Kerberos

KERBEROS V5 ASN.1 Codec

Kerberos v5 grammar

Here is the ASN.1 grammar for SPNEGO V5 as define din RFC 4220 :

```
KerberosV5Spec2 {
        iso(1) identified-organization(3) dod(6) internet(1)
        security(5) kerberosV5(2) modules(4) krb5spec2(2)
} DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- OID arc for KerberosV5
--
-- This OID may be used to identify Kerberos protocol messages
-- encapsulated in other protocols.
--
-- This OID also designates the OID arc for KerberosV5-related OIDs.
--
-- NOTE: RFC 1510 had an incorrect value (5) for "dod" in its OID.
id-krb5         OBJECT IDENTIFIER ::= {
        iso(1) identified-organization(3) dod(6) internet(1)
        security(5) kerberosV5(2)
}

Int32           ::= INTEGER (-2147483648..2147483647)
                    -- signed values representable in 32 bits

UInt32          ::= INTEGER (0..4294967295)
                    -- unsigned 32 bit values

Microseconds    ::= INTEGER (0..999999)
                    -- microseconds

KerberosString  ::= GeneralString (IA5String)

Realm           ::= KerberosString

PrincipalName   ::= SEQUENCE {
        name-type       [0] Int32,
        name-string     [1] SEQUENCE OF KerberosString
}

KerberosTime    ::= GeneralizedTime -- with no fractional seconds

HostAddress     ::= SEQUENCE  {
        addr-type       [0] Int32,
        address         [1] OCTET STRING
}

-- NOTE: HostAddresses is always used as an OPTIONAL field and
-- should not be empty.
HostAddresses   -- NOTE: subtly different from rfc1510,
                -- but has a value mapping and encodes the same
        ::= SEQUENCE OF HostAddress

-- NOTE: AuthorizationData is always used as an OPTIONAL field and
-- should not be empty.
AuthorizationData       ::= SEQUENCE OF SEQUENCE {
        ad-type         [0] Int32,
        ad-data         [1] OCTET STRING
}
```

```
PA-DATA         ::= SEQUENCE {
        -- NOTE: first tag is [1], not [0]
        padata-type     [1] Int32,
        padata-value    [2] OCTET STRING -- might be encoded AP-REQ
}

KerberosFlags   ::= BIT STRING (SIZE (32..MAX))
                    -- minimum number of bits shall be sent,
                    -- but no fewer than 32

EncryptedData   ::= SEQUENCE {
        etype   [0] Int32 -- EncryptionType --,
        kvno    [1] UInt32 OPTIONAL,
        cipher  [2] OCTET STRING -- ciphertext
}

EncryptionKey   ::= SEQUENCE {
        keytype         [0] Int32 -- actually encryption type --,
        keyvalue        [1] OCTET STRING
}

Checksum        ::= SEQUENCE {
        cksumtype       [0] Int32,
        checksum        [1] OCTET STRING
}

Ticket          ::= [APPLICATION 1] SEQUENCE {
        tkt-vno         [0] INTEGER (5),
        realm           [1] Realm,
        sname           [2] PrincipalName,
        enc-part        [3] EncryptedData -- EncTicketPart
}

-- Encrypted part of ticket
EncTicketPart   ::= [APPLICATION 3] SEQUENCE {
        flags                   [0] TicketFlags,
        key                     [1] EncryptionKey,
        crealm                  [2] Realm,
        cname                   [3] PrincipalName,
        transited               [4] TransitedEncoding,
        authtime                [5] KerberosTime,
        starttime               [6] KerberosTime OPTIONAL,
        endtime                 [7] KerberosTime,
        renew-till              [8] KerberosTime OPTIONAL,
        caddr                   [9] HostAddresses OPTIONAL,
        authorization-data      [10] AuthorizationData OPTIONAL
}

-- encoded Transited field
TransitedEncoding       ::= SEQUENCE {
        tr-type         [0] Int32 -- must be registered --,
        contents        [1] OCTET STRING
}

TicketFlags     ::= KerberosFlags
        -- reserved(0),
        -- forwardable(1),
        -- forwarded(2),
        -- proxiable(3),
        -- proxy(4),
        -- may-postdate(5),
        -- postdated(6),
        -- invalid(7),
        -- renewable(8),
        -- initial(9),
        -- pre-authent(10),
        -- hw-authent(11),
-- the following are new since 1510
        -- transited-policy-checked(12),
        -- ok-as-delegate(13)
```

```
AS-REQ          ::= [APPLICATION 10] KDC-REQ

TGS-REQ         ::= [APPLICATION 12] KDC-REQ

KDC-REQ         ::= SEQUENCE {
        -- NOTE: first tag is [1], not [0]
        pvno            [1] INTEGER (5) ,
        msg-type        [2] INTEGER (10 -- AS -- | 12 -- TGS --),
        padata          [3] SEQUENCE OF PA-DATA OPTIONAL
                            -- NOTE: not empty --,
        req-body        [4] KDC-REQ-BODY
}

KDC-REQ-BODY    ::= SEQUENCE {
        kdc-options             [0] KDCOptions,
        cname                   [1] PrincipalName OPTIONAL
                                    -- Used only in AS-REQ --,
        realm                   [2] Realm
                                    -- Server's realm
                                    -- Also client's in AS-REQ --,
        sname                   [3] PrincipalName OPTIONAL,
        from                    [4] KerberosTime OPTIONAL,
        till                    [5] KerberosTime,
        rtime                   [6] KerberosTime OPTIONAL,
        nonce                   [7] UInt32,
        etype                   [8] SEQUENCE OF Int32 -- EncryptionType
                                    -- in preference order --,
        addresses               [9] HostAddresses OPTIONAL,
        enc-authorization-data  [10] EncryptedData OPTIONAL
                                    -- AuthorizationData --,
        additional-tickets      [11] SEQUENCE OF Ticket OPTIONAL
                                        -- NOTE: not empty
}

KDCOptions      ::= KerberosFlags
        -- reserved(0),
        -- forwardable(1),
        -- forwarded(2),
        -- proxiable(3),
        -- proxy(4),
        -- allow-postdate(5),
        -- postdated(6),
        -- unused7(7),
        -- renewable(8),
        -- unused9(9),
        -- unused10(10),
        -- opt-hardware-auth(11),
        -- unused12(12),
        -- unused13(13),
-- 15 is reserved for canonicalize
        -- unused15(15),
-- 26 was unused in 1510
        -- disable-transited-check(26),
--
        -- renewable-ok(27),
        -- enc-tkt-in-skey(28),
        -- renew(30),
        -- validate(31)

AS-REP          ::= [APPLICATION 11] KDC-REP

TGS-REP         ::= [APPLICATION 13] KDC-REP

KDC-REP         ::= SEQUENCE {
        pvno            [0] INTEGER (5),
        msg-type        [1] INTEGER (11 -- AS -- | 13 -- TGS --),
        padata          [2] SEQUENCE OF PA-DATA OPTIONAL
                                -- NOTE: not empty --,
        crealm          [3] Realm,
        cname           [4] PrincipalName,
```

```
        ticket          [5] Ticket,
        enc-part        [6] EncryptedData
                                -- EncASRepPart or EncTGSRepPart,
                                -- as appropriate
}

EncASRepPart    ::= [APPLICATION 25] EncKDCRepPart

EncTGSRepPart   ::= [APPLICATION 26] EncKDCRepPart

EncKDCRepPart   ::= SEQUENCE {
        key             [0] EncryptionKey,
        last-req        [1] LastReq,
        nonce           [2] UInt32,
        key-expiration  [3] KerberosTime OPTIONAL,
        flags           [4] TicketFlags,
        authtime        [5] KerberosTime,
        starttime       [6] KerberosTime OPTIONAL,
        endtime         [7] KerberosTime,
        renew-till      [8] KerberosTime OPTIONAL,
        srealm          [9] Realm,
        sname           [10] PrincipalName,
        caddr           [11] HostAddresses OPTIONAL
}

LastReq         ::=     SEQUENCE OF SEQUENCE {
        lr-type         [0] Int32,
        lr-value        [1] KerberosTime
}

AP-REQ          ::= [APPLICATION 14] SEQUENCE {
        pvno            [0] INTEGER (5),
        msg-type        [1] INTEGER (14),
        ap-options      [2] APOptions,
        ticket          [3] Ticket,
        authenticator   [4] EncryptedData -- Authenticator
}

APOptions       ::= KerberosFlags
        -- reserved(0),
        -- use-session-key(1),
        -- mutual-required(2)

-- Unencrypted authenticator
Authenticator   ::= [APPLICATION 2] SEQUENCE  {
        authenticator-vno       [0] INTEGER (5),
        crealm                  [1] Realm,
        cname                   [2] PrincipalName,
        cksum                   [3] Checksum OPTIONAL,
        cusec                   [4] Microseconds,
        ctime                   [5] KerberosTime,
        subkey                  [6] EncryptionKey OPTIONAL,
        seq-number              [7] UInt32 OPTIONAL,
        authorization-data      [8] AuthorizationData OPTIONAL
}

AP-REP          ::= [APPLICATION 15] SEQUENCE {
        pvno            [0] INTEGER (5),
        msg-type        [1] INTEGER (15),
        enc-part        [2] EncryptedData -- EncAPRepPart
}

EncAPRepPart    ::= [APPLICATION 27] SEQUENCE {
        ctime           [0] KerberosTime,
        cusec           [1] Microseconds,
        subkey          [2] EncryptionKey OPTIONAL,
        seq-number      [3] UInt32 OPTIONAL
}

KRB-SAFE        ::= [APPLICATION 20] SEQUENCE {
        pvno            [0] INTEGER (5),
```

```
        msg-type        [1] INTEGER (20),
        safe-body       [2] KRB-SAFE-BODY,
        cksum           [3] Checksum
}

KRB-SAFE-BODY   ::= SEQUENCE {
        user-data       [0] OCTET STRING,
        timestamp       [1] KerberosTime OPTIONAL,
        usec            [2] Microseconds OPTIONAL,
        seq-number      [3] UInt32 OPTIONAL,
        s-address       [4] HostAddress,
        r-address       [5] HostAddress OPTIONAL
}

KRB-PRIV        ::= [APPLICATION 21] SEQUENCE {
        pvno            [0] INTEGER (5),
        msg-type        [1] INTEGER (21),
                        -- NOTE: there is no [2] tag
        enc-part        [3] EncryptedData -- EncKrbPrivPart
}

EncKrbPrivPart  ::= [APPLICATION 28] SEQUENCE {
        user-data       [0] OCTET STRING,
        timestamp       [1] KerberosTime OPTIONAL,
        usec            [2] Microseconds OPTIONAL,
        seq-number      [3] UInt32 OPTIONAL,
        s-address       [4] HostAddress -- sender's addr --,
        r-address       [5] HostAddress OPTIONAL -- recip's addr
}

KRB-CRED        ::= [APPLICATION 22] SEQUENCE {
        pvno            [0] INTEGER (5),
        msg-type        [1] INTEGER (22),
        tickets         [2] SEQUENCE OF Ticket,
        enc-part        [3] EncryptedData -- EncKrbCredPart
}

EncKrbCredPart  ::= [APPLICATION 29] SEQUENCE {
        ticket-info     [0] SEQUENCE OF KrbCredInfo,
        nonce           [1] UInt32 OPTIONAL,
        timestamp       [2] KerberosTime OPTIONAL,
        usec            [3] Microseconds OPTIONAL,
        s-address       [4] HostAddress OPTIONAL,
        r-address       [5] HostAddress OPTIONAL
}

KrbCredInfo     ::= SEQUENCE {
        key             [0] EncryptionKey,
        prealm          [1] Realm OPTIONAL,
        pname           [2] PrincipalName OPTIONAL,
        flags           [3] TicketFlags OPTIONAL,
        authtime        [4] KerberosTime OPTIONAL,
        starttime       [5] KerberosTime OPTIONAL,
        endtime         [6] KerberosTime OPTIONAL,
        renew-till      [7] KerberosTime OPTIONAL,
        srealm          [8] Realm OPTIONAL,
        sname           [9] PrincipalName OPTIONAL,
        caddr           [10] HostAddresses OPTIONAL
}

KRB-ERROR       ::= [APPLICATION 30] SEQUENCE {
        pvno            [0] INTEGER (5),
        msg-type        [1] INTEGER (30),
        ctime           [2] KerberosTime OPTIONAL,
        cusec           [3] Microseconds OPTIONAL,
        stime           [4] KerberosTime,
        susec           [5] Microseconds,
        error-code      [6] Int32,
        crealm          [7] Realm OPTIONAL,
        cname           [8] PrincipalName OPTIONAL,
        realm           [9] Realm -- service realm --,
```

```
        sname           [10] PrincipalName -- service name --,
        e-text          [11] KerberosString OPTIONAL,
        e-data          [12] OCTET STRING OPTIONAL
}

METHOD-DATA     ::= SEQUENCE OF PA-DATA

TYPED-DATA      ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
        data-type       [0] Int32,
        data-value      [1] OCTET STRING OPTIONAL
}

-- preauth stuff follows

PA-ENC-TIMESTAMP        ::= EncryptedData -- PA-ENC-TS-ENC

PA-ENC-TS-ENC           ::= SEQUENCE {
        patimestamp     [0] KerberosTime -- client's time --,
        pausec          [1] Microseconds OPTIONAL
}

ETYPE-INFO-ENTRY        ::= SEQUENCE {
        etype           [0] Int32,
        salt            [1] OCTET STRING OPTIONAL
}

ETYPE-INFO              ::= SEQUENCE OF ETYPE-INFO-ENTRY

ETYPE-INFO2-ENTRY       ::= SEQUENCE {
        etype           [0] Int32,
        salt            [1] KerberosString OPTIONAL,
        s2kparams       [2] OCTET STRING OPTIONAL
}

ETYPE-INFO2             ::= SEQUENCE SIZE (1..MAX) OF ETYPE-INFO2-ENTRY

AD-IF-RELEVANT          ::= AuthorizationData

AD-KDCIssued            ::= SEQUENCE {
        ad-checksum     [0] Checksum,
        i-realm         [1] Realm OPTIONAL,
        i-sname         [2] PrincipalName OPTIONAL,
        elements        [3] AuthorizationData
}

AD-AND-OR               ::= SEQUENCE {
        condition-count [0] Int32,
        elements        [1] AuthorizationData
}

AD-MANDATORY-FOR-KDC    ::= AuthorizationData

END
```

Decoding Kerberos messages

The Kerberos protocol define many standalone messages. We will define the state machine for each of those messages.

AS-REQ

This is a message sent to the Authentication Service by a client.

It's a KDC-REQ message with a specific type. Here is its grammar :

```
AS-REQ          ::= [APPLICATION 10] KDC-REQ

KDC-REQ         ::= SEQUENCE {
        -- NOTE: first tag is [1], not [0]
        pvno            [1] INTEGER (5) ,
        msg-type        [2] INTEGER (10 -- AS -- | 12 -- TGS --), -- Here, 10
        padata          [3] SEQUENCE OF PA-DATA OPTIONAL
                            -- NOTE: not empty --,
        req-body        [4] KDC-REQ-BODY
}
```

A typical PDU for this message will be something like :

```
0x6A L1               // APPLICATION 10
  0x30 L2             // SEQUENCE
    0xA1 0x03
      0x02 0x01 NN    // pvno
    0xA2 0x03
      0x02 0x01 0x0A  // msg-type, 10
   [0xA3 L3 abcd]     // SEQUENCE OF PA-DATA OPTIONAL
    0xA4 L4 abcd      // KDC-REQ-BODY
```

If the PA-DATA is not absent, then t can be empty.

TGS-REQ

This is a message sent to the Ticket Granting Service by a client.

It's a KDC-REQ message with a specific type. Here is its grammar :

```
TGS-REQ          ::= [APPLICATION 12] KDC-REQ

KDC-REQ         ::= SEQUENCE {
        -- NOTE: first tag is [1], not [0]
        pvno            [1] INTEGER (5) ,
        msg-type        [2] INTEGER (10 -- AS -- | 12 -- TGS --), -- Here, 12
        padata          [3] SEQUENCE OF PA-DATA OPTIONAL
                            -- NOTE: not empty --,
        req-body        [4] KDC-REQ-BODY
}
```

A typical PDU for this message will be something like :

```
0x6C L1               // APPLICATION 12
  0x30 L2             // SEQUENCE
    0xA1 0x03
      0x02 0x01 NN    // pvno
    0xA2 0x03
      0x02 0x01 0x0C  // msg-type, 12
   [0xA3 L3 abcd]     // SEQUENCE OF PA-DATA OPTIONAL
    0xA4 L4 abcd      // KDC-REQ-BODY
```