Kerberos V5 System Administrator's Guide

Release: 1.3
Document Edition: 1.0
Last updated: June 16, 2000

MIT

Copyright

Copyright © 1985-2002 by the Massachusetts Institute of Technology.

> Export of software employing encryption from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Furthermore if you modify this software you must label your software as modified software and not distribute it in such a fashion that it might be confused with the original MIT software. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

The following copyright and permission notice applies to the OpenVision Kerberos Administration system located in kadmin/create, kadmin/dbutil, kadmin/passwd, kadmin/server, lib/kadm5, and portions of lib/rpc:

> Copyright, OpenVision Technologies, Inc., 1996, All Rights Reserved
>
> WARNING: Retrieving the OpenVision Kerberos Administration system source code, as described below, indicates your acceptance of the following terms. If you do not agree to the following terms, do not retrieve the OpenVision Kerberos administration system.
>
> You may freely use and distribute the Source Code and Object Code compiled from it, with or without modification, but this Source Code is provided to you "AS IS" EXCLUSIVE OF ANY WARRANTY, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, OR ANY OTHER WARRANTY, WHETHER EXPRESS OR IMPLIED. IN NO EVENT WILL OPENVISION HAVE ANY LIABILITY FOR ANY LOST PROFITS, LOSS OF DATA OR COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS AGREEMENT, INCLUDING, WITHOUT LIMITATION, THOSE RESULTING FROM THE USE OF THE SOURCE CODE, OR THE FAILURE OF THE SOURCE CODE TO PERFORM, OR FOR ANY OTHER REASON.
>
> OpenVision retains all copyrights in the donated Source Code. OpenVision also retains copyright to derivative works of the Source Code, whether created by OpenVision or by a third party. The OpenVision copyright notice must be preserved if derivative works are made based on the donated Source Code.

OpenVision Technologies, Inc. has donated this Kerberos Administration system to MIT for inclusion in the standard Kerberos 5 distribution. This donation underscores our commitment to continuing Kerberos technology development and our gratitude for the valuable work which has been performed by MIT and the Kerberos community.

The implementation of the Yarrow pseudo-random number generator in src/lib/crypto/yarrow has the following copyright:

Copyright 2000 by Zero-Knowledge Systems, Inc.

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Zero-Knowledge Systems, Inc. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Zero-Knowledge Systems, Inc. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

ZERO-KNOWLEDGE SYSTEMS, INC. DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL ZERO-KNOWLEDGE SYSTEMS, INC. BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTUOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

The implementation of the AES encryption algorithm in src/lib/crypto/aes has the following copyright:

Copyright (c) 2001, Dr Brian Gladman <brg@gladman.uk.net>, Worcester, UK. All rights reserved.

LICENSE TERMS

The free distribution and use of this software in both source and binary form is allowed (with or without changes) provided that:

1. distributions of this source code include the above copyright notice, this list of conditions and the following disclaimer;
2. distributions in binary form include the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other associated materials;
3. the copyright holder's name is not used to endorse products built using this software without specific written permission.

DISCLAIMER

This software is provided 'as is' with no explcit or implied warranties in respect of any properties, including, but not limited to, correctness and fitness for purpose.

Kerberos V5 includes documentation and software developed at the University of California at Berkeley, which includes this copyright notice:

Copyright © 1983 Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

   This product includes software developed by the University of California, Berkeley and its contributors.
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notices and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided also that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions.

1 Introduction

1.1 Why Should I use Kerberos?

Since Kerberos negotiates authenticated, and optionally encrypted, communications between two points anywhere on the internet, it provides a layer of security that is not dependent on which side of a firewall either client is on. Since studies have shown that half of the computer security breaches in industry happen from *inside* firewalls, Kerberos V5 from MIT will play a vital role in the security of your network.

1.2 Documentation for Kerberos V5

This document is one piece of the document set for Kerberos V5. The documents, and their intended audiences, are:

- **Kerberos V5 Installation Guide**: a concise guide for installing Kerberos V5. Kerberos administrators (particularly whoever will be making site-wide decisions about the installation) and the system administrators who will be installing the software should read this guide.
- **Kerberos V5 System Administrator's Guide**: a sysadmin's guide to administering a Kerberos installation. The System Administrator's Guide describes the administration software and suggests policies and procedures for administering a Kerberos installation. Anyone who will have administrative access to your Kerberos database should read this guide.
- **Kerberos V5 UNIX User's Guide**: a guide to using the Kerberos UNIX client programs. All users on UNIX systems should read this guide, particularly the "Tutorial" section.

1.3 Overview of This Guide

The next chapter describes how Kerberos works.

Chapter three describes administration of the principals in the Kerberos database.

Chapter four describes how you can use DNS in configuring your Kerberos realm.

Chapter five describes administrative programs for manipulating the Kerberos database as a whole.

Chapter six describes issues to consider when adding an application server to the database.

Chapter seven describes our problem reporting system.

The appendices include the list of Kerberos error messages, and a complete list of the time zones understood by `kadmin`.

2 How Kerberos Works

This section provides a simplified description of a general user's interaction with the Kerberos system. This interaction happens transparently—users don't need to know and probably don't care about what's going on—but Kerberos administrators might find a schematic description of the process useful. This description glosses over a lot of details; for more information, see *Kerberos: An Authentication Service for Open Network Systems*, a paper presented at Winter USENIX 1988, in Dallas, Texas. This paper can be retreived by FTP from `athena-dist.mit.edu`, in the location: `/pub/ATHENA/kerberos/doc/usenix.PS`.

2.1 Network Services and Their Client Programs

In an environment that provides network services, you use *client* programs to request *services* from *server* programs that are somewhere on the network. Suppose you have logged in to a workstation and you want to '`rlogin`' to a typical UNIX host. You use the local '`rlogin`' client program to contact the remote machine's '`rlogind`' daemon.

2.2 Kerberos Tickets

Under Kerberos, the '`klogind`' daemon allows you to login to a remote machine if you can provide '`klogind`' a Kerberos ticket which proves your identity. In addition to the ticket, you must also have possession of the corresponding ticket session key. The combination of a ticket and the ticket's session key is known as a credential.

Typically, a client program automatically obtains credentials identifying the person using the client program. The credentials are obtained from a Kerberos server that resides somewhere on the network. A Kerberos server maintains a database of user, server, and password information.

2.3 The Kerberos Database

Kerberos will give you credentials only if you have an entry in the Kerberos server's *Kerberos database*. Your database entry includes your Kerberos *principal* (an identifying string, which is often just your username), and your Kerberos password. Every Kerberos user must have an entry in this database.

2.4 Kerberos Realms

Each administrative domain will have its own Kerberos database, which contains information about the users and services for that particular site or administrative domain. This administrative domain is the *Kerberos realm*.

Each Kerberos realm will have at least one Kerberos server, where the master Kerberos database for that site or administrative domain is stored. A Kerberos realm may also have one or more *slave servers*, which have read-only copies of the Kerberos database that are periodically propagated from the master server. For more details on how this is done, see the "Set Up the Slave KDCs for Database Propagation" and "Propagate the Database to Each Slave KDC" sections of the Kerberos V5 Installation Guide.

2.5 The Ticket-Granting Ticket

The '`kinit`' command prompts for your password. If you enter it successfully, you will obtain a *ticket-granting ticket* and a *ticket session key* which gives you the right to use the ticket. This combination of the ticket and its associated key is known as your *credentials*. As illustrated below, client programs use your ticket-granting ticket credentials in order to obtain client-specific credentials as needed.

Your credentials are stored in a *credentials cache*, which is often just a file in `/tmp`. The credentials cache is also called the *ticket file*, especially in Kerberos V4 documentation. Note, however, that a credentials cache does not have to be stored in a file.

2.6 Network Services and the Master Database

The master database also contains entries for all network services that require Kerberos authentication. Suppose that your site has a machine, '`laughter.mit.edu`', that requires Kerberos authentication from anyone who wants to '`rlogin`' to it. The host's Kerberos realm is '`ATHENA.MIT.EDU`'.

This service must be registered in the Kerberos database, using the proper service name, which in this case is the *principal*:

```
host/laughter.mit.edu@ATHENA.MIT.EDU
```

The '`/`' character separates the Kerberos *primary* (in this case, '`host`') from the *instance* (in this case, '`laughter.mit.edu`'); the '`@`' character separates the realm name (in this case, '`ATHENA.MIT.EDU`') from the rest of the principal. The primary, '`host`', denotes the name or type of the service that is being offered: generic host-level access to the machine. The instance, '`laughter.mit.edu`', names the specific machine that is offering this service. There will generally be many different machines, each offering one particular type of service, and the instance serves to give each one of these servers a different Kerberos principal.

2.6.1 The Keytab File

For each service, there must also be a *service key* known only by Kerberos and the service. On the Kerberos server, the service key is stored in the Kerberos database.

On the server host, these service keys are stored in *key tables*, which are files known as *keytabs*.[1] For example, the service keys used by services that run as root are usually stored in the keytab file `/etc/krb5.keytab`. **N.B.:** This service key is the equivalent of the service's password, and must be kept secure. Data which is meant to be read only by the service is encrypted using this key.

2.7 The User/Kerberos Interaction

Suppose that you walk up to a host intending to login to it, and then '`rlogin`' to the machine '`laughter`'. Here's what happens:

1. You login to the workstation and use the '`kinit`' command to get a ticket-granting ticket. This command prompts you for your Kerberos password. (On systems running the Kerberos V5 '`login`' program, this may be done as part of the login process, not requiring the user to run a separate program.)
   A. The '`kinit`' command sends your request to the Kerberos master server machine. The server software looks for your principal name's entry in the Kerberos database.
   B. If this entry exists, the Kerberos server creates and returns a ticket-granting ticket and the key which allows you to use it, encrypted by your password. If '`kinit`' can decrypt the Kerberos reply using the password you provide, it stores this ticket in a credentials cache on your local machine for later use. The name of the credentials cache can be specified in the '`KRB5CCNAME`' environment variable. If this variable is not set, the name of the file will be '`/tmp/krb5cc_<uid>`', where `<uid>` is your UNIX user-id, represented in decimal format.
2. Now you use the '`rlogin`' client to access the machine '`laughter`'.

   `host%` **rlogin laughter**

   A. The '`rlogin`' client checks your ticket file to see if you have a ticket for the '`host`' service for '`laughter`'. You don't, so '`rlogin`' uses the credential cache's ticket-granting ticket to make a request to the master server's ticket-granting service.
   B. This ticket-granting service receives the request for a ticket for '`host/laughter.mit.edu`', and looks in the master database for an entry for '`host/laughter.mit.edu`'. If the entry exists, the ticket-granting service issues you a ticket for that service. That ticket is also cached in your credentials cache.

[1] Keytabs were called *srvtabs* in Kerberos V4.

C. The '`rlogin`' client now sends that ticket to the '`laughter`' '`klogind`' service program. The service program checks the ticket by using its own service key. If the ticket is valid, it now knows your identity. If you are allowed to login to '`laughter`' (because your username matches one in /etc/passwd, or your Kerberos principal is in the appropriate '`.k5login`' file), `klogind` will let you login.

2.8 Definitions

Following are definitions of some of the Kerberos terminology.

client an entity that can obtain a ticket. This entity is usually either a user or a host.

host a computer that can be accessed over a network.

Kerberos in Greek mythology, the three-headed dog that guards the entrance to the underworld. In the computing world, Kerberos is a network security package that was developed at MIT.

KDC Key Distribution Center. A machine that issues Kerberos tickets.

keytab a **key tab**le file containing one or more keys. A host or service uses a *keytab* file in much the same way as a user uses his/her password.

principal a string that names a specific entity to which a set of credentials may be assigned. It can have an arbitrary number of components, but generally has three:

- **primary** the first part of a Kerberos *principal*. In the case of a user, it is the username. In the case of a service, it is the name of the service.
- **instance** the second part of a Kerberos *principal*. It gives information that qualifies the primary. The instance may be null. In the case of a user, the instance is often used to describe the intended use of the corresponding credentials. In the case of a host, the instance is the fully qualified hostname.
- **realm** the logical network served by a single Kerberos database and a set of Key Distribution Centers. By convention, realm names are generally all uppercase letters, to differentiate the realm from the internet domain.

The typical format of a typical Kerberos principal is primary/instance@REALM.

service any program or computer you access over a network. Examples of services include "host" (a host, *e.g.*, when you use `telnet` and `rsh`), "ftp" (FTP), "krbtgt" (authentication; cf. *ticket-granting ticket*), and "pop" (email).

ticket a temporary set of electronic credentials that verify the identity of a client for a particular service.

TGT Ticket-Granting Ticket. A special Kerberos ticket that permits the client to obtain additional Kerberos tickets within the same Kerberos realm.

3 Configuration Files

3.1 Supported Encryption Types

Any tag in the configuration files which requires a list of encryption types can be set to some combination of the following strings.

`des-cbc-crc`
: DES cbc mode with CRC-32

`des-cbc-md4`
: DES cbc mode with RSA-MD4

`des-cbc-md5`
: DES cbc mode with RSA-MD5

`des3-cbc-sha1`
`des3-hmac-sha1`
`des3-cbc-sha1-kd`
: triple DES cbc mode with HMAC/sha1

`des-hmac-sha1`
: DES with HMAC/sha1

`aes256-cts-hmac-sha1-96`
`aes256-cts`
: AES-256 CTS mode with 96-bit SHA-1 HMAC

`aes128-cts-hmac-sha1-96`
`aes128-cts`
: AES-128 CTS mode with 96-bit SHA-1 HMAC

`arcfour-hmac`
`rc4-hmac`
`arcfour-hmac-md5`
: RC4 with HMAC/MD5

`arcfour-hmac-exp`
`rc4-hmac-exp`
`arcfour-hmac-md5-exp`
: exportable RC4 with HMAC/MD5

While aes128-cts and aes256-cts are supported for all Kerberos operations, they are not supported by the GSSAPI. AES GSSAPI support will be added after the necessary standardization work is completed.

By default, AES is enabled on clients and application servers. Because of the lack of support for GSSAPI, AES is disabled in the default KDC supported_enctypes Section 3.4 [kdc.conf], page 25. Sites wishing to use AES encryption types on their KDCs need to be careful not to give GSSAPI services AES keys. If GSSAPI services are given AES keys, then services will start to fail in the future when clients supporting AES for GSSAPI are deployed before updated servers that support AES for GSSAPI. Sites may wish to use AES for user keys and for the ticket granting ticket key, although doing so requires specifying what encryption types are used as each principal is created. Alternatively sites can use the default configuration which will make AES support available in clients and servers but not actually use this support until a future version of Kerberos adds support to GSSAPI.

3.2 Salts

Your Kerberos key is derived from your password. To ensure that people who happen to pick the same password do not have the same key, Kerberos 5 incorporates more information into the key using something called a salt. The supported values for salts are as follows.

`normal` default for Kerberos Version 5

`v4` the only type used by Kerberos Version 4, no salt

`norealm` same as the default, without using realm information

`onlyrealm`
uses only realm information as the salt

`afs3` AFS version 3, only used for compatibility with Kerberos 4 in AFS

`special` only used in very special cases; not fully supported

3.3 krb5.conf

The `krb5.conf` file contains Kerberos configuration information, including 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`'.

The `krb5.conf` file is set up in the style of a Windows INI file. 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 '*' at the end of a line indicates that this is the *final* value for the tag. This means that neither the remainder of this configuration file nor any other configuration file will be checked for any other values for this tag.

For example, if you have the following lines:

```
foo = bar*
foo = baz
```

then the second value of foo (baz) would never be read.

The `krb5.conf` file may contain any or all of the following sections:

libdefaults Contains default values used by the Kerberos V5 library.

login Contains default values used by the Kerberos V5 login program.

appdefaults
Contains default values that can be used by Kerberos V5 applications.

realms Contains subsections keyed by Kerberos realm names. Each subsection describes realm-specific information, including where to find the Kerberos servers for that realm.

domain_realm
Contains relations which map domain names and subdomains onto Kerberos realm names. This is used by programs to determine what realm a host should be in, given its fully qualified domain name.

logging Contains relations which determine how Kerberos programs are to perform logging.

capaths Contains the authentication paths used with direct (nonhierarchical) cross-realm authentication. Entries in this section are used by the client to determine the intermediate realms which may be used in cross-realm authentication. It is also used by the end-service when checking the transited field for trusted intermediate realms.

3.3.1 [libdefaults]

The `libdefaults` section may contain any of the following relations:

default_keytab_name
This relation specifies the default keytab name to be used by application servers such as telnetd and rlogind. The default is /etc/krb5.keytab.

default_realm
Identifies the default Kerberos realm for the client. Set its value to your Kerberos realm. If this is not specified and the TXT record lookup is enabled (see Chapter 4 [Using DNS], page 31), then that information will be used to determine the default

realm. If this tag is not set in this configuration file and there is no DNS information found, then an error will be returned.

default_tgs_enctypes
Identifies the supported list of session key encryption types that should be returned by the KDC. The list may be delimited with commas or whitespace. Kerberos supports many different encryption types, and support for more is planned in the future. (see Section 3.1 [Supported Encryption Types], page 11 for a list of the accepted values for this tag). The default value is aes256-cts-hmac-sha1-96 des3-cbc-sha1 arcfour-hmac-md5 des-cbc-crc des-cbc-md5 des-cbc-md4.

default_tkt_enctypes
Identifies the supported list of session key encryption types that should be requested by the client. The format is the same as for *default_tgs_enctypes*. The default value for this tag is aes256-cts-hmac-sha1-96 des3-cbc-sha1 arcfour-hmac-md5 des-cbc-crc des-cbc-md5 des-cbc-md4.

permitted_enctypes
Identifies all encryption types that are permitted for use in session key encryption. The default value for this tag is aes256-cts-hmac-sha1-96 des3-cbc-sha1 arcfour-hmac-md5 des-cbc-crc des-cbc-md5 des-cbc-md4.

clockskew
Sets the maximum allowable amount of clockskew in seconds that the library will tolerate before assuming that a Kerberos message is invalid. The default value is 300 seconds, or five minutes.

kdc_timesync
If this is set to 1 (for true), then client machines will compute the difference between their time and the time returned by the KDC in the timestamps in the tickets and use this value to correct for an inaccurate system clock. This corrective factor is only used by the Kerberos library. The default is 1.

kdc_req_checksum_type
ap_req_checksum_type
safe_checksum_type
An integer which specifies the type of checksum to use. Used for compatability with DCE security servers which do not support the default RSA MD5 used by this version of Kerberos. The possible values and their meanings are as follows.

1 CRC32

2 RSA MD4

3 RSA MD4 DES

4 DES CBC

7 RSA MD5

8 RSA MD5 DES

9 NIST SHA

12 HMAC SHA1 DES3

-138 Microsoft MD5 HMAC checksum type

ccache_type
Use this parameter on systems which are DCE clients, to specify the type of cache to be created by kinit, or when forwarded tickets are received. DCE and Kerberos can share the cache, but some versions of DCE do not support the default cache as created by this version of Kerberos. Use a value of 1 on DCE 1.0.3a systems, and a value of 2 on DCE 1.1 systems. The default value is 4.

krb4_srvtab
Specifies the location of the Kerberos V4 srvtab file. Default is /etc/srvtab.

krb4_config
Specifies the location of hte Kerberos V4 configuration file. Default is /etc/krb.conf.

krb4_realms
Specifies the location of the Kerberos V4 domain/realm translation file. Default is /etc/krb.realms.

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 information for the realm. (Note that the '`admin_server`' entry must be in the file, because the DNS implementation for it 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 service, 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.

If this option is not specified but '`dns_fallback`' is, that value will be used instead. If neither option is specified, the behavior depends on configure-time options; if none were given, the default is to enable this option. If the DNS support is not compiled in, this entry has no effect.

dns_lookup_realm
Indicate whether DNS TXT records should be used to determine the Kerberos realm of a host.

Enabling this option may permit a redirection attack, where spoofed DNS replies persuade a client to authenticate to the wrong realm, when talking to the wrong host (either by spoofing yet more DNS records or by intercepting the net traffic). Depending on how the client software manages hostnames, however, it could already be vulnerable to such attacks. We are looking at possible ways to minimize or eliminate this exposure. For now, we encourage more adventurous sites to try using Secure DNS.

If this option is not specified but '`dns_fallback`' is, that value will be used instead. If neither option is specified, the behavior depends on configure-time options; if none were given, the default is to disable this option. If the DNS support is not compiled in, this entry has no effect.

dns_fallback
: General flag controlling the use of DNS for Kerberos information. If both of the preceding options are specified, this option has no effect.

extra_addresses
: This allows a computer to use multiple local addresses, in order to allow Kerberos to work in a network that uses NATs. The addresses should be in a comma-separated list.

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_preference_list`. If the message is smaller than `udp_preference_list`, 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 set, then an attempt to get initial credentials will fail if the client machine does not have a keytab. The default for the flag is not set.

renew_lifetime
: The value of this tag is the default renewable lifetime for initial tickets. The default value for the tag is 0.

noaddresses
: Setting this flag causes the initial Kerberos ticket to be addressless. The default for the flag is set.

forwardable
: If this flag is set, initial tickets by default will be forwardable. The default value for this flag is not set.

proxiable
: If this flag is set, initial tickets by default will be proxiable. The default value for this flag is not set.

3.3.2 [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 option2 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 option2 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.

A special application name (afs_krb5) is used by the krb524 service to know whether new format AFS tokens based on Kerberos 5 can be used rather than the older format which used a converted Kerberos 4 ticket. The new format allows for cross-realm authentication without introducing a security hole. It is used by default. Older AFS servers (before OpenAFS 1.2.8) will not support the new format. If servers in your cell do not support the new format, you will need to add an `afs_krb5` relation to the `appdefaults` section. The following config file shows how to disable new format AFS tickets for the `afs.example.com` cell in the `EXAMPLE.COM` realm.

```
[appdefaults]
    afs_krb5 = {
        EXAMPLE.COM = {
            afs/afs.example.com = false
        }
    }
```

3.3.3 [login]

Each tag in the [login] section of the file is an option for login.krb5. This section may contain any of the following relations:

krb5_get_tickets
Indicate whether or not to use a user's password to get V5 tickets. The default value is true.

krb4_get_tickets
Indicate whether or not to user a user's password to get V4 tickets. The default value is false.

krb4_convert
Indicate whether or not to use the Kerberos conversion daemon to get V4 tickets. The default value is false. If this is set to false and krb4_get_tickets is true, then login will get the V5 tickets directly using the Kerberos V4 protocol directly. This does not currently work with non-MIT-V4 salt types (such as the AFS3 salt type). Note that if this is set to true and krb524d is not running, login will hang for approximately a minute under Solaris, due to a Solaris socket emulation bug.

krb_run_aklog
Indicate whether or not to run aklog. The default value is false.

aklog_path
Indicate where to find aklog. The default value is $(prefix)/bin/aklog.

accept_passwd
A true value will cause login not to accept plaintext passwords. The default value is false. This is not yet implemented.

3.3.4 [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:

kdc
The name of a host running a KDC for that realm. An optional port number (separated from the hostname by a colon) may be included. 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 (see Chapter 4 [Using DNS], page 31).

admin_server
Identifies the host where the administration server is running. Typically, this is the master Kerberos server. This tag must be given a value in order to communicate with the kadmin server for the realm.

default_domain
This tag is used for Kerberos 4 compatibility. Kerberos 4 does not require the entire hostname of a server to be in its principal like Kerberos 5 does. This tag provides the domain name needed to produce a full hostname when translating V4 principal names

into V5 principal names. All servers in this realm are assumed to be in the domain given as the value of this tag

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 hostname (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 converting 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.

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

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 translated. The possible values are:

DB:*filename*
The principal will be looked up in the database *filename*. Support for this is not currently compiled in by default.

RULE:*exp* The local name will be formulated from *exp*.

The format for *exp* is `[`*n*`:$`*d*`..`*string*`](`*regexp*`)s/`*pattern*`/`*replacement*`/g`. The integer *n* indicates how many components the target principal should have. If this matches, then a string will be formed by putting together the components of the principal in the order indicated by each integer *d*, and the arbitrary string *string* (i.e. if the principal was johndoe/admin then [2:$2$1foo] would result in the string "adminjohndoefoo". 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/
            RULE:[2:$1;$2](^.*;admin$)s/;admin$//
            RULE:[2:$2](^.*;root)s/^.*$/root/
            DEFAULT
            }
        }
```

would result in any principal without `root` or `admin` as the second component to be translated with the default rule. A principal with a second component of `admin` will become its first component. `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`.

3.3.5 [domain_realm]

The [domain_realm] section provides a translation from a domain name or hostname to a Kerberos realm name. The tag name can be a host name, or a domain name, where domain names are indicated by a prefix of a period ('`.`'). The value of the relation is the Kerberos realm name for that particular host or domain. Host names and domain names should be in lower case.

If no translation entry applies, the host's realm is considered to be the hostname's domain portion converted to upper case. For example, the following [domain_realm] section:

```
[domain_realm]
    .mit.edu = ATHENA.MIT.EDU
    mit.edu = ATHENA.MIT.EDU
    crash.mit.edu = TEST.ATHENA.MIT.EDU
    example.com = EXAMPLE.COM
```

maps crash.mit.edu into the TEST.ATHENA.MIT.EDU realm. All other hosts in the mit.edu domain will map by default to the ATHENA.MIT.EDU realm, and all hosts in the example.com domain will map by default into the EXAMPLE.COM realm. Note the entries for the hosts mit.edu and example.com. Without these entries, these hosts would be mapped into the Kerberos realms '`EDU`' and '`ORG`', respectively.

3.3.6 [logging]

The [logging] section indicates how a particular entity is to perform its logging. The relations in this section assign one or more values to the entity name. Currently, the following entities are used:

kdc These entries specify how the KDC is to perform its logging.

admin_server
These entries specify how the administrative server is to perform its logging.

default These entries specify how to perform logging in the absence of explicit specifications otherwise.

Values are of the following forms:

FILE=<filename>
FILE:<filename>
This value causes the entity's logging messages to go to the specified file. If the '`=`' form is used, the file is overwritten. If the '`:`' form is used, the file is appended to.

STDERR This value causes the entity's logging messages to go to its standard error stream.

CONSOLE
This value causes the entity's logging messages to go to the console, if the system supports it.

DEVICE=<devicename>
This causes the entity's logging messages to go to the specified device.

SYSLOG[**:<severity>**[**:<facility>**]]
This causes the entity's logging messages to go to the system log.

The *severity* argument specifies the default severity of system log messages. This may be any of the following severities supported by the `syslog(3)` call, minus the LOG_ prefix: LOG_EMERG, LOG_ALERT, LOG_CRIT, LOG_ERR, LOG_WARNING, LOG_NOTICE, LOG_INFO, and LOG_DEBUG. For example, a value of '`CRIT`' would specify LOG_CRIT severity.

The facility argument specifies the facility under which the messages are logged. This may be any of the following facilities supported by the syslog(3) call minus the LOG_ prefix: LOG_KERN, LOG_USER, LOG_MAIL, LOG_DAEMON, LOG_AUTH, LOG_LPR, LOG_NEWS, LOG_UUCP, LOG_CRON, and LOG_LOCAL0 through LOG_LOCAL7.

If no severity is specified, the default is ERR. If no facility is specified, the default is AUTH.

In the following example, the logging messages from the KDC will go to the console and to the system log under the facility LOG_DAEMON with default severity of LOG_INFO; and the logging messages from the administrative server will be appended to the file /var/adm/kadmin.log and sent to the device /dev/tty04.

```
[logging]
    kdc = CONSOLE
    kdc = SYSLOG:INFO:DAEMON
    admin_server = FILE:/var/adm/kadmin.log
    admin_server = DEVICE=/dev/tty04
```

3.3.7 [capaths]

In order to perform direct (non-hierarchical) cross-realm authentication, a database is needed to construct the authentication paths between the realms. This section defines that database.

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 section 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 realm, and each tag has subtags for each of the realms. The value of the subtags is an intermediate realm which may participate in the cross-realm authentication. The subtags may be repeated if there is more then one intermediate realm. A value of "." means that the two realms share keys directly, and no intermediate realms should be allowd to participate.

There are n**2 possible entries in this table, but only those entries which will be needed on the client or the server need to be present. The client needs a tag for its local realm, with subtags for all the realms of servers it will need to authenticate with. A server needs a tag for each realm of the clients it will serve.

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
    }
```

In the above examples, the ordering is not important, except when the same subtag name is used more then once. The client will use this to determine the path. (It is not important to the server, since the transited field is not sorted.)

This feature is not currently supported by DCE. DCE security servers can be used with Kerberized clients and servers, but versions prior to DCE 1.1 did not fill in the transited field, and should be used with caution.

3.3.8 Sample krb5.conf File

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

```
[libdefaults]
    default_realm = ATHENA.MIT.EDU
    default_tkt_enctypes = des3-hmac-sha1 des-cbc-crc
    default_tgs_enctypes = des3-hmac-sha1 des-cbc-crc
    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:750
        admin_server = kerberos.mit.edu
        default_domain = 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
    mit.edu = ATHENA.MIT.EDU

[capaths]
    ATHENA.MIT.EDU = {
     EXAMPLE.COM = .
    }
    EXAMPLE.COM = {
     ATHENA.MIT.EDU = .
    }

[logging]
    kdc = SYSLOG:INFO
    admin_server = FILE=/var/kadm5.log
```

3.4 kdc.conf

The `kdc.conf` file contains KDC configuration information, including defaults used when issuing Kerberos tickets. Normally, you should install your `kdc.conf` file in the directory `/usr/local/var/krb5kdc`. You can override the default location by setting the environment variable 'KRB5_KDC_PROFILE'.

The `kdc.conf` file is set up in the same format as the `krb5.conf` file. (See Section 3.3 [krb5.conf], page 12.) The `kdc.conf` file may contain any or all of the following three sections:

kdcdefaults
: Contains default values for overall behavior of the KDC.

realms
: Contains subsections keyed by Kerberos realm names. Each subsection describes realm-specific information, including where to find the Kerberos servers for that realm.

logging
: Contains relations which determine how Kerberos programs are to perform logging.

3.4.1 [kdcdefaults]

The following relation is defined in the [kdcdefaults] section:

kdc_ports
: This relation lists the ports on which the Kerberos server should listen for UDP requests by default. This list is a comma separated list of integers. If this relation is not specified, the compiled-in default is 88,750, the first being the assigned Kerberos port and the second which was used by Kerberos V4.

kdc_tcp_ports
: This relation lists the ports on which the Kerberos server should listen for TCP connections by default. This list is a comma separated list of integers. If this relation is not specified, the compiled-in default is not to listen for TCP connections at all.

  If you wish to change this (which we do not recommend, because the current implementation has little protection against denial-of-service attacks), the standard port number assigned for Kerberos TCP traffic is port 88.

v4_mode
: This string specifies how the KDC should respond to Kerberos 4 packets. The possible values are none, disable, full, and nopreauth. The default value is none.

3.4.2 [realms]

Each tag in the [realms] section of the file names a Kerberos realm. The value of the tag is a subsection where the relations in that subsection define KDC parameters for that particular realm.

For each realm, the following tags may be specified in the [realms] subsection:

acl_file (String.) Location of the access control list (acl) file that kadmin uses to determine which principals are allowed which permissions on the database. The default is `/usr/local/var/krb5kdc/kadm5.acl`.

admin_keytab
(String.) Location of the keytab file that the legacy administration daemons `kadmind4` and `v5passwdd` use to authenticate to the database. The default is `/usr/local/var/krb5kdc/kadm5.keytab`.

database_name
(String.) Location of the Kerberos database for this realm. The default is `/usr/local/var/krb5kdc/principal`.

default_principal_expiration
(Absolute time string.) Specifies the default expiration date of principals created in this realm. The default value for this tag is 0.

default_principal_flags
(Flag string.) Specifies the default attributes of principals created in this realm. The format for this string is a comma-separated list of flags, with '+' before each flag that should be enabled and '-' before each flag that should be disabled. The default is postdateable, forwardable, tgt-based, renewable, proxiable, dup-skey, allow-tickets, and service enabled..

There are a number of possible flags:

postdateable
Enabling this flag allows the principal to obtain postdateable tickets.

forwardable
Enabling this flag allows the principal to obtain forwardable tickets.

tgt-based Enabling this flag allows a principal to obtain tickets based on a ticket-granting-ticket, rather than repeating the authentication process that was used to obtain the TGT.

renewable Enabling this flag allows the principal to obtain renewable tickets.

proxiable Enabling this flag allows the principal to obtain proxy tickets.

dup-skey Enabling this flag allows the principal to obtain a session key for another user, permitting user-to-user authentication for this principal.

allow-tickets
Enabling this flag means that the KDC will issue tickets for this principal. Disabling this flag essentially deactivates the principal within this realm.

preauth If this flag is enabled on a client principal, then that principal is required to preauthenticate to the KDC before receiving any tickets. On a service principal, enabling this flag means that service tickets for this principal will only be issued to clients with a TGT that has the preauthenticated ticket set.

hwauth If this flag is enabled, then the principal is required to preauthenticate using a hardware device before receiving any tickets.

pwchange Enabling this flag forces a password change for this principal.

service Enabling this flag allows the the KDC to issue service tickets for this principal.

pwservice If this flag is enabled, it marks this principal as a password change service. This should only be used in special cases, for example, if a user's password has expired, then the user has to get tickets for that principal without going through the normal password authentication in order to be able to change the password.

dict_file (String.) Location of the dictionary file containing strings that are not allowed as passwords. If none is specified or if there is no policy assigned to the principal, no dictionary checks of passwords will be performed.

kadmind_port
(Port number.) Specifies the port on which the kadmind daemon is to listen for this realm. The assigned port for kadmind is 749.

kpasswd_port
(Port number.) Specifies the port on which the kpasswd daemon is to listen for this realm. The default is 464.

key_stash_file
(String.) Specifies the location where the master key has been stored (via `kdb5_util stash`). The default is `/usr/local/var/krb5kdc/.k5.`*REALM*, where *REALM* is the Kerberos realm.

kdc_ports (String.) Specifies the list of ports that the KDC is to listen to for UDP requests for this realm. By default, the value of kdc_ports as specified in the [kdcdefaults] section is used.

kdc_tcp_ports
(String.) Specifies the list of ports that the KDC is to listen to for TCP requests for this realm. By default, the value of kdc_tcp_ports as specified in the [kdcdefaults] section is used.

master_key_name
(String.) Specifies the name of the principal associated with the master key. The default is K/M.

master_key_type
(Key type string.) Specifies the master key's key type. The default value for this is des3-cbc-sha1. For a list of all possible values, see Section 3.1 [Supported Encryption Types], page 11.

max_life (Delta time string.) Specifes the maximum time period for which a ticket may be valid in this realm. The default value is 10 hours.

max_renewable_life
(Delta time string.) Specifies the maximum time period during which a valid ticket may be renewed in this realm. The default value is 0.

supported_enctypes
List of key:salt strings. Specifies the default key/salt combinations of principals for this realm. Any principals created through `kadmin` will have keys of these types. The default value for this tag is des3-hmac-sha1:normal des-cbc-crc:normal. For lists of possible values, see Section 3.1 [Supported Encryption Types], page 11 and Section 3.2 [Salts], page 12.

reject_bad_transit
A boolean value (`true`, `false`). If set to `true`, the KDC will check the list of transited realms for cross-realm tickets against the transit path computed from the realm names and the `capaths` section of its `krb5.conf` file; if the path in the ticket to be issued contains any realms not in the computed path, the ticket will not be issued, and an error will be returned to the client instead. If this value is set to `false`, such tickets will be issued anyways, and it will be left up to the application server to validate the realm transit path.

If the `disable-transited-check` flag is set in the incoming request, this check is not performed at all. Having the `reject_bad_transit` option will cause such ticket requests to be rejected always.

This transit path checking and config file option currently apply only to TGS requests.

Earlier versions of the MIT release (before 1.2.3) had bugs in the application server support such that the server-side checks may not be performed correctly. We recommend turning this option on, unless you know that all application servers in this realm have been updated to fixed versions of the software, and for whatever reason, you don't want the KDC to do the validation.

This is a per-realm option so that multiple-realm KDCs may control it separately for each realm, in case (for example) one realm has had the software on its application servers updated but another has not.

This option defaults to `true`.

3.4.3 Sample kdc.conf File

Here's an example of a `kdc.conf` file:

```
[kdcdefaults]
    kdc_ports = 88

[realms]
    ATHENA.MIT.EDU = {
        kadmind_port = 749
        max_life = 12h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
        master_key_type = des3-hmac-sha1
        supported_enctypes = des3-hmac-sha1:normal des-cbc-crc:normal des-cbc-crc:v4
    }

[logging]
    kdc = FILE:/usr/local/var/krb5kdc/kdc.log
    admin_server = FILE:/usr/local/var/krb5kdc/kadmin.log
```

4 Using DNS

4.1 Mapping Hostnames onto Kerberos Realms

Mapping hostnames onto Kerberos realms is done in one of two ways.

The first mechanism, which has been in use for years in MIT-based Kerberos distributions, works through a set of rules in the `krb5.conf` configuration file. (See Section 3.3 [krb5.conf], page 12.) You can specify mappings for an entire domain or subdomain, and/or on a hostname-by-hostname basis. Since greater specificity takes precedence, you would do this by specifying the mappings for a given domain or subdomain and listing the exceptions.

The second mechanism works by looking up the information in special `TXT` records in the Domain Name Service. This is currently not used by default because security holes could result if the DNS TXT records were spoofed. If this mechanism is enabled on the client, it will try to look up a `TXT` record for the DNS name formed by putting the prefix `_kerberos` in front of the hostname in question. If that record is not found, it will try using `_kerberos` and the host's domain name, then its parent domain, and so forth. So for the hostname BOSTON.ENGINEERING.FOOBAR.COM, the names looked up would be:

```
_kerberos.boston.engineering.foobar.com
_kerberos.engineering.foobar.com
_kerberos.foobar.com
_kerberos.com
```

The value of the first TXT record found is taken as the realm name. (Obviously, this doesn't work all that well if a host and a subdomain have the same name, and different realms. For example, if all the hosts in the ENGINEERING.FOOBAR.COM domain are in the ENGINEERING.FOOBAR.COM realm, but a host named ENGINEERING.FOOBAR.COM is for some reason in another realm. In that case, you would set up TXT records for all hosts, rather than relying on the fallback to the domain name.)

Even if you do not choose to use this mechanism within your site, you may wish to set it up anyway, for use when interacting with other sites.

4.2 Hostnames for KDCs

MIT recommends that your KDCs have a predefined set of CNAME records (DNS hostname aliases), such as `kerberos` for the master KDC and `kerberos-1`, `kerberos-2`, ... for the slave KDCs. This way, if you need to swap a machine, you only need to change a DNS entry, rather than having to change hostnames.

A new mechanism for locating KDCs of a realm through DNS has been added to the MIT Kerberos V5 distribution. A relatively new record type called `SRV` has been added to DNS. Looked up by a service name and a domain name, these records indicate the hostname and port number to contact for that service, optionally with weighting and prioritization. (See RFC 2782 if you want more information. You can follow the example below for straightforward cases.)

The use with Kerberos is fairly straightforward. The domain name used in the SRV record name is the domain-style Kerberos realm name. (It is possible to have Kerberos realm names that are not DNS-style names, but we don't recommend it for Internet use, and our code does not support it well.) Several different Kerberos-related service names are used:

`_kerberos._udp`
: This is for contacting any KDC by UDP. This entry will be used the most often. Normally you should list port 88 on each of your KDCs.

`_kerberos._tcp`
: This is for contacting any KDC by TCP. The MIT KDC by default will not listen on any TCP ports, so unless you've changed the configuration or you're running another KDC implementation, you should leave this unspecified. If you do enable TCP support, normally you should use port 88.

`_kerberos-master._udp`
: This entry should refer to those KDCs, if any, that will immediately see password changes to the Kerberos database. This entry is used only in one case, when the user is logging in and the password appears to be incorrect; the master KDC is then contacted, and the same password used to try to decrypt the response, in case the user's password had recently been changed and the first KDC contacted hadn't been updated. Only if that fails is an "incorrect password" error given.

  If you have only one KDC, or for whatever reason there is no accessible KDC that would get database changes faster than the others, you do not need to define this entry.

`_kerberos-adm._tcp`
: This should list port 749 on your master KDC. Support for it is not complete at this time, but it will eventually be used by the `kadmin` program and related utilities. For now, you will also need the `admin_server` entry in `krb5.conf`. (See Section 3.3 [krb5.conf], page 12.)

`_kpasswd._udp`
: This should list port 464 on your master KDC. It is used when a user changes her password.

`_kerberos-iv._udp`
: This should refer to your KDCs that serve Kerberos version 4 requests, if you have Kerberos v4 enabled.

Be aware, however, that the DNS SRV specification requires that the hostnames listed be the canonical names, not aliases. So, for example, you might include the following records in your (BIND-style) zone file:

```
$ORIGIN foobar.com.
_kerberos               TXT       "FOOBAR.COM"
kerberos                CNAME     daisy
kerberos-1              CNAME     use-the-force-luke
kerberos-2              CNAME     bunny-rabbit
_kerberos._udp          SRV       0 0 88 daisy
                        SRV       0 0 88 use-the-force-luke
                        SRV       0 0 88 bunny-rabbit
_kerberos-master._udp   SRV       0 0 88 daisy
_kerberos-adm._tcp      SRV       0 0 749 daisy
_kpasswd._udp           SRV       0 0 464 daisy
```

As with the DNS-based mechanism for determining the Kerberos realm of a host, we recommend distributing the information this way for use by other sites that may want to interact with yours using Kerberos, even if you don't immediately make use of it within your own site. If you anticipate installing a very large number of machines on which it will be hard to update the Kerberos configuration files, you may wish to do all of your Kerberos service lookups via DNS and not put the information (except for `admin_server` as noted above) in future versions of your `krb5.conf` files at all. Eventually, we hope to phase out the listing of server hostnames in the client-side configuration files; making preparations now will make the transition easier in the future.

5 Administrating the Kerberos Database

Your Kerberos database contains all of your realm's Kerberos principals, their passwords, and other administrative information about each principal. For the most part, you will use the `kdb5_util` program to manipulate the Kerberos database as a whole, and the `kadmin` program to make changes to the entries in the database. (One notable exception is that users will use the `kpasswd` program to change their own passwords.) The `kadmin` program has its own command-line interface, to which you type the database administrating commands.

`Kdb5_util` provides a means to create, delete, load, or dump a Kerberos database. It also includes a command to stash a copy of the master database key in a file on a KDC, so that the KDC can authenticate itself to the `kadmind` and `krb5kdc` daemons at boot time.

`Kadmin` provides for the maintenance of Kerberos principals, KADM5 policies, and service key tables (keytabs). It exists as both a Kerberos client, `kadmin`, using Kerberos authentication and an RPC, to operate securely from anywhere on the network, and as a local client, `kadmin.local`, intended to run directly on the KDC without Kerberos authentication. Other than the fact that the remote client uses Kerberos to authenticate the person using it, the functionalities of the two versions are identical. The local version is necessary to enable you to set up enough of the database to be able to use the remote version. It replaces the now obsolete `kdb5_edit` (except for database dump and load, which are provided by `kdb5_util`).

The remote version authenticates to the KADM5 server using the service principal `kadmin/admin`. If the credentials cache contains a ticket for the `kadmin/admin` principal, and the '`-c ccache`' option is specified, that ticket is used to authenticate to KADM5. Otherwise, the '`-p`' and '`-k`' options are used to specify the client Kerberos principal name used to authenticate. Once kadmin has determined the principal name, it requests a `kadmin/admin` Kerberos service ticket from the KDC, and uses that service ticket to authenticate to KADM5.

5.1 Kadmin Options

You can invoke `kadmin` or `kadmin.local` with any of the following options:

-r *REALM*
: Use *REALM* as the default Kerberos realm for the database.

-p *principal*
: Use the Kerberos principal *principal* to authenticate to Kerberos. If this option is not given, `kadmin` will append `admin` to either the primary principal name, the environment variable USER, or to the username obtained from `getpwuid`, in order of preference.

-q *query*
: Pass *query* directly to `kadmin`. This is useful for writing scripts that pass specific queries to `kadmin`.

You can invoke `kadmin` with any of the following options:

-k [**-t** *keytab*]
Use the keytab *keytab* to decrypt the KDC response instead of prompting for a password on the TTY. In this case, the principal will be '`host/`*hostname*'. If `-t` is not used to specify a keytab, then the default keytab will be used.

-c *credentials cache*
Use *credentials_cache* as the credentials cache. The credentials cache should contain a service ticket for the `kadmin/admin` service, which can be acquired with the `kinit` program. If this option is not specified, `kadmin` requests a new service ticket from the KDC, and stores it in its own temporary ccache.

-w *password*
Use *password* as the password instead of prompting for one on the TTY. Note: placing the password for a Kerberos principal with administration access into a shell script can be dangerous if unauthorized users gain read access to the script.

-s *admin_server[:port]*
Specifies the admin server that kadmin should contact.

You can invoke `kadmin.local` with an of the follwing options:

-d_ *dbname*
Specifies the name of the Kerberos database.

-e "*enctypes ...*"
Sets the list of cryptosystem and salt types to be used for any new keys created. See Section 3.1 [Supported Encryption Types], page 11 and Section 3.2 [Salts], page 12 for available types.

-m Do not authenticate using a keytab. This option will cause kadmin to prompt for the master database password.

5.2 Date Format

Many of the `kadmin` commands take a duration or time as an argument. The date can appear in a wide variety of formats, such as:

```
"15 minutes"
"7 days"
"1 month"
"2 hours"
"400000 seconds"
"next year"
"this Monday"
"next Monday"
yesterday
tomorrow
now
"second Monday"
fortnight
"3/31/1992 10:00:07 PST"
"January 23, 2007 10:05pm"
"22:00 GMT"
```

Note that if the date specification contains spaces, you must enclose it in double quotes. Note also that you cannot use a number without a unit. (I.e., “"60 seconds"” is correct, but “60” is incorrect.) All keywords are case-insensitive. The following is a list of all of the allowable keywords.

Months january, jan, february, feb, march, mar, april, apr, may, june, jun, july, jul, august, aug, september, sep, sept, october, oct, november, nov, december, dec

Days sunday, sun, monday, mon, tuesday, tues, tue, wednesday, wednes, wed, thursday, thurs, thur, thu, friday, fri, saturday, sat

Units year, month, fortnight, week, day, hour, minute, min, second, sec

Relative tomorrow, yesterday, today, now, last, this, next, first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, tenth, eleventh, twelfth, ago

Time Zones
`kadmin` recognizes abbreviations for most of the world's time zones. A complete listing appears in Section A.2 [kadmin Time Zones], page 76.

12-hour Time Delimiters
am, pm

5.3 Principals

Each entry in the Kerberos database contains a Kerberos principal (see Section 2.8 [Definitions], page 10) and the attributes and policies associated with that principal.

5.3.1 Retrieving Information About a Principal

5.3.1.1 Attributes

To retrieve a listing of the attributes and/or policies associated with a principal, use the `kadmin get_principal` command, which requires the "inquire" administrative privilege. The syntax is:

get_principal *principal*

The `get_principal` command has the alias `getprinc`.

For example, suppose you wanted to view the attributes of the principal `jennifer/root@ATHENA.MIT.EDU`. You would type:

```
shell% kadmin
kadmin: getprinc jennifer/root
Principal: jennifer/root@ATHENA.MIT.EDU
Expiration date: [never]
Last password change: Mon Jan 31 02:06:40 EDT 2002
Password Expiration date: [none]
Maximum ticket life: 0 days 10:00:00
Maximum renewable life: 7 days 00:00:00
Last modified: Wed Jul 24 14:46:25 EDT 2002 (joeadmin/admin@ATHENA.MIT.EDU)
Last successful authentication: Mon Jul 29 18:20:17 EDT 2002
Last failed authentication: Mon Jul 29 18:18:54 EDT 2002
Failed password attempts: 3
Number of keys: 2
Key: vno 2, Triple DES cbc mode with HMAC/sha1, no salt
Key: vno 2, DES cbc mode with CRC-32, no salt
Attributes: DISALLOW_FORWARDABLE, DISALLOW_PROXIABLE
Policy: [none]
kadmin:
```

The `get_principal` command has a `-terse` option, which lists the fields as a quoted, tab-separated string. For example:

```
kadmin: getprinc -terse jennifer/root
jennifer/root@ATHENA.MIT.EDU 0 1027458564
0 36000 (joeadmin/admin@ATHENA.MIT.EDU
1027536385 18 2 0 [none] 604800 1027980137
1027980054 3 2 1 2 16 0 1
2 1 0
kadmin:
```

5.3.1.2 Retrieving a List of Principals

To generate a listing of principals, use the `kadmin list_principals` command, which requires the "list" privilege. The syntax is:

list_principals [*expression*]

where *expression* is a shell-style glob expression that can contain the characters '*', '?', '[', and ']'. All policy names matching the expression are displayed. The `list_principals` command has the aliases `listprincs`, `get_principals`, and `getprincs`. For example:

```
kadmin: listprincs test*
test3@ATHENA.MIT.EDU
test2@ATHENA.MIT.EDU
test1@ATHENA.MIT.EDU
testuser@ATHENA.MIT.EDU
kadmin:
```

If no expression is provided, all principals are printed.

5.3.2 Privileges

Administrative privileges for the Kerberos database are stored in the file `kadm5.acl`.

The format of the file is:

```
Kerberos_principal      permissions     [target_principal] [restrictions]
```

The Kerberos principal (and optional target principal) can include the "*" wildcard, so if you want any principal with the instance "admin" to have full permissions on the database, you could use the principal "`*/admin@REALM`" where "REALM" is your Kerberos realm. `target_principal` can also include backreferences to `Kerberos_principal`, in which "**number*" matches the component *number* in the `Kerberos_principal`.

Note: a common use of an *admin* instance is so you can grant separate permissions (such as administrator access to the Kerberos database) to a separate Kerberos principal. For example, the user `joeadmin` might have a principal for his administrative use, called `joeadmin/admin`. This way, `joeadmin` would obtain `joeadmin/admin` tickets only when he actually needs to use those permissions.

The permissions are represented by single letters; UPPER-CASE letters represent negative permissions. The permissions are:

a allows the addition of principals or policies in the database.
A disallows the addition of principals or policies in the database.
d allows the deletion of principals or policies in the database.
D disallows the deletion of principals or policies in the database.
m allows the modification of principals or policies in the database.
M disallows the modification of principals or policies in the database.
c allows the changing of passwords for principals in the database.
C disallows the changing of passwords for principals in the database.
i allows inquiries to the database.

I disallows inquiries to the database.
l allows the listing of principals or policies in the database.
L disallows the listing of principals or policies in the database.
s allows the explicit setting of the key for a principal
S disallows the explicit setting of the key for a principal
***** All privileges (admcil).
x All privileges (admcil); identical to "*".

The restrictions are a string of flags. Allowed restrictions are:

[+ -]*flagname*
: flag is forced to indicated value. The permissible flags are the same as the `+` and `-` flags for the `kadmin addprinc` and `modprinc` commands.

-clearpolicy
: policy is forced to clear

-policy *pol*
: policy is forced to be *pol*

expire *time*
pwexpire *time*
maxlife *time*
maxrenewlife *time*
: associated value will be forced to MIN(*time*, requested value)

The above flags act as restrictions on any add or modify operation which is allowed due to that ACL line.

Here is an example of a `kadm5.acl` file. Note that order is important; permissions are determined by the first matching entry.

```
*/admin@ATHENA.MIT.EDU  *
joeadmin@ATHENA.MIT.EDU  ADMCIL
joeadmin/*@ATHENA.MIT.EDU il */root@ATHENA.MIT.EDU
*@ATHENA.MIT.EDU cil *1/admin@ATHENA.MIT.EDU
*/*@ATHENA.MIT.EDU  i
*/admin@EXAMPLE.COM * -maxlife 9h -postdateable
```

In the above file, any principal in the ATHENA.MIT.EDU realm with an `admin` instance has all administrative privileges. The user `joeadmin` has all permissions with his `admin` instance, `joeadmin/admin@ATHENA.MIT.EDU` (matches the first line). He has no permissions at all with his `null` instance, `joeadmin@ATHENA.MIT.EDU` (matches the second line). His root instance has *inquire* and *list* permissions with any other principal that has the instance `root`. Any principal in ATHENA.MIT.EDU can inquire, list, or change the password of their `admin` instance, but not any other `admin` instance. Any principal in the realm `ATHENA.MIT.EDU` (except for `joeadmin@ATHENA.MIT.EDU`, as mentioned above) has *inquire* privileges. Finally, any principal with an admin instance in EXAMPLE.COM has all permissions, but any principal that they create or modify will not be able to get postdateable tickets or tickets with a life of longer than 9 hours.

5.3.3 Adding or Modifying Principals

To add a principal to the database, use the kadmin `add_principal` command, which requires the "add" administrative privilege. This function creates the new principal, prompting twice for a password, and, if neither the -policy nor -clearpolicy options are specified and the policy "default" exists, assigns it that policy. The syntax is:

kadmin: `add_principal [`*options*`]` *principal*

To modify attributes of a principal, use the kadmin `modify_principal` command, which requires the "modify" administrative privilege. The syntax is:

kadmin: `modify_principal [`*options*`]` *principal*

`add_principal` has the aliases `addprinc` and `ank`[1]. `modify_principal` has the alias `modprinc`.

The `add_principal` and `modify_principal` commands take the following switches:

-expire *date*
: Sets the expiration date of the principal to *date*.

-pwexpire *date*
: Sets the expiration date of the password to *date*.

-maxlife *maxlife*
: Sets the maximum ticket life of the principal to *maxlife*.

-maxrenewlife *maxrenewlife*
: Sets the maximum renewable life of tickets for the principal to *maxrenewlife*.

-kvno *number*
: Explicity sets the key version number to *number*. MIT does not recommend doing this unless there is a specific reason.

-policy *policy*
: Sets the policy used by this principal. (See Section 5.4 [Policies], page 46.) With `modify_principal`, the current policy assigned to the principal is set or changed. With `add_principal`, if this option is not supplied, the -clearpolicy is not specified, and the policy "default" exists, that policy is assigned. If a principal is created with no policy, `kadmin` will print a warning message.

---

[1] `ank` was the short form of the equivalent command using the deprecated `kadmin5` database administrative tool. It has been kept

-clearpolicy

For `modify_principal`, removes the current policy from a principal. For `add_principal`, suppresses the automatic assignment of the policy "default".

{-|+}allow_postdated

The "-allow_postdated" option prohibits this principal from obtaining postdated tickets. "+allow_postdated" clears this flag. In effect, "-allow_postdated" sets the KRB5_KDB_DISALLOW_POSTDATED flag on the principal in the database.

{-|+}allow_forwardable

The "-allow_forwardable" option prohibits this principal from obtaining forwardable tickets. "+allow_forwardable" clears this flag. In effect, "-allow_forwardable" sets the KRB5_KDB_DISALLOW_FORWARDABLE flag on the principal in the database.

{-|+}allow_renewable

The "-allow_renewable" option prohibits this principal from obtaining renewable tickets. "+allow_renewable" clears this flag. In effect, "-allow_renewable" sets the KRB5_KDB_DISALLOW_RENEWABLE flag on the principal in the database.

{-|+}allow_proxiable

The "-allow_proxiable" option prohibits this principal from obtaining proxiable tickets. "+allow_proxiable" clears this flag. In effect, "-allow_proxiable" sets the KRB5_KDB_DISALLOW_PROXIABLE flag. on the principal in the database.

{-|+}allow_dup_skey

The "-allow_dup_skey" option disables user-to-user authentication for this principal by prohibiting this principal from obtaining a session key for another user. "+allow_dup_skey" clears this flag. In effect, "-allow_dup_skey" sets the KRB5_KDB_DISALLOW_DUP_SKEY flag on the principal in the database.

{-|+}requires_preauth

The "+requires_preauth" option requires this principal to preauthenticate before being allowed to kinit. -requires_preauth clears this flag. In effect, +requires_preauth sets the KRB5_KDB_REQUIRES_PRE_AUTH flag on the principal in the database.

{-|+}requires_hwauth

The "+requires_hwauth" flag requires the principal to preauthenticate using a hardware device before being allowed to kinit. "-requires_hwauth" clears this flag. In effect, "+requires_hwauth" sets the KRB5_KDB_REQUIRES_HW_AUTH flag on the principal in the database.

{-|+}allow_svr

The "-allow_svr" flag prohibits the issuance of service tickets for this principal. "+allow_svr" clears this flag. In effect, "-allow_svr" sets the KRB5_KDB_DISALLOW_SVR flag on the principal in the database.

{-|+}allow_tgs_req

The "-allow_tgs_req" option specifies that a Ticket-Granting Service (TGS) request for a service ticket for this principal is not permitted. You will probably never need to use this option. "+allow_tgs_req" clears this flag. The default is "+allow_tgs_req". In effect, "-allow_tgs_req" sets the KRB5_KDB_DISALLOW_TGT_BASED flag on the principal in the database.

{-|+}allow_tix

The "-allow_tix" option forbids the issuance of any tickets for this principal. "+allow_tix" clears this flag. The default is "+allow_tix". In effect, "-allow_tix" sets the KRB5_KDB_DISALLOW_ALL_TIX flag on the principal in the database.

{-|+}needchange

The "+needchange" option sets a flag in attributes field to force a password change; "-needchange" clears it. The default is "-needchange". In effect, "+needchange" sets the KRB5_KDB_REQUIRES_PWCHANGE flag on the principal in the database.

{-|+}password_changing_service

The "+password_changing_service" option sets a flag in the attributes field marking this principal as a password change service. (Again, you will probably never need to use this option.) "-password_changing_service" clears the flag. The default is "-password_changing_service". In effect, the "+password_changing_service" option sets the KRB5_KDB_PWCHANGE_SERVICE flag on the principal in the database.

-randkey

Sets the key for the principal to a random value (`add_principal` only). MIT recommends using this option for host keys.

-pw *password*

Sets the key of the principal to the specified string and does not prompt for a password (`add_principal` only). MIT does not recommend using this option.

-e *enc:salt...*

Uses the specified list of enctype-salttype pairs for setting the key of the principal. The quotes are necessary if there are multiple enctype-salttype pairs. This will not function against kadmin daemons earlier than krb5-1.2. See Section 3.1 [Supported Encryption Types], page 11 and Section 3.2 [Salts], page 12 for available types.

If you want to just use the default values, all you need to do is:

kadmin: `addprinc jennifer`
WARNING: no policy specified for "jennifer@ATHENA.MIT.EDU";
defaulting to no policy.

Enter password for principal jennifer@ATHENA.MIT.EDU: ⇐ *Type the password.*
Re-enter password for principal jennifer@ATHENA.MIT.EDU: ⇐ *Type it again.*

Principal "jennifer@ATHENA.MIT.EDU" created.
kadmin:

If, on the other hand, you want to set up an account that expires on January 1, 2000, that uses a policy called "stduser", with a temporary password (which you want the user to change immediately), you would type the following. (Note: each line beginning with ⇒ is a continuation of the previous line.)

kadmin: `addprinc david -expire "1/1/2000 12:01am EST" -policy stduser`
⇒ `+needchange`

Enter password for principal david@ATHENA.MIT.EDU: ⇐ *Type the password.*
Re-enter password for principal
david@ATHENA.MIT.EDU: ⇐ *Type it again.*

Principal "david@ATHENA.MIT.EDU" created.
kadmin:

If you will need cross-realm authentication, you need to add principals for the other realm's TGT to each realm. For example, if you need to do cross-realm authentication between the realms ATHENA.MIT.EDU and EXAMPLE.COM, you would need to add the principals '`krbtgt/EXAMPLE.COM@ATHENA.MIT.EDU`' and '`krbtgt/ATHENA.MIT.EDU@EXAMPLE.COM`' to both databases. You need to be sure the passwords and the key version numbers (kvno) are the same in both databases. This may require explicitly setting the kvno with the '`-kvno`' option. See Section 5.6 [Cross-realm Authentication], page 52 for more details.

5.3.4 Deleting Principals

To delete a principal, use the kadmin `delete_principal` command, which requires the "delete" administrative privilege. The syntax is:

delete_principal [-force] *principal*

`delete_principal` has the alias `delprinc`. The `-force` option causes `delete_principal` not to ask if you're sure. For example:

kadmin: `delprinc jennifer`
Are you sure you want to delete the principal
"jennifer@ATHENA.MIT.EDU"? (yes/no): `yes`
Principal "jennifer@ATHENA.MIT.EDU" deleted.
Make sure that you have removed this principal from
all ACLs before reusing.
kadmin:

5.3.5 Changing Passwords

To change a principal's password use the kadmin `change_password` command, which requires the "modify" administrative privilege (unless the principal is changing his/her own password). The syntax is:

change_password [*options*] *principal*

The `change_password` option has the alias `cpw`. `change_password` takes the following options:

-randkey Sets the key of the principal to a random value.

-pw *password*
Sets the password to the string *password*. MIT does not recommend using this option.

-e "*enc:salt...*"
Uses the specified list of enctype-salttype pairs for setting the key of the principal. The quotes are necessary if there are multiple enctype-salttype pairs. This will not function against kadmin daemons earlier than krb5-1.2. See Section 3.1 [Supported Encryption Types], page 11 and Section 3.2 [Salts], page 12 for possible values.

-keepold Keeps the previous kvno's keys around. There is no easy way to delete the old keys, and this flag is usually not necessary except perhaps for TGS keys. Don't use this flag unless you know what you're doing.

For example:

kadmin: `cpw david`

Enter password for principal david@ATHENA.MIT.EDU: ⇐ *Type the new password.*
Re-enter password for principal david@ATHENA.MIT.EDU: ⇐ *Type it again.*

Password for david@ATHENA.MIT.EDU changed.
kadmin:

Note that `change_password` will not let you change the password to one that is in the principal's password history.

5.4 Policies

A policy is a set of rules governing passwords. Policies can dictate minimum and maximum password lifetimes, minimum number of characters and character classes a password must contain, and the number of old passwords kept in the database.

5.4.1 Retrieving Policies

To retrieve a policy, use the kadmin `get_policy` command, which requires the "inquire" administrative privilege. The syntax is:

get_policy **[-terse]** *policy*

The `get_policy` command has the alias `getpol`. For example:

```
kadmin: get_policy admin
Policy: admin
Maximum password life: 180 days 00:00:00
Minimum password life: 00:00:00
Minimum password length: 6
Minimum number of password character classes: 2
Number of old keys kept: 5
Reference count: 17
kadmin:
```

The *reference count* is the number of principals using that policy.

The `get_policy` command has a `-terse` option, which lists each field as a quoted, tab-separated string. For example:

```
kadmin: get_policy -terse admin
admin   15552000        0       6       2       5       17
kadmin:
```

5.4.2 Retrieving the List of Policies

You can retrieve the list of policies with the kadmin `list_policies` command, which requires the "list" privilege. The syntax is:

list_policies [*expression*]

where *expression* is a shell-style glob expression that can contain the characters *, ?, and []. All policy names matching the expression are displayed. The `list_policies` command has the aliases `listpols`, `get_policies`, and `getpols`. For example:

kadmin: `listpols`
test-pol
dict-only
once-a-min
test-pol-nopw

kadmin: `listpols t*`
test-pol
test-pol-nopw
kadmin:

5.4.3 Adding or Modifying Policies

To add a new policy, use the kadmin `add_policy` command, which requires the "add" administrative privilege. The syntax is:

add_policy [*options*] *policy_name*

To modify attributes of a principal, use the kadmin `modify_policy` command, which requires the "modify" administrative privilege. The syntax is:

modify_policy [*options*] *policy_name*

`add_policy` has the alias `addpol`. `modify_poilcy` has the alias `modpol`.

The `add_policy` and `modify_policy` commands take the following switches:

-maxlife *time*
Sets the maximum lifetime of a password to *time*.

-minlife *time*
Sets the minimum lifetime of a password to *time*.

-minlength *length*
Sets the minimum length of a password to *length* characters.

-minclasses *number*
Requires at least *number* of character classes in a password.

-history *number*
Sets the number of past keys kept for a principal to *number*.

5.4.4 Deleting Policies

To delete a policy, use the `kadmin delete_policy` command, which requires the "delete" administrative privilege. The syntax is:

delete_policy [**-force**] *policy_name*

The `delete_policy` command has the alias `delpol`. It prompts for confirmation before deletion. For example:

kadmin: `delete_policy guests`
Are you sure you want to delete the policy "guests"?
(yes/no): `yes`
kadmin:

Note that you must cancel the policy from all principals before deleting it. The `delete_policy` command will fail if it is in use by any principals.

5.5 Global Operations on the Kerberos Database

The `kdb5_util` command is the primary tool for administrating the Kerberos database. The syntax is:

kdb5_util *command* [*kdb5_util_options*] [*command_options*]

The `kdb5_util` command takes the following options, which override the defaults specified in the configuration files:

-r *realm* specifies the the Kerberos realm of the database.

-d *database_name*
specifies the name under which the principal database is stored.

-k *master_key_type*
specifies the key type of the master key in the database.

-M *master_key_name*
specifies the principal name of the master key in the database.

-m indicates that the master database password should be read from the TTY rather than fetched from a file on disk.

-sf *stash_file*
specifies the stash file of the master database password

-P *password*
specifies the master database password. MIT does not recommend using this option.

5.5.1 Dumping a Kerberos Database to a File

To dump a Kerberos database into a file, use the `kdb5_util dump` command on one of the KDCs. The syntax is:

kdb5_util dump [**-old**] [**-b6**] [**-b7**] [**-ov**]
[**-verbose**] [-mkey_convert] [-new_mkey_file] [*filename*
[*principals...*]]

The `kdb5_util dump` command takes the following options:

-old causes the dump to be in the Kerberos 5 Beta 5 and earlier dump format ("kdb5_edit load_dump version 2.0").

-b6 causes the dump to be in the Kerberos 5 Beta 6 format ("kdb5_edit load_dump version 3.0").

-b7 causes the dump to be in the Kerberos 5 Beta 7 format ("kdbt_edit load_dump version 4").

-ov causes the dump to be in ovsec_adm_export format. Currently, the only way to preserve per-principal policy information is to use this in conjunction with a normal dump.

-verbose causes the name of each principal and policy to be printed as it is dumped.

-mkey_convert
prompts for a new master password, and then dumps the database with all keys reencrypted in this new master key

-new_mkey_file
reads a new key from the default keytab and then dumps the database with all keys reencrypted in this new master key

For example:

```
shell% kdb5_util dump dumpfile
shell%

shell% kbd5_util dump -verbose dumpfile
kadmin/admin@ATHENA.MIT.EDU
krbtgt/ATHENA.MIT.EDU@ATHENA.MIT.EDU
kadmin/history@ATHENA.MIT.EDU
K/M@ATHENA.MIT.EDU
kadmin/changepw@ATHENA.MIT.EDU
shell%
```

If you specify which principals to dump, you must use the full principal, as in the following example. (The line beginning with ⇒ is a continuation of the previous line.):

```
shell% kdb5_util dump -verbose dumpfile K/M@ATHENA.MIT.EDU
⇒ kadmin/admin@ATHENA.MIT.EDU
kadmin/admin@ATHENA.MIT.EDU
K/M@ATHENA.MIT.EDU
shell%
```

Otherwise, the principals will not match those in the database and will not be dumped:

```
shell% kdb5_util dump -verbose dumpfile K/M kadmin/admin
shell%
```

If you do not specify a dump file, `kdb5_util` will dump the database to the standard output.

There is currently a bug where the default dump format omits the per-principal policy information. In order to dump all the data contained in the Kerberos database, you must perform a normal dump (with no option flags) and an additional dump using the "-ov" flag to a different file.

5.5.2 Restoring a Kerberos Database from a Dump File

To restore a Kerberos database dump from a file, use the `kdb5_util load` command on one of the KDCs. The syntax is:

> **kdb5_util load [-old] [-b6] [-b7] [-ov] [-verbose]**
> **[-update] [-hash]** *dumpfilename dbname* [*admin_dbname*]

The `kdb5_util load` command takes the following options:

-old requires the dump to be in the Kerberos 5 Beta 5 and earlier dump format ("kdb5_edit load_dump version 2.0").

-b6 requires the dump to be in the Kerberos 5 Beta 6 format ("kdb5_edit load_dump version 3.0").

-b7 requires the dump to be in the Kerberos 5 Beta 7 format ("kdb5_edit load_dump version 4").

-ov requires the dump to be in ovsec_adm_export format.

-verbose causes the name of each principal and policy to be printed as it is loaded.

-update causes records from the dump file to be updated in or added to the existing database. This is useful in conjunction with an ovsec_adm_export format dump if you want to preserve per-principal policy information, since the current default format does not contain this data.

-hash causes the database to be stored as a hash rather than a binary tree.

For example:

```
shell% kdb5_util load dumpfile principal
shell%

shell% kdb5_util load -update dumpfile principal
shell%
```

If the database file exists, and the **-update** flag was not given, `kdb5_util` will overwrite the existing database.

5.5.3 Creating a Stash File

A stash file allows a KDC to authenticate itself to the database utilities, such as `kadmin`, `kadmind`, `krb5kdc`, and `kdb5_util`.

To create a stash file, use the `kdb5_util stash` command. The syntax is:

kdb5_util stash [**-f** *keyfile*]

For example:

```
shell% kdb5_util stash
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key

Enter KDC database master key:  ⇐ Type the KDC database master password.


shell%
```

If you do not specify a stash file, `kdb5_util` will stash the key in the file specified in your `kdc.conf` file.

5.5.4 Creating and Destroying a Kerberos Database

If you need to create a new Kerberos database, use the `kdb5_util create` command. The syntax is:

kdb5_util create [**-s**]

If you specify the '`-s`' option, `kdb5_util` will stash a copy of the master key in a stash file. (See Section 5.5.3 [Creating a Stash File], page 51.) For example:

```
shell% /usr/local/sbin/kdb5_util -r ATHENA.MIT.EDU create -s
kdb5_util: No such file or directory while setting active database to
⇒ '/usr/local/var/krb5kdc/principal'
Initializing database '/usr/local/var/krb5kdc/principal' for
⇒ realm 'ATHENA.MIT.EDU',
master key name 'K/M@ATHENA.MIT.EDU'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.

Enter KDC database master key:  ⇐ Type the master password.
Re-enter KDC database master key to verify:  ⇐ Type it again.


shell%
```

If you need to destroy the current Kerberos database, use the `kdb5_util destroy` command. The syntax is:

kdb5_util destroy **[-f]**

The `destroy` command destroys the database, first overwriting the disk sectors and then unlinking the files. If you specify the '`-f`' option, `kdb5_util` will not prompt you for a confirmation before destroying the database.

shell% `/usr/local/sbin/kdb5_util -r ATHENA.MIT.EDU destroy`

kdb5_util: Deleting KDC database stored in /usr/local/var/krb5kdc/principal, are you sure (type yes to confirm)? ⇐*yes*

OK, deleting database '/usr/local/var/krb5kdc/principal'...

shell%

5.6 Cross-realm Authentication

In order for a KDC in one realm to authenticate Kerberos users in a different realm, it must share a key with the KDC in the other realm. In both databases, there must be krbtgt service principals for realms. These principals should all have the same passwords, key version numbers, and encryption types. For example, if the administrators of ATHENA.MIT.EDU and EXAMPLE.COM wanted to authenticate across the realms, they would run the following commands on the KDCs in *both* realms:

shell%: `kadmin.local -e "des3-hmac-sha1:normal des-cbc-crc:v4"`
kadmin: `add_princ -requires_preauth krbtgt/ATHENA.MIT.EDU@EXAMPLE.COM`
Enter password for principal krbtgt/ATHENA.MIT.EDU@EXAMPLE.COM:
Re-enter password for principal krbtgt/ATHENA.MIT.EDU@EXAMPLE.COM:
kadmin: `add_princ -requires_preauth krbtgt/EXAMPLE.COM@ATHENA.MIT.EDU`
Enter password for principal krbtgt/EXAMPLE.COM@ATHENA.MIT.EDU:
Enter password for principal krbtgt/EXAMPLE.COM@{No value for "PRIMARYREALML"}:
kadmin:

Even if most principals in a realm are generally created with the requires_preauth flag enabled, this flag is not desirable on cross-realm authentication keys because doing so makes it impossible to disable preauthentication on a service-by-service basis. Disabling it as in the example above is recommended.

It is also very important that these principals have good passwords. MIT recommends that TGT principal passwords be at least 26 characters of random ASCII text.

6 Application Servers

If you need to install the Kerberos V5 programs on an application server, please refer to the Kerberos V5 Installation Guide. Once you have installed the software, you need to add that host to the Kerberos database (see Section 5.3.3 [Adding or Modifying Principals], page 41), and generate a *keytab* for that host, that contains the host's key. You also need to make sure the host's clock is within your maximum clock skew of the KDCs.

6.1 Keytabs

A *keytab* is a host's copy of its own keylist, which is analogous to a user's password. An application server that needs to authenticate itself to the KDC has to have a keytab that contains its own principal and key. Just as it is important for users to protect their passwords, it is equally important for hosts to protect their keytabs. You should always store keytab files on local disk, and make them readable only by root, and you should never send a keytab file over a network in the clear. Ideally, you should run the `kadmin` command to extract a keytab on the host on which the keytab is to reside.

6.1.1 Adding Principals to Keytabs

To generate a keytab, or to add a principal to an existing keytab, use the `ktadd` command from `kadmin`, which requires the "inquire" administrative privilege. (If you use the **-glob** *princ_exp* option, it also requires the "list" administrative privilege.) The syntax is:

> **ktadd** [**-k**[**eytab**] *keytab*] [**-q**] [**-e**
> *key:salt_list*] [*principal* | **-glob** *princ_exp*]
> [...]

The `ktadd` command takes the following switches:

-k[**eytab**] *keytab*
: use *keytab* as the keytab file. Otherwise, `ktadd` will use the default keytab file (`/etc/krb5.keytab`).

-e "*enc:salt...*"
: Uses the specified list of enctype-salttype pairs for setting the key of the principal. The quotes are necessary if there are multiple enctype-salttype pairs. This will not function against kadmin daemons earlier than krb5-1.2. See Section 3.1 [Supported Encryption Types], page 11 and Section 3.2 [Salts], page 12 for all possible values.

-q
: run in quiet mode. This causes `ktadd` to display less verbose information.

principal | **-glob** *principal expression*
: add *principal*, or all principals matching *principal expression* to the keytab. The rules for *principal expression* are the same as for the kadmin `list_principals` (see Section 5.3.1.2 [Retrieving a List of Principals], page 38) command.

Here is a sample session, using configuration files that enable only '`des-cbc-crc`' encryption. (The line beginning with ⇒ is a continuation of the previous line.)

```
kadmin: ktadd host/daffodil.mit.edu@ATHENA.MIT.EDU
kadmin: Entry for principal host/daffodil.mit.edu@ATHENA.MIT.EDU with
    kvno 2, encryption type DES-CBC-CRC added to keytab
    WRFILE:/etc/krb5.keytab.
kadmin:
```

```
kadmin: ktadd -k /usr/local/var/krb5kdc/kadmind.keytab
⇒ kadmin/admin kadmin/changepw
kadmin: Entry for principal kadmin/admin@ATHENA.MIT.EDU with
    kvno 3, encryption type DES-CBC-CRC added to keytab
    WRFILE:/usr/local/var/krb5kdc/kadmind.keytab.
kadmin:
```

6.1.2 Removing Principals from Keytabs

To remove a principal from an existing keytab, use the kadmin `ktremove` command. The syntax is:

ktremove [**-k**[**eytab**] *keytab*] [**-q**] *principal* [*kvno* | **all** | **old**]

The `ktremove` command takes the following switches:

-k[**eytab**] *keytab*
: use *keytab* as the keytab file. Otherwise, `ktremove` will use the default keytab file (`/etc/krb5.keytab`).

-q
: run in quiet mode. This causes `ktremove` to display less verbose information.

principal
: the principal to remove from the keytab. (Required.)

kvno
: remove all entries for the specified principal whose Key Version Numbers match *kvno*.

all
: remove all entries for the specified principal

old
: remove all entries for the specified principal except those with the highest kvno.

For example:

```
kadmin: ktremove -k /usr/local/var/krb5kdc/kadmind.keytab kadmin/admin
kadmin: Entry for principal kadmin/admin with kvno 3 removed
    from keytab WRFILE:/usr/local/var/krb5kdc/kadmind.keytab.
kadmin:
```

6.2 Clock Skew

In order to prevent intruders from resetting their system clocks in order to continue to use expired tickets, Kerberos V5 is set up to reject ticket requests from any host whose clock is not within the specified maximum clock skew of the KDC (as specified in the `kdc.conf` file). Similarly, hosts are configured to reject responses from any KDC whose clock is not within the specified maximum clock skew of the host (as specified in the `krb5.conf` file). The default value for maximum clock skew is 300 seconds, or five minutes.

MIT suggests that you add a line to client machines' `/etc/rc` files to synchronize the machine's clock to your KDC at boot time. On UNIX hosts, assuming you had a kdc called `kerberos` in your realm, this would be:

```
gettime -s kerberos
```

If the host is not likely to be rebooted frequently, you may also want to set up a cron job that adjusts the time on a regular basis.

6.3 Getting DNS Information Correct

Several aspects of Kerberos rely on name service. In order for Kerberos to provide its high level of security, it is less forgiving of name service problems than some other parts of your network. It is important that your Domain Name System (DNS) entries and your hosts have the correct information.

Each host's canonical name must be the fully-qualified host name (including the domain), and each host's IP address must reverse-resolve to the canonical name.

Other than the `localhost` entry, make all entries in each machine's `/etc/hosts` file in the following form:

```
IP address      fully-qualified hostname        aliases
```

Here is a sample `/etc/hosts` file:

```
# this is a comment
127.0.0.1       localhost localhost@mit.edu
10.0.0.6       daffodil.mit.edu trillium wake-robin
```

Additionally, on Solaris machines, you need to be sure the "hosts" entry in the file `/etc/nsswitch.conf` includes the source "dns" as well as "file".

Finally, each host's keytab file must include a host/key pair for the host's canonical name. You can list the keys in a keytab file by issuing the command `klist -k`. For example:

```
viola# klist -k
Keytab name: /etc/krb5.keytab
KVNO Principal
---- ------------------------------------------------------------
   1 host/daffodil.mit.edu@ATHENA.MIT.EDU
```

If you telnet to the host with a fresh credentials cache (ticket file), and then `klist`, the host's service principal should be *host/fully-qualified-hostname*`@`*REALM_NAME*.

6.4 Configuring Your Firewall to Work With Kerberos V5

If you need off-site users to be able to get Kerberos tickets in your realm, they must be able to get to your KDC. This requires either that you have a slave KDC outside your firewall, or you configure your firewall to allow UDP requests into at least one of your KDCs, on whichever port the KDC is running. (The default is port 88; other ports may be specified in the KDC's kdc.conf file.) Similarly, if you need off-site users to be able to change their passwords in your realm, they must be able to get to your Kerberos admin server. The default port for the admin server is 749.

If your on-site users inside your firewall will need to get to KDCs in other realms, you will also need to configure your firewall to allow outgoing TCP and UDP requests to port 88. Additionally, if they will need to get to any Kerberos V4 KDCs, you may also need to allow TCP and UDP requests to port 750. If your on-site users inside your firewall will need to get to Kerberos admin servers in other realms, you will also need to allow outgoing TCP and UDP requests to port 749.

If any of your KDCs are outside your firewall, you will need to allow `kprop` requests to get through to the remote KDC. `Kprop` uses the krb5_prop service on port 754 (tcp).

If you need your off-site users to have access to machines inside your firewall, you need to allow TCP connections from their off-site hosts on the appropriate ports for the programs they will be using. The following lines from `/etc/services` show the default port numbers for the Kerberos V5 programs:

```
ftp           21/tcp           # Kerberos ftp and telnet use the
telnet        23/tcp           # default ports
kerberos      88/udp    kdc    # Kerberos V5 KDC
kerberos      88/tcp    kdc    # Kerberos V5 KDC
klogin        543/tcp          # Kerberos authenticated rlogin
kshell        544/tcp   cmd    # and remote shell
kerberos-adm  749/tcp          # Kerberos 5 admin/changepw
kerberos-adm  749/udp          # Kerberos 5 admin/changepw
krb5_prop     754/tcp          # Kerberos slave propagation
eklogin       2105/tcp         # Kerberos auth. & encrypted rlogin
krb524        4444/tcp         # Kerberos 5 to 4 ticket translator
```

By default, Kerberos V5 `telnet` and `ftp` use the same ports as the standard `telnet` and `ftp` programs, so if you already allow telnet and ftp connections through your firewall, the Kerberos V5 versions will get through as well. If you do not already allow telnet and ftp connections through your firewall, but need your users to be able to use Kerberos V5 telnet and ftp, you can either allow ftp and telnet connections on the standard ports, or switch these programs to non-default port numbers and allow ftp and telnet connections on those ports to get through.

Kerberos V5 `rlogin` uses the `klogin` service, which by default uses port 543. Encrypted Kerberos V5 rlogin uses the `eklogin` service, which by default uses port 2105.

Kerberos V5 `rsh` uses the `kshell` service, which by default uses port 544. However, the server must be able to make a TCP connection from the kshell port to an arbitrary port on the client, so if your users are to be able to use `rsh` from outside your firewall, the server they connect to must be able to send outgoing packets to arbitrary port numbers. Similarly, if your users need to run `rsh` from inside your firewall to hosts outside your firewall, the outside server needs to be able to connect to an arbitrary port on the machine inside your firewall. Because Kerberos V5 `rcp` uses `rsh`, the same issues apply. If you need to use `rsh` (or `rcp`) through your firewall and are concerned with the security implications of allowing connections to arbitrary ports, MIT suggests that you have rules that specifically name these applications and, if possible, list the allowed hosts.

The book *UNIX System Security*, by David Curry, is a good starting point for learning to configure firewalls.

7 Backups of Secure Hosts

When you back up a secure host, you should exclude the host's keytab file from the backup. If someone obtained a copy of the keytab from a backup, that person could make any host masquerade as the host whose keytab was compromised. This could be particularly dangerous if the compromised keytab was from one of your KDCs. If the machine has a disk crash and the keytab file is lost, it is easy to generate another keytab file. (See Section 6.1.1 [Adding Principals to Keytabs], page 53.) If you are unable to exclude particular files from backups, you should ensure that the backups are kept as secure as the host's root password.

7.1 Backing Up the Kerberos Database

As with any file, it is possible that your Kerberos database could become corrupted. If this happens on one of the slave KDCs, you might never notice, since the next automatic propagation of the database would install a fresh copy. However, if it happens to the master KDC, the corrupted database would be propagated to all of the slaves during the next propagation. For this reason, MIT recommends that you back up your Kerberos database regularly. Because the master KDC is continuously dumping the database to a file in order to propagate it to the slave KDCs, it is a simple matter to have a cron job periodically copy the dump file to a secure machine elsewhere on your network. (Of course, it is important to make the host where these backups are stored as secure as your KDCs, and to encrypt its transmission across your network.) Then if your database becomes corrupted, you can load the most recent dump onto the master KDC. (See Section 5.5.2 [Restoring a Kerberos Database from a Dump File], page 50.)

8 Bug Reporting

In any complex software, there will be bugs. If you have successfully built and installed Kerberos V5, please use the `krb5-send-pr` program to fill out a Problem Report should you encounter any errors in our software.

Bug reports that include proposed fixes are especially welcome. If you do include fixes, please send them using either context diffs or unified diffs (using '`diff -c`' or '`diff -u`', respectively). Please be careful when using "cut and paste" or other such means to copy a patch into a bug report; depending on the system being used, that can result in converting TAB characters into spaces, which makes applying the patches more difficult.

The `krb5-send-pr` program is installed in the directory `/usr/local/sbin`.

The `krb5-send-pr` program enters the problem report into our Problem Report Management System (PRMS), which automatically assigns it to the engineer best able to help you with problems in the assigned category.

The `krb5-send-pr` program will try to intelligently fill in as many fields as it can. You need to choose the *category*, *class*, *severity*, and *priority* of the problem, as well as giving us as much information as you can about its exact nature.

The PR **category** will be one of:

```
krb5-admin   krb5-appl    krb5-build   krb5-clients
krb5-doc     krb5-kdc     krb5-libs    krb5-misc
pty          telnet       test
```

Choose the category that best describes the area under which your problem falls.

The **class** can be *sw-bug*, *doc-bug*, *change-request*, or *support*. The first two are exactly as their names imply. Use *change-request* when the software is behaving according to specifications, but you want to request changes in some feature or behavior. The *support* class is intended for more general questions about building or using Kerberos V5.

The **severity** of the problem indicates the problem's impact on the usability of Kerberos V5. If a problem is *critical*, that means the product, component or concept is completely non-operational, or some essential functionality is missing, and no workaround is known. A *serious* problem is one in which the product, component or concept is not working properly or significant functionality is missing. Problems that would otherwise be considered *critical* are rated *serious* when a workaround is known. A *non-critical* problem is one that is indeed a problem, but one that is having a minimal effect on your ability to use Kerberos V5. *E.g.*, The product, component or concept is working in general, but lacks features, has irritating behavior, does something wrong, or doesn't match its documentation. The default severity is *serious*.

The **priority** indicates how urgent this particular problem is in relation to your work. Note that low priority does not imply low importance. A priority of *high* means a solution is needed as soon as possible. A priority of *medium* means the problem should be solved no later than the next release. A priority of *low* means the problem should be solved in a future release, but it is not important to your work how soon this happens. The default priority is *medium*.

Note that a given severity does not necessarily imply a given priority. For example, a non-critical problem might still have a high priority if you are faced with a hard deadline. Conversely, a serious problem might have a low priority if the feature it is disabling is one that you do not need.

It is important that you fill in the *release* field and tell us what changes you have made, if any.

A sample filled-out form from a company named “Toasters, Inc.” might look like this:

```
To: krb5-bugs@mit.edu
Subject: misspelled "Kerberos" in title of installation guide
From: jcb
Reply-To: jcb
Cc:
X-send-pr-version: 3.99


>Submitter-Id:  mit
>Originator:  Jeffrey C. Gilman Bigler
>Organization:
mit
>Confidential:  no
>Synopsis:  Misspelled "Kerberos" in title of installation guide
>Severity:  non-critical
>Priority:  low
>Category:  krb5-doc
>Class:  doc-bug
>Release:  1.0-development
>Environment:
<machine, os, target, libraries (multiple lines)>
System: ULTRIX imbrium 4.2 0 RISC
Machine: mips
>Description:
        Misspelled "Kerberos" in title of "Kerboros V5 Installation Guide"
>How-To-Repeat:
        N/A
>Fix:
        Correct the spelling.
```

If the `krb5-send-pr` program does not work for you, or if you did not get far enough in the process to have an installed and working `krb5-send-pr`, you can generate your own form, using the above as an example.

Appendix A Appendix

A.1 Kerberos Error Messages

A.1.1 Kerberos V5 Library Error Codes

This is the Kerberos v5 library error code table. Protocol error codes are ERROR_TABLE_BASE_krb5 + the protocol error code number; other error codes start at ERROR_TABLE_BASE_krb5 + 128.

0. KRB5KDC_ERR_NONE: No error
1. KRB5KDC_ERR_NAME_EXP: Client's entry in database has expired
2. KRB5KDC_ERR_SERVICE_EXP: Server's entry in database has expired
3. KRB5KDC_ERR_BAD_PVNO: Requested protocol version not supported
4. KRB5KDC_ERR_C_OLD_MAST_KVNO: Client's key is encrypted in an old master key
5. KRB5KDC_ERR_S_OLD_MAST_KVNO: Server's key is encrypted in an old master key
6. KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN: Client not found in Kerberos database
7. KRB5KDC_ERR_S_PRINCIPAL_UNKNOWN: Server not found in Kerberos database
8. KRB5KDC_ERR_PRINCIPAL_NOT_UNIQUE: Principal has multiple entries in Kerberos database
9. KRB5KDC_ERR_NULL_KEY: Client or server has a null key
10. KRB5KDC_ERR_CANNOT_POSTDATE: Ticket is ineligible for postdating
11. KRB5KDC_ERR_NEVER_VALID: Requested effective lifetime is negative or too short
12. KRB5KDC_ERR_POLICY: KDC policy rejects request
13. KRB5KDC_ERR_BADOPTION: KDC can't fulfill requested option
14. KRB5KDC_ERR_ETYPE_NOSUPP: KDC has no support for encryption type
15. KRB5KDC_ERR_SUMTYPE_NOSUPP: KDC has no support for checksum type
16. KRB5KDC_ERR_PADATA_TYPE_NOSUPP: KDC has no support for padata type
17. KRB5KDC_ERR_TRTYPE_NOSUPP: KDC has no support for transited type
18. KRB5KDC_ERR_CLIENT_REVOKED: Clients credentials have been revoked
19. KRB5KDC_ERR_SERVICE_REVOKED: Credentials for server have been revoked
20. KRB5KDC_ERR_TGT_REVOKED: TGT has been revoked
21. KRB5KDC_ERR_CLIENT_NOTYET: Client not yet valid - try again later
22. KRB5KDC_ERR_SERVICE_NOTYET: Server not yet valid - try again later
23. KRB5KDC_ERR_KEY_EXP: Password has expired

24. KRB5KDC_ERR_PREAUTH_FAILED: Preauthentication failed
25. KRB5KDC_ERR_PREAUTH_REQUIRED: Additional pre-authentication required
26. KRB5KDC_ERR_SERVER_NOMATCH: Requested server and ticket don't match
27. KRB5PLACEHOLD_27: KRB5 error code 27
28. KRB5PLACEHOLD_28: KRB5 error code 28
29. KRB5PLACEHOLD_29: KRB5 error code 29
30. KRB5PLACEHOLD_30: KRB5 error code 30
31. KRB5KRB_AP_ERR_BAD_INTEGRITY: Decrypt integrity check failed
32. KRB5KRB_AP_ERR_TKT_EXPIRED: Ticket expired
33. KRB5KRB_AP_ERR_TKT_NYV: Ticket not yet valid
34. KRB5KRB_AP_ERR_REPEAT: Request is a replay
35. KRB5KRB_AP_ERR_NOT_US: The ticket isn't for us
36. KRB5KRB_AP_ERR_BADMATCH: Ticket/authenticator don't match
37. KRB5KRB_AP_ERR_SKEW: Clock skew too great
38. KRB5KRB_AP_ERR_BADADDR: Incorrect net address
39. KRB5KRB_AP_ERR_BADVERSION: Protocol version mismatch
40. KRB5KRB_AP_ERR_MSG_TYPE: Invalid message type
41. KRB5KRB_AP_ERR_MODIFIED: Message stream modified
42. KRB5KRB_AP_ERR_BADORDER: Message out of order
43. KRB5KRB_AP_ERR_ILL_CR_TKT: Illegal cross-realm ticket
44. KRB5KRB_AP_ERR_BADKEYVER: Key version is not available
45. KRB5KRB_AP_ERR_NOKEY: Service key not available
46. KRB5KRB_AP_ERR_MUT_FAIL: Mutual authentication failed
47. KRB5KRB_AP_ERR_BADDIRECTION: Incorrect message direction
48. KRB5KRB_AP_ERR_METHOD: Alternative authentication method required
49. KRB5KRB_AP_ERR_BADSEQ: Incorrect sequence number in message
50. KRB5KRB_AP_ERR_INAPP_CKSUM: Inappropriate type of checksum in message
51. KRB5KRB_AP_PATH_NOT_ACCEPTED: Policy rejects transited path
52. KRB5KRB_ERR_RESPONSE_TOO_BIG: Response too big for UDP, retry with TCP
53. KRB5PLACEHOLD_53: KRB5 error code 53
54. KRB5PLACEHOLD_54: KRB5 error code 54
55. KRB5PLACEHOLD_55: KRB5 error code 55
56. KRB5PLACEHOLD_56: KRB5 error code 56
57. KRB5PLACEHOLD_57: KRB5 error code 57

58. KRB5PLACEHOLD_58: KRB5 error code 58
59. KRB5PLACEHOLD_59: KRB5 error code 59
60. KRB5KRB_ERR_GENERIC: Generic error (see e-text)
61. KRB5KRB_ERR_FIELD_TOOLONG: Field is too long for this implementation
62. KRB5PLACEHOLD_62: KRB5 error code 62
63. KRB5PLACEHOLD_63: KRB5 error code 63
64. KRB5PLACEHOLD_64: KRB5 error code 64
65. KRB5PLACEHOLD_65: KRB5 error code 65
66. KRB5PLACEHOLD_66: KRB5 error code 66
67. KRB5PLACEHOLD_67: KRB5 error code 67
68. KRB5PLACEHOLD_68: KRB5 error code 68
69. KRB5PLACEHOLD_69: KRB5 error code 69
70. KRB5PLACEHOLD_70: KRB5 error code 70
71. KRB5PLACEHOLD_71: KRB5 error code 71
72. KRB5PLACEHOLD_72: KRB5 error code 72
73. KRB5PLACEHOLD_73: KRB5 error code 73
74. KRB5PLACEHOLD_74: KRB5 error code 74
75. KRB5PLACEHOLD_75: KRB5 error code 75
76. KRB5PLACEHOLD_76: KRB5 error code 76
77. KRB5PLACEHOLD_77: KRB5 error code 77
78. KRB5PLACEHOLD_78: KRB5 error code 78
79. KRB5PLACEHOLD_79: KRB5 error code 79
80. KRB5PLACEHOLD_80: KRB5 error code 80
81. KRB5PLACEHOLD_81: KRB5 error code 81
82. KRB5PLACEHOLD_82: KRB5 error code 82
83. KRB5PLACEHOLD_83: KRB5 error code 83
84. KRB5PLACEHOLD_84: KRB5 error code 84
85. KRB5PLACEHOLD_85: KRB5 error code 85
86. KRB5PLACEHOLD_86: KRB5 error code 86
87. KRB5PLACEHOLD_87: KRB5 error code 87
88. KRB5PLACEHOLD_88: KRB5 error code 88
89. KRB5PLACEHOLD_89: KRB5 error code 89
90. KRB5PLACEHOLD_90: KRB5 error code 90
91. KRB5PLACEHOLD_91: KRB5 error code 91

92. KRB5PLACEHOLD_92: KRB5 error code 92
93. KRB5PLACEHOLD_93: KRB5 error code 93
94. KRB5PLACEHOLD_94: KRB5 error code 94
95. KRB5PLACEHOLD_95: KRB5 error code 95
96. KRB5PLACEHOLD_96: KRB5 error code 96
97. KRB5PLACEHOLD_97: KRB5 error code 97
98. KRB5PLACEHOLD_98: KRB5 error code 98
99. KRB5PLACEHOLD_99: KRB5 error code 99
100. KRB5PLACEHOLD_100: KRB5 error code 100
101. KRB5PLACEHOLD_101: KRB5 error code 101
102. KRB5PLACEHOLD_102: KRB5 error code 102
103. KRB5PLACEHOLD_103: KRB5 error code 103
104. KRB5PLACEHOLD_104: KRB5 error code 104
105. KRB5PLACEHOLD_105: KRB5 error code 105
106. KRB5PLACEHOLD_106: KRB5 error code 106
107. KRB5PLACEHOLD_107: KRB5 error code 107
108. KRB5PLACEHOLD_108: KRB5 error code 108
109. KRB5PLACEHOLD_109: KRB5 error code 109
110. KRB5PLACEHOLD_110: KRB5 error code 110
111. KRB5PLACEHOLD_111: KRB5 error code 111
112. KRB5PLACEHOLD_112: KRB5 error code 112
113. KRB5PLACEHOLD_113: KRB5 error code 113
114. KRB5PLACEHOLD_114: KRB5 error code 114
115. KRB5PLACEHOLD_115: KRB5 error code 115
116. KRB5PLACEHOLD_116: KRB5 error code 116
117. KRB5PLACEHOLD_117: KRB5 error code 117
118. KRB5PLACEHOLD_118: KRB5 error code 118
119. KRB5PLACEHOLD_119: KRB5 error code 119
120. KRB5PLACEHOLD_120: KRB5 error code 120
121. KRB5PLACEHOLD_121: KRB5 error code 121
122. KRB5PLACEHOLD_122: KRB5 error code 122
123. KRB5PLACEHOLD_123: KRB5 error code 123
124. KRB5PLACEHOLD_124: KRB5 error code 124
125. KRB5PLACEHOLD_125: KRB5 error code 125

126. KRB5PLACEHOLD_126: KRB5 error code 126
127. KRB5PLACEHOLD_127: KRB5 error code 127
128. KRB5_ERR_RCSID: (RCS Id string for the krb5 error table)
129. KRB5_LIBOS_BADLOCKFLAG: Invalid flag for file lock mode
130. KRB5_LIBOS_CANTREADPWD: Cannot read password
131. KRB5_LIBOS_BADPWDMATCH: Password mismatch
132. KRB5_LIBOS_PWDINTR: Password read interrupted
133. KRB5_PARSE_ILLCHAR: Illegal character in component name
134. KRB5_PARSE_MALFORMED: Malformed representation of principal
135. KRB5_CONFIG_CANTOPEN: Can't open/find Kerberos configuration file
136. KRB5_CONFIG_BADFORMAT: Improper format of Kerberos configuration file
137. KRB5_CONFIG_NOTENUFSPACE: Insufficient space to return complete information
138. KRB5_BADMSGTYPE: Invalid message type specified for encoding
139. KRB5_CC_BADNAME: Credential cache name malformed
140. KRB5_CC_UNKNOWN_TYPE: Unknown credential cache type
141. KRB5_CC_NOTFOUND: Matching credential not found
142. KRB5_CC_END: End of credential cache reached
143. KRB5_NO_TKT_SUPPLIED: Request did not supply a ticket
144. KRB5KRB_AP_WRONG_PRINC: Wrong principal in request
145. KRB5KRB_AP_ERR_TKT_INVALID: Ticket has invalid flag set
146. KRB5_PRINC_NOMATCH: Requested principal and ticket don't match
147. KRB5_KDCREP_MODIFIED: KDC reply did not match expectations
148. KRB5_KDCREP_SKEW: Clock skew too great in KDC reply
149. KRB5_IN_TKT_REALM_MISMATCH: Client/server realm mismatch in initial ticket request
150. KRB5_PROG_ETYPE_NOSUPP: Program lacks support for encryption type
151. KRB5_PROG_KEYTYPE_NOSUPP: Program lacks support for key type
152. KRB5_WRONG_ETYPE: Requested encryption type not used in message
153. KRB5_PROG_SUMTYPE_NOSUPP: Program lacks support for checksum type
154. KRB5_REALM_UNKNOWN: Cannot find KDC for requested realm
155. KRB5_SERVICE_UNKNOWN: Kerberos service unknown
156. KRB5_KDC_UNREACH: Cannot contact any KDC for requested realm
157. KRB5_NO_LOCALNAME: No local name found for principal name
158. KRB5_MUTUAL_FAILED: Mutual authentication failed
159. KRB5_RC_TYPE_EXISTS: Replay cache type is already registered

160. KRB5_RC_MALLOC: No more memory to allocate (in replay cache code)
161. KRB5_RC_TYPE_NOTFOUND: Replay cache type is unknown
162. KRB5_RC_UNKNOWN: Generic unknown RC error
163. KRB5_RC_REPLAY: Message is a replay
164. KRB5_RC_IO: Replay I/O operation failed XXX
165. KRB5_RC_NOIO: Replay cache type does not support non-volatile storage
166. KRB5_RC_PARSE: Replay cache name parse/format error
167. KRB5_RC_IO_EOF: End-of-file on replay cache I/O
168. KRB5_RC_IO_MALLOC: No more memory to allocate (in replay cache I/O code)
169. KRB5_RC_IO_PERM: Permission denied in replay cache code
170. KRB5_RC_IO_IO: I/O error in replay cache i/o code
171. KRB5_RC_IO_UNKNOWN: Generic unknown RC/IO error
172. KRB5_RC_IO_SPACE: Insufficient system space to store replay information
173. KRB5_TRANS_CANTOPEN: Can't open/find realm translation file
174. KRB5_TRANS_BADFORMAT: Improper format of realm translation file
175. KRB5_LNAME_CANTOPEN: Can't open/find lname translation database
176. KRB5_LNAME_NOTRANS: No translation available for requested principal
177. KRB5_LNAME_BADFORMAT: Improper format of translation database entry
178. KRB5_CRYPTO_INTERNAL: Cryptosystem internal error
179. KRB5_KT_BADNAME: Key table name malformed
180. KRB5_KT_UNKNOWN_TYPE: Unknown Key table type
181. KRB5_KT_NOTFOUND: Key table entry not found
182. KRB5_KT_END: End of key table reached
183. KRB5_KT_NOWRITE: Cannot write to specified key table
184. KRB5_KT_IOERR: Error writing to key table
185. KRB5_NO_TKT_IN_RLM: Cannot find ticket for requested realm
186. KRB5DES_BAD_KEYPAR: DES key has bad parity
187. KRB5DES_WEAK_KEY: DES key is a weak key
188. KRB5_BAD_ENCTYPE: Bad encryption type
189. KRB5_BAD_KEYSIZE: Key size is incompatible with encryption type
190. KRB5_BAD_MSIZE: Message size is incompatible with encryption type
191. KRB5_CC_TYPE_EXISTS: Credentials cache type is already registered.
192. KRB5_KT_TYPE_EXISTS: Key table type is already registered.
193. KRB5_CC_IO: Credentials cache I/O operation failed XXX

194. KRB5_FCC_PERM: Credentials cache file permissions incorrect
195. KRB5_FCC_NOFILE: No credentials cache found
196. KRB5_FCC_INTERNAL: Internal credentials cache error
197. KRB5_CC_WRITE: Error writing to credentials cache
198. KRB5_CC_NOMEM: No more memory to allocate (in credentials cache code)
199. KRB5_CC_FORMAT: Bad format in credentials cache
200. KRB5_INVALID_FLAGS: Invalid KDC option combination (library internal error) [for dual tgt library calls]
201. KRB5_NO_2ND_TKT: Request missing second ticket [for dual tgt library calls]
202. KRB5_NOCREDS_SUPPLIED: No credentials supplied to library routine
203. KRB5_SENDAUTH_BADAUTHVERS: Bad sendauth version was sent
204. KRB5_SENDAUTH_BADAPPLVERS: Bad application version was sent (via sendauth)
205. KRB5_SENDAUTH_BADRESPONSE: Bad response (during sendauth exchange)
206. KRB5_SENDAUTH_REJECTED: Server rejected authentication (during sendauth exchange)
207. KRB5_PREAUTH_BAD_TYPE: Unsupported preauthentication type
208. KRB5_PREAUTH_NO_KEY: Required preauthentication key not supplied
209. KRB5_PREAUTH_FAILED: Generic preauthentication failure
210. KRB5_RCACHE_BADVNO: Unsupported replay cache format version number
211. KRB5_CCACHE_BADVNO: Unsupported credentials cache format version number
212. KRB5_KEYTAB_BADVNO: Unsupported key table format version number
213. KRB5_PROG_ATYPE_NOSUPP: Program lacks support for address type
214. KRB5_RC_REQUIRED: Message replay detection requires rcache parameter
215. KRB5_ERR_BAD_HOSTNAME: Hostname cannot be canonicalized
216. KRB5_ERR_HOST_REALM_UNKNOWN: Cannot determine realm for host
217. KRB5_SNAME_UNSUPP_NAMETYPE: Conversion to service principal undefined for name type
218. KRB5KRB_AP_ERR_V4_REPLY: Initial Ticket response appears to be Version 4 error
219. KRB5_REALM_CANT_RESOLVE: Cannot resolve KDC for requested realm
220. KRB5_TKT_NOT_FORWARDABLE: Requesting ticket can't get forwardable tickets
221. KRB5_FWD_BAD_PRINCIPAL: Bad principal name while trying to forward credentials
222. KRB5_GET_IN_TKT_LOOP: Looping detected inside krb5_get_in_tkt
223. KRB5_CONFIG_NODEFREALM: Configuration file does not specify default realm
224. KRB5_SAM_UNSUPPORTED: Bad SAM flags in obtain_sam_padata
225. KRB5_KT_NAME_TOOLONG: Keytab name too long
226. KRB5_KT_KVNONOTFOUND: Key version number for principal in key table is incorrect

227. KRB5_APPL_EXPIRED: This application has expired
228. KRB5_LIB_EXPIRED: This Krb5 library has expired
229. KRB5_CHPW_PWDNULL: New password cannot be zero length
230. KRB5_CHPW_FAIL: Password change failed
231. KRB5_KT_FORMAT: Bad format in keytab
232. KRB5_NOPERM_ETYPE: Encryption type not permitted
233. KRB5_CONFIG_ETYPE_NOSUPP: No supported encryption types (config file error?)
234. KRB5_OBSOLETE_FN: Program called an obsolete, deleted function
235. KRB5_EAI_FAIL: unknown getaddrinfo failure
236. KRB5_EAI_NODATA: no data available for host/domain name
237. KRB5_EAI_NONAME: host/domain name not found
238. KRB5_EAI_SERVICE: service name unknown
239. KRB5_ERR_NUMERIC_REALM: Cannot determine realm for numeric host address

A.1.2 Kerberos V5 Database Library Error Codes

This is the Kerberos v5 database library error code table.

0. KRB5_KDB_RCSID: (RCS Id string for the kdb error table)
1. KRB5_KDB_INUSE: Entry already exists in database
2. KRB5_KDB_UK_SERROR: Database store error
3. KRB5_KDB_UK_RERROR: Database read error
4. KRB5_KDB_UNAUTH: Insufficient access to perform requested operation
5. KRB5_KDB_NOENTRY: No such entry in the database
6. KRB5_KDB_ILL_WILDCARD: Illegal use of wildcard
7. KRB5_KDB_DB_INUSE: Database is locked or in use–try again later
8. KRB5_KDB_DB_CHANGED: Database was modified during read
9. KRB5_KDB_TRUNCATED_RECORD: Database record is incomplete or corrupted
10. KRB5_KDB_RECURSIVELOCK: Attempt to lock database twice
11. KRB5_KDB_NOTLOCKED: Attempt to unlock database when not locked
12. KRB5_KDB_BADLOCKMODE: Invalid kdb lock mode
13. KRB5_KDB_DBNOTINITED: Database has not been initialized
14. KRB5_KDB_DBINITED: Database has already been initialized
15. KRB5_KDB_ILLDIRECTION: Bad direction for converting keys
16. KRB5_KDB_NOMASTERKEY: Cannot find master key record in database
17. KRB5_KDB_BADMASTERKEY: Master key does not match database

18. KRB5_KDB_INVALIDKEYSIZE: Key size in database is invalid
19. KRB5_KDB_CANTREAD_STORED: Cannot find/read stored master key
20. KRB5_KDB_BADSTORED_MKEY: Stored master key is corrupted
21. KRB5_KDB_CANTLOCK_DB: Insufficient access to lock database
22. KRB5_KDB_DB_CORRUPT: Database format error
23. KRB5_KDB_BAD_VERSION: Unsupported version in database entry
24. KRB5_KDB_BAD_SALTTYPE: Unsupported salt type
25. KRB5_KDB_BAD_ENCTYPE: Unsupported encryption type
26. KRB5_KDB_BAD_CREATEFLAGS: Bad database creation flags
27. KRB5_KDB_NO_PERMITTED_KEY: No matching key in entry having a permitted enc type
28. KRB5_KDB_NO_MATCHING_KEY: No matching key in entry

A.1.3 Kerberos V5 Magic Numbers Error Codes

This is the Kerberos v5 magic numbers error code table.

0. KV5M_NONE: Kerberos V5 magic number table
1. KV5M_PRINCIPAL: Bad magic number for krb5_principal structure
2. KV5M_DATA: Bad magic number for krb5_data structure
3. KV5M_KEYBLOCK: Bad magic number for krb5_keyblock structure
4. KV5M_CHECKSUM: Bad magic number for krb5_checksum structure
5. KV5M_ENCRYPT_BLOCK: Bad magic number for krb5_encrypt_block structure
6. KV5M_ENC_DATA: Bad magic number for krb5_enc_data structure
7. KV5M_CRYPTOSYSTEM_ENTRY: Bad magic number for krb5_cryptosystem_entry structure
8. KV5M_CS_TABLE_ENTRY: Bad magic number for krb5_cs_table_entry structure
9. KV5M_CHECKSUM_ENTRY: Bad magic number for krb5_checksum_entry structure
10. KV5M_AUTHDATA: Bad magic number for krb5_authdata structure
11. KV5M_TRANSITED: Bad magic number for krb5_transited structure
12. KV5M_ENC_TKT_PART: Bad magic number for krb5_enc_tkt_part structure
13. KV5M_TICKET: Bad magic number for krb5_ticket structure
14. KV5M_AUTHENTICATOR: Bad magic number for krb5_authenticator structure
15. KV5M_TKT_AUTHENT: Bad magic number for krb5_tkt_authent structure
16. KV5M_CREDS: Bad magic number for krb5_creds structure
17. KV5M_LAST_REQ_ENTRY: Bad magic number for krb5_last_req_entry structure
18. KV5M_PA_DATA: Bad magic number for krb5_pa_data structure
19. KV5M_KDC_REQ: Bad magic number for krb5_kdc_req structure

20. KV5M_ENC_KDC_REP_PART: Bad magic number for krb5_enc_kdc_rep_part structure
21. KV5M_KDC_REP: Bad magic number for krb5_kdc_rep structure
22. KV5M_ERROR: Bad magic number for krb5_error structure
23. KV5M_AP_REQ: Bad magic number for krb5_ap_req structure
24. KV5M_AP_REP: Bad magic number for krb5_ap_rep structure
25. KV5M_AP_REP_ENC_PART: Bad magic number for krb5_ap_rep_enc_part structure
26. KV5M_RESPONSE: Bad magic number for krb5_response structure
27. KV5M_SAFE: Bad magic number for krb5_safe structure
28. KV5M_PRIV: Bad magic number for krb5_priv structure
29. KV5M_PRIV_ENC_PART: Bad magic number for krb5_priv_enc_part structure
30. KV5M_CRED: Bad magic number for krb5_cred structure
31. KV5M_CRED_INFO: Bad magic number for krb5_cred_info structure
32. KV5M_CRED_ENC_PART: Bad magic number for krb5_cred_enc_part structure
33. KV5M_PWD_DATA: Bad magic number for krb5_pwd_data structure
34. KV5M_ADDRESS: Bad magic number for krb5_address structure
35. KV5M_KEYTAB_ENTRY: Bad magic number for krb5_keytab_entry structure
36. KV5M_CONTEXT: Bad magic number for krb5_context structure
37. KV5M_OS_CONTEXT: Bad magic number for krb5_os_context structure
38. KV5M_ALT_METHOD: Bad magic number for krb5_alt_method structure
39. KV5M_ETYPE_INFO_ENTRY: Bad magic number for krb5_etype_info_entry structure
40. KV5M_DB_CONTEXT: Bad magic number for krb5_db_context structure
41. KV5M_AUTH_CONTEXT: Bad magic number for krb5_auth_context structure
42. KV5M_KEYTAB: Bad magic number for krb5_keytab structure
43. KV5M_RCACHE: Bad magic number for krb5_rcache structure
44. KV5M_CCACHE: Bad magic number for krb5_ccache structure
45. KV5M_PREAUTH_OPS: Bad magic number for krb5_preauth_ops
46. KV5M_SAM_CHALLENGE: Bad magic number for krb5_sam_challenge
47. KV5M_SAM_KEY: Bad magic number for krb5_sam_key
48. KV5M_ENC_SAM_RESPONSE_ENC: Bad magic number for krb5_enc_sam_response_enc
49. KV5M_SAM_RESPONSE: Bad magic number for krb5_sam_response
50. KV5M_PREDICTED_SAM_RESPONSE: Bad magic number for krb5_predicted_sam_response

51. KV5M_PASSWD_PHRASE_ELEMENT: Bad magic number for passwd_phrase_element
52. KV5M_GSS_OID: Bad magic number for GSSAPI OID
53. KV5M_GSS_QUEUE: Bad magic number for GSSAPI QUEUE

A.1.4 ASN.1 Error Codes

0. ASN1_BAD_TIMEFORMAT: ASN.1 failed call to system time library
1. ASN1_MISSING_FIELD: ASN.1 structure is missing a required field
2. ASN1_MISPLACED_FIELD: ASN.1 unexpected field number
3. ASN1_TYPE_MISMATCH: ASN.1 type numbers are inconsistent
4. ASN1_OVERFLOW: ASN.1 value too large
5. ASN1_OVERRUN: ASN.1 encoding ended unexpectedly
6. ASN1_BAD_ID: ASN.1 identifier doesn't match expected value
7. ASN1_BAD_LENGTH: ASN.1 length doesn't match expected value
8. ASN1_BAD_FORMAT: ASN.1 badly-formatted encoding
9. ASN1_PARSE_ERROR: ASN.1 parse error
10. ASN1_BAD_GMTIME: ASN.1 bad return from gmtime
11. ASN1_MISMATCH_INDEF: ASN.1 non-constructed indefinite encoding
12. ASN1_MISSING_EOC: ASN.1 missing expected EOC

A.1.5 GSSAPI Error Codes

Generic GSSAPI Errors:

0. G_BAD_SERVICE_NAME: No in SERVICE-NAME name string
1. G_BAD_STRING_UID: STRING-UID-NAME contains nondigits
2. G_NOUSER: UID does not resolve to username
3. G_VALIDATE_FAILED: Validation error
4. G_BUFFER_ALLOC: Couldn't allocate gss_buffer_t data
5. G_BAD_MSG_CTX: Message context invalid
6. G_WRONG_SIZE: Buffer is the wrong size
7. G_BAD_USAGE: Credential usage type is unknown
8. G_UNKNOWN_QOP: Unknown quality of protection specified
9. G_BAD_HOSTNAME: Hostname in SERVICE-NAME string could not be canonicalized
10. G_WRONG_MECH: Mechanism is incorrect
11. G_BAD_TOK_HEADER: Token header is malformed or corrupt

12. G_BAD_DIRECTION: Packet was replayed in wrong direction
13. G_TOK_TRUNC: Token is missing data
14. G_REFLECT: Token was reflected
15. G_WRONG_TOKID: Received token ID does not match expected token ID

Kerberos 5 GSSAPI Errors:

0. KG_CCACHE_NOMATCH: Principal in credential cache does not match desired name
1. KG_KEYTAB_NOMATCH: No principal in keytab matches desired name
2. KG_TGT_MISSING: Credential cache has no TGT
3. KG_NO_SUBKEY: Authenticator has no subkey
4. KG_CONTEXT_ESTABLISHED: Context is already fully established
5. KG_BAD_SIGN_TYPE: Unknown signature type in token
6. KG_BAD_LENGTH: Invalid field length in token
7. KG_CTX_INCOMPLETE: Attempt to use incomplete security context
8. KG_CONTEXT: Bad magic number for krb5_gss_ctx_id_t
9. KG_CRED: Bad magic number for krb5_gss_cred_id_t
10. KG_ENC_DESC: Bad magic number for krb5_gss_enc_desc
11. KG_BAD_SEQ: Sequence number in token is corrupt
12. KG_EMPTY_CCACHE: Credential cache is empty
13. KG_NO_CTYPES: Acceptor and Initiator share no checksum types

A.2 kadmin Time Zones

This is a complete listing of the time zones recognized by the `kadmin` command.

gmt Greenwich Mean Time

ut, utc Universal Time (Coordinated).

wet Western European Time. (Same as GMT.)

bst British Summer Time. (1 hour ahead of GMT.)

wat West Africa Time. (1 hour behind GMT.)

at Azores Time. (2 hours behind GMT.)

bst Brazil Standard Time. (3 hours behind GMT.) Note that the abbreviation BST also stands for British Summer Time.

gst Greenland Standard Time. (3 hours behind GMT.) Note that the abbreviation GST also stands for Guam Standard Time.

nft Newfoundland Time. (3.5 hours behind GMT.)

nst Newfoundland Standard Time. (3.5 hours behind GMT.)

ndt Newfoundland Daylight Time. (2.5 hours behind GMT.)

ast	Atlantic Standard Time. (4 hours behind GMT.)
adt	Atlantic Daylight Time. (3 hours behind GMT.)
est	Eastern Standard Time. (5 hours behind GMT.)
edt	Eastern Daylight Time. (4 hours behind GMT.)
cst	Central Standard Time. (6 hours behind GMT.)
cdt	Central Daylight Time. (5 hours behind GMT.)
mst	Mountain Standard Time. (7 hours behind GMT.)
mdt	Mountain Daylight Time. (6 hours behind GMT.)
pst	Pacific Standard Time. (8 hours behind GMT.)
pdt	Pacific Daylight Time. (7 hours behind GMT.)
yst	Yukon Standard Time. (9 hours behind GMT.)
ydt	Yukon Daylight Time. (8 hours behind GMT.)
hst	Hawaii Standard Time. (10 hours behind GMT.)
hdt	Hawaii Daylight Time. (9 hours behind GMT.)
cat	Central Alaska Time. (10 hours behind GMT.)
ahst	Alaska-Hawaii Standard Time. (10 hours behind GMT.)
nt	Nome Time. (11 hours behind GMT.)
idlw	International Date Line West Time. (12 hours behind GMT.)
cet	Central European Time. (1 hour ahead of GMT.)
met	Middle European Time. (1 hour ahead of GMT.)
mewt	Middle European Winter Time. (1 hour ahead of GMT.)
mest	Middle European Summer Time. (2 hours ahead of GMT.)
swt	Swedish Winter Time. (1 hour ahead of GMT.)
sst	Swedish Summer Time. (1 hours ahead of GMT.)
fwt	French Winter Time. (1 hour ahead of GMT.)
fst	French Summer Time. (2 hours ahead of GMT.)
eet	Eastern Europe Time; Russia Zone 1. (2 hours ahead of GMT.)
bt	Baghdad Time; Russia Zone 2. (3 hours ahead of GMT.)
it	Iran Time. (3.5 hours ahead of GMT.)
zp4	Russia Zone 3. (4 hours ahead of GMT.)
zp5	Russia Zone 4. (5 hours ahead of GMT.)
ist	Indian Standard Time. (5.5 hours ahead of GMT.)
zp6	Russia Zone 5. (6 hours ahead of GMT.)
nst	North Sumatra Time. (6.5 hours ahead of GMT.) Note that the abbreviation NST is also used for Newfoundland Stanard Time.
sst	South Sumatra Time; Russia Zone 6. (7 hours ahead of GMT.) Note that SST is also Swedish Summer Time.
wast	West Australian Standard Time. (7 hours ahead of GMT.)
wadt	West Australian Daylight Time. (8 hours ahead of GMT.)
jt	Java Time. (7.5 hours ahead of GMT.)

cct China Coast Time; Russia Zone 7. (8 hours ahead of GMT.)
jst Japan Standard time; Russia Zone 8. (9 hours ahead of GMT.)
kst Korean Standard Time. (9 hours ahead of GMT.)
cast Central Australian Standard Time. (9.5 hours ahead of GMT.)
cadt Central Australian Daylight Time. (10.5 hours ahead of GMT.)
east Eastern Australian Standard Time. (10 hours ahead of GMT.)
eadt Eastern Australian Daylight Time. (11 hours ahead of GMT.)
gst Guam Standard Time; Russia Zone 9. (10 hours ahead of GMT.)
kdt Korean Daylight Time. (10 hours ahead of GMT.)
nzt New Zealand Time. (12 hours ahead of GMT.)
nzst New Zealand Standard Time. (12 hours ahead of GMT.)
nzdt New Zealand Daylight Time. (13 hours ahead of GMT.)
idle International Date Line East. (12 hours ahead of GMT.)

Table of Contents