Kerberos Working Group
INTERNET-DRAFT
Category: Standards Track

Nicolas Williams
Sun Microsystems
Jonathan Trostle
Cisco Systems
Mike Swift
University of WA
John Brezak
Microsoft
Bill Gossman
Cisco Systems

Kerberos Set/Change Password: Version 2

<draft-ietf-krb-wg-kerberos-set-passwd-01.txt>

Status of this Memo

This draft expires on December 31st, 2001. Please send comments to the authors.

Copyright Notice

Abstract

This document specifies an extensible protocol for setting keys and changing the passwords of Kerberos V principals.

Table of Contents

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1 Introduction

Up to this point Kerberos V has lacked a single, standard protocol for changing passwords and keys. While several vendor-specific

---

protocols exist for changing Kerberos passwords/keys, none are properly internationalized and all are incomplete in one respect or another and none are sufficiently extensible to cope with new features that may be added to Kerberos V at some future time.

This document defines a protocol that is somewhat backward-compatible with the "kpasswd" protocol, version 1 [KPASSWDv1] and a derivative defined in [RFC3244] that uses more or less the same protocol framing.

This new protocol is designed to be extensible and properly internationalized.

2 The Protocol

The structure of the protocol is quite similar to that of typical RPC protocols. Each transaction consists of a data structure specific to an operation which is then wrapped in a data structure which is general to all operations of the protocol. These data structures are defined with the Abstract Syntax Notation 1 (ASN.1) [X680] and they are encoded using the Distinguished Encoding Rules (DER) [X690].

The protocol data is wrapped in a KRB-PRIV, or, in some cases, a KRB-ERROR, and framed in a header that is backwards compatible with version 1 [KPASSWDv1] of this protocol and [RFC3244].

[Discussion: Should the v1/rfc3244 framing and port numbers be dropped?]

2.1 Transports

The service SHOULD accept requests on UDP port 464 and TCP port 464. This is the same port used by version 1 [KPASSWDv1] of this protocol, but version 2 is a completely different protocol sharing with [KPASSWDv1] and [RFC3244] only the outer framing.

[Discussion: Should we remove UDP support? I think so [Nico].]

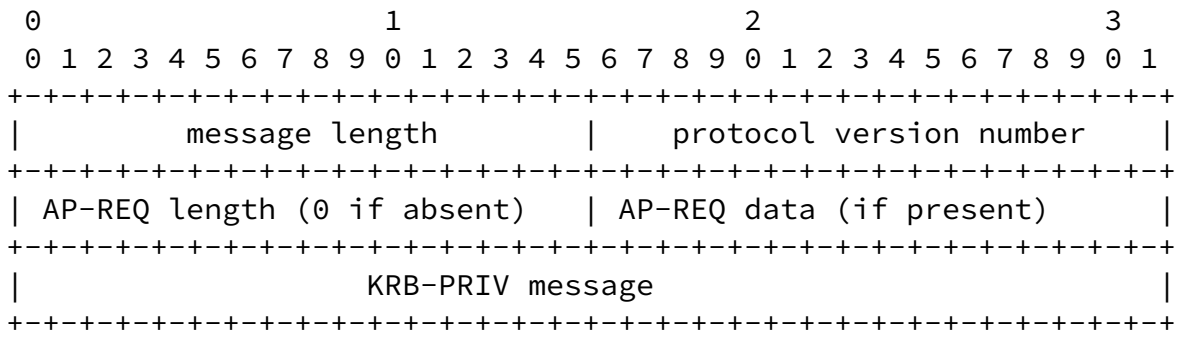
2.2 Protocol Framing

For compatibility with the original Kerberos password changing protocol developed at MIT as well as RFC3244, the first 4 bytes of the message consist of a 2-byte network byte order message length, followed by a 2 byte network byte order protocol version number, followed by a 2 byte network byte order length for an optional AP-REQ, AP-REP or KRB-ERROR, followed by the same, if present, followed by a KRB-PRIV (optional in TCP) containing the actual protocol message encoded in DER [X690].

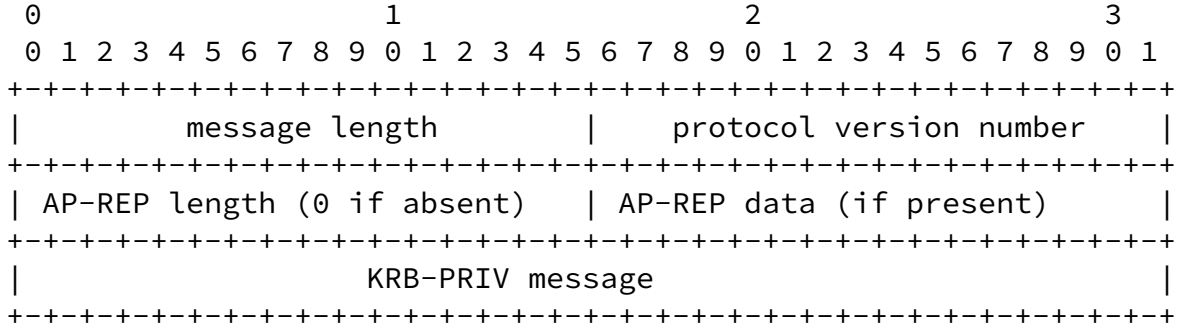
In the case of TCP there is an additional 4 byte network byte order length prepended to the frame described above as per-RFC3244.

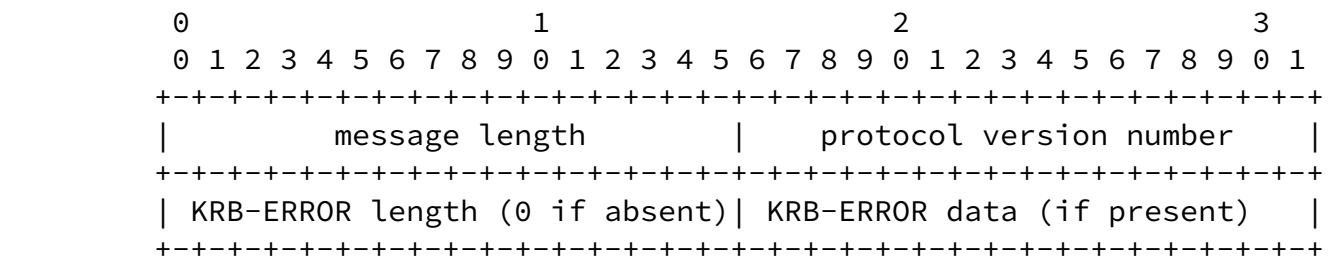
The protocol version number MUST be set to 2 for this protocol.

Bytes on the wire description of the framing:



The same framing applies equally to requests and responses, but responses use AP-REP and/or KRB-ERROR instead of AP-REQ:





For the UDP case the AP-REQ/AP-REP/KRB-ERROR MUST always be included.

Note that this framing is used by version 1 [KPASSWDv1] and version 0xff80 [RFC3244], though the latter does not use the framing when responding with KRB-ERROR messages.

Version 0xff80 servers may respond to requests with an un-framed KRB-ERROR and e-data set as per-RFC3244 [RFC3244], otherwise clients and server MUST always use this framing. See section 2.3.

2.2.1 The protocol over UDP

In the UDP case there is a single message from the client and a single response from the server with no state kept between requests, and each request MUST include a Kerberos AP-REQ and a KRB-PRIV and each response MUST carry an AP-REP, or KRB-ERROR and a KRB-PRIV.

Both the client and server MUST destroy the sub-session key, if any, resulting from the AP exchange after each operation.

UDP clients MUST not request the use of sequence numbers, otherwise they cannot generate the KRB-PRIV prior to receiving the AP-REP.

2.2.2 The protocol over TCP

The initial message from the client MUST carry an AP-REQ and the response to any request bearing an AP-REQ MUST carry an AP-REP.

Subsequent messages MAY involve Kerberos V AP exchanges, but generally the client SHOULD NOT initiate a new AP exchange except when it desires to authenticate as a different principal, when the ticket last used for authentication expires or when the server responds with an error indicating that the client must re-authenticate.

KRB-PRIV messages should use the session or sub-session key

established in the most recent AP exchange performed over the same TCP connection.

After each new AP exchange the client and server MUST destroy the sub-session key, if any, resulting from the previous AP exchange.

Servers MAY close open sessions at any time.

2.3 Protocol version negotiation

There are several major versions of this protocol. Version 2 also introduces a notion of protocol minor versions for use in negotiating protocol extensions. As of this time only one minor version is defined for major version 2: minor version 0, defined herein.

2.3.1 Protocol major version negotiation

Version 2 clients that also support other versions, such as [KPASSWDv1] or [RFC3244] SHOULD attempt to use version 2 of the protocol first and then MAY try other versions if the server responds with either a message framed as described in section 2.2 but with a protocol version number other than 2, or a KRB-ERROR with an error code of KRB5_KPASSWD_BAD_VERSION in the e-data field.

Note that some version 1 servers return a KRB-ERROR indicating that versions other than 1 of the change password protocol are not supported rather than an AP-REP and a KRB-PRIV containing the error data.

Also note that some [RFC3244] implementations do not return any responses to requests for protocol versions other than 0xff80, and in the TCP case close the TCP connection.

As a result change password protocol major version negotiation is subject to downgrade attacks. Therefore major version negotiation is NOT RECOMMENDED.

Version 2 servers SHOULD respond to non-v2 requests using whatever response is appropriate for the versions used by the clients, but if a server does not do this or know how to do this then it MUST respond with an error framed as in section 2.2, using an AP-REP and KRB-PRIV if the client's AP-REQ can be accepted, or a KRB-ERROR (framed) otherwise and using a ProtocolErrorCode value of unsupported-major-version or .

2.3.2 Protocol minor version negotiation

Version 2 clients are free to use whatever protocol minor version and message extensions are available to them in their initial messages to version 2 servers, provided that the minor versions (other than 0) have been defined through IETF documents and registered with the IANA.

Version 2 servers MUST answer with the highest protocol minor version number supported by the server and the client.

Version 2 clients MUST use the protocol minor version used in a server's reply for any subsequent messages in the same TCP session.

See section 2.7 for further description of the protocol's extensibility and its relation to protocol minor versions and the negotiation thereof.

2.4 Use of Kerberos V

This protocol makes use of messages defined in [RFC1510] and [clarifications]. Specifically, AP-REQ, AP-REP, KRB-ERROR and KRB-PRIV.

All operations are to be performed by the server on behalf of the client principal.

Clients SHOULD use "kadmin/setpw" as the principal name of the server for all requests except when changing the client principal's own password, when expired, for which they should use "kadmin/changepw".

The client MUST request mutual authentication and the client MUST NOT request the use of sequence numbers when using the protocol over UDP, but it MUST request the use of sequence numbers when running over TCP.

Clients SHOULD use INITIAL tickets for change password requests. Servers MAY force the use of INITIAL tickets for any request - see section 3.2.

2.5 Use of ASN.1

This protocol's messages are defined in ASN.1, using only features from [X680]. All ASN.1 types defined herein are to be encoded in

DER [X690]. A complete ASN.1 module is given in section 4. The ASN.1 tagging environment for this module is EXPLICIT.

The DER encoding of the ASN.1 PDUs are exchanged wrapped in a KRB-PRIV as described above and/or as e-data in KRB-ERROR messages.

2.6 Internationalization

This protocol's request PDU carries an optional field indicating the languages spoken by the client user; the client SHOULD send its list of spoken languages to the server (once per-TCP session). All strings in the protocol are UTF-8 strings.

The server SHOULD localize all strings intended for users to a language in common with the languages spoken by the client user.

For TCP sessions servers MUST cache the optional language tag lists from prior requests for use with subsequent requests that exclude the language tag list. Clients MAY expect such server behaviour and send the language tag lists only when they change or even just once per-TCP session. Clients SHOULD send the server the language tag list at least once.

Kerberos principal and realm names used in this protocol MUST be constrained as per the specification of the version of Kerberos V used by the client.

2.6.1 Normalization Forms for UTF-8 Strings

No normalization form is required for string types other than for PrincipalName and Realm, which two types are constrained by the specification of the version of Kerberos V used by the client, and the password fields in the change password operation, which MUST be normalized according to [k5stringprep].

2.6.2 Language Negotiation

The server MUST pick a language from the client's input list or the default language tag (see [RFC3066]) for text in its responses which is meant for the user to read.

The server SHOULD use a language selection algorithm such that consideration is first given to exact matches between the client's spoken languages and the server's available locales, followed by "fuzzy" matches where only the first sub-tags of the client's language tag list are used for matching against the servers available locales.

When the server has a message catalog for one of the client's spoken languages the server SHOULD localize any text strings intended for

users to read.

2.7 Protocol Extensibility

The protocol is defined in ASN.1 and uses extensibility markers throughout. As such, the module presented herein can be extended within the framework of [X680].

Typed holes are not used in this protocol as it is very simple and does not require the ability to deal with abstract data types defined in different layers. For this reason, the only way to extend this protocol is by extending the ASN.1 module within the framework of the IETF; all future extensions to this protocol have to be defined in IETF documents unless otherwise specified in a future IETF revision of this protocol.

A protocol minor version number is used to negotiate use of extensions. See section 2.3.2 for the minor version negotiation.

Message extensions are to be closely tied to protocol minor numbers.

Clients MAY use any protocol minor version that they support in initial requests, and MUST use the protocol minor version indicated in the server's initial reply in any subsequent requests in the same TCP session.

Servers SHOULD ignore additions to the ASN.1 types, in initial requests, where the syntax allows them that they do not understand, except for extensions to the "Op-req" type, which MUST result in an error; servers MAY respond with an error (ProtocolErrorCode value of unsupported-minor-version) to clients that use minor versions unsupported by the server in their initial requests.

Servers MUST select the highest minor version in common with their clients for use in replies.

2.8 Protocol Subsets

The structure of the protocol is such that the ASN.1 syntaxes for the various operations supported by the protocol are independent of the each other. Client and server mplementations MAY implement subsets of the overall protocol by removing some alternatives to the Op-req, Op-rep and Op-err CHOICEs from the ASN.1 module given in section 4.

For example, it should be possible to have a password-change only client that cannot set principal's keys - and vice versa.

3 Protocol Elements

The protocol as defined herein supports the following operations relating to the management of Kerberos principal's passwords or keys:

- change password
- set password (administrative)
- set new keys
- generate new keys

- get new, un-committed keys
- commit new keys
- get password policy name and/or description of principal
- list aliases of a principal
- list enctypes and version of Kerberos V supported by realm

These operations are needed to support Kerberos V interoperability between clients and KDCs of different implementation origins.

The operation for retrieving a list of aliases of a principal is needed where KDCs implement aliasing of principal names and allows clients to properly setup their "keytabs" when principal aliasing is in use.

Operations such as creation or deletion of principals are outside the scope of this document, and should be performed via other means, such as through directories or other Kerberos administration protocols.

The individual operations are described in section 3.2.

3.1 PDUs

The types "Request," "Response" and "Error-Response" are the ASN.1 module's PDUs.

The "Request" and "Response" PDUs are always to be sent wrapped in KRB-PRIV messages, except for the "Error-Response" PDU which MUST be sent as KRB-ERROR e-data (see section 2.4.1) when AP exchanges fail, otherwise it MUST be sent wrapped in a KRB-PRIV.

The ASN.1 syntax for the PDUs is given in section 4.

Note that the first field of each PDU is the major version of the protocol, defaulted to 2, meaning that it is never included in version 2 exchanges. Similarly, the second field of each PDU is the minor version, defaulted to 0.

The request, responses and error PDUs consist of an outer structure ("Request," "Response" and "Error-Response") containing fields common to all requests/responses, and an inner structure for fields that are specific to each operation's requests/responses. The inner structure is optional in the case of the Error-Response PDU and need not be included when generic errors occur for which there is a suitable ProtocolErrorCode.

Specifically, the outer Request structure has a field for passing a client user's spoken (read) languages to the server. It also has two optional fields for identifying the requested operation's target principal's name and realm (if not sent then the server MUST use the client principal name and realm from the AP exchange as the target).

The Response and Error PDUs' outer structures include a field indicating the language that the server has chosen for localization

of text intended to be displayed to users; this field is defaulted to "i-default". This language tag applies to all UTF8 strings in the inner structure (Op-rep and Op-err) that are meant to be displayed to users.

The protocol error codes are:

- proto-generic-error

  An operation-specific error ocurred, see the inner Op-error.

- proto-format-error
- proto-unsupported-major-version
- proto-unsupported-minor-version
- proto-unsupported-operation

- proto-wrong-service-principal

  Use kadmin/setpw for the server's principal name.

- proto-re-authentication-required

  The server demands that the client re-authenticate through a new

AP exchange.

- proto-initial-ticket-required

  Use of an INITIAL ticket is required for the requested operation.

- proto-client-and-target-realm-mismatch

  The server requires that the client's principal name and the target principal of the operation share the same realm name.

- proto-target-principal-unknown
- proto-authorization-failed

3.2 Operations

This section describes the semantics of each operation request and response defined in the ASN.1 module in section 4.

3.2.1 Null

NAME

null - Null or "ping" operation

DESCRIPTION

The null request is intended for use with TCP; its purpose is similar to RPC null procedures and is akin to a "ping" operation.

ERRORS

None.

3.2.2 Change Kerberos Password

NAME

change-pw - Change password operation

SYNOPSIS

Req-change-pw(old-pw, [new-pw], [etypes]) ->

```
Rep-change-pw([info-text], [new-pw], [etypes]) |
Err-change-pw([help-text], error code, [error info])
```

DESCRIPTION

Change a principal's password.

The change password request has one required field and three optional fields: "old-pw" (required), "new-pw" and "etypes", corresponding to the target principal's old password, new password and desired enctypes for the new long-term keys.

The server MUST validate the old password and MUST check the quality of the new password, if sent, according the password quality policy associated with the target principal.

The server SHOULD require that the old password be sent or that the client's ticket be INITIAL, or both, when the client principal and the target principal are the same.

A client MAY request that the server generate a new password by excluding the new password from its request, in which case the server MUST either generate a new password or respond with an error indicating that it does not support this feature.

Server-generated passwords MUST meet the target principal's password quality policy. It is RECOMMENDED that server-generated passwords be user-friendly, that is, memorable and that the target principal's preferred languages be taken into account by the password generation alogrithm used by the server.

RETURN

Upon successful password changes the server responds with a Rep-change-pw. The fields of Rep-change-pw are all optional and include:

- 'info-text' which the server can use to send a message to the user such as "Your new password will expire in 90 days," for

---

example.

- 'new-pw' which the server MUST include if the client requested that the server generate a new password; generated passwords MUST pass the target principal's password quality

policy.

- 'etypes' which the server MAY include to indicate which types of long-term keys it created for the target principal and which the server MUST include if the client specified a set of enctypes in its request.

ERRORS

The server may respond to change password requests with protocol or operation errors. See section 3.1 for a description of protocol error codes.

All operation errors include an optional 'help-text' field by which the server can describe the error in a human-readable, localizaed string.

Change password error codes include:

- generic-error

- old-pw-incorrect

- wont-generate-new-pw

  The server will not generate a new password for this principal or does not support password generation in general.

- new-pw-rejected-generic

  The client's proposed new password failed the target principal's password quality policy.

  The server MUST include a description of the password quality policy or aspect of it that the client's proposed new password failed to meet.

  The server MAY generate and send a new password that the client can then use as a new password and which is guaranteed to pass the target principal's current password quality policy.

- etype-not-supported

  The client requested an enctype that the KDC does not support.

3.2.3 Set Kerberos Password

NAME

set-pw - Set password operation

SYNOPSIS

```
Req-set-pw([languages], [new-pw], [etypes]) ->
   Rep-set-pw([info-text], [new-pw], [etypes]) |
   Err-set-pw([help-text], error code, [error info])
```

DESCRIPTION

Administratively set a principal's password.

The change password request has three optional fields: "languages", "new-pw" and "etypes", corresponding to the target principal's preferred languages, new password and desired enctypes for the new long-term keys.

The server MUST check the quality of the new password, if sent, according the password quality policy associated with the target principal.

The server SHOULD require that the client use the change-pw operation instead of set-pw when the client principal and the target principal are the same.

A client MAY request that the server generate a new password by excluding the new password from its request, in which case the server MUST either generate a new password or respond with an error indicating that it does not support this feature.

Server-generated passwords MUST meet the target principal's password quality policy. It is RECOMMENDED that server-generated passwords be user-friendly, that is, memorable and that the target principal's preferred languages be taken into account by the password generation alogrithm used by the server.

RETURN

Upon successful password changes the server responds with a Rep-change-pw. The fields of Rep-change-pw are all optional and include:

- 'info-text' which the server can use to send a message to the user such as "Your new password will expire in 90 days," for example.

- 'new-pw' which the server MUST include if the client requested that the server generate a new password; generated passwords MUST pass the target principal's password quality policy.

- 'etypes' which the server MAY include to indicate which types of long-term keys it created for the target principal and which the server MUST include if the client specified a set of enctypes in its request.

ERRORS

The server may respond to change password requests with protocol or operation errors. See section XYZ for a description of protocol error codes.

All operation errors include an optional 'help-text' field by which the server can describe the error in a human-readable, localizaed string.

Change password error codes include:

- generic-error

- use-change-pw

  The server demands that the client use the change-pw operation for the target principal of the set-pw request.

- wont-generate-new-pw

  The server will not generate a new password for this principal or does not support password generation in general.

- new-pw-rejected-generic

  The client's proposed new password failed the target principal's password quality policy.

  The server MUST include a description of the password quality policy or aspect of it that the client's proposed new password failed to meet.

The server MAY generate and send a new password that the client can then use as a new password and which is guaranteed to pass the target principal's current password quality policy.

- etype-not-supported

  The client requested an enctype that the KDC does not support.

3.2.4 Set Kerberos Keys

NAME

set-keys

SYNOPSIS

```
Req-set-keys(new-keys, commit?, [isupport]) ->
   Rep-set-keys([info-text], kvno, aliases, [isupport])
```

DESCRIPTION

The set-keys request consists of two required fields and one optional field: "new-keys", "commit" (a boolean field - see below) and "isupport", an optional field for indicating to the KDC what Kerberos V features are supported by the target principal.

When "commit" is true the KDC makes the new keys available for issueing tickets encrypted in them immediately. Otherwise the client MUST follow up with a commit-keys request to make the keys available.

If a principal has keys are awaiting commitment when a new set-keys request for that principal s made then the KDC MUST overwrite the deferred keys.

RETURN

For successful set-keys operations the server returns:

- Informational text, optional.

- The new kvno for the target principal.

- A list of aliases of the target principal known to the KDC (optional).

- The set of Kerberos V features supported by the KDC (optional).

ERRORS

The server may respond with the following errors:

- generic
- deferred-commit-no-support
- etype-no-support

3.2.5 Generate Kerberos Keys

NAME

gen-keys

SYNOPSIS

Req-gen-keys(etypes, [entropy], commit?, [isupport]) ->
Rep-set-keys([info-text], key, kvno, aliases, [isupport])

DESCRIPTION

The gen-keys is similar to the set-keys request (see section 3.2.4) but differs in that the server generates keys of client-requested enctypes, rather than the client providing specific keys.

The gen-keys request consists of two required fields and two optional fields: "etypes" (the enctypes of the new keys), "entropy", "commit" and "isupport" (see section 3.2.4).

If a principal has keys are awaiting commitment when a new set-keys request for that principal s made then the KDC MUST overwrite the deferred keys.

RETURN

For successful set-keys operations the server returns:

- Informational text, optional.

- The new kvno for the target principal.

- The new key (only one is needed).

- A list of aliases of the target principal known to the KDC (optional).

- The set of Kerberos V features supported by the KDC (optional).

ERRORS

The server may respond with the following errors:

- generic
- deferred-commit-no-support
- etype-no-support

3.2.6 Get New Keys

NAME

get-keys

SYNOPSIS

```
Req-get-keys(kvno) ->
   Rep-get-keys([info-text], keys, aliases, [isupport]) |
   Err-get-keys([help-text], error code, [error info])
```


DESCRIPTION

This request allows a client to get the keys set or generated in a previous set-keys or gen-keys request with deferred commitment..

RETURN

If the target principal and kvno correspond to uncommitted keys the server MUST respond with the actual keys that would be set by a subsequent commit-keys request. Otherwise the server MUST respond with an error (meaning that this operation cannot be used

to extract keys from the KDC that may be in use).

ERRORS

- generic
- kvno-committed
- no-such-kvno

3.2.7 Commit New Keys

NAME

commit-keys

SYNOPSIS

```
Req-commit-keys(kvno) ->
   Rep-commit-keys() |
   Err-commit-keys([help-text], error code, [error info])
```

DESCRIPTION

The commit-keys operation allows a client to bring a principal's new keys into use at the KDC.

Clients SHOULD make a commit-keys request corresponding to a deferred commitment set-keys/gen-keys operation as soon as the local key database for the target principal is updated.

The target principal name and the kvno MUST match those from a prior set-keys or gen-keys operation.

Servers MAY expire delayed key commitments at will. Servers SHOULD expire uncommitted new keys after a reasonable amount of time (600 seconds is RECOMMENDED).

Servers MUST respond to new set-keys requests for principals with pending, uncommitted new keys by expiring the uncommitted new keys and proceeding as if there had been no expired new keys.

ERRORS

- generic
- new-keys-conflict (A set-keys or gen-keys request succeeded

subsequent to the request that matches this {principal, kvno} tuple.)

3.2.8 Get Password Quality Policy

NAME

get-pw-policy

SYNOPSIS

```
Req-get-pw-policy() ->
   Rep-get-pw-policy([policy name], [policy description])
```

DESCRIPTION

Returns a description of the target principal's associated password quality policy, if any, as a list of localized UTF8String values.

Clients can use this operation in conjunction with the change-pw operation to obtain text that can be displayed to the user before the user actually enters a new password.

It is common for sites to set policies with respect to password quality. It is beyond the scope of this document to describe such policies. Management of password quality policies' actual content is also beyond the scope of this protocol.

ERRORS

No operation errors are defined.

3.2.9 Get Principal Aliases

NAME

get-print-aliases

SYNOPSIS

```
Req-get-princ-aliases() ->
   Rep-get-princ-aliases(aliases)
```

DESCRIPTION

Returns a list of aliases of the target principal.

ERRORS

No operation-specific errors.

3.2.10 Get Realm's Supported Kerberos V Version and Features

NAME

get-realm-krb5-support

SYNOPSIS

```
Req-get-realm-krb5-support() ->
   Rep-get-realm-krb5-support(isupport)
```

DESCRIPTION

Returns set of Kerberos V features support by the target principal's realm's KDCs.

ERRORS

No operation-specific errors.

3.3 Principal Aliases

Applications that use Kerberos often have to derive acceptor principal names from hostnames entered by users. Such hostnames may be aliases, they may be fully qualified, partially qualified or not qualified at all. Some implementations have resorted to deriving principal names from such hostnames by utilizing the names services to canonicalize the hostname first; such practices are not secure unless the name service are secure, which often aren't.

One method for securely deriving principal names from hostnames is to alias principals at the KDC such that the KDC will issue tickets for principal names which are aliases of others. It is helpful for principals to know what are their aliases as known by the KDCs.

Note that changing a principal's aliases is out of scope for this protocol.

3.4 Kerberos V Feature Negotiation

...

4 ASN.1 Module

```
DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- From [clarifications] with modifications
PrincipalName            ::= SEQUENCE {
     name-string [1] SEQUENCE OF UTF8String
}
```



```
Realm                    ::= UTF8String

-- NOTE WELL: Principal and realm names MUST be constrained by the
--            specification of the version of Kerberos V used by the
--            client.
--
-- [Perhaps PrincipalName should be a SEQUENCE of an optional name
--  type and a UTF8String, for simplicity.]

-- From [clarifications]
Int32            ::= INTEGER (-2147483648..2147483647)
UInt32           ::= INTEGER (0..4294967295)

-- Based on [clarifications]
Etype            ::= Int32   -- as in [clarifications]
Key      ::= SEQUENCE {
     enc-type        [0] Etype,      -- from Kerberos
     key             [1] OCTET STRING,
     ...
}

Language-Tag     ::= UTF8String -- Constrained by [RFC3066]

-- Empty, extensible SEQUENCEs are legal ASN.1
Extensible-NULL      ::= SEQUENCE {
     ...
}

-- Kerberos clients negotiate some parameters relating to their peers
-- indirectly through the KDC.  Today this is true of ticket session
-- key enctypes, but in the future this indirect negotiation may also
-- occur with respect to the minor version of Kerberos V to be used
-- between clients and servers.  Additionally, KDCs may need to know
-- what authorization-data types are supported by service principals,
-- both, for compatibility with legacy software and for optimization.
```

```
   --
   -- Thesefore it is important for KDCs to know what features of
   -- Kerberos V each service principal supports.
   --
   -- In version 2.0 of this protocol the clients and servers may notify
   -- each other of their support for:
   --
   --  - KerberosV minor version (rfc1510 -> 0, clarifications ->1)
   --  - enctypes
   --  - authorization data types
   --  - transited encoding data types
   --
   -- All authorization-data types defined in [clarifications] are
   -- assumed to be supported if the minor version is 1 and do not need
   -- to be included in the ad-type list.
   --
   -- Int32 is used for enctype and transited encoding data type
   -- identifiers.
```

```
   --
   -- An extensible CHOICE of Int32 is used for authorization data
   -- types.

   KerberosV-TR-ID              ::= Int32

   KerberosV-AD-ID              ::= CHOICE {
        ad-int          [0] Int32,
        ...
   }

   KerberosVSupportNego ::= SEQUENCE {
        version         [0] ENUM {
                rfc1510,
                clarifications, -- replace with assigned RFC number
                -- add extensions
                ...
        }
        enc-types       [1] SEQUENCE OF Etype,
        ad-types        [2] SEQUENCE OF KerberosV-AD-ID OPTIONAL,
                                -- authorization data types
        tr-enc-types    [3] SEQUENCE OF KerberosV-TR-ID OPTIONAL,
                                -- transited encoding types
        ...
        -- A future extended Kerberos V may have optional features, in
        -- which case there might be an additional choice here, not of
```

```
        -- NULL, but of some other, possibly structured, type.
    }

    Request              ::= [APPLICATION 0] SEQUENCE {
        pvno-minor       [0] INTEGER DEFAULT 0,
        languages        [1] SEQUENCE OF Language-Tag OPTIONAL,
                 -- Should be defaulted to the SEQUENCE of "i-default"
        targ-name        [2] PrincipalName OPTIONAL,
        targ-realm       [3] Realm OPTIONAL,
                 -- If targ-name/realm are missing then the request
                 -- applies to the principal of the client
        operation        [4] Op-req,
        ...
    }

    Response     ::= [APPLICATION 1] SEQUENCE {
        pvno-minor       [0] INTEGER DEFAULT 0,
        language         [1] Language-Tag DEFAULT "i-default",
        result           [2] Op-rep,
        ...
    }

    Error-Response       ::= [APPLICATION 2] SEQUENCE {
        pvno-minor       [0] INTEGER DEFAULT 0,
        language         [1] Language-Tag DEFAULT "i-default",
        error-code       [2] ProtocolErrorCode,
        help-text        [3] UTF8String OPTIONAL,
```



```
        op-error         [4] Op-err OPTIONAL,
        ...
    }

    Op-req               ::= CHOICE {
        null                     [0] Req-null,
        change-pw                [1] Req-change-pw,
        set-pw                   [2] Req-set-pw,
        set-keys                 [3] Req-set-keys,
        gen-keys                 [4] Req-gen-keys,
        get-keys                 [5] Req-get-keys,
        commit-keys              [6] Req-commit-keys,
        get-pw-policy            [7] Req-get-pw-policy,
        get-princ-aliases        [8] Req-get-princ-aliases,
        get-realm-krb5-support   [9] Req-get-realm-krb5-support,
        ...
    }
```

```
   Op-rep               ::= CHOICE {
        null                    [0] Rep-null,
        change-pw               [1] Rep-change-pw,
        set-pw                  [2] Rep-set-pw,
        set-keys                [3] Rep-set-keys,
        gen-keys                [4] Req-gen-keys,
        get-keys                [5] Req-get-keys,
        commit-keys             [6] Rep-commit-keys,
        get-pw-policy           [7] Rep-get-pw-policy,
        get-princ-aliases       [8] Rep-get-princ-aliases,
        get-realm-krb5-support  [9] Rep-get-realm-krb5-support,
        ...
   }

   Op-err       ::= CHOICE {
        null                    [0] Err-null,
        change-pw               [1] Err-change-pw,
        set-pw                  [2] Err-set-pw,
        set-keys                [3] Err-set-keys,
        gen-keys                [4] Err-gen-keys,
        get-keys                [5] Err-get-keys,
        commit-keys             [6] Err-commit-keys,
        get-pw-policy           [7] Err-get-pw-policy,
        get-princ-aliases       [8] Err-get-princ-aliases,
        get-realm-krb5-support  [9] Err-get-realm-krb5-support,
        ...
   }

   ProtocolErrorCode        ::= ENUM {
        proto-format-error,
        proto-unsupported-major-version,
        proto-unsupported-minor-version,
        proto-unsupported-operation,      -- Request CHOICE tag unknown
        proto-generic-see-op-error,       -- See Op-error
        proto-wrong-service-principal,    -- Use kadmin/setpw
```



```
        proto-re-authentication-required,
        proto-initial-ticket-required,
        proto-client-and-target-realm-mismatch,
        proto-target-principal-unknown,
        proto-authorization-failed,
        ...
   }
```

```
--
-- Requests and responses
--

-- NULL request, much like ONC RPC's NULL procedure - NOT extensible
Req-null    ::= NULL

Rep-null    ::= NULL

Err-null    ::= NULL

-- Change password
Req-change-pw        ::= SEQUENCE {
     old-pw          [0] UTF8String,
     new-pw          [1] UTF8String OPTIONAL,
     etypes          [2] SEQUENCE (1..) OF Etype OPTIONAL,
     ...
}

Rep-change-pw        ::= SEQUENCE {
     info-text       [0] UTF8String OPTIONAL,
     new-pw          [1] UTF8String OPTIONAL,
                             -- generated by the server if present
                             -- (and requested by the client)
     etypes          [2] SEQUENCE (1..) OF Etype OPTIONAL,
     ...
}

Err-change-pw        ::= SEQUENCE {
     help-text               [0] UTF8String OPTIONAL,
     error                   [1] CHOICE {
             op-generic-error                [0] Extensible-NULL,
             op-old-pw-incorrect             [1] Extensible-NULL,
             op-wont-generate-new-pw         [2] Extensible-NULL,
             op-new-pw-rejected-generic      [3] SEQUENCE {
                     policy              [1] SEQUENCE OF UTF8String,
                     suggested-pw        [2] UTF8String OPTIONAL,
                     ...
             }
             op-etype-not-supported          [4] SEQUENCE {
                     supported-etypes    [1] SEQUENCE OF Etype,
                     ...
             },
             ...
     },
```

```
     ...
}

-- Set password
Req-set-pw   ::= SEQUENCE {
     languages       [0] SEQUENCE OF Language-Tag OPTIONAL,
     new-pw          [1] UTF8String OPTIONAL,
     etypes          [2] SEQUENCE (1..) OF Etype OPTIONAL,
     ...
}

Rep-set-pw   ::= SEQUENCE {
     info-text       [0] UTF8String OPTIONAL,
     new-pw          [1] UTF8String OPTIONAL,
                         -- generated by the server if present
                         -- (and requested by the client)
     etypes          [2] SEQUENCE (1..) OF Etype OPTIONAL,
     ...
}

Err-set-pw   ::= SEQUENCE {
     help-text               [0] UTF8String OPTIONAL,
     error                   [1] CHOICE {
             op-generic-error                [0] Extensible-NULL,
             op-use-change-pw                [1] Extensible-NULL,
             op-wont-generate-new-pw         [2] Extensible-NULL,
             op-new-pw-rejected-generic      [3] SEQUENCE {
                     policy              [1] SEQUENCE OF UTF8String,
                     suggested-pw        [2] UTF8String OPTIONAL,
                     ...
             }
             op-etype-not-supported          [4] SEQUENCE {
                     supported-etypes    [1] SEQUENCE OF Etype,
                     ...
             },
             ...
     },
     ...
}

-- Set keys
Req-set-keys ::= SEQUENCE {
     keys            [0] SEQUENCE OF Key,
     commit          [1] BOOLEAN,
                         -- TRUE  -> install keys now
                         --
                         -- FALSE -> require set-keys-commit
                         --          before issuing tickets
                         --          encrypted with these keys.
                         --
```

```
                                -- See commit-keys op
        isupport        [2] KerberosVSupportNego OPTIONAL,
                                -- For future Kerberos V extensions KDCs
```



```
                                -- may need to know what krb5 version is
                                -- supported by individual service
                                -- principals.  This field provides a
                                -- way to tell the KDC what version of
                                -- Kerberos V the target principal
                                -- supports.
        ...
    }

    Rep-set-keys ::= SEQUENCE {
        info-text       [0] UTF8String OPTIONAL,
        kvno            [1] UInt32,
        aliases         [2] SEQUENCE OF SEQUENCE {
                name        [0] PrincipalName,
                realm       [1] Realm OPTIONAL,
                ...
        },
        isupport        [3] KerberosVSupportNego OPTIONAL,
        ...
        -- Should there be ETYPE-INFO2 stuff here?
    }

    Err-set-keys ::= SEQUENCE {
        help-text       [0] UTF8String OPTIONAL, -- Reason for rejection
        error           [1] CHOICE {
                op-generic                      [0] Extensible-NULL,
                op-deferred-commit-no-support   [1] Extensible-NULL,
                op-etype-no-support             [2] SEQUENCE OF {
                        supported-etypes        [1] SEQUENCE OF Etype,
                        ...
                }
                ...
        }
    }

    -- Generate keys
    Req-gen-keys ::= SEQUENCE {
        etypes          [0] SEQUENCE (1..) OF Etype,
        entropy         [1] OCTET STRING OPTIONAL,
        commit          [2] BOOLEAN,
                                -- TRUE  -> install keys now
```

```
                                --
                                -- FALSE -> require set-keys-commit
                                --           before issuing tickets
                                --           encrypted with these keys.
                                --
                                -- See commit-keys op
        isupport        [3] KerberosVSupportNego OPTIONAL,
                                -- For future Kerberos V extensions KDCs
                                -- may need to know what krb5 version is
                                -- supported by individual service
                                -- principals.  This field provides a
                                -- way to tell the KDC what version of
```



```
                                -- Kerberos V the target principal
                                -- supports.
        ...
   }

   Rep-gen-keys ::= SEQUENCE {
        info-text       [0] UTF8String OPTIONAL,
        kvno            [1] UInt32,
        key             [2] Key,
        aliases         [3] SEQUENCE OF SEQUENCE {
                name        [0] PrincipalName,
                realm       [1] Realm OPTIONAL,
                ...
        },
        isupport        [4] KerberosVSupportNego OPTIONAL,
        ...
        -- Should there be ETYPE-INFO2 stuff here?
   }

   Err-gen-keys ::= Err-set-keys

   -- Get un-committed key request
   Req-get-keys ::= SEQUENCE {
        kvno            [0] UInt32,
        ...
   }

   Rep-get-keys ::= SEQUENCE {
        info-text       [0] UTF8String OPTIONAL,
        keys            [1] SEQUENCE OF Key,
        aliases         [2] SEQUENCE OF SEQUENCE {
                name        [0] PrincipalName,
```

```
                realm       [1] Realm OPTIONAL,
                ...
        },
        isupport        [3] KerberosVSupportNego OPTIONAL,
        ...
        -- Should there be ETYPE-INFO2 stuff here?
    }

    Err-get-keys ::= SEQUENCE {
        help-text       [0] UTF8String OPTIONAL, -- Reason for rejection
        error           [1] CHOICE {
                op-generic              [0] Extensible-NULL,
                op-kvno-committed       [1] Extensible-NULL,
                op-no-such-kvno         [1] Extensible-NULL,
                ...
        }
    }

    -- Commit a set keys request
    Req-commit-keys ::= SEQUENCE {
        kvno            [0] UInt32,
```



```
        ...
    }

    Rep-commit-keys ::= Extensible-NULL

    Err-commit-keys ::= SEQUENCE {
        help-text       [0] UTF8String OPTIONAL, -- Reason for rejection
        error           [1] CHOICE {
                op-generic                  [0] Extensible-NULL,
                op-kvno-expired             [1] Extensible-NULL,
                    -- The client took too long to respond.
                op-kvno-unknown             [2] Extensible-NULL,
                    -- The targ princ/kvno is invalid or unknown to the
                    -- server (perhaps it lost track of state)
                op-new-keys-conflict        [3] Extensible-NULL,
                    -- A new-keys/commit-keys request subsequent to the
                    -- new-keys that produced the kvno has completed
                    -- and incremented the principal's kvno
                ...
        }
        ...
    }
```

```
   -- Get password policy
   Req-get-pw-policy   ::= Extensible-NULL

   Rep-get-pw-policy   ::= SEQUENCE {
        policy-name     [0] UTF8String OPTIONAL,
        description     [1] SEQUENCE OF UTF8String OPTIONAL,
        ...
   }

   Err-get-pw-policy   ::= Extensible-NULL

   -- Get principal aliases
   Req-get-princ-aliases        ::= Extensible-NULL

   Rep-get-princ-aliases        ::= SEQUENCE {
        help-text       [0] UTF8String OPTIONAL,
        aliases         [1] SEQUENCE OF SEQUENCE {
                name        [0] PrincipalName,
                realm       [1] Realm OPTIONAL,
                ...
        },
        ...
   }

   Err-get-princ-aliases        ::= Extensible-NULL

   -- Get list of enctypes supported by KDC for new keys
   Req-get-realm-krb5-support   ::=     Extensible-NULL

   Rep-get-realm-krb5-support   ::= SEQUENCE {
```



```
        isupport        [0] KerberosVSupportNego,
                                -- Version of Kerberos V supported by
                                -- the target realm.
        ...
   }

   Err-get-realm-krb5-support   ::= Extensible-NULL

   END
```

6 IANA Considerations

None.

7 Security Considerations

Implementors and site administrators should note that the redundancy of UTF-8 encodings varies by Unicode codepoint used. Password quality policies should, therefore, take this into account when estimating the amount of redundancy and entropy in a proposed new password. [?? It's late at night - I think this is correct.]

Kerberos set/change password/key protocol major version negotiation cannot be done securely. A downgrade attack is possible against clients that attempt to negotiate the protocol major version to use with a server. It is not clear at this time that the attacker would gain much from such a downgrade attack other than denial of service (DoS) by forcing the client to use a protocol version which does not support some feature needed by the client (Kerberos V in general is subject to a variety of DoS attacks anyways [RFC1510]).

This protocol does not have Perfect Forward Security (PFS). As a result, any passive network snooper watching password/key changing operations who has stolen a principal's password or long-term keys can find out what the new ones are.

[More text needed]

8 Acknowledgements

The authors would like to thank Raeburn, Tom Yu, Martin Rex, Sam Hartman, Tony Andrea, Paul W. Nelson, Marcus Watts, Love, Joel N. Weber II, Jeffrey Hutzelman and other participants from the IETF Kerberos Working Group for their assistance.

9 References

9.1 Normative References

[RFC2026]
S. Bradner, RFC2026: "The Internet Standard Process - Revision 3," October 1996, Obsoletes - RFC 1602, Status: Best Current

---

Practice.

[RFC2119]
S. Bradner, RFC2119 (BCP14): "Key words for use in RFCs to Indicate Requirement Levels," March 1997, Status: Best Current

Practice.

[X680]
Abstract Syntax Notation One (ASN.1): Specification of Basic Notation, ITU-T Recommendation X.680 (1997) | ISO/IEC International Standard 8824-1:1998.
http://www.itu.int/ITU-T/studygroups/com17/languages/X680_0702.pdf

[X690]
ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), ITU-T Recommendation X.690 (1997)| ISO/IEC International Standard 8825-1:1998.
http://www.itu.int/ITU-T/studygroups/com17/languages/X690_0702.pdf

[clarifications]
RFC-Editor: To be replaced by RFC number for draft-ietf-krb-wg-kerberos-clarifications.

[k5stringprep]
RFC-Editor: To be replaced by RFC number for draft-ietf-krb-wg-utf8-profile.

[RFC3066]
H. Alvestrand, RFC3066 (BCP47): "Tags for the Identification of Languages," January 2001, Obsoletes RFC1766, Status: Best Current Practice.

[KPASSWDv1]
(Reference needed!)

9.2 Informative References

[RFC3244]
M. Swift, J. Trostle, J. Brezak, RFC3244: "Microsoft Windows 2000 Kerberos Change Password and Set Password Protocols," February 2002, Status: Informational.

10 Authors' Addresses

Nicolas Williams
Sun Microsystems
5300 Riata Trace Ct
Austin, TX 78727
Email: nicolas.williams@sun.com

Jonathan Trostle
Cisco Systems

170 W. Tasman Dr.
San Jose, CA 95134
Email: jtrostle@cisco.com

Mike Swift
University of Washington
Seattle, WA
Email: mikesw@cs.washington.edu

John Brezak
Microsoft
1 Microsoft Way
Redmond, WA 98052
Email: jbrezak@microsoft.com

Bill Gossman
Cisco Systems
500 108th Ave. NE, Suite 500
Bellevue, WA 98004
Email: bgossman@cisco.com

11 Notes to the RFC Editor

This document has two KRB WG drafts as normative references and cannot progress until those drafts progress, but no other draft depends on this one.

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.