Rocky Enterprise Linux 9.2 Manual Pages on command 'krb5.conf.5'

$ man krb5.conf.5

KRB5.CONF(5) MIT Kerberos KRB5.CONF(5)

NAME

krb5.conf - Kerberos configuration file

The krb5.conf file contains Kerberos configuration information, includ?
ing the locations of KDCs and admin servers for the Kerberos realms of
interest, defaults for the current realm and for Kerberos applications,
and mappings of hostnames onto Kerberos realms. Normally, you should
install your krb5.conf file in the directory /etc. You can override
the default location by setting the environment variable KRB5_CONFIG.
Multiple colon-separated filenames may be specified in KRB5_CONFIG; all
files which are present will be read. Starting in release 1.14, direc?
tory names can also be specified in KRB5_CONFIG; all files within the
directory whose names consist solely of alphanumeric characters,
dashes, or underscores will be read.

STRUCTURE

The krb5.conf file is set up in the style of a Windows INI file. Lines
beginning with '#' or ';' (possibly after initial whitespace) are ig?
nored as comments. Sections are headed by the section name, in square

brackets. Each section may contain zero or more relations, of the form:

```
foo = bar
```

or:

```
fubar = {
    foo = bar
    baz = quux
}
```

Placing a '*' after the closing bracket of a section name indicates that the section is final, meaning that if the same section appears within a later file specified in KRB5_CONFIG, it will be ignored. A subsection can be marked as final by placing a '*' after either the tag name or the closing brace.

The krb5.conf file can include other files using either of the follow? ing directives at the beginning of a line:

```
include FILENAME
includedir DIRNAME
```

FILENAME or DIRNAME should be an absolute path. The named file or di? rectory must exist and be readable. Including a directory includes all files within the directory whose names consist solely of alphanumeric characters, dashes, or underscores. Starting in release 1.15, files with names ending in ".conf" are also included, unless the name begins with ".". Included profile files are syntactically independent of their parents, so each included file must begin with a section header. Starting in release 1.17, files are read in alphanumeric order; in pre? vious releases, they may be read in any order.

The krb5.conf file can specify that configuration should be obtained from a loadable module, rather than the file itself, using the follow? ing directive at the beginning of a line before any section headers:

```
module MODULEPATH:RESIDUAL
```

MODULEPATH may be relative to the library path of the krb5 installa? tion, or it may be an absolute path. RESIDUAL is provided to the mod? ule at initialization time. If krb5.conf uses a module directive,

kdc.conf(5) should also use one if it exists.

SECTIONS

The krb5.conf file may contain the following sections:

```
????????????????????????????????????????????????
?[libdefaults]  ? Settings  used by the Ker? ?
?               ? beros V5 library           ?
????????????????????????????????????????????????
?[realms]       ? Realm-specific contact in? ?
?               ? formation and settings     ?
????????????????????????????????????????????????
?[domain_realm] ? Maps  server  hostnames to ?
?               ? Kerberos realms            ?
????????????????????????????????????????????????
?[capaths]      ? Authentication  paths  for ?
?               ? non-hierarchical           ?
?               ? cross-realm                ?
????????????????????????????????????????????????
?[appdefaults]  ? Settings used by some Ker? ?
?               ? beros V5 applications      ?
????????????????????????????????????????????????
?[plugins]      ? Controls   plugin   module ?
?               ? registration               ?
????????????????????????????????????????????????
```

Additionally, krb5.conf may include any of the relations described in kdc.conf(5), but it is not a recommended practice.

[libdefaults]

The libdefaults section may contain any of the following relations:

allow_weak_crypto

If this flag is set to false, then weak encryption types (as noted in Encryption_types in kdc.conf(5)) will be filtered out of the lists default_tgs_enctypes, default_tkt_enctypes, and permitted_enctypes. The default value for this tag is false.

canonicalize

If this flag is set to true, initial ticket requests to the KDC will request canonicalization of the client principal name, and answers with different client principals than the requested principal will be accepted. The default value is false.

ccache_type

This parameter determines the format of credential cache types created by kinit(1) or other programs. The default value is 4, which represents the most current format. Smaller values can be used for compatibility with very old implementations of Kerberos which interact with credential caches on the same host.

clockskew

Sets the maximum allowable amount of clockskew in seconds that the library will tolerate before assuming that a Kerberos mes? sage is invalid. The default value is 300 seconds, or five min? utes.

The clockskew setting is also used when evaluating ticket start and expiration times. For example, tickets that have reached their expiration time can still be used (and renewed if they are renewable tickets) if they have been expired for a shorter dura? tion than the clockskew setting.

default_ccache_name

This relation specifies the name of the default credential cache. The default is FILE:/tmp/krb5cc_%{uid}. This relation is subject to parameter expansion (see below). New in release 1.11.

default_client_keytab_name

This relation specifies the name of the default keytab for ob? taining client credentials. The default is FILE:/var/ker? beros/krb5/user/%{euid}/client.keytab. This relation is subject to parameter expansion (see below). New in release 1.11.

default_keytab_name

This relation specifies the default keytab name to be used by application servers such as sshd. The default is

FILE:/etc/krb5.keytab. This relation is subject to parameter expansion (see below).

default_rcache_name

This relation specifies the name of the default replay cache. The default is dfl:. This relation is subject to parameter ex? pansion (see below). New in release 1.18.

default_realm

Identifies the default Kerberos realm for the client. Set its value to your Kerberos realm. If this value is not set, then a realm must be specified with every Kerberos principal when in? voking programs such as kinit(1).

default_tgs_enctypes

Identifies the supported list of session key encryption types that the client should request when making a TGS-REQ, in order of preference from highest to lowest. The list may be delimited with commas or whitespace. See Encryption_types in kdc.conf(5) for a list of the accepted values for this tag. Starting in re? lease 1.18, the default value is the value of permitted_enc? types. For previous releases or if permitted_enctypes is not set, the default value is aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 aes256-cts-hmac-sha384-192 aes128-cts-hmac-sha256-128 arcfour-hmac-md5 camellia256-cts-cmac camellia128-cts-cmac.

Do not set this unless required for specific backward compati? bility purposes; stale values of this setting can prevent clients from taking advantage of new stronger enctypes when the libraries are upgraded.

default_tkt_enctypes

Identifies the supported list of session key encryption types that the client should request when making an AS-REQ, in order of preference from highest to lowest. The format is the same as for default_tgs_enctypes. Starting in release 1.18, the default value is the value of permitted_enctypes. For previous releases

or if permitted_enctypes is not set, the default value is aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 aes256-cts-hmac-sha384-192 aes128-cts-hmac-sha256-128 arc? four-hmac-md5 camellia256-cts-cmac camellia128-cts-cmac. Do not set this unless required for specific backward compati? bility purposes; stale values of this setting can prevent clients from taking advantage of new stronger enctypes when the libraries are upgraded.

dns_canonicalize_hostname

Indicate whether name lookups will be used to canonicalize host? names for use in service principal names. Setting this flag to false can improve security by reducing reliance on DNS, but means that short hostnames will not be canonicalized to fully-qualified hostnames. If this option is set to fallback (new in release 1.18), DNS canonicalization will only be per? formed the server hostname is not found with the original name when requesting credentials. The default value is true.

dns_lookup_kdc

Indicate whether DNS SRV records should be used to locate the KDCs and other servers for a realm, if they are not listed in the krb5.conf information for the realm. (Note that the ad? min_server entry must be in the krb5.conf realm information in order to contact kadmind, because the DNS implementation for kadmin is incomplete.)

Enabling this option does open up a type of denial-of-service attack, if someone spoofs the DNS records and redirects you to another server. However, it's no worse than a denial of ser? vice, because that fake KDC will be unable to decode anything you send it (besides the initial ticket request, which has no encrypted data), and anything the fake KDC sends will not be trusted without verification using some secret that it won't know.

dns_uri_lookup

Indicate whether DNS URI records should be used to locate the KDCs and other servers for a realm, if they are not listed in the krb5.conf information for the realm. SRV records are used as a fallback if no URI records were found. The default value is true. New in release 1.15.

enforce_ok_as_delegate

If this flag to true, GSSAPI credential delegation will be dis? abled when the ok-as-delegate flag is not set in the service ticket. If this flag is false, the ok-as-delegate ticket flag is only enforced when an application specifically requests en? forcement. The default value is false.

err_fmt

This relation allows for custom error message formatting. If a value is set, error messages will be formatted by substituting a normal error message for %M and an error code for %C in the value.

extra_addresses

This allows a computer to use multiple local addresses, in order to allow Kerberos to work in a network that uses NATs while still using address-restricted tickets. The addresses should be in a comma-separated list. This option has no effect if noad? dresses is true.

forwardable

If this flag is true, initial tickets will be forwardable by de? fault, if allowed by the KDC. The default value is false.

ignore_acceptor_hostname

When accepting GSSAPI or krb5 security contexts for host-based service principals, ignore any hostname passed by the calling application, and allow clients to authenticate to any service principal in the keytab matching the service name and realm name (if given). This option can improve the administrative flexi? bility of server applications on multihomed hosts, but could compromise the security of virtual hosting environments. The

default value is false. New in release 1.10.

k5login_authoritative

If this flag is true, principals must be listed in a local user's k5login file to be granted login access, if a .k5login(5) file exists. If this flag is false, a principal may still be granted login access through other mechanisms even if a k5login file exists but does not list the principal. The default value is true.

k5login_directory

If set, the library will look for a local user's k5login file within the named directory, with a filename corresponding to the local username. If not set, the library will look for k5login files in the user's home directory, with the filename .k5login. For security reasons, .k5login files must be owned by the local user or by root.

kcm_mach_service

On macOS only, determines the name of the bootstrap service used to contact the KCM daemon for the KCM credential cache type. If the value is -, Mach RPC will not be used to contact the KCM daemon. The default value is org.h5l.kcm.

kcm_socket

Determines the path to the Unix domain socket used to access the KCM daemon for the KCM credential cache type. If the value is -, Unix domain sockets will not be used to contact the KCM dae? mon. The default value is /var/run/.heim_org.h5l.kcm-socket.

kdc_default_options

Default KDC options (Xored for multiple values) when requesting initial tickets. By default it is set to 0x00000010 (KDC_OPT_RENEWABLE_OK).

kdc_timesync

Accepted values for this relation are 1 or 0. If it is nonzero, client machines will compute the difference between their time and the time returned by the KDC in the timestamps in the tick?

ets and use this value to correct for an inaccurate system clock when requesting service tickets or authenticating to services. This corrective factor is only used by the Kerberos library; it is not used to change the system clock. The default value is 1.

noaddresses

If this flag is true, requests for initial tickets will not be made with address restrictions set, allowing the tickets to be used across NATs. The default value is true.

permitted_enctypes

Identifies the encryption types that servers will permit for session keys and for ticket and authenticator encryption, or? dered by preference from highest to lowest. Starting in release 1.18, this tag also acts as the default value for de? fault_tgs_enctypes and default_tkt_enctypes. The default value for this tag is aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 aes256-cts-hmac-sha384-192 aes128-cts-hmac-sha256-128 arc? four-hmac-md5 camellia256-cts-cmac camellia128-cts-cmac.

plugin_base_dir

If set, determines the base directory where krb5 plugins are lo? cated. The default value is the krb5/plugins subdirectory of the krb5 library directory. This relation is subject to parame? ter expansion (see below) in release 1.17 and later.

preferred_preauth_types

This allows you to set the preferred preauthentication types which the client will attempt before others which may be adver? tised by a KDC. The default value for this setting is "17, 16, 15, 14", which forces libkrb5 to attempt to use PKINIT if it is supported.

proxiable

If this flag is true, initial tickets will be proxiable by de? fault, if allowed by the KDC. The default value is false.

qualify_shortname

If this string is set, it determines the domain suffix for sin?

gle-component hostnames when DNS canonicalization is not used (either because dns_canonicalize_hostname is false or because forward canonicalization failed). The default value is the first search domain of the system's DNS configuration. To dis? able qualification of shortnames, set this relation to the empty string with qualify_shortname = "". (New in release 1.18.)

radius_md5_fips_override

Downstream-only option to enable use of MD5 in RADIUS communica? tion (libkrad). This allows for local (or protected tunnel) communication with a RADIUS server that doesn't use krad (e.g., freeradius) while in FIPS mode.

rdns If this flag is true, reverse name lookup will be used in addi? tion to forward name lookup to canonicalizing hostnames for use in service principal names. If dns_canonicalize_hostname is set to false, this flag has no effect. The default value is true.

realm_try_domains

Indicate whether a host's domain components should be used to determine the Kerberos realm of the host. The value of this variable is an integer: -1 means not to search, 0 means to try the host's domain itself, 1 means to also try the domain's imme? diate parent, and so forth. The library's usual mechanism for locating Kerberos realms is used to determine whether a domain is a valid realm, which may involve consulting DNS if dns_lookup_kdc is set. The default is not to search domain com? ponents.

renew_lifetime

(duration string.) Sets the default renewable lifetime for ini? tial ticket requests. The default value is 0.

spake_preauth_groups

A whitespace or comma-separated list of words which specifies the groups allowed for SPAKE preauthentication. The possible values are:

?edwards25519 ? Edwards25519 curve (RFC ?

? ? 7748) ?

??

?P-256 ? NIST P-256 curve (RFC ?

? ? 5480) ?

??

?P-384 ? NIST P-384 curve (RFC ?

? ? 5480) ?

??

?P-521 ? NIST P-521 curve (RFC ?

? ? 5480) ?

??

The default value for the client is edwards25519. The default value for the KDC is empty. New in release 1.17.

ticket_lifetime

(duration string.) Sets the default lifetime for initial ticket requests. The default value is 1 day.

udp_preference_limit

When sending a message to the KDC, the library will try using TCP before UDP if the size of the message is above udp_prefer? ence_limit. If the message is smaller than udp_prefer? ence_limit, then UDP will be tried before TCP. Regardless of the size, both protocols will be tried if the first attempt fails.

verify_ap_req_nofail

If this flag is true, then an attempt to verify initial creden? tials will fail if the client machine does not have a keytab. The default value is false.

client_aware_channel_bindings

If this flag is true, then all application protocol authentica? tion requests will be flagged to indicate that the application supports channel bindings when operating over a secure channel. The default value is false.

[realms]

Each tag in the [realms] section of the file is the name of a Kerberos realm. The value of the tag is a subsection with relations that define the properties of that particular realm. For each realm, the following tags may be specified in the realm's subsection:

admin_server

Identifies the host where the administration server is running. Typically, this is the primary Kerberos server. This tag must be given a value in order to communicate with the kadmind(8) server for the realm.

auth_to_local

This tag allows you to set a general rule for mapping principal names to local user names. It will be used if there is not an explicit mapping for the principal name that is being trans? lated. The possible values are:

RULE:exp

The local name will be formulated from exp.

The format for exp is [n:string](regexp)s/pattern/re? placement/g. The integer n indicates how many components the target principal should have. If this matches, then a string will be formed from string, substituting the realm of the principal for $0 and the n'th component of the principal for $n (e.g., if the principal was john? doe/admin then [2:$2$1foo] would result in the string ad? minjohndoefoo). If this string matches regexp, then the s//[g] substitution command will be run over the string. The optional g will cause the substitution to be global over the string, instead of replacing only the first match in the string.

DEFAULT

The principal name will be used as the local user name. If the principal has more than one component or is not in the default realm, this rule is not applicable and the

conversion will fail.

For example:

```
[realms]
    ATHENA.MIT.EDU = {
        auth_to_local = RULE:[2:$1](johndoe)s/^.*$/guest/
        auth_to_local = RULE:[2:$1;$2](^.*;admin$)s/;admin$//
        auth_to_local = RULE:[2:$2](^.*;root)s/^.*$/root/
        auth_to_local = DEFAULT
    }
```

would result in any principal without root or admin as the sec? ond component to be translated with the default rule. A princi? pal with a second component of admin will become its first com? ponent. root will be used as the local name for any principal with a second component of root. The exception to these two rules are any principals johndoe/*, which will always get the local name guest.

auth_to_local_names

This subsection allows you to set explicit mappings from princi? pal names to local user names. The tag is the mapping name, and the value is the corresponding local user name.

default_domain

This tag specifies the domain used to expand hostnames when translating Kerberos 4 service principals to Kerberos 5 princi? pals (for example, when converting rcmd.hostname to host/host? name.domain).

disable_encrypted_timestamp

If this flag is true, the client will not perform encrypted timestamp preauthentication if requested by the KDC. Setting this flag can help to prevent dictionary attacks by active at? tackers, if the realm's KDCs support SPAKE preauthentication or if initial authentication always uses another mechanism or al? ways uses FAST. This flag persists across client referrals dur? ing initial authentication. This flag does not prevent the KDC

from offering encrypted timestamp. New in release 1.17.

http_anchors

When KDCs and kpasswd servers are accessed through HTTPS prox?
ies, this tag can be used to specify the location of the CA cer?
tificate which should be trusted to issue the certificate for a
proxy server. If left unspecified, the system-wide default set
of CA certificates is used.

The syntax for values is similar to that of values for the
pkinit_anchors tag:

FILE: filename

filename is assumed to be the name of an OpenSSL-style ca-bundle
file.

DIR: dirname

dirname is assumed to be an directory which contains CA certifi?
cates. All files in the directory will be examined; if they
contain certificates (in PEM format), they will be used.

ENV: envvar

envvar specifies the name of an environment variable which has
been set to a value conforming to one of the previous values.

For example, ENV:X509_PROXY_CA, where environment variable
X509_PROXY_CA has been set to FILE:/tmp/my_proxy.pem.

kdc The name or address of a host running a KDC for that realm. An
optional port number, separated from the hostname by a colon,
may be included. If the name or address contains colons (for
example, if it is an IPv6 address), enclose it in square brack?
ets to distinguish the colon from a port separator. For your
computer to be able to communicate with the KDC for each realm,
this tag must be given a value in each realm subsection in the
configuration file, or there must be DNS SRV records specifying
the KDCs.

kpasswd_server

Points to the server where all the password changes are per?
formed. If there is no such entry, DNS will be queried (unless

forbidden by dns_lookup_kdc). Finally, port 464 on the ad?
min_server host will be tried.

master_kdc

The name for primary_kdc prior to release 1.19. Its value is used as a fallback if primary_kdc is not specified.

primary_kdc

Identifies the primary KDC(s). Currently, this tag is used in only one case: If an attempt to get credentials fails because of an invalid password, the client software will attempt to contact the primary KDC, in case the user's password has just been changed, and the updated database has not been propagated to the replica servers yet. New in release 1.19.

v4_instance_convert

This subsection allows the administrator to configure exceptions to the default_domain mapping rule. It contains V4 instances (the tag name) which should be translated to some specific host? name (the tag value) as the second component in a Kerberos V5 principal name.

v4_realm

This relation is used by the krb524 library routines when con? verting a V5 principal name to a V4 principal name. It is used when the V4 realm name and the V5 realm name are not the same, but still share the same principal names and passwords. The tag value is the Kerberos V4 realm name.

[domain_realm]

The [domain_realm] section provides a translation from hostnames to Kerberos realms. Each tag is a domain name, providing the mapping for that domain and all subdomains. If the tag begins with a period (.) then it applies only to subdomains. The Kerberos realm may be identi? fied either in the realms section or using DNS SRV records. Tag names should be in lower case. For example:

```
[domain_realm]
    crash.mit.edu = TEST.ATHENA.MIT.EDU
```

.dev.mit.edu = TEST.ATHENA.MIT.EDU

mit.edu = ATHENA.MIT.EDU

maps the host with the name crash.mit.edu into the TEST.ATHENA.MIT.EDU realm. The second entry maps all hosts under the domain dev.mit.edu into the TEST.ATHENA.MIT.EDU realm, but not the host with the name dev.mit.edu. That host is matched by the third entry, which maps the host mit.edu and all hosts under the domain mit.edu that do not match a preceding rule into the realm ATHENA.MIT.EDU.

If no translation entry applies to a hostname used for a service prin? cipal for a service ticket request, the library will try to get a re? ferral to the appropriate realm from the client realm's KDC. If that does not succeed, the host's realm is considered to be the hostname's domain portion converted to uppercase, unless the realm_try_domains setting in [libdefaults] causes a different parent domain to be used.

[capaths]

In order to perform direct (non-hierarchical) cross-realm authentica? tion, configuration is needed to determine the authentication paths be? tween realms.

A client will use this section to find the authentication path between its realm and the realm of the server. The server will use this sec? tion to verify the authentication path used by the client, by checking the transited field of the received ticket.

There is a tag for each participating client realm, and each tag has subtags for each of the server realms. The value of the subtags is an intermediate realm which may participate in the cross-realm authentica? tion. The subtags may be repeated if there is more then one intermedi? ate realm. A value of "." means that the two realms share keys di? rectly, and no intermediate realms should be allowed to participate.

Only those entries which will be needed on the client or the server need to be present. A client needs a tag for its local realm with sub? tags for all the realms of servers it will need to authenticate to. A server needs a tag for each realm of the clients it will serve, with a subtag of the server realm.

For example, ANL.GOV, PNL.GOV, and NERSC.GOV all wish to use the ES.NET realm as an intermediate realm. ANL has a sub realm of TEST.ANL.GOV which will authenticate with NERSC.GOV but not PNL.GOV. The [capaths] section for ANL.GOV systems would look like this:

```
[capaths]
    ANL.GOV = {
        TEST.ANL.GOV = .
        PNL.GOV = ES.NET
        NERSC.GOV = ES.NET
        ES.NET = .
    }
    TEST.ANL.GOV = {
        ANL.GOV = .
    }
    PNL.GOV = {
        ANL.GOV = ES.NET
    }
    NERSC.GOV = {
        ANL.GOV = ES.NET
    }
    ES.NET = {
        ANL.GOV = .
    }
```

The [capaths] section of the configuration file used on NERSC.GOV systems would look like this:

```
[capaths]
    NERSC.GOV = {
        ANL.GOV = ES.NET
        TEST.ANL.GOV = ES.NET
        TEST.ANL.GOV = ANL.GOV
        PNL.GOV = ES.NET
        ES.NET = .
    }
```

```
        ANL.GOV = {
            NERSC.GOV = ES.NET
        }
        PNL.GOV = {
            NERSC.GOV = ES.NET
        }
        ES.NET = {
            NERSC.GOV = .
        }
        TEST.ANL.GOV = {
            NERSC.GOV = ANL.GOV
            NERSC.GOV = ES.NET
        }
```

When a subtag is used more than once within a tag, clients will use the order of values to determine the path. The order of values is not important to servers.

[appdefaults]

Each tag in the [appdefaults] section names a Kerberos V5 application or an option that is used by some Kerberos V5 application[s]. The value of the tag defines the default behaviors for that application.

For example:

```
    [appdefaults]
        telnet = {
            ATHENA.MIT.EDU = {
                option1 = false
            }
        }
        telnet = {
            option1 = true
            option2 = true
        }
        ATHENA.MIT.EDU = {
            option2 = false
```

}

option2 = true

The above four ways of specifying the value of an option are shown in order of decreasing precedence. In this example, if telnet is running in the realm EXAMPLE.COM, it should, by default, have option1 and op? tion2 set to true. However, a telnet program in the realm ATHENA.MIT.EDU should have option1 set to false and option2 set to true. Any other programs in ATHENA.MIT.EDU should have option2 set to false by default. Any programs running in other realms should have op? tion2 set to true.

The list of specifiable options for each application may be found in that application's man pages. The application defaults specified here are overridden by those specified in the realms section.

[plugins]

? pwqual interface

? kadm5_hook interface

? clpreauth and kdcpreauth interfaces

Tags in the [plugins] section can be used to register dynamic plugin modules and to turn modules on and off. Not every krb5 pluggable in? terface uses the [plugins] section; the ones that do are documented here.

New in release 1.9.

Each pluggable interface corresponds to a subsection of [plugins]. All subsections support the same tags:

disable

This tag may have multiple values. If there are values for this tag, then the named modules will be disabled for the pluggable interface.

enable_only

This tag may have multiple values. If there are values for this tag, then only the named modules will be enabled for the plug? gable interface.

module This tag may have multiple values. Each value is a string of

the form modulename:pathname, which causes the shared object lo?
cated at pathname to be registered as a dynamic module named
modulename for the pluggable interface. If pathname is not an
absolute path, it will be treated as relative to the
plugin_base_dir value from [libdefaults].

For pluggable interfaces where module order matters, modules registered with a module tag normally come first, in the order they are regis? tered, followed by built-in modules in the order they are documented below. If enable_only tags are used, then the order of those tags overrides the normal module order.

The following subsections are currently supported within the [plugins] section:

ccselect interface

The ccselect subsection controls modules for credential cache selection within a cache collection. In addition to any registered dynamic mod? ules, the following built-in modules exist (and may be disabled with the disable tag):

k5identity
Uses a .k5identity file in the user's home directory to select a client principal

realm Uses the service realm to guess an appropriate cache from the collection

hostname
If the service principal is host-based, uses the service host? name to guess an appropriate cache from the collection

pwqual interface

The pwqual subsection controls modules for the password quality inter? face, which is used to reject weak passwords when passwords are changed. The following built-in modules exist for this interface:

dict Checks against the realm dictionary file

empty Rejects empty passwords

hesiod Checks against user information stored in Hesiod (only if Ker? beros was built with Hesiod support)

princ Checks against components of the principal name

kadm5_hook interface

The kadm5_hook interface provides plugins with information on principal creation, modification, password changes and deletion. This interface can be used to write a plugin to synchronize MIT Kerberos with another database such as Active Directory. No plugins are built in for this interface.

kadm5_auth interface

The kadm5_auth section (introduced in release 1.16) controls modules for the kadmin authorization interface, which determines whether a client principal is allowed to perform a kadmin operation. The follow? ing built-in modules exist for this interface:

acl This module reads the kadm5.acl(5) file, and authorizes opera? tions which are allowed according to the rules in the file.

self This module authorizes self-service operations including pass? word changes, creation of new random keys, fetching the client's principal record or string attributes, and fetching the policy record associated with the client principal.

clpreauth and kdcpreauth interfaces

The clpreauth and kdcpreauth interfaces allow plugin modules to provide client and KDC preauthentication mechanisms. The following built-in modules exist for these interfaces:

pkinit This module implements the PKINIT preauthentication mechanism.

encrypted_challenge

This module implements the encrypted challenge FAST factor.

encrypted_timestamp

This module implements the encrypted timestamp mechanism.

hostrealm interface

The hostrealm section (introduced in release 1.12) controls modules for the host-to-realm interface, which affects the local mapping of host? names to realm names and the choice of default realm. The following built-in modules exist for this interface:

profile

This module consults the [domain_realm] section of the profile for authoritative host-to-realm mappings, and the default_realm variable for the default realm.

dns This module looks for DNS records for fallback host-to-realm mappings and the default realm. It only operates if the dns_lookup_realm variable is set to true.

domain This module applies heuristics for fallback host-to-realm mappings. It implements the realm_try_domains variable, and uses the uppercased parent domain of the hostname if that does not produce a result.

localauth interface

The localauth section (introduced in release 1.12) controls modules for the local authorization interface, which affects the relationship between Kerberos principals and local system accounts. The following built-in modules exist for this interface:

default

This module implements the DEFAULT type for auth_to_local values.

rule This module implements the RULE type for auth_to_local values.

names This module looks for an auth_to_local_names mapping for the principal name.

auth_to_local

This module processes auth_to_local values in the default realm's section, and applies the default method if no auth_to_local values exist.

k5login

This module authorizes a principal to a local account according to the account's .k5login(5) file.

an2ln This module authorizes a principal to a local account if the principal name maps to the local account name.

certauth interface

The certauth section (introduced in release 1.16) controls modules for the certificate authorization interface, which determines whether a

certificate is allowed to preauthenticate a user via PKINIT. The fol? lowing built-in modules exist for this interface:

pkinit_san

This module authorizes the certificate if it contains a PKINIT Subject Alternative Name for the requested client principal, or a Microsoft UPN SAN matching the principal if pkinit_allow_upn is set to true for the realm.

pkinit_eku

This module rejects the certificate if it does not contain an Extended Key Usage attribute consistent with the pkinit_eku_checking value for the realm.

dbmatch

This module authorizes or rejects the certificate according to whether it matches the pkinit_cert_match string attribute on the client principal, if that attribute is present.

PKINIT OPTIONS

NOTE:

The following are PKINIT-specific options. These values may be specified in [libdefaults] as global defaults, or within a realm-specific subsection of [libdefaults], or may be specified as realm-specific values in the [realms] section. A realm-specific value overrides, not adds to, a generic [libdefaults] specification.

The search order is:

1. realm-specific subsection of [libdefaults]:

```
[libdefaults]
    EXAMPLE.COM = {
        pkinit_anchors = FILE:/usr/local/example.com.crt
    }
```

2. realm-specific value in the [realms] section:

```
[realms]
    OTHERREALM.ORG = {
        pkinit_anchors = FILE:/usr/local/otherrealm.org.crt
    }
```

3. generic value in the [libdefaults] section:

[libdefaults]

pkinit_anchors = DIR:/usr/local/generic_trusted_cas/

Specifying PKINIT identity information

The syntax for specifying Public Key identity, trust, and revocation information for PKINIT is as follows:

FILE:filename[,keyfilename]

This option has context-specific behavior.

In pkinit_identity or pkinit_identities, filename specifies the name of a PEM-format file containing the user's certificate. If keyfilename is not specified, the user's private key is expected to be in filename as well. Otherwise, keyfilename is the name of the file containing the private key.

In pkinit_anchors or pkinit_pool, filename is assumed to be the name of an OpenSSL-style ca-bundle file.

DIR:dirname

This option has context-specific behavior.

In pkinit_identity or pkinit_identities, dirname specifies a di? rectory with files named *.crt and *.key where the first part of the file name is the same for matching pairs of certificate and private key files. When a file with a name ending with .crt is found, a matching file ending with .key is assumed to contain the private key. If no such file is found, then the certificate in the .crt is not used.

In pkinit_anchors or pkinit_pool, dirname is assumed to be an OpenSSL-style hashed CA directory where each CA cert is stored in a file named hash-of-ca-cert.#. This infrastructure is en? couraged, but all files in the directory will be examined and if they contain certificates (in PEM format), they will be used.

In pkinit_revoke, dirname is assumed to be an OpenSSL-style hashed CA directory where each revocation list is stored in a file named hash-of-ca-cert.r#. This infrastructure is encour? aged, but all files in the directory will be examined and if

they contain a revocation list (in PEM format), they will be used.

PKCS12:filename

filename is the name of a PKCS #12 format file, containing the user's certificate and private key.

PKCS11:[module_name=]modname[:slotid=slot-id][:token=token-label][:cer?tid=cert-id][:certlabel=cert-label]

All keyword/values are optional. modname specifies the location of a library implementing PKCS #11. If a value is encountered with no keyword, it is assumed to be the modname. If no mod?ule-name is specified, the default is p11-kit-proxy.so. slotid= and/or token= may be specified to force the use of a particular smard card reader or token if there is more than one available. certid= and/or certlabel= may be specified to force the selec?tion of a particular certificate on the device. See the pkinit_cert_match configuration option for more ways to select a particular certificate to use for PKINIT.

ENV:envvar

envvar specifies the name of an environment variable which has been set to a value conforming to one of the previous values. For example, ENV:X509_PROXY, where environment variable X509_PROXY has been set to FILE:/tmp/my_proxy.pem.

PKINIT krb5.conf options

pkinit_anchors

Specifies the location of trusted anchor (root) certificates which the client trusts to sign KDC certificates. This option may be specified multiple times. These values from the config file are not used if the user specifies X509_anchors on the com?mand line.

pkinit_cert_match

Specifies matching rules that the client certificate must match before it is used to attempt PKINIT authentication. If a user has multiple certificates available (on a smart card, or via

other media), there must be exactly one certificate chosen be? fore attempting PKINIT authentication. This option may be spec? ified multiple times. All the available certificates are checked against each rule in order until there is a match of ex? actly one certificate.

The Subject and Issuer comparison strings are the RFC 2253 string representations from the certificate Subject DN and Is? suer DN values.

The syntax of the matching rules is:

[relation-operator]component-rule ...

where:

relation-operator

can be either &&, meaning all component rules must match, or ||, meaning only one component rule must match. The default is &&.

component-rule

can be one of the following. Note that there is no punc? tuation or whitespace between component rules.

<SUBJECT>regular-expression

<ISSUER>regular-expression

<SAN>regular-expression

<EKU>extended-key-usage-list

<KU>key-usage-list

extended-key-usage-list is a comma-separated list of re? quired Extended Key Usage values. All values in the list must be present in the certificate. Extended Key Usage values can be:

? pkinit

? msScLogin

? clientAuth

? emailProtection

key-usage-list is a comma-separated list of required Key Usage values. All values in the list must be present in

the certificate. Key Usage values can be:

? digitalSignature

? keyEncipherment

Examples:

pkinit_cert_match = ||<SUBJECT>.*DoE.*<SAN>.*@EXAMPLE.COM

pkinit_cert_match = &&<EKU>msScLogin,clientAuth<ISSUER>.*DoE.*

pkinit_cert_match = <EKU>msScLogin,clientAuth<KU>digitalSignature

pkinit_eku_checking

This option specifies what Extended Key Usage value the KDC cer?

tificate presented to the client must contain. (Note that if

the KDC certificate has the pkinit SubjectAlternativeName en?

coded as the Kerberos TGS name, EKU checking is not necessary

since the issuing CA has certified this as a KDC certificate.)

The values recognized in the krb5.conf file are:

kpKDC This is the default value and specifies that the KDC must

have the id-pkinit-KPKdc EKU as defined in RFC 4556.

kpServerAuth

If kpServerAuth is specified, a KDC certificate with the

id-kp-serverAuth EKU will be accepted. This key usage

value is used in most commercially issued server certifi?

cates.

none If none is specified, then the KDC certificate will not

be checked to verify it has an acceptable EKU. The use

of this option is not recommended.

pkinit_dh_min_bits

Specifies the size of the Diffie-Hellman key the client will at?

tempt to use. The acceptable values are 1024, 2048, and 4096.

The default is 2048.

pkinit_identities

Specifies the location(s) to be used to find the user's X.509

identity information. If this option is specified multiple

times, each value is attempted in order until certificates are

found. Note that these values are not used if the user speci?

fies X509_user_identity on the command line.

pkinit_kdc_hostname

The presence of this option indicates that the client is willing to accept a KDC certificate with a dNSName SAN (Subject Alterna? tive Name) rather than requiring the id-pkinit-san as defined in RFC 4556. This option may be specified multiple times. Its value should contain the acceptable hostname for the KDC (as contained in its certificate).

pkinit_pool

Specifies the location of intermediate certificates which may be used by the client to complete the trust chain between a KDC certificate and a trusted anchor. This option may be specified multiple times.

pkinit_require_crl_checking

The default certificate verification process will always check the available revocation information to see if a certificate has been revoked. If a match is found for the certificate in a CRL, verification fails. If the certificate being verified is not listed in a CRL, or there is no CRL present for its issuing CA, and pkinit_require_crl_checking is false, then verification suc? ceeds.

However, if pkinit_require_crl_checking is true and there is no CRL information available for the issuing CA, then verification fails.

pkinit_require_crl_checking should be set to true if the policy is such that up-to-date CRLs must be present for every CA.

pkinit_revoke

Specifies the location of Certificate Revocation List (CRL) in? formation to be used by the client when verifying the validity of the KDC certificate presented. This option may be specified multiple times.

PARAMETER EXPANSION

Starting with release 1.11, several variables, such as de?

fault_keytab_name, allow parameters to be expanded. Valid parameters are:

```
??????????????????????????????????????????????????
?%{TEMP}          ? Temporary directory       ?
??????????????????????????????????????????????????
?%{uid}           ? Unix real UID or Windows ?
?                 ? SID                       ?
??????????????????????????????????????????????????
?%{euid}          ? Unix effective user ID or ?
?                 ? Windows SID               ?
??????????????????????????????????????????????????
?%{USERID}        ? Same as %{uid}            ?
??????????????????????????????????????????????????
?%{null}          ? Empty string              ?
??????????????????????????????????????????????????
?%{LIBDIR}        ? Installation library di? ?
?                 ? rectory                   ?
??????????????????????????????????????????????????
?%{BINDIR}        ? Installation binary direc? ?
?                 ? tory                      ?
??????????????????????????????????????????????????
?%{SBINDIR}       ? Installation admin binary ?
?                 ? directory                 ?
??????????????????????????????????????????????????
?%{username}      ? (Unix) Username of effec? ?
?                 ? tive user ID              ?
??????????????????????????????????????????????????
?%{APPDATA}       ? (Windows) Roaming applica? ?
?                 ? tion data for current user ?
??????????????????????????????????????????????????
?%{COMMON_APPDATA} ? (Windows) Application data ?
?                 ? for all users             ?
??????????????????????????????????????????????????
```

?%{LOCAL_APPDATA} ? (Windows) Local applica? ?

? ? tion data for current user ?

??

?%{SYSTEM} ? (Windows) Windows system ?

? ? folder ?

??

?%{WINDOWS} ? (Windows) Windows folder ?

??

?%{USERCONFIG} ? (Windows) Per-user MIT ?

? ? krb5 config file directory ?

??

?%{COMMONCONFIG} ? (Windows) Common MIT krb5 ?

? ? config file directory ?

??

SAMPLE KRB5.CONF FILE

Here is an example of a generic krb5.conf file:

```
[libdefaults]
    default_realm = ATHENA.MIT.EDU
    dns_lookup_kdc = true
    dns_lookup_realm = false
[realms]
    ATHENA.MIT.EDU = {
        kdc = kerberos.mit.edu
        kdc = kerberos-1.mit.edu
        kdc = kerberos-2.mit.edu
        admin_server = kerberos.mit.edu
        primary_kdc = kerberos.mit.edu
    }
    EXAMPLE.COM = {
        kdc = kerberos.example.com
        kdc = kerberos-1.example.com
        admin_server = kerberos.example.com
    }
```

```
        [domain_realm]
            mit.edu = ATHENA.MIT.EDU
        [capaths]
            ATHENA.MIT.EDU = {
                EXAMPLE.COM = .
            }
            EXAMPLE.COM = {
                ATHENA.MIT.EDU = .
            }
```

FILES

/etc/krb5.conf

SEE ALSO

syslog(3)

AUTHOR

MIT

COPYRIGHT