## NAME

krb5_fileformats - File formats

## File formats

This section documents the diffrent file formats that are used in Heimdal and other Kerberos implementations.

### keytab

The keytab binary format is not a standard format. The format has evolved and may continue to. It is however understood by several Kerberos implementations including Heimdal, MIT, Sun’s Java ktab and are created by the ktpass.exe utility from Windows. So it has established itself as the defacto format for storing Kerberos keys.

The following C-like structure definitions illustrate the MIT keytab file format. All values are in network byte order. All text is ASCII.

```
keytab {
    uint16_t file_format_version;            # 0x502
    keytab_entry entries[*];
};

keytab_entry {
    int32_t size;
    uint16_t num_components;   # subtract 1 if version 0x501
    counted_octet_string realm;
    counted_octet_string components[num_components];
    uint32_t name_type;       # not present if version 0x501
    uint32_t timestamp;
    uint8_t vno8;
    keyblock key;
    uint32_t vno; #only present if >= 4 bytes left in entry
    uint32_t flags; #only present if >= 4 bytes left in entry
};

counted_octet_string {
    uint16_t length;
    uint8_t data[length];
};

keyblock {
    uint16_t type;
    counted_octet_string;
};
```

All numbers are stored in network byteorder (big endian) format.

The keytab file format begins with the 16 bit file_format_version which at the time this document was authored is 0x502. The format of older keytabs is described at the end of this document.

The file_format_version is immediately followed by an array of keytab_entry structures which are prefixed with a 32 bit size indicating the number of bytes that follow in the entry. Note that the size should be evaluated as signed. This is because a negative value indicates that the entry is in fact empty (e.g. it has been deleted) and that the negative value of that negative value (which is of course a positive value) is the offset to the next keytab_entry. Based on these size values alone the entire keytab file can be traversed.

The size is followed by a 16 bit num_components field indicating the number of counted_octet_string components in the components array.

The num_components field is followed by a counted_octet_string representing the realm of the principal.

A counted_octet_string is simply an array of bytes prefixed with a 16 bit length. For the realm and name components, the counted_octet_string bytes are ASCII encoded text with no zero terminator.

Following the realm is the components array that represents the name of the principal. The text of these components may be joined with slashs to construct the typical SPN representation. For example, the service principal HTTP/www.foo.net@FOO.NET would consist of name components ’HTTP’ followed by ’www.foo.net’.

Following the components array is the 32 bit name_type (e.g. 1 is KRB5_NT_PRINCIPAL, 2 is KRB5_NT_SRV_INST, 5 is KRB5_NT_UID, etc). In practice the name_type is almost certainly 1 meaning KRB5_NT_PRINCIPAL.

The 32 bit timestamp indicates the time the key was established for that principal. The value represents the number of seconds since Jan 1, 1970.

The 8 bit vno8 field is the version number of the key. This value is overridden by the 32 bit vno field if it is present. The vno8 field is filled with the lower 8 bits of the 32 bit protocol kvno field.

The keyblock structure consists of a 16 bit value indicating the encryption type and is a counted_octet_string containing the key. The encryption type is the same as the Kerberos standard (e.g. 3 is des-cbc-md5, 23 is arcfour-hmac-md5, etc).

The last field of the keytab_entry structure is optional. If the size of the keytab_entry indicates that there are at least 4 bytes remaining, a 32 bit value representing the key version number is present. This value supersedes the 8 bit vno8 value preceeding the keyblock.

Older keytabs with a file_format_version of 0x501 are different in three ways:

- All integers are in host byte order [1].

- The num_components field is 1 too large (i.e. after decoding, decrement by 1).

- The 32 bit name_type field is not present.

[1] The file_format_version field should really be treated as two separate 8 bit quantities representing the major and minor version number respectively.

## Heimdal database dump file

Format of the Heimdal text dump file as of Heimdal 0.6.3:

Each line in the dump file is one entry in the database.

Each field of a line is separated by one or more spaces, with the exception of fields consisting of principals containing spaces, where space can be quoted with \ and \ is quoted by \.

Fields and their types are:

Quoted princial (quote character is  [string]
Keys [keys]
Created by [event]
Modified by [event optional]
Valid start time [time optional]
Valid end time [time optional]
Password end valid time [time optional]
Max lifetime of ticket [time optional]
Max renew time of ticket [integer optional]
Flags [hdb flags]
Generation number [generation optional]

Extensions [extentions optional]

Fields following these silently are ignored.

All optional fields will be skipped if they fail to parse (or comprise the optional field marker of ’-’, w/o quotes).

Example:

```
fred@CODE.COM 27:1:16:e8b4c8fc7e60b9e641dcf4cff3f08a701d982a2f89ba373733d26ca59ba6c789666f6b8bfcf16
```

Encoding of types are as follows:

- keys

```
kvno:[masterkvno:keytype:keydata:salt]{zero or more separated by :}
```

kvno is the key version number.

keydata is hex-encoded

masterkvno is the kvno of the database master key. If this field is empty, the kadmin load and merge operations will encrypt the key data with the master key if there is one. Otherwise the key data will be imported asis.

salt is encoded as ’-’ (no/default salt) or

```
salt-type /
salt-type / ’string’
salt-type / hex-encoded-data
```

keytype is the protocol enctype number; see enum ENCTYPE in include/krb5_asn1.h for values.

Example:

```
27:1:16:e8b4c8fc7e60b9e641dcf4cff3f08a701d982a2f89ba373733d26ca59ba6c789666f6b8bfcf169412bb1e5dceb9b3


kvno=27,{key: masterkvno=1,keytype=des3-cbc-sha1,keydata=..., default salt}...
```

- time

Format of the time is: YYYYmmddHHMMSS, corresponding to strftime format ’%Y%m%d%k%M%S’.

Time is expressed in UTC.

Time can be optional (using -), when the time 0 is used.

Example:

```
 20041221112428
```

- event

```
        time:principal
```

time is as given in format time

principal is a string. Not quoting it may not work in earlier versions of Heimdal.

Example:

```
 20041221112428:bloggs@CODE.COM
```

- hdb flags

Integer encoding of HDB flags, see HDBFlags in lib/hdb/hdb.asn1. Each bit in the integer is the same as the bit in the specification.

- generation:

```
 time:usec:gen
```

usec is a the microsecond, integer. gen is generation number, integer.

The generation can be defaulted (using ’-’) or the empty string

- extensions:

```
 first-hex-encoded-HDB-Extension[:second-...]
```

HDB-extension is encoded the DER encoded HDB-Extension from lib/hdb/hdb.asn1. Consumers HDB extensions should be aware that unknown entires needs to be preserved even thought the ASN.1 data content might be unknown. There is a critical flag in the data to show to the KDC that the entry MUST be understod if the entry is to be used.