies one or more users that the agent will ignore for lookup in Active Directory. This configuration parameter ignores listed users for authentication and NSS lookups. Because this parameter allows you to intentionally skip looking up an account in Active Directory, it allows faster lookup for system accounts such as `tty`, `root`, and `bin` and local login accounts.

**Note:** Starting with Centrify DB2 agent 5.2.3, the `db2.implement.pam.ignore.users` parameter controls whether the agent checks `pam.ignore.users`. The `pam.ignore.users` parameter is checked only if `db2.implement.pam.ignore.users` is set to `true`. If `db2.implement.pam.ignore.users` is set to `false`, `pam.ignore.users` is not checked, and all users are authenticated in Active Directory. See [db2.implement.pam.ignore.users](#) for more information about `db2.implement.pam.ignore.users`.

In most cases, you set this configuration parameter using group policy. This list is then stored in the `/etc/centrifydc/user.ignore` file and used to disable lookups in Active Directory for the users specified. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value should be one or more user names, separated by a space, or the `file:` keyword and a file location. For example, to specify a list of users to authenticate locally:

```
pam.ignore.users: root sys tty
```

To specify a file that contains a list of the users to ignore:

```
pam.ignore.users: file:/etc/centrifydc/users.ignore
```

If this parameter is not defined in the configuration file, no users are specified.



## Skipping Active Directory authentication for local AIX users

By default, the agent modifies the AIX Loadable Authentication Module (LAM) for the SYSTEM user attribute to look like this:

```
SYSTEM=CENTRIFYDC OR CENTRIFYDC[NOTFOUND] AND compat
```

This setting specifies that the first attempt to authenticate a user should be passed to Active Directory through the agent. In some cases, however, you may have local user accounts that you only want to authenticate locally. Although there are parameters in the access control configuration file (`centrifydc.conf`) that enable you to ignore Active Directory authentication for specific local users, these parameters are not completely applicable on computers running AIX. To exclude any local user account from Active Directory authentication on AIX, you can run the following command for the user:

```
chuser SYSTEM=compat username
```

Alternatively, you can edit the `/etc/security/user` file and change the stanza for a particular user's SYSTEM attribute to:

```
SYSTEM=compat
```

If you later decide you want to migrate the local user account to use Active Directory, you can run the following command for the user to reset the default authentication:

```
chuser SYSTEM= username
```

**Note:** To reset the user account to be authenticated through Active Directory, there must be a space after the equal sign (=) in the command line.

## pam.mapuser.username

This configuration parameter maps a local UNIX user account to an Active Directory account. Local user mapping allows you to set password policies in Active Directory even when a local UNIX account is used to log in. This parameter is most commonly used to map local system or application service accounts to an Active Directory account and password, but it can be used for any local user account. For more information about mapping local accounts to Active Directory users, see “Mapping local UNIX accounts to Active Directory in the Administrator’s Guide.



In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, you should note that the local account name you want to map to Active Directory is specified as the last portion of the configuration parameter name. The parameter value is the Active Directory account name for the specified local user. For example, the following parameter maps the local UNIX account `oracle` to the Active Directory account `oracle_storm@acme.com` if the host computer's name is `storm`:

```
pam.mapuser.oracle: oracle_${HOSTNAME}@acme.com
```

You can specify the user name in the configuration file with any of the following valid formats:

- Standard Windows format: `domain\user_name`
- Universal Principal Name (UPN): `user_name@domain`
- Alternate UPN: `alt_user_name@alt_domain`
- UNIX user name: `user`

You must include the domain name in the format if the user account is not in the local computer's current Active Directory domain.

If this parameter is not defined in the configuration file, no local UNIX user accounts are mapped to Active Directory accounts.

## pam.mfa.program.ignore

This configuration parameter specifies a list of programs for which multi-factor authentication is ignored. If you have configured roles to require multi-factor authentication, users assigned to those roles will be required to provide two types of authentication to access PAM applications. However, some PAM applications do not support more than one authentication challenge.

You can use this parameter to add the program names that do not support multi-factor authentication. When users access the PAM applications you specify for this parameter, the multi-factor authentication requirement is ignored so that users can log on rather than be denied access.

For example, if you have configured a role with the `login-all` PAM application right and the `Require multi-factor authentication system` right, you can use this parameter to skip multi-factor authentication for specific PAM applications—



such as `xscreensaver` and `vsftpd`—where multi-factor authentication is not needed or not supported.

```
pam.mfa.program.ignore: xscreensaver vsftpd
```

You can specify multiple options separated by spaces.

By default, `ftpd`, `proftpd`, `vsftpd`, `java`, `httpd`, `cdc_chkpwd`, `kdm`, and `unix2_chkpwd` are all added to this parameter.

## pam.ntlm.auth.domains

This configuration parameter specifies the list of domains that should use NTLM authentication instead of Kerberos authentication. This parameter enables you to authenticate users behind a firewall when the Kerberos ports are blocked, but a trust relationship exists between domains inside and outside the firewall. When you set this parameter, the local domain controller outside of the firewall passes its authentication requests through the transitive trust chain for authentication inside of the firewall.

The parameter value must be one or more fully-qualified Active Directory domain names. The Active Directory domain names must be mapped to NTLM domain names, either automatically if the firewall does not prevent the mapping from being discovered, or manually by modifying the contents of the `/etc/centrifydc/domains.conf` file if the firewall prevents the mapping from automatically being discovered.

If firewall constraints prevent the automatic discovery of Active Directory to NTLM domain mapping, you can manually configure how Active Directory domain names map to NTLM domains by editing the `/etc/centrifydc/domains.conf` file to consist of a list of colon-separated values in the form of:

```
AD_DomainName:NTLM_DomainName
```

For example, the `domains.conf` file should consist of entries similar to the following:

```
AJAX.ORG:AJAX
```

```
FIREFLY.COM:FIREFLY
```

```
HR1.FIREFLY.COM:HR1
```

You can then use the `adclient.ntlm.domains` parameter using the `file:` keyword to specify the location of this file. For example:



```
adclient.ntlm.domains: file:/etc/centrifydc/domains.conf
```

**Note:** If you don't want to define the Active Directory to NTLM mapping in a separate file, you can set the `adclient.ntlm.domains` parameter to map domain names using the format `AD_DomainName:NTLM_DomainName`. For example:

```
adclient.ntlm.domains: AJAX.ORG:AJAX FIREFLY.COM:FIREFLY
```

After you have configured the mapping, you can list the Active Directory domain names for this parameter. For example, to specify that the Active Directory domains `AJAX.ORG` and `FIREFLY.COM`, which are outside of the firewall with a one-way trust to the forest inside the firewall, should use NTLM authentication, you could set the parameter like this:

```
pam.ntlm.auth.domains: AJAX.ORG, FIREFLY.COM
```

For more information about manually defining the mapping of Active Directory domains to NTLM domains, see `adclient.ntlm.domains`.

Alternatively, you can set the group policy **Computer Configuration > Centrify Settings > DirectControl Settings > Pam Settings > Specify NTLM authentication domains**.

## `pam.password.change.mesg`

This configuration parameter specifies the text displayed by a PAM-enabled application when it requests a user to change a password.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

The parameter value must be an ASCII string. UNIX special characters and environment variables are allowed. For example:

```
pam.password.change.mesg: Changing Active Directory password for\
```

If this parameter is not present, its default value is "Change password for".

## `pam.password.change.required.mesg`

This configuration parameter specifies the message displayed if the user enters the correct password, but the password must be changed immediately.



For example:

```
pam.password.change.required.mesg: \  
You are required to change your password immediately
```

## pam.password.confirm.mesg

This configuration parameter specifies the text displayed by a PAM-enabled application when it requests a user to confirm his new password by entering it again.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

The parameter value must be an ASCII string. UNIX special characters and environment variables are allowed. For example:

```
pam.password.confirm.mesg: Confirm new Active Directory  
password:\
```

If this parameter is not present, its default value is “Confirm new password:”.

## pam.password.empty.mesg

This configuration parameter specifies the message displayed if the user enters an empty password.

For example:

```
pam.password.empty.mesg: Empty password not allowed
```

## pam.password.enter.mesg

This configuration parameter specifies the text displayed by a PAM-enabled application when it requests a user to enter his password.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

The parameter value must be an ASCII string. UNIX special characters and environment variables are allowed. For example:



`pam.password.enter.mesg`: Active Directory password:\nIf this parameter is not present, its default value is "Password:".

## `pam.password.expiry.warn`

This configuration parameter specifies how many days before a password expires the PAM-enabled applications should start issuing the `pam.password.expiry.warn.mesg` to the user.

The parameter value must be a positive integer. For example, to issue a password expiration warning 10 days before a password is set to expire:

```
pam.password.expiry.warn: 10
```

If this parameter is not present, the default value is 14 days.

## `pam.password.expiry.warn.mesg`

This configuration parameter specifies the text displayed by a PAM-enabled application to warn the user that her password will expire in `pam.password.expiry.warn` days or less.

When the message is displayed, the '%d' token is replaced with the number of days until expiration. The message must contain exactly one '%d' token and no other '%' characters.

For example:

```
pam.password.expiry.warn.mesg: Password will expire in %d days
```

## `pam.password.new.mesg`

This configuration parameter specifies the text displayed by a PAM-enabled application when it requests a user to enter his new password during a password change.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.



The parameter value must be an ASCII string. UNIX special characters and environment variables are allowed. For example:

```
pam.password.new.mesg: Enter new Active Directory password:\
```

If this parameter is not present, its default value is “Enter new password:”.

## pam.password.new.mismatch.mesg

This configuration parameter specifies the message displayed during password change when the two new passwords do not match each other.

For example:

```
pam.password.new.mismatch.mesg: New passwords don't match
```

## pam.password.old.mesg

This configuration parameter specifies the message displayed by a PAM-enabled application when it requests a user to enter his old password during a password change.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

The parameter value must be an ASCII string. UNIX special characters and environment variables are allowed. For example:

```
pam.password.old.mesg: (current) Active Directory password:\
```

If this parameter is not present, its default value is “(current) password:”.

## pam.policy.violation.mesg

This configuration parameter specifies the message displayed during password change if the operation fails because of a domain password policy violation. For example, if the user attempts to enter a password that doesn't contain the minimum number of characters or doesn't meet complexity requirements, this message is displayed.

For example:



```
pam.policy.violation.mesg: \  
The password change operation failed due to a policy restriction  
set by the Active Directory administrator.  
This may be due to the new password length, lack of complexity or  
a minimum age for the current password.
```

## pam.setcred.respect.sufficient

This configuration parameter overrides an anomaly in the operation of the PAM interface on some platforms that denies access to a user who has entered the correct password. The default setting depends upon the platform as follows:

- For HPUX and Mac OSX platforms the default is `true`
- For all other platforms the default is `false`.

**Note:** Some Solaris 2.6 and Solaris 8 users have reported getting the error message `PAM_AUTH_ERR` after entering the correct password. If this occurs, set `pam.setcred.respect.sufficient: true`.

## pam.setcred.support.refresh

This parameter specifies whether the PAM flag `PAM_REFRESH_CRED` is supported and can be used to trigger creation of the credential cache and renew Kerberos tickets. The default is `false`, in which case the `PAM_ESTABLISH_CRED` flag is used to trigger creation of the credential cache and renew Kerberos tickets. For example:

```
pam.setcred.support.refresh: false
```

## pam.setcred.support.reinitialize

This parameter specifies whether the PAM flag `PAM_REINITIALIZE_CRED` is supported and can be used to trigger creation of the credential cache and renew Kerberos tickets. The default is `false`, in which case the `PAM_ESTABLISH_CRED` flag is used to trigger creation of the credential cache and renew Kerberos tickets.

For example:

```
pam.setcred.support.reinitialize: false
```



## pam.sync.mapuser

This configuration parameter controls whether the password synchronization service keeps passwords synchronized for local users that are mapped to an Active Directory account.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you set this parameter in the configuration file, the parameter value should be a list of local user accounts that are mapped to Active Directory accounts. For example:

```
pam.sync.mapuser: root oracle sanchez
```

If you set this parameter and a mapped user changes his password, PAM updates the password hash for the corresponding local UNIX account in the local `/etc/shadow` file so that the passwords match. Synchronizing the passwords in this way ensures that local users can still log on even if there are problems with the network, Active Directory, or the `adclient` process. For example, if Active Directory is not available, the mapped user can log on as a local user by appending `@localhost` to the user name:

```
sanchez@localhost
```

Password synchronization requires you to do the following:

- Install either the Centrify Password Synchronization component or the Microsoft Password Synchronization Service on all domain controllers.  
If you do not have the Microsoft Password Synchronization Service installed on your domain controllers, you can install and use the Centrify Password Synchronization extension instead. You can install the Centrify Password Synchronization extension when you install other Centrify Management Services using the setup program or by running the standalone password extension installation program.
- Configure the zone properties for the computer's zone to support agentless clients and to use the proper NIS domain name and Active Directory attribute for storing the user's password hash.
- Map the specified local users to Active Directory using either the `pam.mapuser.username` configuration parameter or group policy.
- Verify the Active Directory user to which the local user is mapped has a profile in the zone you have configured for agentless authentication.



This parameter has no effect on Mac OS X systems.

## pam.uid.conflict

This configuration parameter specifies how you want the agent to respond if a user logs on with an Active Directory account and either the Active Directory user name or Active Directory UID conflicts with a local user account. The purpose of detecting a duplicate user name or duplicate UID is to prevent an Active Directory user from signing on and receiving privileges to modify files created by a different local user.

The `pam.uid.conflict` configuration parameter determines what happens when this type of conflict is found. The parameter value must be set to one of the following valid options:

Use this value	To do this
<code>ignore</code>	Do not report duplicate user names or UID conflicts. If detected, log the conflict at the <code>info</code> level if logging is enabled.
<code>warn</code>	Warn the user of the user name or UID conflict after a successful login. Log the conflict at <code>warning</code> level if logging is enabled. This is the default value.
<code>error</code>	Report UID conflict to user after user name is entered. Don't accept password. Don't allow log in. Log conflict at <code>error</code> level.

For example:

```
pam.uid.conflict: warn
```

**Note:** If both the Active Directory user name and Active Directory UID are the same as a local user name and UID, the accounts do not conflict and the user can log on regardless of how you set this parameter. Although this situation is rare, you should avoid using Active Directory user names and UIDs that duplicate local user names and UIDs but apply to different individual users.

If this parameter is not present, its default value is `warn`.

## pam.workstation.denied.mesg

This configuration parameter specifies the message displayed if a user attempting to log on is denied access because of a workstation restriction.



For example:

```
pam.workstation.denied.mesg: \  
Your account is configured to prevent you from using this  
computer. Please try another computer.
```

## microsoft.pam.privilege.escalation.enabled

The configuration parameter specifies if the Microsoft Privileged Access Management (PAM) Privilege Escalation feature is supported or not within the Centrify environment.

If `microsoft.pam.privilege.escalation.enabled` is `true`, then, when an Active Directory user logs in, the configured privilege that's granted to the user through PAMGroup takes effect until the granted period has elapsed.

The Privileged Access Management (PAM) Privilege Escalation feature can be enabled or disabled through Group Policy. Select **Computer Configuration > Centrify Settings > DirectControl Settings > Enable Active Directory PAM Privilege Escalation feature**

The Microsoft PAM Privilege Escalation feature specifies if Centrify DirectControl uses Microsoft PAM Privilege Escalation feature in the computer.

For example:

```
microsoft.pam.privilege.escalation.enabled: true
```

Default is `false`, the Microsoft PAM Privilege Escalation feature support is disabled. Setting it to `true` enables grants the Active Directory user, at log in, the same configured privilege as the user's PAMGroup. This is in effect until the grant period expires.



# Customizing group policy configuration parameters

This section describes the configuration parameters that affect group policy support on the local host computer.

gp.disable.all .....	204
gp.disable.machine .....	204
gp.disable.user .....	204
gp.disk.space.check.folders .....	205
gp.disk.space.min .....	205
gp.mappers.certgp.pl.additional.cafiles .....	205
gp.mappers.certgp.pl.exclude.cacerts .....	206
gp.mappers.directory.machine .....	206
gp.mappers.directory.user .....	206
gp.mappers.error_file .....	207
gp.mappers.machine .....	207
gp.mappers.runmappers .....	208
gp.mappers.timeout .....	208
gp.mappers.timeout.all .....	209
gp.mappers.umask .....	209
gp.mappers.user .....	210
gp.refresh.disable .....	210
gp.reg.directory.machine .....	211



gp.reg.directory.user .....	211
gp.use.user.credential.for.user.policy .....	211
gp.user.login.run .....	212

## gp.disable.all

This configuration parameter can be used to disable both computer and user group policies on a local computer. If set to `true`, all group policy settings are ignored.

For example:

```
gp.disable.all: true
```

If this parameter is not defined in the configuration file, its default value is `false`.

## gp.disable.machine

This configuration parameter can be used to disable computer-based group policies on a local computer. If set to `true`, all computer-based group policy settings are ignored.

For example:

```
gp.disable.machine: true
```

If this parameter is not defined in the configuration file, its default value is `false`.

## gp.disable.user

This configuration parameter can be used to disable user-based group policies on a local computer. If set to `true`, all user-based group policy settings are ignored.

For example:

```
gp.disable.user: true
```



If this parameter is not defined in the configuration file, its default value is `false`.

## `gp.disk.space.check.folders`

This configuration parameter specifies the folders that need the free disk space check. If the free space in any specified folder is less than the value in `gp.disk.space.min`, then group policy settings will not be updated.

Specify a comma-separated list of folders; for example, the default is:

```
gp.disk.space.check.folders: /,/etc,/var
```

## `gp.disk.space.min`

This configuration parameter specifies the minimum free disk space in kilobytes (KB) that is required for a group policy update. If the free disk space in any folder specified in `gp.disk.space.check.folders` is less than this value, then group policy settings will not be updated.

When updating the configuration file, the Perl mapper scripts create a temp file, print to it, and replace the original file. If the disk is full, the mapper cannot write to the temp file, so the temp file is empty, and the original file is replaced by the empty temp file. This configuration parameter and `gp.disk.space.min` prevent the mapper writing to a temp file when disk space is low.

The default value is 5120 KBytes. Set this parameter to 0 to not check free disk space.

## `gp.mappers.certgp.pl.additional.cafiles`

This setting defines a list of certificate files which will be included in the `certgp.pl` install, if found.

It can be a list of certificates to be added. For example:

```
gp.mappers.certgp.pl.additional.cafiles: <ca-file> <ca-file> ...
```

It can also point to a file that contains a list of certificate files to be added. For example:



```
gp.mappers.certgp.pl.additional.cafiles:  
file:/etc/centrifydc/cert_included.list
```

The default value is empty.

## gp.mappers.certgp.pl.exclude.cacerts

This setting defines a certificate list which will be excluded from the `certgp.pl` install, if matched.

It can be a list of fingerprints of the certificates to be excluded. For example:

```
gp.mappers.certgp.pl.exclude.cacerts: <fingerprint> <fingerprint>  
...
```

It can also point to a file that contains a list of fingerprints of the certificates to be excluded. For example:

```
gp.mappers.certgp.pl.exclude.cacerts: file:/etc/centrifydc/cert_  
excluded.list
```

The default value is empty.

## gp.mappers.directory.machine

This configuration parameter specifies the root directory that contains all of the mapping programs for computer-based group policy settings. Individual programs map entries from the virtual registry into configuration settings in the appropriate files on the local computer.

The parameter value must be a path name. For example:

```
gp.mappers.directory.machine:  
/usr/share/centrifydc/mappers/machine
```

If this parameter is not defined in the configuration file, its default value is `/usr/share/centrifydc/mappers/machine`.

## gp.mappers.directory.user

This configuration parameter specifies the root directory that contains all of the mapping programs for user-based group policy settings. Individual programs



map entries from the virtual registry into configuration settings in the appropriate files on the local computer.

The parameter value must be a path name. For example:

```
gp.mappers.directory.machine: /usr/share/centrifdc/mappers/user
```

If this parameter is not defined in the configuration file, its default value is `/usr/share/centrifdc/mappers/user`.

## gp.mappers.error\_file

This configuration parameter specifies the name of the file where the group policy mapper programs write error messages.

For example:

```
gp.mappers.error_file: mapper.errors
```

## gp.mappers.machine

This configuration parameter specifies the list of mapping programs to run to configure computer-based policies. The mapping programs are contained in the root directory specified by `gp.mappers.directory.machine` (`/usr/share/centrifdc/mappers/machine` by default). The mapping programs are executed in the order in which they are specified. The mapping program `centrifdc.conf.pl` will always run even if unspecified and does not run only if you specify that it not run (described later).

In most cases you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you want to temporarily override group policy.

To specify mapping programs to run, you can list each individual program name literally, or you can use wild card characters that are a subset of regular expression wild card characters:

- An asterisk (\*) specifies any set of zero or more characters. “map\*”, for example, specifies any program names starting with “map”. “set\*.pl” specifies any program names starting with “set” and ending with “.pl”. And “\*dc\*” specifies any program names that include “dc”. “\*” means all programs.



- A question mark (?) specifies any single character. “map???” , for example, specifies any six-character program name starting with “map”.
- Square brackets ([ ]) enclosing a set of characters specifies a single character that is one of the enclosed characters. “mapprogram[123]” , for example, matches the program names mapprogram1, mapprogram2, and mapprogram3.

You can specify a program name not to execute by preceding it with an exclamation point (!). If you specify “\* !mapprogram1” , for example, you specify that all mapping programs in the mapping program root directory should execute except for “mapprogram1” . Note that the only way you can stop the automatically executing program `centrifdc.conf.pl` from executing is to specify “!centrifdc.conf.pl” in this parameter.

You can combine all of these rules to give you precise control over which mapping programs run. Some examples:

`gp.mappers.machine: *` specifies all mapping programs in the mapping program parent directory.

`gp.mappers.machine: mapgp* !mapgp2` specifies all mapping programs in the mapping program parent directory that start with “mapgp” except for “mapgp2” . Note that `centrifdc.conf.pl` will execute because it hasn’t been specified not to execute and so executes automatically.

## gp.mappers.runmappers

This configuration parameter specifies the location of the runmappers program. The runmappers program is started by the agent and invokes individual mapping programs for computers, users or both.

The parameter value must be a path name. For example:

`gp.mappers.runmappers: /usr/share/centrifdc/mappers/runmappers`

If this parameter is not defined in the configuration file, its default value is `/usr/share/centrifdc/mappers/runmappers`.

## gp.mappers.timeout

This configuration parameter specifies the maximum time, in seconds, to allow for a single mapping program to complete execution. If a mapping program



takes longer than this period to successfully complete its execution, the process is stopped and the next mapping program is started.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be a positive integer that is less than the value set for the `gp.mappers.timeout.all` parameter. For example, to set the timeout interval to 60 seconds:

```
gp.mappers.timeout: 60
```

The default value for this parameter on Mac is 120 seconds.

The default value for this parameter on all other platforms is 30 seconds.

## gp.mappers.timeout.all

This configuration parameter specifies the maximum time, in seconds, to allow for all mapping programs to complete execution. The parameter value must be a positive integer that is less than the value set for the `1rpc.timeout` parameter.

The default value for this parameter on Mac is ten minutes (600 seconds). For example:

```
gp.mappers.timeout.all: 600
```

The default value for this parameter on all other platforms is four minutes (240 seconds). For example:

```
gp.mappers.timeout.all: 240
```

## gp.mappers.umask

This configuration parameter specifies the default `umask` for mapping programs that create files. The default value for this parameter sets the following read and write permissions for mapping programs that create files:

```
u=rwx  
g=rx  
o=
```



The parameter value specifies these permissions using numeric mode. For example:

```
gp.mappers.umask: 0027
```

## gp.mappers.user

This configuration parameter specifies the mapping programs that map user-based policy settings to run. The mapping programs are executed in the order in which they are specified.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you want to temporarily override group policy.

In defining the list of mapping programs to run, you can use an asterisk (\*) as a wild card to match a set of program names. For example, you can specify `a*` to match all programs with names that start with the letter `a`. You can use square brackets ([ ]) to match any character within the brackets. For example, you can specify `mapprogram[123]` to match the program names of `mapprogram1`, `mapprogram2`, and `mapprogram3`. You can also use an exclamation point (!) with a program name to exclude a program from the list. For example, you can specify `!mysample` to prevent the mapping program `mysample` from running.

To run all of the mapping programs for user-based policy settings, you can specify:

```
gp.mappers.user: *
```

To run a subset of the mapping program, you can explicitly define the order and which programs to run. For example, to run the program `mapgp1`, followed by `mapgp4` and `mapgp3`, but skipping the execution of `mapgp2`:

```
gp.mappers.user: mapgp1 !mapgp2 mapgp4 mapgp3
```

## gp.refresh.disable

This configuration parameter specifies whether to disable the background processing of group policy updates. This configuration parameter applies to both computer- and user-based policies.



In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you want to temporarily override group policy. For example:

```
gp.refresh.disable: false
```

## gp.reg.directory.machine

This configuration parameter specifies the root directory of the virtual registry for computer-based group policies. The parameter value must be a path name. For example:

```
gp.reg.directory.machine: /var/centrifydc/reg/machine
```

If this parameter is not defined in the configuration file, its default value is `/var/centrifydc/reg/machine`.

## gp.reg.directory.user

This configuration parameter specifies the root directory of the virtual registry for user-based group policies.

The parameter value must be a path name. For example:

```
gp.reg.directory.user: /var/centrifydc/reg/users
```

If this parameter is not defined in the configuration file, its default value is `/var/centrifydc/reg/users`.

## gp.use.user.credential.for.user.policy

This configuration parameter specifies whether to use the user's credentials to retrieve user group policies. By default, all group policies are retrieved using the computer account credentials, which are associated with the `adcli` process rather than the user who is currently logged on. In most cases, this default behavior is sufficient because most of the Centrify group policies are computer configuration policies. However, if the computer account does not have permission to access the Group Policy Object where user policies are defined, the default behavior prevents user policies from being applied.



You can set this configuration parameter to `true` to use the user's credentials to retrieve user group policies. For example:

```
gp.use.user.credentials.for.user.policy: true
```

If this parameter is not defined in the configuration file, its default value is `false`.

## gp.user.login.run

This configuration parameter specifies when user-based group policies should run. By default, user-based group policies are applied when a user first logs on to a computer, then at a regular interval in background to check for updates and changes while the user's session remains active. However, running group policies at every login and refresh interval for users who are already logged on can impact performance on computers where there are a large number of group policies being applied. You can use this parameter to reduce the load on those computers by customizing when group policies should be applied.

This configuration parameter enables you to specify whether the user-based group policies should be applied:

- Only once when the user first logs on and not again until the user logs off and logs back on.
- When the user first logs on and regularly at the refresh interval for as long as the user remains logged on.
- Never when the user logs on, but periodically at the refresh interval thereafter.

The valid parameter values for this configuration parameter are `once`, `always`, and `never`.

For example, to specify that user-based group policies should only run once when the user first logs on but not thereafter, you can set this parameter to `once`:

```
gp.user.login.run: once
```

If this parameter is not defined in the configuration file, its default value is `always` to apply the user group policies when a user first logs on and periodically refresh the policies in the background for as long the user remains logged on.



# Customizing NSS-related configuration parameters

This section describes the configuration parameters that affect the operation of NSS-related activity on the local host computer.

**Note:** On AIX, the NSS configuration parameters described in this chapter may apply to interfaces in the AIX Loadable Authentication Module (LAM). For consistency across platforms, most of the parameter names are the same and retain the reference to NSS settings they configure, but NSS is not used on AIX.

nss.alias.source .....	214
nss.gecos.attribute .....	214
nss.gid.ignore .....	215
nss.group.ignore .....	216
nss.group.override .....	217
nss.group.skip.members .....	218
nss.nobody.gid .....	218
nss.nobody.group .....	219
nss.nobody.uid .....	219
nss.nobody.user .....	219
nss.passwd.hash .....	219
nss.passwd.info.hide .....	220
nss.passwd.override .....	220
nss.program.ignore .....	222



nss.shell.emergency.enabled .....	223
nss.shell.nologin .....	223
nss.split.group.membership .....	224
nss.squash.root .....	225
nss.uid.ignore .....	225
nss.user.ignore .....	226
nss.user.ignore.all .....	227
lam.attributes.group.ignore .....	227
lam.attributes.user.ignore .....	228
lam.max.group.count .....	228
lam.max.user.count .....	228

## nss.alias.source

This configuration parameter specifies the source to look up aliases, and you specify one of the following values:

- nismaps (default)
- mail
- proxyaddresses

To look up the alias from an Active Directory user object, use the mail or proxyAddresses value. Because proxyaddresses is a custom attribute, you need to also include it in the adclient.custom.attributes.user parameter or else the alias source reverts to nismaps.

Using the mail or proxyAddresses values don't work with users in a one-way trusted forest.

## nss.gecos.attribute

This configuration parameter specifies the Active Directory user object attribute to use for the GECOS field. The default value for this parameters is the geCOS attribute in the Active Directory RFC2307 schema.



The order of precedence for the GECOS field setting is:

1. The GECOS setting for the UNIX service connection point (SCP) in Active Directory.
2. The `nss.gecos.attribute` setting.
3. The `displayName` attribute of the user object.

If `nss.gecos.attribute` is set and GECOS is not set for the UNIX SCP, the user attribute specified by `nss.gecos.attribute` is used for the GECOS field in UNIX profiles and NSS lookups. If `nss.gecos.attribute` is not defined or the Active Directory RFC2307 schema is not used, the user object's `displayName` attribute is used as the GECOS field for UNIX profiles.

If you set this configuration parameter, the parameter value is case-sensitive and must exactly match the case used for the attribute name in Active Directory. For example:

```
nss.gecos.attribute: displayName
```

## nss.gid.ignore

This configuration parameter specifies a set of one or more group identifiers that the Centrify NSS module will ignore for lookup in Active Directory.

In most cases, this configuration parameter's value is generated automatically by group policy.

If you select the **Specify group names to ignore** policy and click **Enabled**, you can type the list of local group names not stored in Active Directory. The list you specify for the group policy is then stored in the `/etc/centrifydc/group.ignore` file and used to automatically generate the `/etc/centrifydc/gid.ignore` file. These files are then used to disable looking up account information in Active Directory for the groups specified, which results in faster name lookup service for system group accounts such as `tty` and `disk`.

You can, however, define this parameter manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you manually set this parameter, the parameter value should be one or more group identifiers, separated by a space, or the `file:` keyword and a file location. For example:

```
nss.gid.ignore: 0 20 5861  
nss.gid.ignore=file:/etc/centrifydc/gid.ignore
```



A default set of groups to ignore are defined in sample `/etc/centrifydc/group.ignore` and `/etc/centrifydc/gid.ignore` files. If you edit either file, be sure to run the `adreload` command after modifying the file to have the changes take effect.

## nss.group.ignore

This configuration parameter specifies a set of one or more groups that the Centrify NSS module will ignore for lookup in Active Directory.

In most cases, this configuration parameter's value is generated automatically by group policy.

If you select the **Specify group names to ignore** policy and click **Enabled**, you can type the list of local group names not stored in Active Directory. The list you specify for the group policy is then stored in the `/etc/centrifydc/group.ignore` file and used to automatically generate the `/etc/centrifydc/gid.ignore` file. These files are then used to disable looking up account information in Active Directory for the groups specified, which results in faster name lookup service for system group accounts such as `tty` and `disk`.

You can, however, set this parameter manually in the configuration file if you aren't using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value should be one or more group names, separated by a space, or the `file:` keyword and a file location. For example:

```
nss.group.ignore: maintenance apps  
nss.group.ignore=file:/etc/centrifydc/group.ignore
```

A default set of groups to ignore are defined in sample `/etc/centrifydc/group.ignore` and `/etc/centrifydc/gid.ignore` files. If you are not using group policies, you can uncomment the `nss.group.ignore` parameter in the `/etc/centrifydc/centrifydc.conf` file to ignore the default set of groups.

**Note:** If you plan to edit the `group.ignore` file, be sure to run the `adreload` command after modifying the file to have the changes take effect.



## nss.group.override

This configuration parameter allows you to override group profile entries for zone groups. Using this parameter is similar to defining override filters for local groups in the `/etc/group` file. By defining override filters, you can use this parameter to give you fine-grain control over the groups that can access a local computer. You can also use the override controls to modify the information for specific fields in each group entry on the local computer. For example, you can override the group ID or member list for a specific group on the local computer without modifying the group entry itself.

In most cases, you set this configuration parameter using group policy. The entries created by group policy are then stored in the `/etc/centrifydc/group.ovr` file and used to filter group access to a local computer. You can, however, set this parameter manually in the configuration file if you are not using group policy or want to temporarily override group policy.

The syntax for overriding group entries is similar to the syntax used for overriding NIS. You use `+` and `-` entries to allow or deny access for specific groups on the local system. Additional fields correspond to the standard `/etc/group` fields separated by colons (`:`).

In most cases, the `nss.group.override` parameter is used to identify a file location of an override file that contains all of group override entries you want to use on the local computer. For example:

```
nss.group.override: file:/etc/centrifydc/group.ovr
```

Within the override file, you use the following format:

```
+zone_group_name:group_name:group_password:group_id:member_list  
-zone_group_name:group_name:group_password:group_id:member_list
```

For example:

```
+users::::  
+admins::::jdoe,bsmith,frank  
+ftusers:ftp::300:  
-webusers  
+::::
```

**Note:** Changes to the group password field are ignored.

For more information about overriding group entries, see the sample group override file `/etc/centrifydc/group.ovr`.



**Note:** If you make changes to this parameter or the override file, you should run `adflush` to clear the cache to ensure your changes take effect.

## nss.group.skip.members

This configuration parameter allows you to skip the retrieval of group membership information for specific groups. Retrieving group membership information from Active Directory can be a very time-consuming and memory-intensive operation for groups with a large number of users, or when using nested groups, but in many cases this information is not needed to perform common UNIX operations. Using this configuration parameter to skip the retrieval of group membership information for specific groups can greatly improve performance for groups with a large number of members.

The parameter value should be a comma-separated list of the UNIX commands for which you can skip group member expansion in the `getgrent()` call.

The default setting for this configuration parameter is the following for most systems:

```
ls, chown, find, ps, chgrp, dtaction, dtwm, pt_
chmod, id, login, sshd, sshd2, getty, dtlogin, su, adsetgrps, adid
```

For AIX system, the default is the following:

```
nss.group.skip.members=ls, chown, find, ps, chgrp, dtaction, dtwm, pt_
chmod, id, login, sshd, sshd2, getty, dtlogin, su, adsetgrps, adid
```

**Note:** Setting this parameter does not affect the information returned when the `nscd` or `pwgrd` daemon is running on a system. The `nscd` or `pwgrd` daemons provide a cache for faster user and group lookups, but when the response comes from this cache, the agent cannot modify the response to skip the members listed with this parameter.

## nss.nobody.gid

This configuration parameter specifies the group ID (GID) of the system's `nobody` group.

For example:

```
nss.nobody.gid: 99
```



## nss.nobody.group

This configuration parameter specifies the group name of the system's nobody group.

For example:

```
nss.nobody.group: nobody
```

## nss.nobody.uid

This configuration parameter specifies the user ID (UID) of the system's nobody user.

For example:

```
nss.nobody.uid: 99
```

## nss.nobody.user

This configuration parameter specifies the user name of the system's nobody user.

For example:

```
nss.nobody.user: nobody
```

## nss.passwd.hash

This configuration parameter specifies whether to include the UNIX password hash in response to the `getpw*` commands. The parameter value can be `true` or `false`. The default value for the parameter is `false` because the password hash is sensitive information and can make a system vulnerable to a brute force attack. However, if you have applications, such as Informix, that validate users based on the password hash retrieved from NSS, you can set this parameter to `true` to accommodate those applications.

If you set this parameter to `true`, however, you must also install a password synchronization service on all of the domain controllers in the domain. The password synchronization service can be the Centrify Password Filter, or the



Password Synchronization Service provided by Microsoft in Windows Server 2003 R2 or in the Microsoft Services for UNIX (SFU) package.

## nss.passwd.info.hide

This configuration parameter specifies whether to hide the following password attributes from non-root users:

- Maximum Password Age
- Password Expiration Date
- Minimum Password Age
- Change Password Needed
- Password Last Changed On

The parameter value can be `true` or `false`. When this parameter is set to `true`, only users with root permissions can view the password attributes shown above. When this parameter is set to `false`, users without root permissions can view the password attributes.

The default value for this parameter is `true` on all UNIX operating systems except HP-UX. On HP-UX, the default is `false` because HP-UX does not support hiding these attributes.

## nss.passwd.override

This configuration parameter allows you to override user profile entries for zone users. Using this parameter is similar to defining override filters for local users in the `/etc/passwd` file. By defining override filters, you can use this parameter to give you fine-grain control over the user accounts that can access a local computer. You can also use the override controls to modify the information for specific fields in each `/etc/passwd` entry on the local computer. For example, you can override the user ID, primary group ID, default shell, or home directory for specific login accounts on the local computer without modifying the account entry itself.

In most cases, you set this configuration parameter using group policy. The entries created by group policy are then stored in the `/etc/centrifydc/passwd.ovr` file and used to filter user access to a local computer. You can, however, set this parameter manually in the configuration



file if you are not using group policy or want to temporarily override group policy.

The syntax for overriding `passwd` entries is similar to the syntax used for overriding NIS. You use `+` and `-` entries to allow or deny access for specific users on the local system. Additional fields correspond to the standard `/etc/passwd` fields separated by colons (`:`).

In most cases, the `nss.passwd.override` parameter is used to identify a file location of an override file that contains all of `passwd` override entries you want to use on the local computer. For example:

```
nss.group.override: file:/etc/centrifydc/passwd.ovr
```

**Note:** Although the `passwd.ovr` file is generated from the list of override entries you specify using group policy, you can also manually create or update the override file on any local computer, if needed. A sample illustrating the syntax is provided in the `/etc/centrifydc/passwd.ovr.sample` file.

Within the override file, you use the following format for entries:

```
+zone_username:username:password:uid:gid:GECOS:home_
directory:shell
```

For example:

```
+mike:::::::::/usr/local/ultrabash
+kris:kdavis:x:6:6:Kris Davis:/home/kdavis:/bin/bash
+janedoe@centrify.test:jdoe::300:300:::
+@sysadmins:::::::::
-ftp
+@staff:::::::::
+@rejected-users:::32767:32767:::/bin/false
+:::::::::/sbin/nologin
+:::::::::
```

**Note:** Overriding the password hash field is ignored. Changing this field in the override file does not affect zone user passwords. In overriding `passwd` entries, users accounts must be enabled for UNIX in the zone, but the groups do not need to be UNIX-enabled.

In the example above, the `@` symbol denotes an Active Directory name. It may be an Active Directory group name, a zone name, or some other container name. You may also specify an Active Directory user principal name instead of the zone name.



Entries in the override file are evaluated in order from first to last with the first match taking precedence. This means the system will only use the first entry that matches a particular user. For example, if the user `cruz` is a member of both the `staff` group and the `rejected-users` group and you have defined the override entries as listed in the example above, the `cruz` user account is allowed to log on to the computer because the `staff` entry is evaluated and matched before the `rejected-users` entry. If the order were reversed in the override file, the `cruz` account would be flagged as a `rejected-users` account and denied access.

**Note:** If you manually create the `passwd.ovr` file, you must include the following as the last line in the file:

```
+:::~::~:
```

For more information about overriding group entries, see the sample `passwd` override file `/etc/centrifydc/passwd.ovr`. For information about using the NSS Overrides group policy to generate and maintain the `passwd.ovr` file, see the Access Manager online help.

**Note:** If you make changes to this parameter or the override file, you should run `adf1ush` to clear the cache to ensure your changes take effect.

## nss.program.ignore

This configuration parameter specifies one or more programs that should not look up account information in Active Directory. The programs you specify for this parameter do not use the agent to contact Active Directory.

Setting this parameter helps to ensure that local programs that create, manage, or use local user and group information do not attempt to look up conflicting information in Active Directory. For example, you can specify programs such as `adduser` and `addgroup` to ensure those programs can still be used to create and update local accounts independent of Active Directory:

```
nss.program.ignore: addgroup,adduser
```

The specific programs you should include in the list vary by platform and the specific operating environment you are using. The default setting for this configuration parameter includes the most common program names that shouldn't make calls to Active Directory through the agent.



If you have auditing enabled, the agent's auditing service maintains a cache of user information for performance reasons. When you have auditing enabled, you can also use this parameter to circumvent the agent accessing its local cache when you use commands that manipulate local user information directly. For example, you would want the agent to skip checking its local cache when you use commands such as `useradd`, `userdel`, `adduser`, `usermod`, `mkuser`, `rmuser`, `chuser`, and any other programs that directly access the local `/etc/passwd` file.

**Note:** Setting this parameter does not affect the information returned when the `nscd` or `pwgrd` daemon is running on a system. The `nscd` and `pwgrd` daemons provide a cache for faster user and group lookups, but when the response comes from this cache, the agent cannot modify the response to skip the programs listed with this parameter.

You can also set this configuration parameter using `group` policy.

## nss.shell.emergency.enabled

This configuration parameter specifies whether to use the default login shell when a user or group attempting to access the computer is not allowed to log in.

The default no-login shell and its location is typically platform-specific. For example, on machines running Red Hat Linux, the default shell for users who are denied access is:

```
/sbin/nologin
```

The default for this parameter is `false`, which means that the `nologin` shell configured in `nss.shell.nologin` is returned.

## nss.shell.nologin

This configuration parameter specifies the default login shell to use when a user or group attempting to access the computer is not allowed to log on. The default no-login shell and its location is typically platform-specific. For example, on Red Hat Linux the default shell for users who are denied access is `/sbin/nologin`.

For example:



```
nss.shell.nologin: /sbin/nologin
```

**Note:** If you make changes to this parameter, you should run `adflush` to clear the cache to ensure your changes take effect.

## nss.split.group.membership

This configuration parameter specifies whether to split up or truncate large groups when you use the `getent group` UNIX command to retrieve group information.

In operating environments that do not support large groups, commands that return group information could fail or return incomplete results when a group has a membership list exceeds the maximum size allowed. Typically, the maximum size allowed for groups is 1024 bytes, which is roughly equivalent to 125 users. If your environment contains large groups that exceed the 1024-byte limit, you can set this parameter to `true` to have those groups automatically split into multiple groups when they reach the maximum size.

When this parameter is set to `true` and you issue the `getent group` command without specifying a group name, large groups are split into sublists, and all sublists are returned. When this parameter is set to `false`, large groups are truncated, and only the truncated results of the group list (typically the first 1024 bytes) are returned.

**Note:** This policy has no effect in Mac OS X environments.

**Note:** This configuration parameter takes effect only when you do not specify a group name on the `getent group` command line. Because of the way in which group information is queried in NSS, group lists are always truncated (and not split) when you specify a group name on the `getent group` command line (for example, `getent group group_name`).

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

The default value is `true` for Solaris, HP-UX, and IRIX, but `false` for all other operating environments. For example:

```
nss.split.group.membership: true
```



## nss.squash.root

This configuration parameter specifies whether you want to force `root` and `wheel` super-user accounts to be defined locally. If you set this parameter to `true`, Active Directory users with a UID of 0, a GID of 0, a user or group name of `root`, or a group name of `wheel` are not permitted to log on. Because the agent cannot prevent Active Directory users or groups from being assigned a UID or GID of 0, which would give those users or groups root-level access to the computers in a zone, you can use this parameter to prevent any Active Directory users with a UID or GID of 0 from logging on. Setting this parameter to `true` forces the privileged accounts to be defined as local accounts and not authenticated through Active Directory.

For example:

```
nss.squash.root: true
```

If you set this parameter to `false`, you should use other configuration parameters, such as `pam.ignore.users` or `user.ignore` to skip Active Directory authentication for system accounts so that Active Directory users cannot be granted root access on the computers in the zones they are permitted to access.

The default value for this parameter is `true`. It is possible, however, for an Active Directory administrator to override this setting through the use of group policy applied to a local computer, for example, by using the **Sudo rights** group policy. There is no way to effectively prevent the setting from being changed, except by disabling computer-based group policies in the local `centrifydc.conf` file or by strictly controlling who has permission to enable and apply group policies to computers that join an Active Directory domain. For information about disabling group policies using parameters in the local `centrifydc.conf` file, see [gp.disable.all](#) or [gp.disable.machine](#) in [Customizing group policy configuration parameters](#)

## nss.uid.ignore

This configuration parameter specifies a set of one or more user identifiers that the Centrify NSS module will ignore for lookup in Active Directory.

In most cases, this configuration parameter's value is generated automatically by group policy.



If you select the **Specify user names to ignore** group policy and click **Enabled**, you can type the list of local user names not stored in Active Directory. The list you specify for the group policy is then stored in the `/etc/centrifydc/user.ignore` file and used to automatically generate the `/etc/centrifydc/uid.ignore` file. These files are then used to disable looking up account information in Active Directory for the users specified, which results in faster name lookup service for system user accounts such as `tty` and `disk`.

You can, however, define this parameter manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you manually set this parameter, the parameter value should be one or more user identifiers, separated by a space, or the `file:` keyword and a file location. For example:

```
nss.uid.ignore: 0 20 5861
nss.uid.ignore=file:/etc/centrifydc/uid.ignore
```

A default set of system user accounts to ignore is defined in the sample `/etc/centrifydc/user.ignore` file and in the `/etc/centrifydc/uid.ignore` file. If you edit either file, be sure to run the `adreload` command after modifying the file to have the changes take effect.

## nss.user.ignore

This configuration parameter specifies one or more users that the Centrify NSS module will ignore for lookup in Active Directory. Because this parameter allows you to intentionally skip looking up specific accounts in Active Directory, it allows faster lookup for system accounts such as `tty`, `root`, and `bin`.

**Note:** This configuration parameter only ignores the listed users for NSS lookups. To ignore users for authentication and NSS lookups, use the `pam.ignore.users` configuration parameter.

In most cases, this configuration parameter's value is generated automatically by group policy.

If you select the **Specify user names to ignore** policy and click **Enabled**, you can type the list of local user names not stored in Active Directory. This list is then stored in the `/etc/centrifydc/user.ignore` file and used to automatically generate the `/etc/centrifydc/uid.ignore` file. These files are then used to disable looking



up account information in Active Directory for the users specified, which results in faster name lookup service for system user accounts such as `tty` and `disk`.

You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value should be one or more user names, separated by a space, or the `file:` keyword and a file location. For example:

```
nss.user.ignore: root sys tty
nss.user.ignore=file:/etc/centrifydc/user.ignore
```

A default set of users to ignore are defined in sample `/etc/centrifydc/user.ignore` and `/etc/centrifydc/uid.ignore` files. If you are not using group policies, you can uncomment the `nss.user.ignore` parameter in the `/etc/centrifydc/centrifydc.conf` file to ignore the default set of users.

**Note:** If you plan to edit the `user.ignore` file, be sure to run the `adreload` command after modifying the file to have the changes take effect.

## nss.user.ignore.all

This configuration parameter specifies how the list of users in `nss.user.ignore` is applied during lookups.

The parameter value can be `true` or `false`.

When you set this parameter to `true`, lookups generated by NSS, `ldapproxy`, or JAPI ignore the Active Directory users listed in `nss.user.ignore`.

When you set this parameter to `false`, only lookups generated by NSS ignore the Active Directory users listed in `nss.user.ignore`.

The default value is `false`.

## lam.attributes.group.ignore

This parameter points to a file containing a list of AIX group attributes that the `lam` module should ignore and let AIX handle it to either provide the default value or return `ENOATTR`. The default is `file:/etc/centrifydc/attributes.group.ignore`.



## lam.attributes.user.ignore

This parameter points to a file containing a list of AIX user attributes that the lam module should ignore and let AIX handle it to either provide the default value or return ENOATTR. The default is `file:/etc/centrifydc/attributes.user.ignore`.

## lam.max.group.count

This configuration parameter applies to the AIX Loadable Authentication Module (LAM) and specifies the maximum number of Active Directory groups that the `lsgroup ALL` command will return.

The parameter value must be an integer. The default value for this parameter is 1000 groups. If you specify 0 or a negative value (for example, -1), there is no limit on the number of groups returned. For example:

```
lam.max.group.count: 100
```

Before changing this parameter setting or using a value of 0, you should consider its impact on your environment. Increasing the value of this parameter may provide more complete information about the number of Active Directory UNIX groups, but may result in slower performance if there are more Active Directory UNIX groups in the zone than the maximum you specify. Similarly, if you do not set a limit, you may experience performance problems if you have a large number of Active Directory groups. Decreasing the value of this parameter may provide better response time if there are more Active Directory UNIX groups in the zone than the maximum you specify, but further limits how much information is returned.

If this parameter is not defined in the configuration file, its default value is 1000 groups.

## lam.max.user.count

This configuration parameter applies to the AIX Loadable Authentication Module (LAM) and specifies the maximum number of Active Directory users that the `lsuser ALL` command will return. This value also limit the results returned by the `getpwent()` and `nextuser()` functions.



The parameter value must be an integer. The default value for this parameter is 1000 users. If you specify 0 or a negative value (for example, -1), there is no limit on the number of users returned. For example:

```
Tam.max.user.count: 100
```

Before changing this parameter setting or using a value of 0, you should consider its impact on your environment. Increasing the value of this parameter may provide more complete information about the number of Active Directory UNIX users, but may result in slower performance if there are more Active Directory UNIX users in the zone than the maximum you specify. Similarly, if you do not set a limit, you may experience performance problems if you have a large number of Active Directory users. Decreasing the value of this parameter may provide better response time if there are more Active Directory UNIX users in the zone than the maximum you specify, but further limits how much information is returned.

If this parameter is not defined in the configuration file, its default value is 1000 users.



# Customizing NIS configuration parameters

This section describes the configuration parameters that affect the operation of the Centrify Network Information Service on the local host computer. The Centrify Network Information Service—`adnisd`—provides a mechanism for responding to NIS client requests from computers not managed by a Centrify agent.

<code>log.adnisd</code> .....	231
<code>log.adnisd.netgroup</code> .....	231
<code>logger.facility.adnisd</code> .....	232
<code>nisd.domain.name</code> .....	232
<code>nisd.exclude.maps</code> .....	232
<code>nisd.largegroup.name.length</code> .....	233
<code>nisd.largegroup.suffix</code> .....	233
<code>nisd.maps</code> .....	234
<code>nisd.maps.max</code> .....	235
<code>nisd.net_addr</code> .....	235
<code>nisd.passwd.expired.allow</code> .....	236
<code>nisd.port.tcp</code> .....	236
<code>nisd.port.udp</code> .....	237
<code>nisd.securenets</code> .....	237
<code>nisd.server.switch.delay</code> .....	238
<code>nisd.startup.delay</code> .....	238



nisd.threads .....	239
nisd.update.interval .....	239

## log.adnisd

This configuration parameter specifies the logging level for the Centrify Network Information Service. The default logging level is the logging level set for the `log` configuration parameter or `INFO` if neither parameter is defined in the configuration file. For example, to diagnose problems with the Centrify Network Information Service without changing the logging level for other components:

```
log.adnisd: DEBUG
```

## log.adnisd.netgroup

This configuration parameter specifies the logging level for `netgroup` processing of the Centrify Network Information Service. The default logging level is the logging level set for the `log.adnisd` parameter if that parameter is defined. This parameter value can be set to `DEBUG` to log `netgroup` diagnostics or to `INFO` to suppress messages.

For example:

```
log.adnisd.netgroup: INFO
```

You can also set lower-level logging for `netgroup` processing using the following parameters:

Use this parameter	To log
<code>log.adnisd.netgroup.syntax</code>	Syntax warnings and errors for <code>netgroup</code> processing. The default value is the value defined for the <code>log.adnisd.netgroup</code> parameter.
<code>log.adnisd.netgroup.inv</code>	Inversion processing. The default value is the value defined for the <code>log.adnisd.netgroup</code> parameter. This parameter value can be set to <code>DEBUG</code> to log <code>netgroup</code> diagnostics or to <code>INFO</code> to suppress messages.



## logger.facility.adnisd

This configuration parameter specifies the `syslog` facility to use for logging `adnisd` operations. This parameter enables you to log `adnisd` messages using a different `syslog` facility than the facilities used for logging general `adclient` messages or `adclient` audit messages. This parameter's value can be any valid `syslog` facility. For example, you can set this parameter to log messages to `auth`, `authpriv`, `daemon`, `security`, or `localn` facilities. The default is the `auth` facility. For example:

```
logger.facility.adnisd: auth
```

## nisd.domain.name

This configuration parameter specifies the NIS domain name for the `adnisd` process to use when communicating with NIS clients.

For example, to specify that you want to use `euro-all` as the NIS domain name in the zone named `Europe-00-Zone`, you can set this parameter as follows:

```
nisd.domain.name: euro-all
```

If this parameter is not defined in the configuration file, the zone name is used by default.

## nisd.exclude.maps

This configuration parameter specifies the name of the NIS maps you want to prevent the NIS service from using in response to NIS clients. This parameter enables you to exclude specific maps rather than explicitly specifying the maps you want to make available. For example, if you have a large number of `automount` maps or other network information that you want to make available to NIS clients but do not want to use agentless authentication, you can use this parameter to exclude the `passwd` and `group` maps but respond to `automount` or `netgroup` requests.

To use this configuration parameter, you must add the parameter name to the `/etc/centrifydc.conf` configuration file, then define its value. The parameter value must be a list of valid NIS map names, separated by spaces. For example:

```
nisd.exclude.maps: group passwd
```



This parameter excludes the named map and all derived maps. For example, if you specify `group`, the derived maps, `group.byname`, and `group.bygid`, are excluded. If this parameter is not defined in the configuration file, all NIS maps found in Active Directory are retrieved and available for service.

This configuration parameter overrides the setting of the `nisd.maps` parameter. If the same map is specified for both the `nisd.exclude.maps` and `nisd.maps` parameters, the map is excluded.

## `nisd.largegroup.name.length`

This configuration parameter specifies the maximum number of characters to use in group names when groups with a large number of members are split into multiple new groups. Because some devices that submit NIS requests have limitations on the length of group names, you can use this parameter to specify the maximum length for group names.

When the `adnisd` process splits the group membership for a large group into multiple smaller groups, it truncates the original group name as needed to append the suffix defined in the `nisd.largegroup.suffix` parameter and not exceed the number of characters specified by this parameter. For example, if you have a large group named `worldwide-all-corp`, and have defined the suffix string as `-all` and the maximum length for group names as 10, when the `worldwide-all-corp` group membership is split into multiple groups, the groups are named as follows:

```
world-all1  
world-all2  
world-all3  
world-all3
```

For example, to set the maximum length for group names to 20 characters:

```
nisd.largegroup.name.length: 20
```

If this parameter is not defined in the configuration file, the maximum group name length is 1024 characters by default.

## `nisd.largegroup.suffix`

This configuration parameter specifies the suffix string or character to use in group names when automatically splitting up a group with large number of members.



Because `group.bygid` and `group.byname` NIS maps can often contain membership lists that exceed the 1024 limit for how much NIS data can be served to clients, the `adnsd` process will automatically truncate the membership list when this limit is reached. To allow the additional membership data to be retrieved, you can configure the Centrifry Network Information Service to automatically split a large group into as many new groups as needed to deliver the complete membership list.

If you specify any value for the `nisd.largegroup.suffix` parameter, you enable the `adnsd` process to automatically split a large group into multiple new groups. When a group's data size exceeds 1024 data limit, a new group is created. The new group name is formed using the original group name, followed by the string defined for the `nisd.largegroup.suffix` parameter and ending in a number that represents the numeric order of the new group created.

For example, if you have a large group named `performix-worldwide-corp`, and have defined the suffix string as `-a11` and the maximum length for group names as 10, when the `performix-worldwide-corp` group membership is split into multiple groups, the groups are named as follows:

```
performix-worldwide-corp-a111  
performix-worldwide-corp-a112  
performix-worldwide-corp-a113  
performix-worldwide-corp-a114
```

All of the new groups have the same group identifier (GID) as the original group. If the new group names would exceed the maximum length for group names on a platform, you can use the `nisd.largegroup.name.length` parameter to set the maximum length for the new groups created.

If this configuration parameter is not set, the `adnsd` process truncates the group membership list such that each group entry is under 1024 characters.

## nisd.maps

This configuration parameter specifies the name of the NIS maps currently available for NIS service. When the `adnsd` daemon connects to Active Directory, it retrieves the list of NIS maps available for the local computer's zone, creates a local map data store, and updates this configuration parameter, if necessary, to indicate the maps retrieved. If any NIS client requests a map that is not in the list specified by this parameter, the daemon refuses the request.



The parameter value must be a list of NIS map names. If the parameter is included in the configuration file but no value is set, no maps are retrieved from Active Directory or available for service.

For example, to make the `netgroup` maps available, but no other maps, you can set this parameter as follows:

```
nisd.maps: netgroup,netgroup.byhost,netgroup.byuser
```

**Note:** You must specify all maps, including the derived maps.

If this parameter is not defined in the configuration file, all NIS maps found in Active Directory are retrieved and available for service.

## nisd.maps.max

This configuration parameter specifies the number of alternate sets of NIS maps to retain. A new set of NIS maps is normally created when `adnisd` switches to an alternate domain controller. Keeping these alternate sets of maps allows Centrify Network Information Service to more efficiently switch between domain controllers.

The parameter value must be an integer greater than zero. The default is 2 map sets. For example:

```
nisd.maps.max: 2
```

## nisd.net\_addr

This configuration parameter sets the IP address the `adnisd` process uses for the NIS client socket. For example, the following sets the IP address to 192.168.212.11:

```
nisd.net_addr: 192.168.212.11
```

On systems with multiple Ethernet interfaces, `adnisd` configures RPC to the first interface. If an NIS client is trying to communicate on a different interface, `adnisd` will not receive the request.

Before creating sockets, `adnisd` reads `centrifydc.conf` file to see if an IP address and TCP and UDP ports are specified. If not, it uses localhost and random port numbers assigned by the operating system.



Use the `nisd.port.udp` and `nisd.port.tcp` parameters to complete the NIS port assignment.

## `nisd.passwd.expired.allow`

This configuration parameter specifies whether a user with an expired Active Directory password should be allowed to log on to computers authenticated through NIS requests. The parameter value can be set to `true` or `false`.

By default, when a user's Active Directory password expires the password hash field in the `passwd` NIS map is replaced by two exclamation marks (`!!`), and the user is not allowed to log on to the local NIS client computer without first logging on to a Windows computer or an agent-managed computer running `adcli`ent to update the expired password. You can use this parameter to allow the user to log on locally using the expired password.

If you set the parameter value to `true`, users with an existing password hash in the `passwd` map generated from Active Directory do not have their password hash replaced by the exclamation marks and they can continue to log on using the expired password until they update their password in Active Directory. Once they update their password in Active Directory, the NIS map is updated with a new password hash and users can log on with the new password. If a user never updates the Active Directory password by logging on to a Windows or agent-managed computer, however, the user's expired password may be used indefinitely.

The default value for this parameter is `false`. For example:

```
nisd.passwd.expired.allow: false
```

## `nisd.port.tcp`

This configuration parameter sets the TCP port number the `adnisd` process uses to create the socket for NIS client communications. For example, the following sets the TCP port to 2556:

```
nisd.port.tcp: 2556
```

By default, no port number is specified. If you do not specify the port number, the operating system assigns a random port number.



Use the `nisd.port.udp` and `nisd.net_addr` parameters to complete the NIS client socket configuration.

## `nisd.port.udp`

This configuration parameter sets the UDP port number the `asnisd` process uses to create the socket for NIS client communications. For example, the following sets the UDP port to 2555

```
nisd.port.udp: 2555
```

By default, no port number is specified. If you do not specify the port number, the operating system assigns a random port number.

Use the `nisd.port.tcp` and `nisd.net_addr` parameters to complete the NIS client socket configuration.

## `nisd.securenets`

This configuration parameter specifies a list of one or more subnets from which the daemon will accept NIS requests. You use this parameter to restrict access to the Centrify Network Information Service by IP address. NIS requests that do not come from the IP addresses specified in this configuration parameters are refused by the `asnisd` daemon.

**Note:** You do not need to specify the local IP address for this parameter. The Centrify Network Information Service will always accept local NIS client requests.

The parameter value must include both the specific IP address or subnet and the subnet mask, separated by a forward slash. For example:

```
nisd.securenets: 192.168.111.0/255.255.255.0
```

You can specify multiple IP addresses by separating each IP address-subnet mask pair with a comma or a space. For example:

```
nisd.securenets:  
192.68.11.0/255.255.255.0,192.147.10.0/255.255.255.0
```

If this parameter is not defined in the configuration file, only local NIS client requests are accepted by the `asnisd` process.



## nisd.server.switch.delay

This configuration parameter specifies how long, in seconds, to wait before loading maps from a backup domain controller when the connection to the primary domain controller is lost. If the Centrify Network Information Service is unable to connect to its primary Active Directory domain controller, it will respond to NIS client requests using information in the local cache until the switch to the backup domain controller is complete.

The parameter value must be an integer equal to or greater than zero. If the value is zero, then the delay is disabled. For example, to set the delay period to 2 hours:

```
nisd.server.switch.delay: 7200
```

If this parameter is not defined in the configuration file, the default delay for switching to the backup domain controller is ten minutes (600 seconds).

## nisd.startup.delay

This configuration parameter specifies the maximum number of seconds that the `adnisd` process should wait before responding to NIS client requests.

While `adnisd` retrieves and generates its NIS maps, it does not respond to client requests for the maximum number of seconds specified by this parameter. At the end of the startup delay time, `adnisd` will respond to NIS client requests whether all maps are loaded or not. Therefore, setting this parameter enables the `adnisd` process to begin responding to NIS clients requests before all NIS maps are loaded or created. You should be aware, however, that if the delay time is reached before all of the NIS maps are available, NIS clients may receive partial or empty answers to their requests.

**Note:** If all of the NIS maps are loaded or created in less time than specified by this parameter, `adnisd` will begin responding to NIS requests without any startup delay.

By default, the maximum startup delay is 180 seconds. If you set this configuration parameter to zero, the `adnisd` process will only respond to NIS client requests after all NIS maps have been loaded or created. Therefore, in most cases, the parameter value should be a positive integer. For example, to set the startup delay to two minutes, you would set the parameter value to 120:



```
nisd.startup.delay: 120
```

## nisd.threads

This configuration parameter specifies the maximum number of threads to allocate for processing NIS client requests.

The parameter value must be a positive integer within the valid range of 1 to 200. If you want to increase or decrease the number of threads used, you should stop the `adnisd` process, modify this parameter and save the configuration file, then restart the `adnisd` process.

The default value for this parameter is 4 threads. For example:

```
nisd.threads: 4
```

## nisd.update.interval

This configuration parameter specifies the interval, in seconds, that the `adnisd` daemon waits between connections to Active Directory. At each interval, the `adnisd` daemon connects to Active Directory, gets the latest NIS maps for the local computer's zone, and updates its local NIS map data store.

The parameter value must be an integer equal to or greater than zero. If the value is zero, then the update interval is disabled and the local NIS map data store is not updated. For example, to set the interval for getting NIS maps to 1 hour:

```
nisd.update.interval: 3600
```

If this parameter is not defined in the configuration file, the default interval is 30 minutes (1800 seconds).



# Customizing AIX configuration parameters

This section describes the configuration parameters that affect the administration of users and groups on AIX computers.

Setting extended attributes .....	241
aix.cache.extended.attr.enable .....	243
aix.user.attr.admgroups .....	244
aix.user.attr.admin .....	244
aix.user.attr.auditclasses .....	244
aix.user.attr.core .....	245
aix.user.attr.cpu .....	245
aix.user.attr.data .....	245
aix.user.attr.daemon .....	246
aix.user.attr.fsize .....	246
aix.user.attr.nofiles .....	246
aix.user.attr.nprocs .....	247
aix.user.attr.rlogin .....	247
aix.user.attr.rss .....	247
aix.user.attr.stack .....	248
aix.user.attr.su .....	248
aix.user.attr.sugroups .....	248
aix.user.attr.threads .....	249



aix.user.attr.tpath .....	249
aix.user.attr.ttys .....	250
aix.user.attr.umask .....	250

## Setting extended attributes

AIX provides extended user and group attributes that enable administrators to specify user or group characteristics, such as the ability to login remotely to a user account, use the system resource controller (SRC) to execute programs, and so on. You can define these attributes for specific users and groups or for all user and group accounts on a local computer by editing specific configuration files such as `/etc/security/user`, `/etc/security/group`, and `/etc/security/limits`. The specific extended attributes available depend on the version of AIX you are using. For information about the extended attributes available for users and groups, see the AIX documentation for the security configuration files.

You can centralize administration of AIX computers by setting extended attributes for individual AIX users and groups in Active Directory. You can also set configuration parameters to set default extended attribute values for all Active Directory users or groups on a particular AIX computer.

**Note:** Certain extended attributes, such as the system privileges, or capabilities attributes, are only supported by methods in the Loadable Authentication Module (LAM) version 5.2 or later.

The agent configuration file can include AIX configuration parameters that correspond to AIX extended attributes. For example:

AIX attribute	Parameter
admin	<code>aix.user.attr.admin</code>
daemon	<code>aix.user.attr.daemon</code>
rlogin	<code>aix.user.attr.rlogin</code>
su	<code>aix.user.attr.su</code>

Each configuration parameter has a hard-coded default value. You can edit the `centrifdc.conf` configuration file on any computer to change its default value. You should note that changes you make in the `centrifdc.conf` file only affect Active Directory users and groups. The settings do not affect local users or groups. Local users and groups get their extended attributes from the



settings in the AIX configuration files, such as `/etc/security/user` and `/etc/security/limits`.

## Enforcing access rights on AIX computers

If you are using the AIX Loadable Authentication Module (LAM), users who do not have the PAM `login-all` right can still log in. For example, an Active Directory user joined to the zone with the AIX computer and assigned to a role that does NOT include the `login-all` right can, in fact, log in to the AIX servers using the LAM interface. This is because the LAM interface does not use the rights defined in the user's Centrify role to control access. If the same server is configured with the PAM authentication module, that user would not be able to log in.

To control user log in activity, you have two choices:

- Keep the LAM interface and use one of the following PAM configuration parameters to define who has or does not have access:
  - **pam.allow.groups**: This configuration parameter specifies the groups allowed to access PAM-enabled applications.
  - **pam.allow.users**: This configuration parameter specifies the users who are allowed to access PAM-enabled applications.
  - **pam.deny.groups**: This configuration parameter specifies the groups that should be denied access to PAM-enabled applications.
  - **pam.deny.users**: This configuration parameter specifies the users that should be denied access to PAM-enabled applications.
- Replace the LAM interface with PAM. See the [IBM AIX documentation](#) for the instructions. The conversion procedure is fairly simple, however, you should test all applications on the server to ensure that they work the same with PAM. In addition, if you are using Centrify OpenSSH there are two versions: one for LAM and one for PAM. Both a LAM and PAM versions are distributed in the package. If you convert to PAM, uninstall the LAM version and install the PAM version.

## Setting extended attributes

To set an extended attribute for an individual user, you can use `adedit` commands.



For example, to set the value of the extended attributes `aix.ttys` and `aix.rlogin` for the user `joe`, you might run commands similar to the following after binding to a domain and selecting a zone:

```
select_zone_user joe@ajax.centriky.test
set_zone_user_field aix.ttys r1,r2,r3
set_zone_user_field aix.rlogin true
```

To verify the value of the extended attributes you have set, you might run commands similar to the following:

```
get_zone_user_field aix.ttys
r1,r2,r3
save_zone_user
```

You can also use `adedit` abbreviations to set and get extended attribute values. For example:

```
s1zu joe@ajax.centriky.test
szuf aix.fsize 209715
szuf aix.core 2097151
szuf aix.cpu -1
szuf aix.data 262144
```

Alternatively, you can also use configuration parameters to supplement the settings in the AIX `/etc/security/user` file. For example, if you have not explicitly defined the `aix.rlogin` attribute in `/etc/security/user`, you can set the following parameter in the `centrikydc.conf` file:

```
aix.user.attr.rlogin: false
```

You can use `adquery` and the keyword `help` to view a list of the supported extended attributes. For example:

```
adquery user --extattr help
```

## **aix.cache.extended.attr.enable**

Use this parameter to specify whether to cache extended attribute default values. Caching extended attribute default values improves performance of the `Isuser` command.

The default value is `false`.



## `aix.user.attr.admgroups`

This configuration parameter specifies the groups that the user account administers.

For the parameter value, enter a comma-separated list of groups; for example:

```
aix.user.attr.admgroups: unixAdmins,dnsAdmins
```

This parameter corresponds to the `aix.admingroups` attribute in the `/etc/security.user` file.

The default value is the empty string (no groups).

## `aix.user.attr.admin`

This configuration parameter specifies the administrative status of the user.

Set the parameter value to `true` to define the user as an administrator; for example:

```
aix.user.attr.admin: true
```

Set the value to `false` to specify that the user is not an administrator. This is the default value.

This parameter corresponds to the `aix.admin` attribute in the `/etc/security.user` file.

## `aix.user.attr.auditclasses`

This configuration parameter specifies the audit classes for the user.

You may enter a list of audit classes separated by commas, or the keyword `ALL` or an asterisk (`*`) to specify all audit classes. For example:

```
aix.user.attr.auditclasses: general,system
```

Place an exclamation point in front of a class to exclude it. For example, the following setting specifies all classes except `system`:

```
aix.user.attr.auditclasses: ALL,!system
```

This parameter corresponds to the `aix.auditclasses` attribute in the `/etc/security.user` file.



The default value is the empty string (no audit classes).

## **aix.user.attr.core**

This configuration parameter specifies the soft limit for the largest core file that the user can create. Use -1 to set an unlimited size.

For example, to set the value to 2097151:

```
aix.user.attr.core: 2097151
```

This parameter corresponds to the `aix.core` attribute in the `/etc/security.limits` file.

The default value is 2097151.

## **aix.user.attr.cpu**

This configuration parameter specifies the soft limit (in seconds) for the amount of system time that a user's process can use.

Use -1 to set an unlimited size; for example, to set the limit to one hour:

```
aix.user.attr.cpu: 3600
```

This parameter corresponds to the `aix.cpu` attribute in the `/etc/security.limits` file.

The default value is -1.

## **aix.user.attr.data**

This configuration parameter specifies the soft limit for the largest data-segment for a user's process. Use -1 to set an unlimited size.

For example, to set the value to 2097151:

```
aix.user.attr.data: 2097151
```

This parameter corresponds to the `aix.data` attribute in the `/etc/security.limits` file.

The default value is 2097151.



## **aix.user.attr.daemon**

This configuration parameter specifies whether the user can execute programs using the system resource controller (SRC), which manages system daemons and sub systems such as `adcli`ent and NFS.

Set the parameter value to `true` to allow users to execute programs using the SRC. Set the value to `false` to prevent users from executing programs using the SRC.

This parameter corresponds to the `aix.daemon` attribute in the `/etc/security.user` file.

The default value is `false`, which prevents a user from executing programs using SRC.

## **aix.user.attr.fsize**

This configuration parameter specifies the soft limit for the largest file that the user process can create. Use `-1` to set an unlimited size.

For example, to set the value to 2097151:

```
aix.user.attr.fsize: 2097151
```

This parameter corresponds to the `aix.fsize` attribute in the `/etc/security.limits` file.

The default value is 2097151.

## **aix.user.attr.nfiles**

This configuration parameter specifies the soft limit for the number of file descriptors that the user's process may have open at one time.

Use `-1` to set an unlimited size.

For example, to set the limit to 2000:

```
aix.user.attr.nfiles: 2000
```

This parameter corresponds to the `aix.nfiles` attribute in the `/etc/security.limits` file.

The default value is `-1`.



## **aix.user.attr.nprocs**

This configuration parameter specifies the soft limit on the number of processes a user can have running at one time.

Use -1 to specify the maximum number allowed by the system; for example:

```
aix.user.attr.nprocs: -1
```

This parameter corresponds to the `aix.nprocs` attribute in the `/etc/security/limits` file.

The default value is -1.

## **aix.user.attr.rlogin**

This configuration parameter specifies whether remote users can access the user account through `rlogin` and `telnet`.

Set the parameter value to `true` to allow remote access to the user account.

Set the parameter value to `false` to prevent remote access to the user account.

This parameter corresponds to the `aix.rlogin` attribute in the `/etc/security.user` file.

The default value is `true`, which allows remote access to the user account.

## **aix.user.attr.rss**

This configuration parameter specifies the soft limit for the largest amount of system memory that the user process can allocate. Use -1 to set an unlimited size.

For example, to set the value to 2097151:

```
aix.user.attr.rss: 2097151
```

This parameter corresponds to the `aix.rss` attribute in the `/etc/security.limits` file.

The default value is 65536.



## **aix.user.attr.stack**

This configuration parameter specifies the soft limit for the largest stack segment for the user's process. Use `-1` to set an unlimited size.

For example, to set the value to `2097151`:

```
aix.user.attr.stack: 2097151
```

This parameter corresponds to the `aix.stack` attribute in the `/etc/security.limits` file.

The default value is `65536`.

## **aix.user.attr.su**

This configuration parameter specifies whether other users can use the `su` command to switch to this user account.

Set the parameter value to `true` to allow other users to switch to this user account.

Set the value to `false` to prevent users from switching to this user account.

This parameter corresponds to the `aix.su` attribute in the `/etc/security.user` file.

The default value is `true`, which allows other users to switch to this user account.

## **aix.user.attr.sugroups**

This configuration parameter specifies the groups that can use the `su` command to switch to this user account.

You may enter a list of groups separated by commas, or the keyword `ALL` or an asterisk (`*`) to specify all groups. For example:

```
aix.user.attr.sugroups:  
admins,unixAdmins,dnsAdmins,enterpriseAdmins
```

Place an exclamation point in front of a group to exclude it. For example, the following setting specifies all groups except `dnsAdmins`:

```
aix.user.attr.sugroups: ALL,!dnsAdmins
```



This parameter corresponds to the `aix.sugroups` attribute in the `/etc/security.user` file.

The default value is `ALL`, which allows all groups to switch to the user account.

## `aix.user.attr.threads`

This configuration parameter specifies the soft limit for the largest number of threads that a user process can create.

Use `-1` to specify an unlimited number; for example:

```
aix.user.attr.threads: -1
```

This parameter corresponds to the `aix.thread` attribute in the `/etc/security/limits` file.

The default value is `-1`, which specifies an unlimited number of threads.

## `aix.user.attr.tpath`

This configuration parameter specifies the status of the user's trusted path. The trusted path prevents unauthorized programs from reading data from the user terminal.

Set one of the following values for this parameter:

Use this value	To do this
<code>always</code>	Allows the user to execute trusted processes only, which means the that the user's initial program must be in the trusted shell or another trusted process.
<code>notsh</code>	Prevents the user from invoking the trusted shell on a trusted path. Entering the secure attention key (SAK) causes the login session to terminate.
<code>nosak</code>	Disables the secure attention key (SAK) for all processes run by the user. Specify <code>nosak</code> if the user transfers binary that may contain the SAK. This is the default value.
<code>on</code>	Provides the user with normal trusted path characteristics; the user can invoke a trusted path (enter a trusted shell) with the secure attention key (SAK).

This parameter corresponds to the `aix.tpath` attribute in the `/etc/security.user` file.

The default value is `nosak`.



## `aix.user.attr.tty`s

This configuration parameter specifies the terminals that can access the user account.

You may enter a list of terminals separated by commas, or the keyword ALL or an asterisk (\*) to specify all terminals. For example:

```
aix.user.attr.tty: /dev/pts
```

**Note:** You must specify /dev/pts or ALL for network logins to work.

Place an exclamation point in front of a group to exclude it.

This parameter corresponds to the `aix.tty`s attribute in the `/etc/security.user` file. The default value is ALL, which allows all terminals to access the user account.

## `aix.user.attr.umask`

This configuration parameter specifies the default umask to define permissions for the user. The umask value along with the permissions of the creating process determine the permissions for a new file.

This parameter corresponds to the `aix.tty`s attribute in the `/etc/security.user` file.

The parameter value can be set to a three-digit octal value. The default value is 022.

# Customizing Centrifify UNIX programs configuration parameters

This section describes the configuration parameters that affect the operation of Centrifify UNIX command line programs on the local host computer.

<code>adjoin.adclient.wait.seconds</code> .....	251
<code>adjoin.krb5.conf.file</code> .....	252
<code>adjoin.samaccountname.length</code> .....	252
<code>adpasswd.account.disabled.mesg</code> .....	253
<code>adpasswd.account.invalid.mesg</code> .....	253
<code>adpasswd.password.change.disabled.mesg</code> .....	253
<code>adpasswd.password.change.perm.mesg</code> .....	253

## `adjoin.adclient.wait.seconds`

This configuration parameter specifies the number of seconds the `adjoin` command should wait before exiting to ensure that the agent is available to complete the join operation.

For example, to configure the `adjoin` command to wait 10 seconds:

```
adjoin.adclient.wait.seconds: 10
```



## adjoin.krb5.conf.file

This configuration parameter specifies the path to a customized Kerberos configuration file you want to use to join a domain.

The parameter value must be a path name. For example:

```
adjoin.krb5.conf.file: /etc/centrifydc/krb5_join.conf
```

## adjoin.samaccountname.length

This configuration parameter specifies the maximum number of characters to use when the `adjoin` command must generate a pre-Windows 2000 computer name by truncating the host name. This parameter also determines how `adjoin` creates the computer account in Active Directory.

The default value is 15 characters to conform to the maximum length allowed by the NetLogon service, which is the preferred service for `adcli` to use for NTLM pass-through authentication. NetLogon is fast and automatically returns a user's group membership.

The maximum length allowed for the pre-Windows 2000 computer name, which is stored in the `sAMAccountName` attribute for the computer account in Active Directory, is 19 characters. However, if you specify more than 15 characters (up to the 19 character limit) `adcli` will use slower NTLM authentication methods, and will use additional LDAP searches to fetch the user's group membership.

**Note:** This configuration parameter is ignored if you run the `adjoin` command with the `--prewin2k` option to manually specify the pre-Windows 2000 computer name.

The parameter value should be a positive integer in the valid range of 1 to 19 characters. For example:

```
adjoin.samaccountname.length: 15
```

If you specify a value greater than 19, the parameter setting is ignored and the computer name is truncated at 19 characters in the `sAMAccountName` attribute for the computer account.

If the computer's host name size exceeds the specified value for this parameter, `adjoin` will use LDAP (and require administrative privileges) to create computer accounts, instead of MS-RPC. In any case, if the computer's short



host name exceeds 19 characters, then it is no longer possible to create computer accounts by using MS-RPC methods and LDAP will be used instead.

## adpasswd.account.disabled.msg

This configuration parameter specifies the message displayed by the adpasswd program when users cannot change their password because their account is locked.

For example:

```
adpasswd.account.disabled.msg:  
Account cannot be accessed at this time. Please contact your  
system administrator.
```

## adpasswd.account.invalid.msg

This configuration parameter specifies the message displayed by the adpasswd program when a user account is unrecognized or the password is invalid.

For example:

```
adpasswd.account.invalid.msg: \  
Invalid username or password
```

## adpasswd.password.change.disabled.msg

This configuration parameter specifies the message displayed by the adpasswd program when users are not allowed to change their password because password change for these users has been disabled in Active Directory.

For example:

```
adpasswd.password.change.disabled.msg: \  
Password change for this user has been disabled in Active  
Directory
```

## adpasswd.password.change.perm.msg

This configuration parameter specifies the message displayed by the adpasswd program when a user cannot change another user's password because of



insufficient permissions.

For example:

```
adpasswd.password.change.perm.mesg:  
You do not have permission to change this users password. Please  
contact your system administrator.
```

# Customizing smart card configuration parameters

This section describes the configuration parameters that affect the use of Centrify access control smart cards on the local host computer.

<code>rhel.smartcard.pkcs11.module</code> .....	255
<code>smartcard.allow.noeku</code> .....	256
<code>smartcard.name.mapping</code> .....	256

## `rhel.smartcard.pkcs11.module`

This configuration parameter specifies the path to the PKCS #11 module to be used by smart card components on the computer.

By default, smart card components use the Centrify Coolkey PKCS #11 module. However, Coolkey does not support all smart cards so you may specify a different module if necessary by specifying the absolute path to your PKCS #11 module with this parameter. For example:

```
rhel.smartcard.pkcs11.module /usr/$LIB/pkcs11/opensc-pkcs11.so
```

**Note:** In the path specification, this parameter supports the use of the `$LIB` environment variable, which allows a single path specification to work for 32-bit and 64-bit systems. At run time, on 32-bit systems, `$LIB` resolves to `lib`, while on 64-bit systems it resolves to `lib64`.

After changing the value of this parameter, you must re-enable smart card support by running the following `sctool` commands as root:

```
[root]$ sctool --disable  
[root]$ sctool --enable
```



Also, refresh the GNOME desktop by running the following command as root:

```
[root]$ /usr/sbin/gdm-safe-restart
```

In most cases, you set this configuration parameter using group policy.

## smartcard.allow.noeku

This configuration parameter allows the use of certificates that do not have the Extended Key Usage (EKU) attribute. Normally, smart card use requires certificates with the EKU attribute. The value of this parameter can be `true` or `false`.

If you set this parameter to `true`, certificates without an EKU attribute can be used for SmartCard logon, and certificates with the following attributes can also be used to log on with a smart card:

- Certificates with no EKU
- Certificates with an All Purpose EKU
- Certificates with a Client Authentication EKU

If you set this parameter to `false`, only certificates that contain the smart card logon object identifier can be used to log on with a smart card. The default value of this parameter is `false`.

After changing the value of this parameter, you must re-enable smart card support by running the following `sctool` command as root:

```
[root]$ sctool -E
```

When you run `sctool` with the `-E` option, you must also specify the `-a` or `-k` option. You can also control this feature using group policy.

## smartcard.name.mapping

This configuration parameter turns on support for multi-user smart cards.

By default, this parameter is set to `False`, which prevents the use of multi-user smart cards.

To allow the use of multi-user smart cards, set this parameter to `True`:

```
smartcard.name.mapping: true
```



# Customizing authorization configuration parameters

This section describes the configuration parameters that affect the operation of authorization features (privilege elevation service) on the local host computer. You can configure authorization rules by defining specific command- or application-level rights, combining those rights into roles, and assigning users to those roles to control the operations they are allowed to perform on specific computers in a zone.

adclient.azman.refresh.interval .....	258
adclient.cache.flush.interval.dz .....	259
adclient.dzdo.clear.passwd.timestamp .....	259
adclient.refresh.interval.dz .....	260
adclient.sudo.clear.passwd.timestamp .....	260
adclient.sudo.timestampdir .....	262
audittrail.dz.command.with.args .....	262
dz.auto.anchors .....	262
dz.enabled .....	263
dz.system.path .....	263
dz.user.path .....	264
dzdo.always_set_home .....	264
dzdo.badpass_message .....	264
dzdo.command_alias .....	265
dzdo.edit.checkdir .....	265
dzdo.edit.follow .....	266



dzdo.env_check .....	266
dzdo.env_delete .....	267
dzdo.env_keep .....	268
dzdo.lecture .....	268
dzdo.lecture_file .....	269
dzdo.legacyzone.mfa.enabled .....	269
dzdo.log_good .....	269
dzdo.passprompt .....	270
dzdo.passwd_timeout .....	270
dzdo.path_info .....	271
dzdo.search_path .....	271
dzdo.requiretty .....	272
dzdo.secure_path .....	272
dzdo.set_home .....	273
dzdo.set.runas.explicit .....	273
dzdo.timestampdir .....	274
dzdo.timestamp_timeout .....	275
dzdo.tty_tickets .....	275
dzdo.use.realpath .....	275
dzdo.user.command.timeout .....	276
dzdo.validator .....	276
dzdo.validator.required .....	277
dzsh.roleswitch.silent .....	277

## adclient.azman.refresh.interval

This configuration parameter is deprecated and is replaced by the `adclient.refresh.interval.dz` parameter. See [adclient.refresh.interval.dz](#)



for details about `adclient.refresh.interval.dz`.

The Centrify Authentication Service, Privilege Elevation Service, and Audit & Monitoring Service upgrade utility renames this parameter if it is being used.

## `adclient.cache.flush.interval.dz`

This configuration parameter specifies the frequency (in seconds) with which the Centrify UNIX agent flushes its authorization cache. You should note that this parameter only forces periodic updates to the authorization cache. It does not affect the agent's primary domain controller cache.

The default value is 0, which completely disables periodic flushing of the authorization cache.

The parameter value must be a positive integer. For example, to force the authorization cache to be cleared every 30 minutes, set the parameter as follows:

```
adclient.cache.flush.interval.dz: 1800
```

## `adclient.dzdo.clear.passwd.timestamp`

This configuration parameter specifies whether users must re-authenticate with `dzdo` after logging out.

When a user authenticates with `dzdo`, a ticket is temporarily created that allows `dzdo` to run without re-authentication for a short period of time (set by the `dzdo.timestamp_timeout` parameter). If a user logs out, the ticket is reused when the user logs back in.

The parameter value can be `true` or `false`. Setting this parameter to `true` clears the ticket and requires users to re-authenticate to use `dzdo` after logging out and back in. The default parameter value is `false`.

For example:

```
adclient.dzdo.clear.passwd.timestamp: true
```

You can also set this parameter using group policy.



## adclient.refresh.interval.dz

**Note:** Starting with agent version 5.1.3, this configuration parameter replaces the deprecated `adclient.azman.refresh.interval` parameter.

This configuration parameter specifies the maximum number of minutes to keep access control information from the authorization store cached before refreshing the data from Active Directory. Access control information consists of rights, roles, and role assignments that the Centrify Privilege Elevation Service uses to control access to `dzdo` privileged commands, `dzsh` restricted environments, PAM-enabled applications, and some third-party application.

Because the agent handles connecting to and retrieving information from Active Directory, this configuration parameter controls how frequently `adclient` checks for updates to the privilege elevation service set of information from Active Directory. If any privilege elevation service information has been modified, the cache is refreshed with the new information.

If local account management is enabled, this configuration parameter also specifies how often `etc/group` and `etc/passwd` are updated on individual computers based on the local group and local user settings that you configure in Access Manager.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

If you are manually setting this parameter, the parameter value must be a positive integer. The following example sets the cache expiration time to 30 minutes:

```
adclient.refresh.interval.dz: 30
```

If this parameter is not defined in the configuration file, its default value is 30 minutes.

## adclient.sudo.clear.passwd.timestamp

This configuration parameter is used together with the `tty_tickets` parameter in the `sudoers` configuration file (`/etc/sudoers`) to specify whether users must re-authenticate with `sudo` after logging out.



When a user authenticates with `sudo`, a ticket is temporarily created that allows `sudo` to run without re-authentication for a short period of time. If a user logs out and the ticket is not cleared, the ticket is reused when the user logs back in, and the user does not need to re-authenticate. If a user logs out and the ticket is cleared, the user must re-authenticate with `sudo` when logging back in.

Starting with release 2015, the way that you configure whether re-authentication is required depends on the `tty_tickets` parameter in the `sudoers` configuration file (`/etc/sudoers.conf`). In some situations, re-authentication requirements are also controlled by this parameter. Details are as follows:

- If `tty_tickets` is enabled, tickets are always removed when a `sudo` user logs out, regardless of whether this parameter is set to `true` or `false`. That is, when `tty_tickets` is enabled, this parameter has no effect, and `sudo` users must always re-authenticate.
- If `tty_tickets` is disabled, the requirement for `sudo` users to re-authenticate is controlled by this parameter and the **Force sudo re-authentication when relogin** group policy.

Tickets are cleared, and `sudo` re-authentication is required, under these scenarios:

- The `tty_ticket` parameter in the `sudoers` configuration file is enabled (it is enabled by default), or
- The `tty_ticket` parameter in the `sudoers` configuration file is disabled and the `adcliient.sudo.clear.passwd.timestamp` parameter is set to `true`, or
- The `tty_ticket` parameter in the `sudoers` configuration file is disabled and the **Force sudo re-authentication when relogin** group policy is enabled.

Tickets are not cleared, and `sudo` re-authentication is not required, under these scenarios:

- The `tty_ticket` parameter in the `sudoers` configuration file is disabled and the `adcliient.sudo.clear.passwd.timestamp` parameter is set to `false`, or
- The `tty_ticket` parameter in the `sudoers` configuration file is disabled and the **Force sudo re-authentication when relogin** group policy is disabled.

The default parameter value is `false`.



For example:

```
adclient.sudo.clear.passwd.timestamp: false
```

You can also set this parameter using group policy.

## adclient.sudo.timestampdir

This configuration parameter specifies the directory where authentication tickets reside. By default, the directory is `/var/run/sudo`. Some platforms use a different directory for tickets, such as `/var/db/sudo/user` (RHEL) or `/var/lib/sudo/user` (Ubuntu), which you can specify in this parameter.

The default value of this parameter is `/var/run/sudo`.

For example:

```
adclient.sudo.timestampdir: /var/run/sudo
```

## audittrail.dz.command.with.args

This configuration parameter specifies whether to show command parameters in the audit log for `dzdo` and `dzsh` or just the command name. The default (`false`) is to show only the command name. For example, to keep passwords entered on the command line out of the log, leave this parameter set to `false`.

Set to `true` to show the command parameters as well as the command name.

For example:

```
audittrail.dz.command.with.args: true
```

## dz.auto.anchors

This configuration parameter specifies whether you want to add anchors (`$`) automatically to the regular expressions you define as command rights and use in role definitions. The default setting is `true` to avoid matching unintended paths or commands if the regular expression pattern is not carefully set. If you set this parameter to `false`, you should carefully review all regular expressions used as command rights to identify all possible matches for the pattern defined.

For example:



```
dz.auto.anchors: true
```

## dz.enabled

This configuration parameter is only applicable for classic zones to specify whether authorization services are enabled or disabled. In hierarchical zones, which must have agents version 5.x or later, this parameter is not applicable and is ignored. In classic zones, however, authorization is an optional feature that can be explicitly enabled or disabled.

In classic zones, users can log on as long as they have a profile in a zone. In hierarchical zones, users must be assigned to a role that grants them permission to log on. If you have agents that are joined to a classic zone, you can set this parameter to `false` to explicitly prevent the agent from looking up authorization information to reduce network traffic.

If you have agents from version 4.x, the default value for this parameter is `true`. This parameter is not defined for agents version 5.x and later.

For example:

```
dz.enabled: false
```

## dz.system.path

This configuration parameter specifies the list of common System paths for locating commands in the local operating environment. The paths specified for this parameter define the program locations searched when the System match path option is selected for `dzdo` and `dzsh` commands.

This configuration parameter enables an administrator to define rights to run commands found in the user's path, the system path, or a specific location, even though the default or most commonly used paths may be different in different operating environments.

The default value for this parameter lists the most common locations for finding command line programs in the system path. For example:

```
dz.system.path: "/sbin:/usr/sbin:/usr/local/sbin"
```



## dz.user.path

This configuration parameter specifies the list of common User paths for locating commands in the local operating environment. The paths specified for this parameter define the program locations searched when the User match path option is selected for dzdo and dzsh commands.

This configuration parameter enables an administrator to define rights to run commands found in the user's path, the system path, or a specific location, even though the default or most commonly used paths may be different in different operating environments.

The default value for this parameter lists the most common locations for finding command line programs in the user's path. For example:

```
dz.user.path: "/bin:/usr/bin:/usr/local/bin"
```

## dzdo.always\_set\_home

This configuration parameter specifies whether privileged commands run with dzdo commands should set the HOME environment variable to the home directory of the target user (which is root by default). The parameter value can be true or false. Setting this parameter to true effectively implies that the -H command line option should always be used. The default parameter value is false.

For example:

```
dzdo.always_set_home: false
```

This configuration parameter provides functionality equivalent to the always\_set\_home flag for configuring the sudoers file and sudo operation.

You can also set this parameter using group policy.

## dzdo.badpass\_message

This configuration parameter specifies the message that should be displayed if a user enters an incorrect password. The parameter value can be any text string enclosed by quotation marks.

For example:



`dzdo.badpass_message`: “The password provided is not valid.”

The default value is “Sorry, try again.”

This configuration parameter provides functionality equivalent to the `badpass_message` flag for configuring the `sudoers` file and `sudo` operation.

You can also set this parameter using group policy.

## `dzdo.command_alias`

This configuration parameter specifies a mapping file containing mappings between command aliases and command files for all of the command aliases that a customer uses. If you specify a mapping file in `dzdo.command_alias` and then issue a `dzdo` command using a command alias, `dzdo` searches the mapping file to see if the first `dzdo` parameter matches any of the aliases.

If there is a match, the command path specified for the alias in the mapping file is used by `dzdo` to perform command matching to determine whether the command is allowed to run.

The parameter value has the following syntax:

```
dzdo.command_alias: aliasfile_full_pathname
```

For example, the following line in `centrifdc.conf` results in the default mapping file (`dzdo.commandalias.map`) being used:

```
dzdo.command_alias: /etc/centrifdc/dzdo.commandalias.map
```

The syntax of the content within the mapping file is:

```
command_alias_1: command_path [arguments]
command_alias_n: command_path [arguments]
```

For example, a mapping file could contain the following, which defines two command aliases—`oracle_startup` and `centrifdc_startup`:

```
oracle_startup: /opt/oracle/startup
centrifdc_startup: /opt/centrifdec/startup
```

Actual mapping files can contain any number of aliases.

## `dzdo.edit.checkdir`

This configuration parameter prevents a user from editing files using the `dzedit` command in a directory that the user already has permissions to edit



using their own account.

If a user who can already write to a directory with their own account uses `dzedit` to edit the same directory, it is possible that they may unintentionally edit arbitrary files in the directory if wild cards are used to specify the files the user intends to edit.

Set this configuration parameter to `true` to specify whether `dzedit` will check the user's directory permissions, and deny the user the ability to modify files in the directory when they run as root if they have sufficient permissions to edit the directory using their own account.

For example:

```
dzdo.edit.checkdir: true
```

The default value of this configuration parameter is `true`.

## dzdo.edit.follow

This configuration parameter prevents users from editing a file in a directory using `dzedit` that is reached by following a symbolic link (`symlink`) if the user already has permissions to edit the directory. Edits are also prevented on all sub-directories on the file path.

In some cases, if a user that already has permissions to write to a directory but invokes `dzedit` to edit a file in that directory which contains a `symlink`, they may edit the linked file as well.

If set to `false`, this configuration parameter will not allow a user to edit files reached by symbolic link by using `dzedit`.

It is strongly recommended that you keep the specified value as `false`.

For example:

```
dzdo.edit.follow: false
```

The default value of this configuration parameter is `false`.

## dzdo.env\_check

This configuration parameter specifies the list of environment variables that the `dzdo` process should check for the special characters, `%` or `/`, in the value. If the



dzdo process finds environment variable values containing the special characters, it removes those variables from the user's environment. Variables with % or / characters are removed regardless of whether you have selected the **Reset environment variables** option for the command in Access Manager.

The default list of variables to check is displayed when you run `dzdo -v` command as root. You can customize the list by modifying this configuration parameter in the `centrifdc.conf` file.

The parameter value can be a comma-separated list of environment variable names.

For example:

```
dzdo.env_check: COLORTERM, LANG, LANGUAGE, LC_*, LINGUAS, TERM
```

This configuration parameter provides functionality equivalent to the `env_reset` flag for configuring the `sudoers` file and `sudo` operation.

You can also set this parameter using group policy.

## dzdo.env\_delete

This configuration parameter specifies the default list of environment variables to be removed from the user's environment. This configuration parameter only applies if you have selected the **Remove unsafe environment variables** option for the command in the Access Manager. The variables specified with this parameter are removed in addition to the default list of variables displayed when you run the `dzdo -v` command as root.

The parameter value can be a comma-separated list of environment variable names.

For example:

```
dzdo.env_delete: IFS, CDPATH, LOCALDOMAIN, RES_OPTIONS, HOSTALIASES, \
NLSPATH, PATH_LOCALE, LD_*, _RLD*, TERMINFO, TERMINFO_DIRS, \
TERMPATH, TERMCAP, ENV, BASH_ENV, PS4, GLOBIGNORE, SHELLOPTS, \ JAVA_
TOOL_OPTIONS, PERLIO_DEBUG, PERLLIB, PERL5LIB, \
PERL5OPT, PERL5DB, FPATH, NULLCMD, READNULLCMD, ZDOTDIR, TMPPREFIX, \
PYTHONHOME, PYTHONPATH, PYTHONINSPECT, RUBYLIB, RUBYOPT, KRB5_CONFIG, \
KRB5_KTNAME, VAR_ACE, USR_ACE, DLC_ACE, SHLIB_PATH, LDR_*, \
LIBPATH, DYLD_*
```

This configuration parameter provides functionality equivalent to the `env_delete` flag for configuring the `sudoers` file and `sudo` operation.

You can also set this parameter using group policy.



## dzdo.env\_keep

This configuration parameter specifies the default list of environment variables to preserve in the user's environment. This configuration parameter only applies if you have selected the **Reset environment variables** option for the command in the Access Manager. The variables specified with this parameter are preserved in addition to the default list of variables displayed when you run the `dzdo -v` command as root.

The parameter value can be a comma-separated list of environment variable names.

For example:

```
dzdo.env_keep: COLORS,DISPLAY,HOME,HOSTNAME,KRB5CCNAME,LS_
COLORS,MAIL,PATH,PS1,PS2,TZ,XAUTHORITY,XAUTHORIZATION
```

This configuration parameter provides functionality equivalent to the `env_keep` flag for configuring the `sudoers` file and `sudo` operation.

You can also set this parameter using group policy.

## dzdo.lecture

This configuration parameter specifies whether `dzdo` displays a warning message about using the program before displaying the password prompt. The valid parameter values are:

Use this value	To do this
<code>once</code>	To display the warning message only the first time the command is run.
<code>never</code>	To never display a warning message.
<code>always</code>	To display the warning message every time the program is invoked.

The default parameter value is `once`. For example:

```
dzdo.lecture: once
```

This configuration parameter provides functionality equivalent to the `lecture` flag for configuring the `sudoers` file and `sudo` operation.

You can also set this parameter using group policy.



## **dzdo.lecture\_file**

This configuration parameter specifies the full path to a file containing the warning message you want displayed. If this parameter is not set, a default message is displayed.

For example, to use a custom message in the file `dzdo_warning`:

```
dzdo.lecture_file: /etc/custom/dzdo_warning
```

This configuration parameter provides functionality equivalent to the `lecture_file` flag for configuring the `sudoers` file and `sudo` operation.

You can also set this parameter using group policy.

## **dzdo.legacyzone.mfa.enabled**

Enable this configuration parameter to require multi-factor authentication for users to run the `dzdo` command. If you enable this parameter, users will be required to authenticate with MFA if they are required to re-authenticate to run `dzdo`, and are listed in either `adcli.legacyzone.mfa.required.users` or `adcli.legacyzone.mfa.required.groups`.

You must enable `adcli.legacyzone.mfa.enabled` for this policy to take effect.

This configuration parameter does not support rescue rights; users listed in `adcli.legacyzone.mfa.rescue.users` will not be able to run `dzdo` without MFA.

To enable this policy, set this parameter to `true`. The default value for this parameter is `false`.

For example:

```
dzdo.legacyzone.mfa.enabled: true
```

## **dzdo.log\_good**

This configuration parameter specifies whether you want to log messages for successful command execution. By default, the `dzdo` program logs both valid and invalid command execution. To log information about only invalid



command execution, set this parameter to `false`. The default value for this parameter is `true`.

For example:

```
dzdo.log_good: true
```

The `dzdo` program typically logs messages to the file `/var/log/secure`.

You can also set this parameter using group policy.

## dzdo.passprompt

This configuration parameter lets you specify the password prompt displayed when running privileged commands. This parameter serves the same function as the `dzdo -p` command.

You can use the following escapes in the prompt:

Escape	Description
<code>%u</code>	Expands to the invoking user's login name
<code>%U</code>	Expands to the login name of the user the command will be run as. If not specified, defaults to root
<code>%h</code>	Expands to the local hostname without the domain name
<code>%H</code>	Expands to the local hostname including the domain name
<code>%p</code>	Expands to the user whose password is asked for
<code>%%</code>	Collapses to a single <code>%</code> character

The default prompt is `[dzdo] password for %p:` where `%p` is root unless specified otherwise.

For example,

```
dzdo.passprompt: "[dzdo] Enter password for %U@%h"
```

You can also set this parameter using group policy.

## dzdo.passwd\_timeout

This configuration parameter specifies the number of minutes before the `dzdo` password prompt times out. The default parameter value is 5 minutes. You can set this parameter to zero (0) to have the password prompt never timeout.

For example:



```
dzdo.passwd_timeout: 5
```

This configuration parameter provides functionality equivalent to the `passwd_timeout` flag for configuring the sudoers file and sudo operation.

You can also set this parameter using group policy.

## dzdo.path\_info

This configuration parameter specifies whether the dzdo program should inform the user when it cannot find a command in the user's PATH. By default, the parameter value is true and the program will display an error statement indicating that the command could not be found in the user's PATH. You can set this configuration parameter to false if you want to prevent dzdo from indicating whether a command was not allowed or simply not found.

For example:

```
dzdo.path_info: true
```

This configuration parameter provides functionality equivalent to the `path_info` flag for configuring the sudoers file and sudo operation.

You can also set this parameter using group policy.

## dzdo.search\_path

This configuration parameter specifies the search path for the dzdo program to use to look for commands and scripts that require privileges to run. You can specify a list of directories for the dzdo program to search for commands and scripts. If you configure this parameter, the dzdo program will search in the specified directories no matter which path the command rights are configured to use in the Access Manager **System search path** option.

If commands are configured to use the path defined in the Access Manager **System search path** option and the `dzdo.search_path` parameter is not defined, the following actions take place:

- The current user's path is used to search for the commands.
- Only the commands located under the System path are allowed to execute.

There is no default value for this parameter.



The parameter value can be a list of directories or the name of a file that contains the list of directories. For example, you can specify a file that contains the directories to search using the `file: keyword` and a file location:

```
dzdo.search_path: file:/etc/centrifdc/customized_dzdo_
directories
```

If you specify a file name for this parameter, you should ensure the file is owned by root and not accessible to any other users.

You can also set this parameter using group policy.

## dzdo.requiretty

This configuration parameter specifies whether a user needs to be logged in to a valid tty session in order to run dzdo. If you set this parameter to true, this means that the user can run dzdo only from a login session and not from a cgi-bin or cron(8) script.

By default, this parameter is set to false so that it's not required to run a tty session in order to run dzdo.

## dzdo.secure\_path

This configuration parameter specifies the path for the dzdo program to use when executing commands and scripts that require privileges to run. If you specify a directory using this parameter, the dzdo program will only execute commands and scripts that are found in that directory.

Setting both the `dzdo.search_path` and `dzdo.secure_path` parameters to the same value is equivalent to setting the `secure_path` parameter in the `sudoers` configuration file.

There is no default value for this parameter.

The parameter value can be a list of directories or the name of a file that contains the list of directories. For example, you can specify a file that contains the directories to search using the `file: keyword` and a file location:

```
dzdo.secure_path: file:/etc/centrifdc/customized_dzdo_
directories
```

Within the file, lines should contain path separated by colons. For example, a file specifying two paths might look like this:



```
/etc/centrifydc/reports/exec_report_cmds:/usr/sbin/ora_cmds
```

If you specify a file name for this parameter, you should ensure the file is owned by root and not accessible to any other users.

You can also set this parameter using group policy.

## dzdo.set\_home

This configuration parameter sets the HOME environment variable to the home directory of the target user when the `-s` command line option is used. The parameter value can be `true` or `false`. The default parameter value is `false`.

For example:

```
dzdo.set_home: false
```

This configuration parameter provides functionality equivalent to the `set_home` flag for configuring the `sudoers` file and `sudo` operation.

You can also set this parameter using group policy.

## dzdo.set.runas.explicit

This configuration parameter specifies whether a user must explicitly identify the 'runas' user when executing a command with `dzdo`.

The parameter value can be `true` or `false`; the default value is `true`.

When the parameter value is `true`, if a user executes a command with `dzdo` and does not explicitly identify the user or group to run as (with the `-u` or `-g` option), `adcli` assumes that the command should be run as root. If the user is not authorized to run the command as root, `dzdo` fails to execute the command and issues an error message; for example:

```
User u1 is authorized to run adinfo as user qa1
dzdo.set.runas.explicit: true
...
[u1@rh6]$dzdo adinfo
Sorry, user u1 is not allowed to execute '/usr/bin/adinfo' as
root on rh6.
```

When the parameter value is `false`, if a user executes a command with `dzdo` and does not explicitly identify the user or group to run as (with the `-u` or `-g` option), `adcli` attempts to resolve the user. If the command defines a single



runas user, dzdo executes the specified command and sends a message to the log file; for example:

```
User u1 is authorized to run adinfo as user qa1
dzdo.set.runas.explicit: false
...
[u1@rh6]$dzdo adinfo
Local host name:      rh6
Joined to domain      acme.com
...
```

If the command defines multiple runas users, dzdo cannot resolve the user to run as and attempts to run the command as root. Since the user is not authorized to run the command as root, dzdo fails to execute the command and issues an error message; for example:

```
User u1 is authorized to run adinfo as users qa1 and adm
dzdo.set.runas.explicit: true
...
[u1@rh6]$dzdo adinfo
Sorry, user u1 is not allowed to execute '/usr/bin/adinfo' as
root on rh6.
```

In all cases, a user can execute a command successfully with dzdo by using the `-u` option to explicitly identify the runas user; for example:

```
[u1@rh6]$dzdo -u qa1 adinfo
Local host name:      rh6
Joined to domain      acme.com
...
```

You can also set this parameter using group policy.

## dzdo.timestampdir

This configuration parameter specifies the directory where dzdo stores the user's login timestamp files. The default is directory is `/var/run/dzdo`.

For example:

```
dzdo.timestampdir: /var/run/dzdo
```

This configuration parameter provides functionality equivalent to the `timestampdir` flag for configuring the sudoers file and sudo operation.

You can also set this parameter using group policy.



## dzdo.timestamp\_timeout

This configuration parameter specifies the maximum number of minutes allowed between operations before prompting the user to re-enter a password. The default parameter value is 5 minutes. You can set this parameter to zero (0) to always prompt for a password when users run privileged commands with dzdo. If set to a value less than 0 the user's timestamp never expires.

For example:

```
dzdo.timestamp_timeout: 5
```

This configuration parameter provides functionality equivalent to the `timestamp_timeout` flag for configuring the `sudoers` file and `sudo` operation.

You can also set this parameter using group policy.

## dzdo.tty\_tickets

This configuration parameter specifies whether dzdo should require authentication once per-tty rather than once per user. The parameter value can be `true` or `false`. The default parameter value is `false`.

For example:

```
dzdo.tty_tickets: false
```

This configuration parameter provides functionality equivalent to the `tty_tickets` flag for configuring the `sudoers` file and `sudo` operation.

You can also set this parameter using group policy.

## dzdo.use.realpath

This configuration parameter specifies whether dzdo uses command paths resolved by `realpath` when searching for commands. The default parameter value is `false`, meaning that `realpath` is not used.

When set to `true`, this parameter specifies that `realpath` is used to expand all symbolic links and resolve references to:



- `./`
- `../`
- extra / characters

You can also set this parameter using group policy.

## `dzdo.user.command.timeout`

When set to `true`, this parameter specifies a timeout on the DZDO command line with a `-T` option. If the timeout expires before the command has exited, the command is terminated.

The default setting is `false`.

## `dzdo.validator`

This configuration parameter specifies the full path to a script that is executed each time the `dzdo` command is run. The script is run synchronously under the user's Active Directory name.

The `dzdo` command always runs the `/usr/share/centrifydc/sbin/dzcheck` script before it executes the command specified. However, the distribution package does not include a `dzcheck` script.

You do not need to create a `dzcheck` script to use `dzdo`. You only need to create a script if you want to modify `dzdo` behavior—for example, to prompt the user to enter some information before executing the command. To incorporate your modification, you would write the script, name it `dzcheck` and put it in `/usr/share/centrifydc/sbin`.

Use the `dzdo.validator` command only if you need to specify a different path or file name. (If you name your script `dzcheck` and store it at the default location, you do not need to use `dzdo.validator`.) For example, if the script was named `myvalidator` and it was in the `/etc/centrifydc` directory, you would add the following command in `centrifydc.conf`:

```
dzdo.validator: /etc/centrifydc/myvalidator
```

The `dzdo` command sets three environment variables:



- `DZDO_USER`: the Active Directory name of the user invoking `dzdo`
- `DZDO_COMMAND`: the command
- `DZDO_RUNASUSER`: the user name that the command will be run as

The script should return one of the following values:

- 0 Success. `dzdo` will continue and run the command.
- non-zero Failure: `dzdo` will not run the command. In this event, `dzdo` does NOT show a message on the console. If you want to notify the user of the failure, include the message in the script.

When the logging level is set to `DEBUG`, the call to the script and the return value are logged in `var/log/centrifdc.log`. If `DEBUG` is off, the call to the script and return value are logged in `/var/log/messages`.

## `dzdo.validator.required`

This configuration parameter specifies whether `dzdo` is required to run the validator script. The default value is `false`.

**Note:** The `dzdo` command skips the validator script if the script is not available, is not owned by root, or is group/world writable. By default, `dzdo` continues to run the command even if the validator script is skipped. When this parameter is set to `true`, `dzdo` does not run the command if validator script is skipped.

## `dzsh.roleswitch.silent`

This configuration parameter specifies whether to display role information when changing from one role to another is a restricted shell.

By default, changing from one role to another displays a message indicating that you have changed your current role. For most commands that run in a restricted shell, displaying this message has no affect on the execution of the command.

There are cases, however, where a command—such as `sftp` or `git`—expects a specific type of response. Because the role change message is not the expected response, the message can cause the command to fail.



You can use this parameter to address those cases where the role change message would cause a command to fail.

Set this configuration parameter to `true` to prevent the role switch information from being displayed when running commands in a restricted shell. The default value is `false`.

For example:

```
dzsh.roleswitch.silent: true
```



# Customizing Auto Zone configuration parameters

This section describes the configuration parameters that affect the operation of a local host computer joined to Auto Zone. These parameters have no effect if the computer is not joined to Auto Zone. ↵

auto.schema.allow.groups .....	280
auto.schema.allow.users .....	282
auto.schema.apple_scheme .....	284
auto.schema.domain.prefix .....	284
auto.schema.groups .....	285
auto.schema.homedir .....	287
auto.schema.primary.gid .....	287
auto.schema.private.group .....	288
auto.schema.shell .....	289
auto.schema.use.adhomedir .....	289
auto.schema.remote.file.service .....	289
auto.schema.name.format .....	290
auto.schema.separator .....	290
auto.schema.search.return.max .....	291
auto.schema.name.lower .....	291
auto.schema.iterate.cache .....	291
auto.schema.override.uid .....	291



auto.schema.uid.conflict .....	292
auto.schema.homedir.illegal_chars .....	292
auto.schema.unix.name.disallow.chars .....	293
auto.schema.substitute.chars .....	293
auto.schema.max.unix.name.length .....	294

## auto.schema.allow.groups

This configuration parameter specifies a list of Active Directory groups that define which Active Directory users are valid users in the Auto Zone. Members of the specified groups are considered valid users in the Auto Zone.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

### Adding zone users based on group membership

By default, all Active Directory users are included in the Auto Zone. If you specify one or more groups using this parameter the only users who can log in using their Active Directory account are members of the specified groups, members of nested groups, users whose primary group is set to one of the groups specified, and all users specified in `auto.schema.allow.users`.

For example, to specify that only the members of the `sf-adms` and `sf-apps` groups should be allowed to log on to computers in Auto Zone, you would enter the following:

```
auto.schema.allow.groups: sf-adms sf-apps
```

The groups you specify for the `auto.schema.allow.groups` parameter must be security groups, but can be domain local, global, or universal groups. Distribution groups are not supported.

You can separate each group by a space or a comma and you can use double quotes or escape characters to include spaces or special characters in group names. For example: in group names. For example,

```
auto.schema.allow.groups: centrify_users, "Domain Admins",  
Domain\ Users
```



You should note that this parameter does not add the Active Directory groups you specify to Auto Zone. It does not assign the groups a numeric identifier (GID) or make the groups available to use as a primary group for any of the users added to Auto Zone. This parameter simply enables UNIX profiles for the users that are members of the specified groups. You can use the `auto.schema.groups` parameter to specify the Active Directory groups to include in the Auto Zone and assign it a GID. You can configure the primary group for users using the `auto.schema.primary.gid` parameter.

## Supported group name formats

You can specify groups by name or you can list the group names in a file using any of the following formats:

- SAM account name: `SAMAccountName@domain`
- User Principal Name: `name@domain`
- NTLM: `DOMAIN/SAMAccountName`
- Full DN: `CN=commonName, . . . ,DC=domain_component,DC=domain_component`
- Canonical Name: `domain/container/cn`

The `adcli` process writes any group name that is not recognized to the agent log file.

## Specifying the parameter value in a separate file

To specify a file that contains a list of Active Directory group names, you can set the parameter value using the `file:` keyword and a file location. For example:

```
auto.schema.allow.groups: file:/etc/centrifydc/auto_user_
groups.allow
```

In the `/etc/centrifydc/auto_user_groups.allow` file, you would type each group name on its own line using any of the supported name formats. For example:

```
centrify_users
"Domain Admins"
Domain Users
CN=group6,CN=Users,DC=domain,DC=com
```



## Limitations of this parameter

Auto Zone does not support one-way trusts. If there are any users in a specified group who belong to a domain that has a one-way trust relationship to the joined domain, they will not become valid users on the computer.

If you set this parameter, you should be aware of search limit defined for the `auto.schema.search.return.max` parameter. The setting for that parameter will limit the number of users returned in search results and stored in the cache. For example, if the `auto.schema.search.return.max` parameter is set to 100, and you use this parameter to specify an Active Directory group with 200 members, a query would only return results for the first 100 users. The remaining members of the group will still be allowed to log on to computers in the Auto Zone, but the results of queries might be misleading.

If desired, you can disable the `auto.schema.search.return.max` parameter by setting the parameter value to 0. Disabling the search limit ensures that all of the users in the specified Active Directory groups are listed as valid zone users when you run queries whether the number of users exceeds or falls short of the number specified for the `auto.schema.search.return.max` parameter. If you are not concerned about whether search results accurately reflect the users in the Active Directory groups you have defined for the `auto.schema.allow.groups` parameter, however, you don't need to modify the `auto.schema.search.return.max` parameter.

## `auto.schema.allow.users`

This configuration parameter specifies which Active Directory users to include in the Auto Zone.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

## Adding specific Active Directory users to Auto Zone

By default, all Active Directory users in a forest are included in the Auto Zone. If you specify one or more users using this parameter, however, only the specified users and members of the groups specified in the `auto.schema.allow.groups` parameter can log in using their Active Directory account.



For example, to specify that only the users `jane` and `sai.wu` should be allowed to log on to computers in `Auto Zone`:

```
auto.schema.allow.users: jane.doe sai.wu@ajax.org
```

You can separate each user name by a space or comma and use double quotes or escape characters to include spaces or special characters in group names. For example,

```
auto.schema.allow.groups: jane.doe, "Alex Adams", jae\ chin
```

## Supported user name formats

You can specify users by name or you can list the user names in a file in any of the following formats:

- SAM account name: `SAMAccountName@domain`
- User Principal Name: `name@domain`
- NTLM: `DOMAIN/SAMAccountName`
- Full DN: `CN=commonName, . . . ,DC=domain_component,DC=domain_component`
- Canonical Name: `domain/container/cn`

The `adcli` process writes any user name that is not recognized to the agent log file.

## Specifying the parameter value in a separate file

To specify a file that contains a list of Active Directory user names, you can set the parameter value using the `file:` keyword and a file location. For example:

```
auto.schema.allow.users: file:/etc/centrifydc/auto_user_
users.allow
```

In the `/etc/centrifydc/auto_user_users.allow` file, you would type each user name on its own line using any of the supported name formats. For example:

```
jane.doe
sai/wu@ajax.org
CN=Alex Adams,CN=Users,DC=ajax,DC=org
```



## auto.schema.apple\_scheme

This configuration parameter specifies that you want to use the Apple algorithm to automatically generate user and group identifiers. The Apple algorithm for generating identifiers is based on the `objectGuid` attribute for the user or group object. The Centrify mechanism for automatically generating UIDs and GIDs is based on the security identifier for the user or group objects. Both methods ensure a globally unique and consistent identifier for the user or group.

By default, this parameter value is set to `false`. If you want to use the Apple algorithm, set the parameter value to `true`. For example:

```
auto.schema.apple_scheme: true
```

If you set this parameter to use the Apple algorithm, you must use `adflush` to clear the cache, then restart the `adcli` process to update UIDs, GIDs, and user primary GIDs. Note that the user's primary group must be available in Auto Zone. If a user's primary group is not in the zone, the user will have an incomplete profile and unable to log on. If a user is provisioned with an incomplete profile, an error is recorded in the Window Event log.

After clearing the cache and restarting the agent, run the `fixhome.pl` script to correct conflicts between the new user UID and the home directory ownership.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

## auto.schema.domain.prefix

This configuration parameter specifies a unique prefix for a trusted domain. You must specify a whole number in the range of 0 to 511.

The Centrify algorithm for generating unique identifiers combines the prefix with the lower 22 bits of each user or group RID (relative identifier) to create unique UNIX user (UID) and group (GID) IDs for each user and group in the forest and in any two-way trusted forests.

Ordinarily, you do not need to set this parameter because the Centrify agent automatically generates the domain prefix from the user or group security identifier (SID). However, in a forest with a large number of domains, domain prefix conflicts are possible. When you join a computer to a domain, the



Centrify agent checks for conflicting domain prefixes. If any conflicts are found, the join fails with a warning message. You can then set a unique prefix for the conflicting domains.

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

To set this parameter, append the domain name and specify a prefix in the range 0 - 511. For example:

```
auto.schema.domain.prefix.acme.com: 3
auto.schema.domain.prefix.finance.com: 4
auto.schema.domain.prefix.corp.com: 5
```

The default behavior, if you do not set this parameter, is for the agent to automatically generate the domain prefix from the user or group security identifier (SID).

## auto.schema.groups

This configuration parameter specifies the Active Directory groups to include in the Auto Zone. When you specify one or more groups in this parameter, the groups specified are assigned a group ID on this computer.

The command syntax is:

```
auto.schema.groups: groupname [, groupname, groupname, ...]
```

By default all Active Directory groups are included.

**Note:** If an Active Directory user specified in `auto.schema.allow.users` is a member of a group and that group is NOT specified in `auto.schema.groups`, that group is ignored.

Any groups listed under `auto.schema.groups` can be domain local, global or universal security groups. Distribution groups are not supported.

You specify each group by name or you can list the groups in a file. The group name can be specified in any of the following formats:

- SAM account name: `SAMAccountName@domain`  
(specify the domain if the group is not in the current domain)
- User Principal Name: `name@domain`



- NTLM: DOMAIN/SAMAccountName

**Note:** Use the `adclient.ntlm.separators` parameter to specify different NTLM separators.

- Full DN: CN=commonName, . . . ,DC=domain\_component,DC=domain\_component
- Canonical Name: domain/container/cn

adclient writes any name that is not recognized to the agent log file.

You can also define the groups using group policy.

Examples:

```
auto.schema.groups: finance_users
auto.schema.groups: "Mktg Users"
auto.schema.groups: ops@domain.com
```

You can specify multiple groups in a single command. Separate each group by a comma and use escape characters to include, for example, spaces, backslashes, or a comma in the group specification. For example,

```
auto.schema.allow.groups: centrify_users, "Domain Admins",
Domain\ Users, \ group1, group2@domain.com, domain\\group3,
domain+group4, \ domain/group5,
CN=group6\,CN=Users\,DC=domain\,DC=com, \
domain/Users/group7
```

You can also use a file instead. The syntax is `file:/path`. For example,

```
auto.schema.allow.groups: file:/etc/centrifydc/auto_user_
groups.allow
```

In the file, enter each group line by line. However, you do not need the escape characters. For example, the following list enters the same groups as the previous example:

```
centrify_users
"Domain Admins"
Domain Users
group1
group2@domain.com
domain\group3
domain+group4
domain/group5
CN=group6,CN=Users,DC=domain,DC=com
domain/Users/group7
```

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.



## auto.schema.homedir

This configuration parameter specifies the home directory for logged in users. The default, if you do not specify this parameter, is:

- Mac OS X: `/Users/{user}`.
- Linux, HP-UX, AIX: `/home/{user}`
- Solaris: `/export/home/{user}`

The syntax `{user}` specifies the logon name of the user. For example, in the `centrifdc.conf` configuration file, if you add:

```
auto.schema.homedir:/Users/{user}
```

and `jsmith` logs on to a Mac OS X machine, the home directory is set to `/Users/jsmith`.

If the parameter `auto.schema.use.adhomedir` is `true`, the home directory is set to the value in Active Directory for the user, if one is defined. If `auto.schema.use.adhomedir` is `false` or if a home directory is not specified for the user in Active Directory, the home directory is set to the value defined for this parameter, `auto.schema.homedir`.

**Note:** The configuration parameter `auto.schema.homedir.illegal_chars` defines characters that are not allowed in home directory names. Any illegal characters in the specified name are removed from the home directory name on the computer.

You can also specify the home directory on all machines joined to Auto Zone by using group policy.

## auto.schema.primary.gid

This configuration parameter specifies the primary GID for Auto Zone users. The `auto.schema.private.group` parameter must be set to `false` to use this parameter.

**Note:** On Mac OS X, the default value of `auto.schema.private.group` is `false`. On Linux, HP-UX, Solaris, and AIX, the default value of `auto.schema.private.group` is `true`.



To specify the GID for an existing group, you must first find the GID for the group. To find the GID for a group, you can use the `adquery` command or `adedit`. For example, to find the GID for the group `Support`, open a terminal session and type:

```
>adquery group --gid Support
```

The command returns the GID for the Active Directory group `Support`:

```
1003
```

You can then set this parameter to the value returned. For example:

```
auto.schema.primary.gid: 1003
```

If you do not set this parameter, the value defaults to the following:

- On Mac OS X:  

```
auto.schema.primary.gid: 20
```
- On Linux, HP-UX, Solaris, and AIX:  

```
auto.schema.primary.gid: -1
```

If you are using the Apple algorithm to automatically generate user and group identifiers, including the group identifier for primary groups, set this parameter to `-1` to disable it. For example:

```
auto.schema.primary.gid: -1
```

In most cases, you set this configuration parameter using group policy. You can, however, set it manually in the configuration file if you are not using group policy or want to temporarily override group policy.

## auto.schema.private.group

This configuration parameter specifies whether to use dynamic private groups.

Specify `true` to create dynamic private groups. If you specify `true`, the primary GID is set to the user's UID and a group is automatically created with a single member.

Specify `false` to not create private groups.

On Mac OS X, the default value of this parameter is `false`. On Linux, HP-UX, Solaris, and AIX, the default value is `true`.

If you specify `false`, the primary GID is set to the value of `auto.schema.primary.gid`. On Mac OS X, the default value of



`auto.schema.primary.gid` is 20. On Linux, HP-UX, Solaris, and AIX, the default value of `auto.schema.primary.gid` is -1.

## auto.schema.shell

This configuration parameter specifies the default shell for the logged in user. The default value is:

- `/bin/bash` on Mac OS X and Linux systems
- `/bin/sh` on UNIX systems, including Solaris, HPUX, AIX.

You can also set the default shell on all machines joined to Auto Zone by using group policy.

## auto.schema.use.adhomedir

**Note:** This configuration parameter applies to Mac OS X computers only.

This configuration parameter specifies whether or not to use the Active Directory value for the home directory if one is defined. Set to `true` to use the Active Directory value (the default), or `false` to not use the Active Directory value. If you set the value to `false`, or if you set the value to `true` but a home directory is not specified in Active Directory, the value for `auto.schema.homedir` is used.

## auto.schema.remote.file.service

**Note:** This configuration parameter applies to Mac OS X computers only.

This configuration parameter specifies the type of remote file service to use for the network home directory. The options are: SMB (default) and AFP.

When you type a path for the network home directory in Active Directory, it requires a specific format: `/server/share/path`, but on Mac OS X, the format for mounting a network directory requires the remote file service type: `/type/server/share/path`. By identifying the remote file-service type, you can



type the network path in the format required by Active Directory, and the agent converts the path into the format required by Mac OS X.

For example:

```
auto.schema.remote.file.service:SMB
```

You can also specify the remote file service type on machines joined to Auto Zone by using group policy.

## auto.schema.name.format

This configuration parameter specifies how the Active Directory username is transformed into a UNIX name (short name in Mac OS X). The options are

- SAM (default)  
An example SAM name is joe
- SAM@domainName  
An example SAM@domainName is joe@acme.com
- NTLM  
An example NTLM name is acme.com-joe

## auto.schema.separator

**Note:** This configuration parameter has been deprecated in favor of `adclient.ntlm.separators`, which applies whenever NTLM format is used. The `auto.schema.separator` parameter only applies when the computer is connected to Auto Zone.

This configuration parameter specifies the separator to be used between the domain name and the user name if NTLM format is used. The default separator is a plus (+). For example:

```
auto.schema.separator:+
```

which results in a name such as:

```
acme.com+jcool
```



## auto.schema.search.return.max

This configuration parameter specifies the number of users that will be returned for searches by utilities such as `dsc1` and the Workgroup Manager application. Because Auto Zone enables access to all users in a domain, a search could potentially return tens of thousands of users. This parameter causes the search to truncate after the specified number of users.

The default is 1000 entries.

## auto.schema.name.lower

This configuration parameter converts all usernames and home directory names to lower case in Active Directory.

Set to `true` to convert usernames and home directory names to lowercase.

Set to `false` to leave usernames and home directories in their original case, upper, lower, or mixed.

The default for a new installation is `true`.

## auto.schema.iterate.cache

This configuration parameter specifies that user and group iteration take place only over cached users and groups.

Set the value for `auto.schema.iterate.cache` to `true` to restrict iteration to cached users and groups.

Set the value for `auto.schema.iterate.cache` to `false` to iterate over all users and groups. The default value is `false`.

## auto.schema.override.uid

This configuration parameter specifies which Active Directory user attribute to use to generate the UID. There are three Active Directory user attribute options:



- `employeeID`
- `employeeNumber`
- `uidNumber`

If you enter a unsupported attribute the setting is ignored.

Only numeric values are supported. If the user attribute contains a non-numeric value, that user is ignored.

Examples:

```
auto.schema.override.uid: employeeID
auto.schema.override.uid: uidNumber
```

## auto.schema.uid.conflict

This configuration parameter specifies what is done if adclient discovers that an Active Directory user already exists with the same UID. There are two options:

- `allow`: Allow the duplicate UID; an information message is logged.
- `disallow`: If a duplicate UID already exists, the second user with the same UID is ignored; a warning message is logged.

Examples:

```
auto.schema.uid.conflict: allow
auto.schema.uid.conflict: disallow
```

## auto.schema.homedir.illegal\_chars

This configuration parameter specifies the characters in a home directory name that are not allowed in UNIX, Linux or Mac OS X home directory names. Each character in a home directory name that matches one of the specified characters is simply removed from the name; for example:

```
/home/user$34      /* illegal $ character
/home/user34       /* illegal character removed
```

The default setting in `centrifydc.conf` for UNIX (HP-UX, Solaris, AIX) and Linux systems is the following:

```
auto.schema.homedir.illegal_chars: \t\n /\$><?*%|\\"'` []
```



The default setting in `centrifdc.conf` for Mac OS X systems is the following (space is omitted):

```
auto.schema.homedir.illegal_chars: \t\n/\\" data-bbox="113 134 691 152"/>
```

Run the `adflush` command after you change the value to flush the cache.

## auto.schema.unix.name.disallow.chars

This configuration parameter specifies the characters in an Active Directory user or group name that are not allowed in UNIX, Linux or Mac OS X names. Each character in the name that matches the characters specified is replaced in the corresponding UNIX name by the character specified in `auto.schema.substitute.chars`.

**Note:** Be sure to specify the replacement character in `auto.schema.substitute.chars`. Otherwise, the offending character is simply removed from the name, and you run the risk of duplicate UNIX names.

The default setting in `centrifdc.conf` for UNIX (HP-UX, Solaris, AIX) and Linux systems is the following:

```
auto.schema.unix.name.disallow.chars: \t\n /\" data-bbox="113 520 764 551"/>
```

The default setting in `centrifdc.conf` for Mac OS X systems is the following (space is omitted):

```
auto.schema.unix.name.disallow.chars: \t\n /\" data-bbox="113 611 859 644"/>
```

Run the `adflush` command after you change the value to flush the cache.

## auto.schema.substitute.chars

This configuration parameter specifies the character that replaces any characters specified in `auto.schema.unix.name.disallow.chars` encountered in an Active Directory user or group name in the corresponding UNIX name.

The default setting in `centrifdc.conf` is the following:

```
auto.schema.substitute.chars: _ (underbar)
```

Run the `adflush` command after you change the value to flush the cache.



## auto.schema.max.unix.name.length

This configuration parameter specifies the maximum length for a generated UNIX user or group name. The UNIX names are generated from the Active Directory user and group names.

The default setting in `centrifydc.conf` is the following:

```
auto.schema.max.unix.name.length: 33
```

Run the `adflush` command after you change the value to flush the cache.



# Customizing auditing configuration parameters

This section describes the configuration parameters that you can set on audited computers that have a Centrify UNIX agent installed. These parameters affect the operation of the auditing service (dad) and the UNIX shell wrapper (cdash).

The parameters are defined in a text file named `centrifyda.conf` in `/etc/centrifyda` on each audited computer.

**Note:** For information about specifying an audit trail target in the `centrifydc.conf` file, see [audittrail.targets](#).

<code>agent.max.missed.update.tolerance</code> .....	297
<code>agent.send.hostname</code> .....	298
<code>agent.video.capture</code> .....	298
<code>autofix.nss.conf</code> .....	299
<code>cache.enable</code> .....	299
<code>cache.max.size</code> .....	300
<code>cache.time.to.live</code> .....	300
<code>dad.client.idle.timeout</code> .....	300
<code>dad.collector.connect.timeout</code> .....	300
<code>dad.dumpcore</code> .....	301
<code>dad.gssapi.seal</code> .....	301
<code>dad.gssapi.sign</code> .....	301
<code>dad.process.fdlimit</code> .....	301



dad.resource.cpulimit .....	302
dad.resource.cpulimit.tolerance .....	302
dad.resource.fdlimit .....	303
dad.resource.memlimit .....	303
dad.resource.restart .....	304
dad.resource.timer .....	305
dad.timer.diskspace .....	305
dad.timer.monitor.nss.conf .....	305
dash.allinvoked .....	306
dash.auditstdin .....	306
dash.auditstdin.except .....	306
dash.cmd.audit.blacklist .....	307
dash.cmd.audit.show.actual.user .....	307
dash.cont.without.dad .....	307
dash.force.audit .....	307
dash.loginrecord .....	308
dash.obfuscate.pattern .....	308
dash.obfuscate.regex .....	310
dash.parent.skiplist .....	310
dash.reconnect.dad.retry.count .....	311
dash.reconnect.dad.wait.time .....	311
dash.shell.env.var.set .....	311
dash.ssh.command.skiplist .....	311
dash.user.alwaysallowed.list .....	312
dash.user.skiplist .....	312
event.execution.monitor .....	312
event.execution.monitor.user.skiplist .....	313



event.file.monitor .....	313
event.file.monitor.process.skiplist .....	314
event.file.monitor.user.skiplist .....	314
event.monitor.commands .....	315
event.monitor.commands.user.skiplist .....	315
lang_setting .....	315
lrpc2.message.signing .....	316
lrpc2.timeout .....	316
lrpc2.rebind.timeout .....	316
nss.alt.zone.auditlevel .....	316
nss.nologin.shell .....	317
nss.user.conflict.auditlevel .....	318
nss.user.override.auditlevel .....	318
nss.user.override.userlist .....	318
preferred.audit.store .....	320
spool.diskspace.logstate.reset.threshold .....	320
spool.diskspace.min .....	320
spool.diskspace.softlimit .....	321
spool.maxdbsize .....	322
uid.ignore .....	322
user.ignore .....	323
user.ignore.audit.level .....	323

## agent.max.missed.update.tolerance

This configuration parameter specifies the number of unsuccessful attempts the Centrifuy auditing agent makes to join a collector before displaying a



notification that the agent is not joined to a collector. Each attempt is made after an interval of 5 minutes.

For example, if you want the agent to warn you that it is not connected to a collector after 3 attempts, you would enter the following:

```
agent.max.missed.update.tolerance: 3
```

The default value for this parameter is 4.

You can use this parameter with the `dad.collector.connect.timeout` parameter which specifies the amount of time, in seconds, the agent waits during each connection attempt before it determines that it cannot connect to a collector.

## agent.send.hostname

This configuration parameter enables audited sessions to display the host name specified by the agent on audited computers instead of the host name resolved by the collector through DNS. This configuration parameter is useful in configurations where the DNS servers used by the collectors cannot reliably resolve host names from IP addresses. The most common scenarios that might require you to use this configuration parameter are when the agents are in a virtual environment using network address translation (NAT) or in a perimeter network outside of a firewall.

You can set this parameter to `true` if you want the agent to determine the host name used. If you set this configuration parameter to `false`, the collector determines the agent's host name based on its IP address.

For example:

```
agent.send.hostname: true
```

## agent.video.capture

This configuration parameter enables or disables saving the video capture for a specific agent, which overrides the video capture settings configured for the entire DirectAudit installation. If video capture is disabled, the collector does not display the video output and does not save it to the database.

The default value uses the settings that you have configured for the entire installation.



To save the captured video output to the database, you set this parameter to `enabled`. To not save the captured video output, set this parameter to `disabled`.

For example:

```
agent.video.capture: enabled
```

## autofix.nss.conf

This configuration parameter enables `dad` to fix `/etc/nsswitch.conf` automatically if anything goes wrong. If set to `true`, the `dad` process configures the `/etc/nsswitch.conf` file automatically. If set to `false`, the `/etc/nsswitch.conf` file is left unmodified.

The default value is `true`.

For example, to disable changes to the `/etc/nsswitch.conf` file:

```
autofix.nss.conf: false
```

## cache.enable

This configuration parameter controls whether the `dad` process caches name service query results about users and groups. If set to `true`, the `dad` process stores query results, for example, from user lookup requests, in memory for better performance. If set to `false`, query results are not saved and must be retrieved whenever they are needed.

If set to `true`, you can use the `cache.max.size` and `cache.time.to.live` parameters to control the number and duration of entries in the cache. You can also use the `daflush` command to clear the cache manually when you want to ensure you get updated information. For example, if you remove the UNIX Login role for an Active Directory user, some information for that user might remain in the cache and be returned when you run a command such as `getent passwd`. You can run `daflush` to ensure the user is removed completely from the local computer cache, including the auditing name service cache.

The default value for this parameter is `true`.

For example, to disable the name service cache on an audited computer:

```
cache.enable: true
```



## cache.max.size

This configuration parameter controls the maximum number of entries that can be stored in the dad name service cache. query results about users and groups. This parameter is only applicable if the `cache.enable` parameter is set to true. The dad process stores query results up to the value set for this parameter in memory for better performance.

The default value for this parameter is 80,000 entries.

For example, to increase the maximum number of name service results that can be stored on an audited computer:

```
cache.max.size: 85000
```

## cache.time.to.live

This configuration parameter controls the length of time entries should remain valid in the name service cache. You can specify the maximum number of seconds cached query result should be available in the cache. This parameter is only applicable if the `cache.enable` parameter is set to true.

The default value for this parameter is 10 minutes (600 seconds).

For example, to increase the number of seconds query results are available in the cache on an audited computer:

```
cache.time.to.live: 900
```

## dad.client.idle.timeout

This configuration parameter determines the time interval within which dad checks for disconnected dash sessions.

The default value for this parameter is 30 minutes (1800 seconds).

## dad.collector.connect.timeout

This configuration parameter specifies the amount of time, in seconds, the agent waits during each connection attempt before it determines that it cannot connect to a collector.



The default value for this parameter is 60 seconds.

You can use this parameter with the `agent.max.missed.update.tolerance` parameter which allows you to specify the number of unsuccessful attempts that the agent can make to connect to a collector before notifying the user that it is not connected to a collector.

## `dad.dumpcore`

This configuration parameter enables dad to do a core dump if an audited computer crashes. This parameter overrides the default `u1imit` setting. If set to `true`, the agent will generate a core dump if the computer crashes. If set to `false`, no core dump is generated.

The default value is **false**.

## `dad.gssapi.seal`

This configuration parameter specifies whether the auditing service seals network communications with the collector using a secure GSSAPI connection. If set to **true**, the network connection is sealed and cannot be read. If set to **false**, the connection is not sealed and is human-readable.

The default value is **true**.

## `dad.gssapi.sign`

This configuration parameter specifies whether the auditing service signs network communications with the collector over a secure GSSAPI connection. If set to **true**, the network connection is signed. If set to **false**, the connection is not signed.

The default value is **true**.

## `dad.process.fdlimit`

This configuration parameter specifies the number of file descriptors that can be used for audited sessions.



For some UNIX platforms, such as Solaris, the default number of available file descriptors for each process is insufficient of auditing sessions, because the Centrify agent requires two descriptors per session.

Use this parameter to increase the number of file descriptors available. For example:

```
dad.process.fdlimit: 2048
```

This configuration parameter can also be set using Group Policy.

## dad.resource.cpulimit

This configuration parameter specifies the maximum percentage of CPU usage that dad can use before dad is restarted, or before the event is logged in `/var/log/centrifydc.log`. Whether dad is restarted when the threshold is exceeded is controlled by [dad.resource.restart](#).

The default value of this parameter is 50, meaning that dad is restarted or the event is logged when dad CPU usage exceeds 50%. For example:

```
dad.resource.cpulimit: 50
```

The dad resource monitor automatically checks the usage of various dad resources during runtime. For each resource that is monitored, you can configure the threshold value that triggers a dad restart or a log entry.

When dad is restarted, the client is purged, and counters for resources such as CPU usage, file descriptors, and memory are reset. See [dad.resource.restart](#) for more information about the advantages of setting a threshold that is lower than the default system value.

## dad.resource.cpulimit.tolerance

This configuration parameter specifies the number of times that CPU usage can exceed the threshold set in [dad.resource.cpulimit](#) before dad is restarted, or before the event is logged in `/var/log/centrifydc.log`. Whether dad is restarted when the value of this parameter is exceeded is controlled by [dad.resource.restart](#).

The default value of this parameter is 5, meaning that the CPU usage set in [dad.resource.cpulimit](#) can be exceeded four times before dad is restarted on the fifth instance or the event is logged. For example:



```
dad.resource.cputolerance: 5
```

The dad resource monitor automatically checks the usage of various dad resources during runtime. For each resource that is monitored, you can configure the threshold value that triggers a dad restart or a log entry.

When dad is restarted, the client is purged, and counters for resources such as CPU usage, file descriptors, and memory are reset. See [dad.resource.restart](#) for more information about the advantages of setting a threshold that is lower than the default system value.

## dad.resource.fdlimit

This configuration parameter specifies the maximum file descriptors that dad can use before dad is restarted, or before the event is logged in `/var/log/centrifdc.log`. Whether dad is restarted when the threshold is exceeded is controlled by [dad.resource.restart](#).

The default value of this parameter is 500, meaning that dad is restarted or the event is logged when dad file descriptor usage exceeds 500. For example:

```
dad.resource.fdlimit: 500
```

The dad resource monitor automatically checks the usage of various dad resources during runtime. For each resource that is monitored, you can configure the threshold value that triggers a dad restart or a log entry.

When dad is restarted, the client is purged, and counters for resources such as CPU usage, file descriptors, and memory are reset. See [dad.resource.restart](#) for more information about the advantages of setting a threshold that is lower than the default system value.

## dad.resource.memlimit

This configuration parameter specifies the maximum memory usage (in bytes) that dad can use before dad is restarted, or before the event is logged in `/var/log/centrifdc.log`. Whether dad is restarted when the threshold is exceeded is controlled by [dad.resource.restart](#).

The default value of this parameter is 104857600 (100 MB), meaning that dad is restarted or the event is logged when dad memory usage exceeds that value. For example:

```
dad.resource.memlimit: 104857600
```



The dad resource monitor automatically checks the usage of various dad resources during runtime. For each resource that is monitored, you can configure the threshold value that triggers a dad restart or a log entry.

When dad is restarted, the client is purged, and counters for resources such as CPU usage, file descriptors, and memory are reset. See `dad.resource.restart` for more information about the advantages of setting a threshold that is lower than the default system value.

## dad.resource.restart

This configuration parameter specifies whether dad restarts or just logs the event when a resource threshold is exceeded. Events that are logged are recorded in `/var/log/centrifdc.log`.

The default value of this parameter is `false`, meaning that when a resource threshold is exceeded, the event is logged, but dad is not restarted. For example:

```
dad.resource.restart: false
```

The dad resource monitor automatically checks the usage of various dad resources during runtime. For each resource that is monitored, you can configure the threshold value that triggers a dad restart or a log entry.

You can set resource thresholds in these parameters:

- `dad.resource.cpulimit`
- `dad.resource.cpulimit.tolerance`
- `dad.resource.fdlimit`
- `dad.resource.memlimit`
- `dad.resource.timer`

When dad is restarted, the client is purged, and counters for resources such as CPU usage, file descriptors, and memory are reset.

Setting a low threshold for restarting dad and purging the client can avoid problems with resources being consumed prematurely. For example, when `cdash` calls another `cdash` recursively, dad receives a large number of client requests. Solaris has only 256 file descriptors for `ulimit` by default. Unless you configure a threshold lower than 256, `cdash` does not stop recursive calling



until 257 calls, and all of dad's file descriptors could be consumed by that one operation.

## `dad.resource.timer`

This configuration parameter specifies how often (in seconds) the dad resource monitor checks dad resource usage.

The default value of this parameter is 600 (10 minutes), meaning that the dad monitor checks dad resource usage every 10 minutes. For example:

```
dad.resource.timer: 600
```

The dad resource monitor automatically checks the usage of various dad resources during runtime. For each resource that is monitored, you can configure the threshold value that triggers a dad restart or a log entry.

When dad is restarted, the client is purged, and counters for resources such as CPU usage, file descriptors, and memory are reset. See `dad.resource.restart` for more information about the advantages of setting a threshold that is lower than the default system value.

## `dad.timer.diskspace`

This configuration parameter specifies the number of seconds between checks of disk space when the disk space reserved for offline storage is less than the value specified in the `spool.diskspace.min` parameter. At each check, a warning message is written to the log file.

The default value is 360 seconds.

## `dad.timer.monitor.nss.conf`

This configuration parameter controls how frequently the dad process checks the `/etc/nsswitch.conf` file for changes. Set this parameter to the number of seconds between checks.

The default value is 60 seconds.



## dash.allinvoked

This configuration parameter was previously required to support auditing of shells invoked in scripts and command-level auditing. The parameter is no longer required and can be removed if you upgrade the agent to the latest version and enable command-level auditing through NSS. If you do not update the agent, you can use this parameter to specify whether to audit all shell invocations. If set to **true**, all login and non-login shells are audited. If set to **false**, only login shells and login sub-shells are audited. If set to `false`, invoked shells are not audited.

The default value is **false**.

## dash.auditstdin

This configuration parameter specifies whether the agent captures standard input (`stdin`). If set to `true`, the auditing service records all session input and output, including `stdin` data. If set to `false`, the auditing service records all session activity to standard output, but does not capture `stdin` data.

The default value is `true`.

## dash.auditstdin.except

This configuration parameter specifies strings that `cdash` should ignore when capturing `stdin` data. For security, typed passwords are always ignored by default. Use regular expressions and do not include quotes. Leading and trailing spaces are ignored, spaces in the middle are not affected. For example:

```
dash.auditstdin.except: (prompt1|prompt2)
```

will match strings like these:

```
This is prompt1:  
Prompt2 asks for password:
```

The default value is empty to only ignore the passwords that users enter. For more information about specifying exceptions, see the comments in the `centrifyda.conf` file.



## dash.cmd.audit.blacklist

This configuration parameter specifies whether certain privileged command patterns are skipped while auditing is enabled.

To add a command pattern to skip, list the regular expression you wish to skip to this parameter. When enabled, an audit trail is still sent by the agent, but the specified command and arguments will not be captured. For example, the following list will skip auditing of the “date” command:

```
dash.cmd.audit.blacklist: date
```

By default, no command patterns are skipped while auditing.

## dash.cmd.audit.show.actual.user

This configuration parameter specifies whether command-based auditing records will display the actual user account used to run a privileged command that requires auditing, as well as the run-as account.

By default, the value of this parameter is set to `false`, and only the run-as account used to execute privileged commands is shown in auditing records. To enable this parameter, set the value to `true`. For example:

```
dash.cmd.audit.show.actual.user: true
```

## dash.cont.without.dad

This configuration parameter specifies whether `cdash` prompts the user to restart auditing when it determines that `dad` is not running. If set to `true`, `cdash` does not prompt the user to restart auditing and continues without the `dad` process. If set to `false`, `cdash` prompts the user to restart auditing to continue.

The default value is `false`.

## dash.force.audit

This configuration parameter specifies one or more session binary files to audit. This parameter was previously required to support command-level auditing on



managed computers. The parameter is no longer required and can be removed if you upgrade the agent to the latest version.

Instead of setting this parameter, you must run the following command to enable auditing for specific command-line programs:

```
dacontrol --enable --command cmd_path
```

If you do not update the agent, you can use this parameter to specify commands to be audited by appending `.audit` to the file name. For example, to audit secure shell (ssh) sessions:

```
dash.force.audit: /usr/share/centrifydc/bin/ssh.audit
```

However, you still must run the `dacontrol` command to enable auditing for specific commands.

You can separate entries by typing a space or a comma. You can escape spaces or commas in file names using the backslash character (`\`). This parameter is not included in the configuration file by default.

## dash.loginrecord

This configuration parameter specifies whether the auditing service should add `utmp` entries for the `cdash` pseudo terminals (pty). The setting of this parameter affects the results of `whoami` and `who` commands.

If set to **true**, the auditing service adds `utmp` entries for `cdash` pseudo terminals processes. With this setting, `whoami` in an audited shell works as expected, but `who` commands list logged-in users twice.

**If set to false**, the auditing service does not create additional `utmp` entries. With this setting, the `whoami` command in an audited shell cannot determine complete user information. Workaround: on some operating systems: `who --lookup` works, but the `who` command lists users only once.

The default value is `false`.

## dash.obfuscate.pattern

This configuration parameter enables you to hide sensitive information in the standard output (stdout) in audit results by using patterns to define the hidden information.



Beginning with release 2015.1, each pattern that you create must be embedded inside double quote characters (“”). For example:

“nnnn-nnnn-nnnn-nnnn”

**Note:** If there is a delay in the display of standard output information that you have designated to be hidden by defining a pattern, the agent may not recognize the pattern, and the information may be shown. To avoid delays, your obfuscation string should not exceed the size of the Cdash standard output buffer, which is 4KB.

**Note:** In releases earlier than 2015.1, patterns could be embedded inside double quote characters (“”) or slash characters (/). If slash characters were used, they are converted automatically to double quote characters when you upgrade from 2015 to 2015.1.

Each single character in a pattern corresponds to one character in actual session data.

Supported characters in a pattern are as follows:

a	Any lower case letter.
A	Any upper case letter.
d	Any character.
D	Any letter.
n	Any decimal digit character.
Symbols, such as the following:	
s	~ ` ! @ # \$ % ^ & * ( - _ = +
	[ { ] }   \ : ; ' < , > . ? /
-	Separator for exact matching in session data.
_	Separator for exact matching in session data.
(	Separator for exact matching in session data.
)	Separator for exact matching in session data.
,	Separator for exact matching in session data.
.	Separator for exact matching in session data.

If you define more than one pattern, separate the patterns with spaces. For example:

“nnnn-nnnn” “A-nnnn”

By default, this parameter does not contain any patterns.



## dash.obfuscate.regex

This configuration parameter enables you to hide sensitive information in the standard output (stdout) in audit results by using regular expressions to define the hidden information patterns.

Beginning with release 2015.1, each regular expression that you create must be embedded inside double quote characters (“”). For example:

```
“[A-Z][0-9]{6}\\([0-9A-Z]\\)”
```

If you define more than one regular expression, separate the regular expressions with spaces. For example:

```
“[0-9]-[0-9]” “[a-z]-[0-9]”
```

**Note:** If there is a delay in the display of standard output information that you have designated to be hidden by defining a pattern, the agent may not recognize the pattern, and the information may be shown. To avoid delays, your obfuscation string should not exceed the size of the Cdash standard output buffer, which is 4KB.

**Note:** In releases earlier than 2015.1, patterns could be embedded inside double quote characters (“”) or slash characters (/). If slash characters were used, they are converted automatically to double quote characters when you upgrade from 2015 to 2015.1.

By default, this parameter does not contain any regular expressions.

## dash.parent.skiplist

This configuration parameter lists the names of parent processes that should not be audited. If the name of a process’s parent is in this list, cdash will drop out without auditing.

You can add parent processes to the list or remove the default parent processes if you do not want to skip auditing for these processes. List entries must be separated by spaces.

For example, to skip auditing for the `sapstartsrv`, `gdm-binary`, `gdm-session-wor`, `kdm`, and `sdt_shell` parent processes:



```
dash.parent.skiplist: sapstartsrv gdm-binary gdm-session-wor kdm  
sdt_shell
```

You can also set this parameter using group policy.

## dash.reconnect.dad.retry.count

This configuration parameter specifies how many times cdash attempts to connect to dad after dad has started.

The default value is 3 retry attempts.

## dash.reconnect.dad.wait.time

This configuration parameter specifies the number of seconds to wait after restarting dad before attempting to reconnect to dad.

The default value is 1 second.

## dash.shell.env.var.set

This configuration parameter specifies whether cdash should set the SHELL environment variable to the user's actual shell or the audit shell.

If set to `true`, the SHELL environment variable is set to the user's actual shell. If set to `false`, the SHELL environment variable is set to the audited shell.

The default is `true`.

## dash.ssh.command.skiplist

This configuration parameter specifies the commands that can be executed using a secure shell (ssh) connection without being audited. You can use this parameter to prevent the auditing service from capturing unwanted session information. For example, by setting this parameter, you can avoid recording all of the binary data sent to and from the server when you execute file transfer commands such as `rsync`, `sftp`, or `scp` through a secure shell connection. By default, the parameter is configured to skip auditing for the `rsync`, `sftp` and `scp` commands, which are the most commonly used file transfer programs.



You can add programs to the list or remove the default programs if you don't want to skip auditing for these sessions. If you remove file transfer programs from the list, however, long data streams might cause problems when transferred to collector service.

For example, to skip auditing for `ftp`, `rsync`, `sftp`, `scp`, and `wget` commands:

```
dash.ssh.command.skiplist: ftp rsync sftp scp wget
```

## dash.user.alwaysallowed.list

This configuration parameter lists the names of UNIX users who are allowed to use a session even if the computer cannot be audited due to environment setup issues.

By default, `root` is the only user allowed to use an unaudited session.

To use this parameter, specify a space-separated list of UNIX user names.

You can also set this parameter using group policy.

## dash.user.skiplist

This configuration parameter lists the names of UNIX users and Active Directory users with a UNIX login who should not be audited. You can separate user names by typing a space or a comma. For example:

```
dash.user.skiplist: MaeJones kelly,dmorris,BookerJames
```

The default value is empty.

When you list a user in the `dash.user.skiplist`, this overrides the user's audit level. For example, if the user is set as "Audit Required" and also in `dash.user.skiplist`, the user might log in successfully but without being audited.

## event.execution.monitor

Use the `event.execution.monitor` parameter to monitor all programs that users run in an audited session.

To use this parameter, you must have enabled the agent to perform advanced monitoring with the command `dacontrol -m`.



The default value for the `event.execution.monitor` parameter is `false`.

In the `audit.log` file, you can find these events by looking for the `cda_sys_execve` messages. In the `cdc.log` file you can find them by looking for the `Emit COMMAND_HISTORY`.

## event.execution.monitor.user.skiplist

Use the `event.execution.monitor.user.skiplist` parameter to specify a list of users to exclude from advanced monitoring for program execution. For these users, the auditing service does not record any programs that they run, even when the parameter `event.execution.monitor` is set to `true`.

For users specified in this list, the auditing service checks for the user account that the program is run by, also known as the “run as” user account.

To use this parameter, you must have enabled the agent to perform advanced monitoring with the command `dacontrol -m`.

## event.file.monitor

Use the `event.file.monitor` parameter to enable advanced monitoring for configuration files. To use this parameter, you must have enabled the agent to perform advanced monitoring with the command `dacontrol -m`.

If advanced monitoring is enabled for files, the auditing service monitors any activity in the following folders:

- `/etc/`
- `/var/centrify/`
- `/var/centrifydc/`
- `/var/centrifyda/`

The default value for the `event.file.monitor` parameter is `true`.

In the `audit.log` file, you can find these events by looking for the `cda_file_monitor_write` messages. In the `cdc.log` file you can find them by looking for the `Emit AUDIT_TRAIL`.



## event.file.monitor.process.skiplist

Use the `event.file.monitor.process.skiplist` parameter to specify which processes to not monitor that are in an area that you've already configured to provide detailed file monitoring. To use this parameter, you must have enabled the agent to perform advanced monitoring with the command `dacontrol -m`.

When either `adclient` or `dad` processes or one of their sub-processes access or alter one of the files specified in the `event.file.monitor` list, this activity is automatically excluded from advanced monitoring.

The default value for this parameter is `daspool`.

## event.file.monitor.user.skiplist

Use the `event.file.monitor.user.skiplist` parameter to specify a list of users to exclude from advanced monitoring for files. For these users, the auditing service does not record any write access to directories specified in [event.file.monitor](#).

For users specified in this list, the auditing service checks this list against the original login user.

For example:

The `event.file.monitor.user.skiplist` parameter does not include the user `dwirth`. `Dwirth` uses the following command:

```
dzdo cp /tmp/badfile /etc/badfile
```

This activity generates the following audit event:

```
user dwirth run as root opened the file /etc/badfile using the /bin/cp command.
```

To use this parameter, you must have enabled the agent to perform advanced monitoring with the command `dacontrol -m`.

The default value for this parameter is `root`.



## event.monitor.commands

Use the `event.monitor.commands` parameter to specify a list of commands to monitor. Be sure to list each command with the full path. The auditing service generates an audit trail event when a user runs any of these monitored commands, and ignores any commands listed in the `event.monitor.commands.user.skiplist`.

To use this parameter, you must have enabled the agent to perform advanced monitoring with the command `dacontrol -m`. Otherwise, you will not get any report or audit trail event results.

In the `audit.log` file, you can find these events by looking for the `cda_cmd_exec` messages. In the `cdc.log` file you can find them by looking for the `Emit AUDIT_TRAIL` and `Emit COMMAND_HISTORY` messages.

## event.monitor.commands.user.skiplist

Use the `event.monitor.commands.user.skiplist` parameter to specify any users for whom you do not want to monitor the commands that they run. Any user that you specify in this parameter will not generate an audit trail event when they run any command, even if the command is listed in `event.monitor.commands`.

## lang\_setting

This configuration parameter specifies the code page that is used by DirectAudit for character encoding.

You can set this parameter to one of the following valid values:

- `utf8`
- `iso8859-1`

The default value is `utf8`.

You can also set this configuration parameter using group policy.



## lrpc2.message.signing

Use the `lrpc2.message.signing` to manage message signing behavior.

You can set this parameter to one of the following values:

- `Disabled` (default) to not ever sign LRPC2 messages.
- `Allowed` does not require LRPC2 message signing but may be chosen if required.
- `Required` must sign LRPC2 messages.

## lrpc2.timeout

This configuration parameter specifies the number of seconds `cdash` and `dainfo` waits while trying to contact the `dad` service before timing out.

The default value is 30 seconds.

You can also set this configuration parameter using group policy.

## lrpc2.rebind.timeout

This configuration parameter specifies the number of seconds that `dareload` (-b) waits while trying to connect to the `dad` service before timing out.

The default value is 300 seconds.

You can also set this configuration parameter using **for rebinding** group policy.

## nss.alt.zone.auditlevel

This configuration parameter enables you to specify a default audit level for all users who do not have an audit level explicitly defined using the `nss.user.override.userlist` parameter. Note that this parameter is only applicable in classic zones. You should not set this parameter if you are using hierarchical zones.

You can set this parameter to one of the following valid values:



- `use_sysrights`
- `audit_if_possible`
- `no_audit`
- `audit_required`

The default value is `use_sysrights`. This setting determines the audit level by communicating with the `adclient` process.

The effective audit level for a user is determined in the following order:

1. If the user is included in the list specified by the `nss.user.override.userlist` parameter and the audit level is specified for the user, the specified audit level is used.
2. If the user is included in the list specified by the `nss.user.override.userlist` parameter and the audit level is not specified for the user, the value specified by the `nss.user.override.auditlevel` parameter is used.
3. If the user is not included in the list specified by the `nss.user.override.userlist` parameter, the audit level specified for the `nss.alt.zone.auditlevel` parameter is used.

## `nss.nologin.shell`

This configuration parameter enables you to specify one or more non-login shells for audited users. In most cases, when audited users log on, they are placed in a wrapper shell so that their activity can be captured and sent to a collector. To use a real shell instead of the wrapper shell, you can specify shells to be non-login shells for audited users to access. After you set this parameter, you should restart the auditing service (`dad`).

For example, to define the shells `/sbin/shell_test1` and `/bin/shell_test2` as a non-login shells, you would type:

```
nss.nologin.shell: /sbin/shell_test1 /bin/shell_test2
```

If this parameter is not configured, the default no-login shells `/sbin/nologin` and `/bin/false` are used.



## nss.user.conflict.auditlevel

This configuration parameter enables you to override a user's audit level when the user is listed in the `user.ignore` list and has the `use_sysrights` audit level. Valid parameter values are:

`audit_if_possible`  
`no_audit`  
`audit_required`

By default, this parameter is set to `audit_if_possible`.

## nss.user.override.auditlevel

This configuration parameter specifies the default audit level for any users specified for the `nss.user.override.userlist` without an audit level defined. This parameter replaces the deprecated `user.ignore.audit.level` parameter.

You can set this parameter to one of the following valid values:

- `use_sysrights`
- `audit_if_possible`
- `no_audit`
- `audit_required`

The default value is `use_sysrights`. If there are classic zone users not included in `nss.user.override.userlist` parameter, the default audit level is the value specified for the `nss.alt.zone.auditlevel` parameter.

## nss.user.override.userlist

This configuration parameter enables you to specify an audit level for a list of users that will bypass Active Directory. In most cases, the auditing service connects to Active Directory to get user profile and audit level information. You can use this parameter to bypass Active Directory, for example, to specify local user accounts that do not have a corresponding user account in Active Directory, but for which you want to audit session activity. This parameter replaces the deprecated `user.ignore` parameter.



You can specify the parameter value by typing individual entries using the format `user_name[:audit_level]`, separated by spaces, or by using the `file:` keyword and a file location.

You can set the `audit_level` to one of the following valid values:

- `use_sysrights`
- `audit_if_possible`
- `no_audit`
- `audit_required`

The `use_sysrights` setting indicates that you want to use the audit level information associated with the user's role. If you don't specify an audit level for a user with this parameter, the default audit level is to the audit level you specify for the `nss.user.override.audit_level` parameter. For example, you can set the value using individual user name entries like this:

```
nss.user.override.userlist: maya:use_sysrights tai:no_audit
carlos
```

Alternatively, you can use the `file:` keyword and a file that has one `user_name[:audit_level]` per line. For example:

```
nss.user.override.userlist: file:/etc/centrifyda/user_auditing_
classiczones
```

Be sure to run the `dareload` command after modifying the configuration file to have the changes take effect.

Note that this parameter is most commonly used to specify the audit level for local user accounts. However, you can use it to specify both local and Active Directory users, if needed. To include Active Directory users in the list of users specified with this parameter, you must specify the Active Directory user's UNIX login name as a parameter value in the user list you define with this parameter.

**Note:** For computers that have only the Centrify Client for Linux installed, there is a sample file that you can use to specify users outside of Active Directory. The sample file is `/etc/centrifyda/nss.user.override.userlist.sample`. To point the client to this sample file, include the following line in your `centrifyda.conf` file:

```
nss.user.override.userlist:
/etc/centrifyda/nss.user.override.userlist.sample
```



## preferred.audit.store

If your UNIX or Linux computer has multiple IP addresses that match the criteria for multiple audit stores, use the `preferred.audit.store` parameter to specify the preferred audit store that the DirectAudit agent will use.

If you have this kind of installation and you don't specify the preferred audit store, the collector may or may not connect to the correct audit store.

## spool.diskspace.logstate.reset.threshold

This configuration parameter specifies a threshold percentage of disk space that is added to the minimum percentage of disk space (set in the `spool.diskspace.min` parameter) that determines when the information/warning/error log state is reset. Message logging resumes only after the log state is reset.

When disk space drops below the minimum percentage (for example, 10%), a warning is logged. Additional warnings are not logged until disk space has risen above the minimum percentage + threshold percentage (for example,  $10\% + 2\% = 12\%$ ), and then drops again below the minimum percentage (10%).

Setting a threshold percentage is useful to prevent unnecessary log messages when disk space hovers near the minimum percentage and would otherwise trigger a log message every time the minimum percentage is crossed.

The default value is 2 percent of disk space.

You can also set this parameter using group policy.

## spool.diskspace.min

This configuration parameter specifies the minimum volume of disk space required on the partition containing the offline spool file before spooling stops.

You can set this value as a percentage of the disk space, or you can set it as an exact size. To set the value as an exact size, specify the unit value after the number value. The unit values are not case-sensitive.



For example, to set the minimum volume of disk space to 12 gigabytes, you would enter the following:

```
spool.diskspace.min: 12 GB
```

You can specify the following unit values:

- B (byte)
- KB (kilobytes)
- MB (megabytes)
- GB (gigabytes)
- TB (terabytes)

To specify the value as a percentage, you can either use the percent (%) symbol, or enter a number with no unit value.

The default value is 10 percent of disk space.

You can also set this parameter using group policy.

## spool.diskspace.softlimit

This configuration parameter specifies the volume of disk space required on the partition containing the offline spool file to avoid warnings in the log. If available disk falls below the level specified in this parameter, a warning is logged and auditing will continue until disk space falls below the level specified in `spool.diskspace.min`.

You can set this value as a percentage of the disk space, or you can set it as an exact size. To set the value as an exact size, specify the unit value after the number value. The unit values are not case-sensitive.

For example, to set the minimum volume of disk space to 5 kilobytes, you would enter the following:

```
spool.diskspace.min: 5 kb
```

You can specify the following unit values:

- B (byte)
- KB (kilobytes)
- MB (megabytes)



- GB (gigabytes)
- TB (terabytes)

To specify the value as a percentage, you can either use the percent (%) symbol, or enter a number with no unit value.

The value of this parameter must be greater than or equal to the value of `spool.diskspace.min`.

The default value is 12 percent of disk space.

You can also set this parameter using group policy.

## spool.maxdbsize

This configuration parameter specifies maximum disk space in bytes to allocate to the offline storage database. Use 0 to designate no limit.

The default value is 0 (unlimited).

You can also set this parameter using group policy.

## uid.ignore

This configuration parameter specifies one or more numeric user identifiers (UID) that you want to ignore for authentication and lookup requests in Active Directory. In most cases, you use this parameter to specify local user accounts that do not have a corresponding user account in Active Directory, but for which you want to audit session activity. You can specify the parameter value by typing individual user identifiers, separated by spaces, or by using the `file:` keyword and a file location. For example, you can set the value using individual UID values like this:

```
uid.ignore: 0 500 5861
```

Alternatively, you can use the `file:` keyword and the sample `uid.ignore` file that is installed with the Centrify agent. The sample `uid.ignore` file ignores the most common default system accounts. For example:

```
uid.ignore: file:/etc/centrifydc/uid.ignore
```

If you edit the `/etc/centrifydc/uid.ignore` file, be sure to run the `adreload` command after modifying the file to have the changes take effect.



## user.ignore

This configuration parameter specifies one or more user names that you want to ignore for authentication and lookup requests in Active Directory. In most cases, you use this parameter to specify local user accounts that do not have a corresponding user account in Active Directory, but for which you want to audit session activity. You can specify the parameter value by typing individual user names, separated by spaces, or by using the `file:` keyword and a file location. For example, you can set the value using individual user name values like this:

```
user.ignore: tai carlos games gopher
```

You can also specify the user's audit level by adding the value after the user names added to `user.ignore`. For example:

```
user.ignore: tai carlos:audit_if_possible
```

Alternatively, you can use the `file:` keyword and the sample `user.ignore` file that is installed with the Centrify agent. The sample `user.ignore` file ignores the most common default system accounts. For example:

```
user.ignore: file:/etc/centrifydc/user.ignore
```

If you edit the `/etc/centrifydc/user.ignore` file, be sure to run the `adreload` command after modifying the file to have the changes take effect.

## user.ignore.audit.level

This configuration parameter is no longer used except for backward compatibility. It has been replaced by the `nss.user.override.audit.level` parameter. This configuration parameter specifies the audit level setting for the user that you want to ignore for authentication and lookup requests in Active Directory. In most cases, you use this parameter when you have specified local user accounts that do not have a corresponding user account in Active Directory, but for which you want to audit session activity. By default, the users you specify for the `uid.ignore` or `user.ignore` parameter are audited if auditing is enabled and the auditing service (dad) is running on the local computer (audit if possible). You can disable the auditing of user activity for the users specified by the `uid.ignore` or `user.ignore` parameter by setting this parameter value to one (1). For example, if you don't want to audit activity for the users specified in `uid.ignore` or `user.ignore` list, set the parameter as follows:

```
user.ignore.audit.level: 1
```



You cannot require auditing for the users specified in `uid.ignore` or `user.ignore` list because those users would be unable to log on if the auditing service stops running or is removed from a local computer. To prevent users from being locked out, you can only set this parameter to audit if possible (0) or no auditing (1).



# Customizing LDAP Proxy configuration parameters

This section describes the configuration parameters that affect the use of Centrify direct access LDAP proxy handling on the local host computer.

<code>ldaproxy.cache.credential.expire</code> .....	325
<code>ldaproxy.netgroup.use.rfc2307nisnetgroup</code> .....	326
<code>ldaproxy.performance.log.interval</code> .....	326

## `ldaproxy.cache.credential.expire`

This configuration parameter specifies how long `ldaproxy` holds the `ldaproxy` credential cache. The cache expires in the specified number of seconds. Default is 300 seconds:

```
ldaproxy.cache.credential.expires <seconds>
```

The `ldaproxy` credential cache retains the user login Active Directory verification.

- If a user logs in within the expiration window, `ldaproxy` does not require another `adclient` verification.
- If a user logs in and the `ldaproxy` credential cache is expired, `ldaproxy` sends a request for authentication to `adclient`. `adclient` can authenticate using its own cache or send to AD for authentication.

Restart `centrify-ldaproxy` to apply changes.

```
[root]$ /usr/share/centrifydc/bin/centrify-ldaproxy restart
```



## ldaproxy.netgroup.use.rfc2307nisnetgroup

This configuration parameter specifies which type of netgroup object that the LDAP proxy searches. By default, this parameter is set to false and the LDAP proxy searches the NIS map netgroup objects.

```
ldaproxy.netgroup.use.rfc2307nisnetgroup false
```

If you set this `ldaproxy.netgroup.use.rfc2307nisnetgroup` parameter to true, then the LDAP proxy searches RFC2307 NIS netgroup objects.

Restart `centrify-ldaproxy` to apply changes.

```
[root]$ /usr/share/centrifydc/bin/centrify-ldaproxy restart
```

## ldaproxy.performance.log.interval

This configuration parameter specifies the interval between each log dump of `ldaproxy` events. The interval is number of seconds. The default value, 0, disables the parameter.

```
ldaproxy.cache.enabled true
```

```
ldaproxy.performance.log.interval <seconds>
```

The `ldaproxy` statistics collected include:

- Search cache statistics, such as: cache hit, cache negative hit, cache miss, cache expires, cache overflow and object clone error.

Search cache statistics require the `ldaproxy.cache.enabled` parameter is set to true and the `ldaproxy.performance.log.interval` is set to a non-zero value.

- Authentication statistics, such as: authentication requests, satisfied from cache, actual authentication to adclient, and average response time of actual authentication through adclient.

Authentication statistics require the `ldaproxy.performance.log.interval` is set to a non-zero value.

Restart `centrify-ldaproxy` to apply changes.

```
[root]$ /usr/share/centrifydc/bin/centrify-ldaproxy restart
```