[MS-SFU]:
Kerberos Protocol Extensions:
Service for User and Constrained Delegation Protocol Specification

---

Revision Summary

Date	Revision History	Revision Class	Comments
10/22/2006	0.01		MCPP Milestone 1 Initial Availability
01/19/2007	1.0		MCPP Milestone 1
03/02/2007	1.1		Monthly release

Date	Revision History	Revision Class	Comments
04/03/2007	1.2		Monthly release
05/11/2007	1.3		Monthly release
06/01/2007	1.3.1	Editorial	Revised and edited the technical content.
07/03/2007	1.3.2	Editorial	Revised and edited the technical content.
07/20/2007	1.3.3	Editorial	Revised and edited the technical content.
08/10/2007	1.3.4	Editorial	Revised and edited the technical content.
09/28/2007	1.3.5	Editorial	Revised and edited the technical content.
10/23/2007	1.3.6	Editorial	Revised and edited the technical content.
11/30/2007	1.3.7	Editorial	Revised and edited the technical content.
01/25/2008	1.4	Minor	Updated the technical content.

Table of Contents

1 Introduction

Service for User (S4U) specifies two extensions to the Kerberos Protocol. Collectivley, these two extensions enable an application service to obtain a Kerberos service ticket on behalf of a user. The resulting service ticket can be used for:

- The requesting service's own information.
- Access control local to the service's machine, impersonating the user.
- Requests to some other service, impersonating the user.

There are two different S4U mechanisms. The first is Service-for-User-to-Self (S4U2self) that allows a service to obtain a Kerberos service ticket to itself on behalf of a user. This enables the service to obtain the user's authorization data that is then used in authorization decisions in the local service.

The second S4U mechanism is Service-for-User-to-Proxy (S4U2proxy). This Kerberos extension enables a service to obtain a service ticket on behalf of the user to a second, back-end service. This allows back-end services to use Kerberos user credentials as if the user had obtained the service ticket and sent it to the back-end service directly. Local policy at the ticket-granting-service (TGS) can be used to limit the scope of the S4U2proxy extension.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Active Directory (AD)
Authorization
Authorization Data
Constrained Delegation
Domain
Forwardable
Kerberos Principal
Key
KRB_AP_REQ/KRB_AP_REP
KRB_AS_REQ/KRB_AS_REP
KRB_TGS_REQ/KRB_TGS_REP
Principal
Privilege Attribute Certificate (PAC)
Realm
Service
Service for User (S4U)
Service for User to Proxy (S4U2proxy)
Service for User to Self (S4U2self)
Service Ticket
Session Key
Ticket
Ticket Granting Service (TGS)
Ticket Granting Ticket (TGT)

The following terms are specific to this document:

Key Distribution Center (KDC): A network **service** that supplies **tickets** to entities to authenticate other entities. Specifically, the Kerberos **KDC** is the Kerberos **ticket-granting service (TGS)** specified in the Kerberos protocol. The Kerberos **service** that implements the authentication and **TGS** is specified in the Kerberos protocol.<1>

Pre-authentication: In Kerberos, the act of proving identity (and knowledge of a **key**) before the issuance of the initial **ticket-granting ticket (TGT)**, as specified in [RFC4120] sections 5.2.7 and 7.5.2.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-DTYP] Microsoft Corporation, "Windows Data Types", January 2007.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-KILE] Microsoft Corporation, "Kerberos Protocol Extensions", January 2007.

[MS-PAC] Microsoft Corporation, "Privilege Attribute Certificate Data Structure", January 2007.

[Referrals] Raeburn, K., Zhu, L., and Jaganathan, K., "Generating KDC Referrals to Locate Kerberos Realms", June 2006, http://tools.ietf.org/html/draft-ietf-krb-wg-kerberos-referrals-08

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC3280] Housley, R., Polk, W., Ford, W., and Solo, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002, http://www.ietf.org/rfc/rfc3280.txt

[RFC3961] Raeburn, K., "Encryption and Checksum Specifications for Kerberos 5", RFC 3961, February 2005, http://www.ietf.org/rfc/rfc3961.txt

[RFC4120] Neuman, C., Yu, T., Hartman, S., and Raeburn, K., "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005, http://www.ietf.org/rfc/rfc4120.txt

[RFC4757] Jaganathan, K., Zhu, L., and Brezak, J., "The RC4-HMAC Kerberos Encryption Types Used by Microsoft Windows", RFC 4757, December 2006, http://www.ietf.org/rfc/rfc4757.txt

1.2.2 Informative References

None.

1.3 Protocol Overview & Synopsis

Microsoft's implementation of Kerberos [MS-KILE] defines the Windows implementation of Kerberos relative to the specification in [RFC4120]. This protocol extends the Windows implementation of Kerberos by specifying **Service for User (S4U)** extensions in relation to [RFC4120]. The reader should be familiar with [MS-KILE] and [RFC4120] as the basis for understanding S4U.

S4U supports two sub-protocols: **Service for User to Self (S4U2self)** and **Service for User to Proxy (S4U2proxy)**. Both of these allow a **service** to request a **ticket** from the **Key Distribution Center (KDC)** on behalf of a user. A ticket can be retrieved by the service to itself via S4U2self or to another service via S4U2proxy. The client name, **realm**, and **authorization data** in the **service ticket** using these extensions are of the user, not of the service making the S4U request. This is in contrast to the Kerberos Protocol specified in [RFC4120] where any service tickets requested by a service will have the client name, realm, and authorization data of that requesting service.

1.3.1 S4U2self

The S4U2self extension allows a service to obtain a service ticket to itself on behalf of a user. The user is identified to the KDC using the user's name and realm. Alternatively, the user may be identified based on the user's certificate. The Kerberos **ticket-granting service (TGS)** request and response messages, **KRB_TGS_REQ** and KRB_TGS_REP, are used along with one of two new data structures. The new PA-FOR-USER data structure is used when the user is identified to the KDC by the user name and realm name. The other structure, PA-S4U-X509-USER, is used when the user certificate is presented to the DC to obtain the **authorization** information. By obtaining a service ticket to itself on behalf of the user, the service receives the user's authorization data in the ticket.

1.3.2 S4U2proxy

The Service-for-User-to-Proxy (S4U2proxy) extension provides a service that obtains a service ticket to another service on behalf of a user. The Kerberos ticket-granting service (TGS) Request and Response messages, KRB_TGS_REQ and KRB_TGS_REP, are used along with the new CNAME-IN-ADDL-TKT and S4U_DELEGATION_INFO data structures. The second service is typically a proxy performing some work on behalf of the first service, and the proxy is doing that work under the authorization context of the user.

The S4U2proxy extension requires that the service ticket to the first service has the **forwardable** flag set (see Service1 in Figure 1, section 1.3.3). This ticket may be obtained through an S4U2self protocol exchange.

This feature differs from the Kerberos forwarded-TGT delegation mechanism and the proxy-service-ticket delegation mechanism ([RFC4120] section 2.5) in the following ways:

- The service does not require the user to forward either the user's **ticket-granting ticket (TGT)** or the proxy ticket and the associated **session key**.
- The user does not need to authenticate through Kerberos (the S4U2self extension can be used instead but this is not a requirement). In other words, the user does not need to have a TGT or a proxy service ticket.
- Local policy can be used to limit the services that can be delegated. This is contradictory to the forwarding-TGT delegation mechanism, as specified in [RFC4120] section 2.6, where a service can delegate to any other service. This is similar to the proxy ticket delegation, as specified in [RFC4120] section 2.5, except the client is not involved in making the delegation decision.
- The client has no control over whether a service can delegate on behalf of the user or not. The client does not request delegation nor does it pass a forwardable TGT to the service. The client is unaware that delegation will be, or has been, performed. If local policy allows the service to perform S4U2proxy delegation, this delegation is performed solely at the discretion of the service.

When using the S4U2proxy delegation and forwarded-TGT delegation mechanisms, the delegation is invoked when the server impersonates the client and performs operations on a remote server (such as ldap_bind() or RPC_bind()). Kerberos SSP will first detect if the forwarded-TGT delegation

mechanism is available (by checking if there is a forwarded TGT in the local ticket cache); if no forwarded TGT is available, it will then try to perform the S4U2proxy delegation.

1.3.3 Protocol Overview

Figure 1 shows the message sequence diagram for Kerberos delegation with a forwarded ticket-granting ticket (TGT). This is background information designed to show the workings of Kerberos delegation, as specified in [RFC4120] section 2.8. This mechanism is then compared to the Service for User (S4U) extensions.

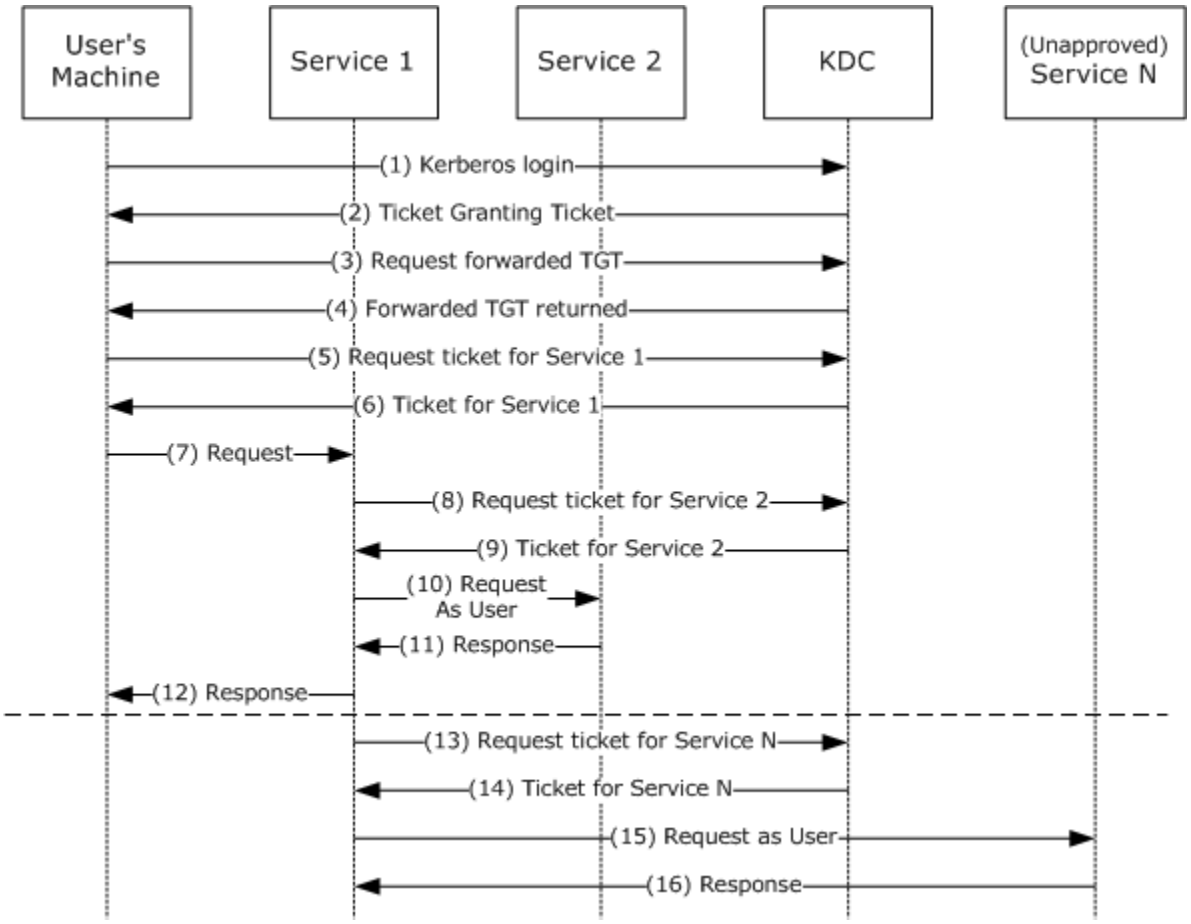

Figure 1: Kerberos Delegation with forwarded TGT

Figure 1 depicts the following protocol steps:

1. The user authenticates to the Key Distribution Center (KDC) by a **KRB_AS_REQ** and requests a forwardable TGT.

2. The KDC returns a forwardable TGT in the KRB_AS_REP.

3. The user requests a forwarded TGT based on the forwardable TGT from step 2. This is done by the KRB_TGS_REQ.

4. The KDC returns a forwarded TGT for the user in the KRB_TGS_REP.

5. The user makes a request for a service ticket to Service 1 using the TGT returned in step 2. This is done by the KRB_TGS_REQ.

6. The ticket-granting service (TGS) returns the service ticket in a KRB_TGS_REP.

7. The user makes a request to Service 1 by a **KRB_AP_REQ**, presenting the service ticket, the forwarded TGT, and the session key for the forwarded TGT.

8. To fulfill the user's request, Service 1 needs Service 2 to perform some action on behalf of the user. Service 1 uses the forwarded TGT of the user and sends that in a KRB_TGS_REQ to the KDC, asking for a ticket for Service 2 in the name of the user.

9. The KDC returns a ticket for Service 2 to Service 1 in a KRB_TGS_REP, along with an authentication **Key** that Service 1 can use. The ticket identifies the client as the user, not as Service 1.

10. Service 1 makes a request to Service 2 by a KRB_AP_REQ, acting as the user.

11. Service 2 responds.

12. With that response, Service 1 can now respond to the user's request in step 7.

13. The TGT forwarding delegation mechanism as described here does not constrain Service 1's use of the forwarded TGT. Service 1 can ask the KDC for a ticket for any other service– in the name of the user.

14. The KDC will return the requested ticket.

15. Service 1 can then continue to impersonate the user with Service N. This can pose a risk if, for example, Service 1 is compromised. Service 1 can continue to masquerade as a legitimate user to other services.

16. Service N will respond to Service 1 as if it was the user's process.

Server-for-User-to-Self (S4U2self) is intended to be used when the user authenticates to the service in some way other than Kerberos. For example, a user could authenticate to a Web server by some means private to the Web server. The Web server could then use S4U2self extension to get a ticket, with authorization data, just as if the user had used Kerberos originally. This simplifies the server's authorization decision by making all decision paths behave as though Kerberos was used. S4U2self primarily uses the KDC to get information about the user for the caller's own benefit. Service-for-User-to-Proxy (S4U2proxy) allows the caller to contact some other service, acting on behalf of the user. The detailed overview is given in Figure 2.

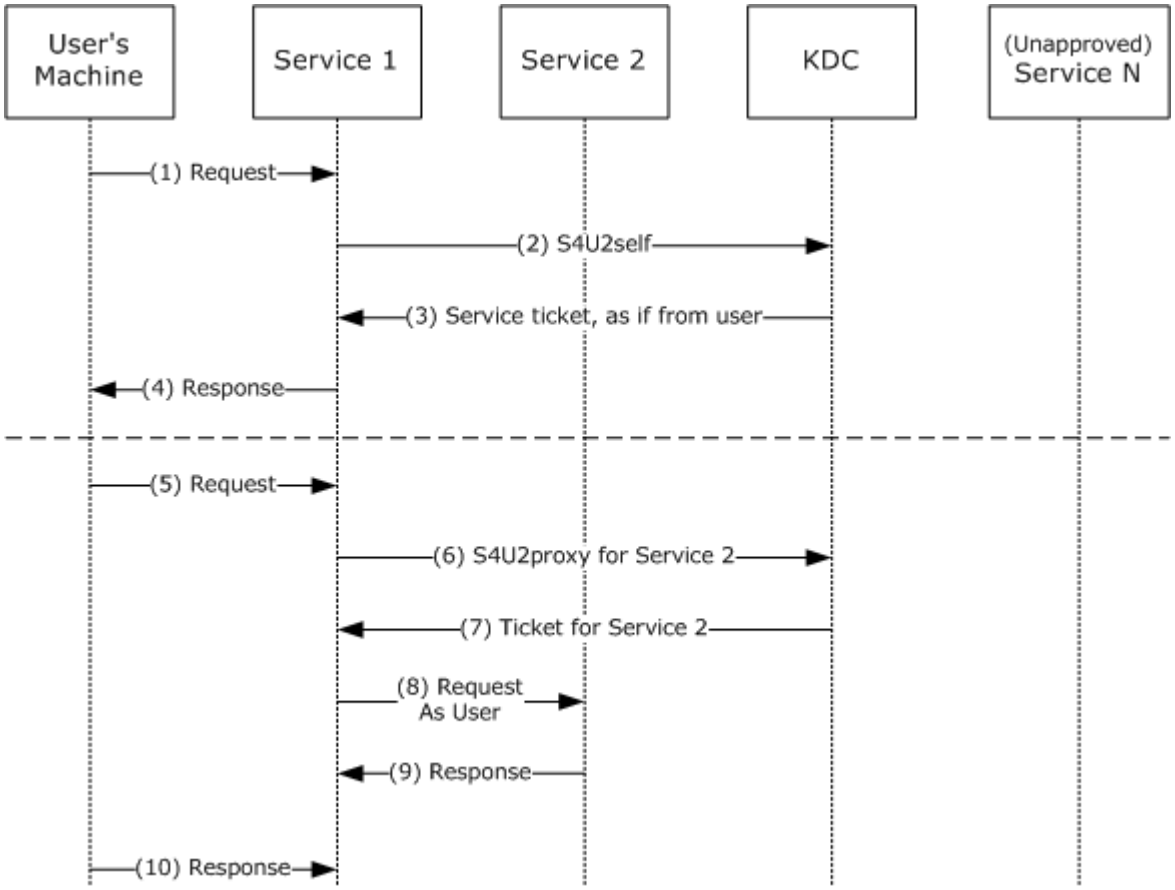

Figure 2: S4U2self and S4U2proxy

S4U2self is described in the top half of Figure 2. Using this extension, the service receives a service ticket to the service itself (a ticket that cannot be used elsewhere).

Figure 2 depicts the following protocol steps:

1. The user's machine makes a request to Service 1. The user is authenticated but Service 1 does not have the user's authorization data. Typically this is due to the authentication being performed by some means other than Kerberos.

2. Service 1, which has already authenticated with the KDC and has obtained its TGT, asks for a service ticket to itself on behalf of the named user by the S4U2self extension. The user is identified by the user name and the user's realm name in the S4U2self data (as specified in section 2.2.1). Alternatively, if service 1 is in possession of the user's certificate, it may use it to identify the user to the KDC using the PA-S4U-X509-USER structure.

3. The KDC returns a service ticket addressed to Service 1 as if it had been requested from the user with the user's own TGT. The service ticket MAY contain the authorization data of the user.

4. Service 1 can use the authorization data from the service ticket to fulfill the user's request. The service then responds to the user.

   While S4U2self provides information about the user to Service 1, this extension does not allow Service 1 to make requests of other services on the user's behalf. That is the role of S4U2proxy.

5. The user's machine makes a request to Service 1. Service 1 needs to access resources on Service 2 as the user. However, Service 1 does not have a forwarded TGT from the user to perform delegation by a forwarded TGT, as described in Figure 1.

6. Two preconditions apply to this step. First, Service 1 has already authenticated with the KDC and has a valid TGT. Second, Service 1 has a forwardable service ticket from the user to Service 1. This **forwardable** service ticket may have been obtained by a KRB_AP_REQ, as specified in [RFC4120] section 3.2, or by an S4U2self request.

7. Service 1 requests a service ticket to Service 2 on behalf of the named user. The user is identified by the client name and the client realm in the service ticket for Service 1. The authorization data in the ticket to be returned is also copied from the service ticket. Service 1 and Service 2 MUST be in the same realm. The user, however, can be in a different realm.

8. The KDC validates the **privilege attribute certificate (PAC)** by checking the signature data of PAC structure, as specified in [MS-PAC] section 2.2.8. If valid, the KDC returns a service ticket for Service 2, but the client identity stored in the **cname** and **crealm** fields of the service ticket are that of the user, not Service 1.<2>

9. Service 1 uses the service ticket to make a request to Service 2. Service 2 treats this request as coming from the user and believes that the user was authenticated by the KDC.<3>

10. Service 2 responds to the request.

11. Service 1 responds to the user's request of message 5.<4>

1.4 Relationship to Other Protocols

The S4U extensions are based on Kerberos, as specified in [RFC4120] and [MS-KILE]. The [MS-KILE] specification also details the dependence on lower-layer protocols such as TCP and UDP. Applications using other protocols may use S4U to create a common authorization path within the application.

The S4U2self extension is used to obtain a privilege attribute certificate (PAC), as specified in [MS-PAC], to determine the authorization capabilities of the user. In addition, the PAC is used in S4U2proxy to validate that S4U2proxy service tickets have not been misused.

The referral mechanism, as specified in [Referrals], is used in the S4U2self protocol extension if the user's realm is different from that of the service trying to obtain an S4U2self service ticket.

1.5 Prerequisites/Preconditions

All Key Distribution Centers (KDCs)) and Kerberos servers sending or receiving the Service for User (S4U) extensions in the KRB_TGS_REQ and KRB_TGS_REP messages MUST understand the protocol extensions. Services can detect whether the KDC supports these extensions by checking the client name of the returned ticket. KDCs that do not understand these extensions will return the client name as the service that is making the request. KDCs that understand these extensions either return an error or return a service ticket that contains the client name as the user, not the service that is making the request.<5>

To support the lookup of users based on a supplied certificate, the KDC MUST have an accounts database available to it that supports looking up user accounts using one or more fields present in the certificate.

1.6 Applicability Statement

The Service-for-User-to-Proxy (S4U2proxy) extension supports delegation that is transparent to the client. Activities are performed under the user's identity in one or more services. Local policy may be used to limit this functionality and control which services can use this feature.

1.7 Versioning and Capability Negotiation

There is no version information in the Service for User (S4U) extensions. A service that uses these extensions will send the new options or data structures in the KRB_TGS_REQ and KRB_TGS_REP messages. Detecting whether a given Key Distribution Center (KDC) can support the extensions is specified in section 1.5.

1.8 Vendor-Extensible Fields

There are no vendor-extensible fields in the Service for User (S4U) extensions.

2 Messages

The following sections specify how messages are transported and common data types for this protocol.

2.1 Transport

For more information on the Kerberos Protocol as well as dependencies on lower-level protocols, see [MS-KILE], specifically section 2.1.

2.2 Message Syntax

The Service for User (S4U) extensions use new structures conforming to the extensibility mechanisms provided in the Kerberos RFC, as specified in [RFC4120] section 1.5, and new values for options already specified by Kerberos in [RFC4120] section 1.1. The following sections describe these new structures and values.

2.2.1 PA-FOR-USER

In a KRB_TGS_REQ/KRB_TGS_REP sub-protocol sequence, as specified in [RFC4120] section 3.3, a **Kerberos principal** uses its ticket-granting ticket (TGT) to request a service ticket to a service. The TGS uses the requesting **principal's** identity from the TGT passed in the KRB_TGS_REQ to create the service ticket.

In the S4U2self KRB_TGS_REQ/KRB_TGS_REP protocol extension, a service requests a service ticket to itself on behalf of a user. The user is identified to the KDC by the user's name and realm. Alternatively, the user may be identified using the user's certificate. The service uses its own TGT and adds a new type of padata. The padata type is specified in [RFC4120] section 5.2.7.

If the user, on whose behalf the service requests the service ticket, is identified using the user name and user realm then the padata type, PA-FOR-USER (ID 129), is used. This padata type contains a unique identifier that indicates the user's identity. This unique identifier consists of the user name and user realm.

The PA-FOR-USER padata value is protected with the help of a keyed checksum, using a key in the PA-TGS-REQ padata-type field of the request message.

The following code defines the ASN.1 structure of the PA-FOR-USER padata type.

```
padata-type     ::= PA-FOR-USER
       -- value 129
padata-value    ::= EncryptedData
                      -- PA-FOR-USER-ENC

PA-FOR-USER-ENC ::= SEQUENCE {
   userName[0] PrincipalName,
   userRealm[1] Realm,
   cksum[2] Checksum,
   auth-package[3] KerberosString,
}
```

userName: the PrincipalName type is discussed in detail in [RFC4120] section 5.2.2. It consists of a name type and name string. The default value for name type is NT_UNKNOWN as specified in

[RFC4120] section 6.2. The name string is a sequence of strings encoded as KerberosString, as specified in [RFC4120] section 5.2.1, that (together with the userRealm) represents a user principal.

userRealm: A KerberosString that represents the realm in which the user account is located. This value is not case-sensitive.

cksum: A hash of a salt value, the service's TGT session key, userName, userRealm, and authpackage. The Checksum type is specified in [RFC4120] section 5.2.9.

The checksum operation is computed according to [RFC3961]. The data for which the checksum is computed is calculated as follows: the PA-FOR-USER-ENC ASN.1 structure is decoded into in-memory structures, and the contents of the member fields are then concatenated in the order in which these fields are defined. The **userName** field is a sequence of strings; the values of the strings in the sequence are simply concatenated. The resulting string is then appended with the value of userRealm, which is another string, then appended with the value of the auth-package, which is yet another string.

The Message type or key usage number being used in this operation is 17 and the key used is the session key of the serviceTGT.

In the preceding calculation, all strings MUST be in UTF-8 format with no multiple-byte encodings of a possible single byte value. That is, while it may be legal to represent the character "ä" as 0x61 0xa8 (or, "a<dieresis mark>"), it MUST be represented as the single byte code 0xe4. No folding, mapping, or other manipulation of the string is performed prior to the hashing, so successful authentication depends on the strings being created in the same fashion by both parties.

auth-package: A string name of the authentication mechanism used to authenticate the user. This must be set to the string, "Kerberos". This value is not case-sensitive.

2.2.2 PA_S4U_X509_USER

If the service possesses the user certificate it may obtain a service ticket to itself on that user's behalf using the S4U2self KRB_TGS_REQ/KRB_TGS_REP protocol extension, with a new padata type PA-S4U-X509-USER (ID 130).<6> This padata type contains a unique identifier that indicates the user's identity. This unique identifier consists of the user's certificate and, optionally, the user's name and realm.

The following code defines the structure of the PA-S4U-X509-USER padata type.

```
Message Type      padata-type    Contents of padata-value

AS-REQ            130            X509 certificate encoded per
                                 [RFC3280].

TGS-REQ/TGS-REP   130            PA-S4U-X509-USER ASN.1 structure


The corresponding data contains the DER encoded PA-S4U-X509-USER
structure.


PA-S4U-X509-USER::= SEQUENCE {
   user-id[0] S4UUserID,
   checksum[1] Checksum,
}
```

```
S4UUserID ::= SEQUENCE {
   nonce [0] INTEGER, -- the nonce in KDC-REQ-BODY
   cname [1] PrincipalName OPTIONAL,
                -- Certificate mapping hints
   crealm [2] Realm,
   subject-certificate [3] OCTET STRING OPTIONAL,
   options [4] BIT STRING OPTIONAL...
}
```

user-id: Contains the user identifiers. This can be either the user name and realm or the user's certificate.

checksum: This is the signature of the **user-id** field above.

nonce: This contains the identically-named field in the KDC body of the containing request.

cname: the PrincipalName type is discussed in detail in [RFC4120] section 5.2.2. It consists of a name type and name string. The default value for name-type is NT_UNKNOWN as specified in [RFC4120] section 6.2. The name string is a sequence of strings encoded as KerberosString, as specified in [RFC4120] section 5.2.1, that (together with the userRealm) represents a user principal.

crealm: A KerberosString that represents the realm in which the user account is located. This value is not case-sensitive.

subject-certificate: This optional field contains the user's certificate that is encoded as specified in [RFC3280].

options: Specifies the additional options in the S4U request. Currently, only one option is defined:

Value	Meaning
0x40000000	This option causes the KDC to check logon hour restrictions for the user.

In [MS-SFU], the client needs to be able to locate the KDC of the client's realm. If the S4U call is based on the certificate and no user name is supplied, the client uses a PA_S4U_X509_USER padata and the corresponding data contains the clients X509 certificate encoded per [RFC3280].

2.2.3 CNAME-IN-ADDL-TKT

This is a new Key Distributor Center (KDC) option that must be set in a KRB_TGS_REQ to request Service-for-User-to-Proxy (S4U2proxy) functionality. The kdc-options flags are specified in [RFC4120] section 5.4.1, and the new cname-in-addl-tkt option is defined as the KDC option with bit position 14.

```
KDCOptions      ::= KerberosFlags
        -- cname-in-addl-tkt (14)
```

2.2.4 S4U_DELEGATION_INFO

The S4U_DELEGATION_INFO structure ([MS-PAC] section 2.2.9) lists the services that have been delegated by this client and subsequent services or servers. The list is meaningful as the Service-

for-User-to-Proxy (S4U2proxy) feature could be used multiple times in succession from service to service. This is useful for auditing purposes.

3 Protocol Details

The following sections specify protocol details including client role, server details, and algorithms.

3.1 Service Details

This section defines the message processing for an application service (see Service 1 in Figure 3, section 3.1.4) using the Service for User (S4U) extensions as well as the ticket-granting service (TGS) that responds to the S4U requests.

3.1.1 Timers

There are no timers added for these extensions.

3.1.2 Initialization

Before sending a KRB_TGS_REQ with a Service for User (S4U) extension, the service MUST have already authenticated to the Key Distributor Center (KDC) and received a ticket-granting ticket (TGT).

3.1.3 Higher-Layer Triggered Events

This section contains the following information.

- S4U2self Triggered Events
- S4U2proxy Triggered Events

3.1.3.1 S4U2self Triggered Events

A service (see Service 1 in figure 3, section 3.1.4) uses a KRB_TGS_REQ with the S4U2self extension when the service wants to make a local access check for a user. This typically occurs when the user has sent some request to the service through a non-Kerberos protocol. The service uses the S4U2self KRB_TGS_REQ/KRB_TGS_REP to obtain authorization data about the user from the Key Distributor Center (KDC).

3.1.3.2 S4U2proxy Triggered Events

A service uses a KRB_TGS_REQ with the Service-for-User-to-Proxy (S4U2proxy) extension when the service determines that it needs to contact another service on behalf of a user for which it has a service ticket. S4U2proxy is used when the request to the second service must use the user's credentials, not the credentials of the first service. The service sends a KRB_TGS_REQ with the S4U2proxy information to obtain a service ticket to another service.

3.1.4 Message Processing and Sequencing Rules

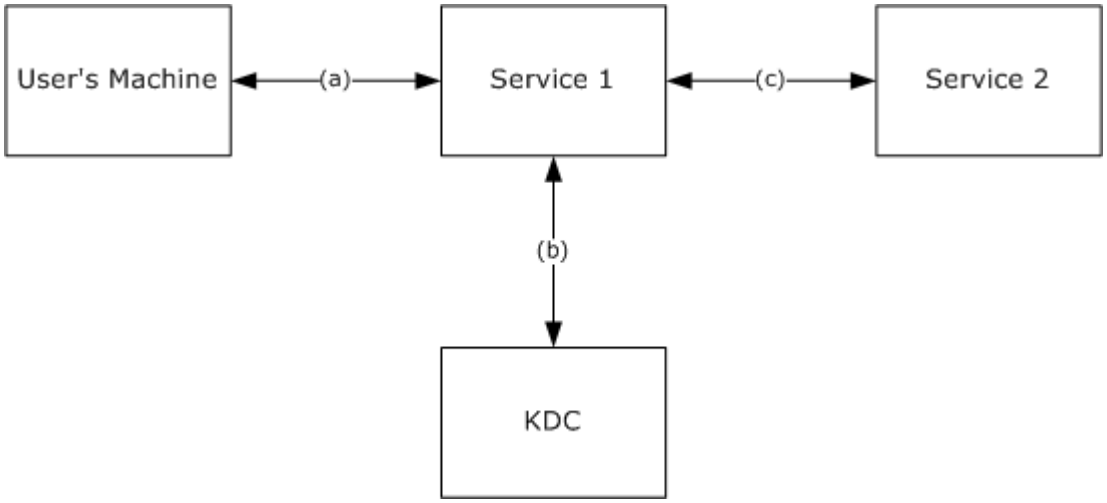

Figure 3: Entities involved in Service for User (S4U) protocols

Figure 3 shows the entities involved in S4U protocols and the principal communications between them. In the following discussions of processing the S4U messages, it is assumed that Service 1 has started up and has already authenticated itself to its own Key Distributor Center (KDC) via the standard KRB_AS_REQ/KRB_AS_REP exchange (b). In addition, the user has contacted the service and authenticated through some mechanism (a) other than using the KDC.

3.1.4.1 S4U2self Message Processing

The Service for User to Self (S4U2self) extensions allow Service 1 to use the service's ticket-granting ticket (TGT) in a Kerberos KRB_TGS_REQ message to retrieve a service ticket to the service itself, as if the ticket was originally requested by the user.

In the S4U2self request, the user is identified by the user realm and the user name or alternatively using the user's certificate if the service has it as specified in sections 3.1.4.1.1 and 3.1.4.1.2.

Services can detect whether or not the Key Distribution Center (KDC) supports these extensions by checking the client name of the returned ticket. For KDCs that do not understand these extensions, the KDC returns the client name as the service that is making the request. For example, Windows 2000 Server ignores the S4U2self and S4U2proxy data and return a service ticket with the client name set to the name of the service that made the request, as specified in [RFC4120], section 2.5. KDCs that understand these extensions either return an error or a service ticket that contains the client name as the user, not the service that is making the request.<7>

3.1.4.1.1 Using the User's Realm and User Name to Identify the User

Service 1 uses the name and realm of the user to locate the appropriate DC to provide the authorization information for the user. The user's realm may be found by local policy, or, if the user name is a User Principal Name, by using KRB_AS_REQ and KRB_AS_REP as follows. Service 1 sends a KRB_AS_REQ without any **pre-authentication** to Service 1's Key Distributor Center (KDC). If this KDC holds the user's account, then it will return KDC_ERR_PREAUTH_REQUIRED, and the user's realm is handled by the KDC. Otherwise, the KDC can refer Service 1 to another realm that may contain the user account or that may have better information about the realm of the user account, as specified in [Referrals] section 4. The KDC does this by returning a KDC_ERR_WRONG_REALM

error, as specified in [RFC4120] section 7.5.9, in the KDC_AS_REP and setting the **crealm** field to the next realm to try. Service 1 then sends a KRB_AS_REQ to the next realm, repeating the process until it reaches a KDC in the user's realm or receives some other error.

Once the realm with the user's account is identified, Service 1 begins the protocol to get the service ticket on behalf of the user. The first step is for the service to retrieve a TGT to the ticket-granting service (TGS) in the user's realm.

If the user's realm is the same as Service 1's realm, the service already has the TGT it needs. If the user's account is in a different realm, the service constructs a KRB_TGS_REQ with the service's TGS as the sname in the request. The cname and crealm are set to the name and realm of Service 1. See [RFC4120] section 5.3 for the use of sname and cname. If there is not a direct trust relationship with an inter-realm key between Service 1's realm and the user's realm, the service's TGS will return a TGT to a realm closer to the user's realm. This process is repeated until Service 1 obtains a TGT to a TGS in the user's realm.

Using the TGT to the TGS in the user's realm, Service 1 requests a service ticket to itself. The S4U2self information in the KRB_TGS_REQ consists of: padata-type = PA-FOR-USER, which consists of four fields: **userName**, **userRealm**, **cksum**, and **auth-package**. Service 1 sets these fields as follows: The userName is a structure consisting of a name type and a sequence of name string (as specified in section 6.2 of [RFC4120]). The name type and name string fields are set to indicate the name of the user. The default name-type is NT_UNKNOWN. The userRealm is the realm of the user account. If unknown, Service 1 SHOULD use its own realm name. The auth-package field MUST be set to the string, "Kerberos". The auth-package field is not case sensitive.

The TGS first checks the user account. If the account is not found, a KRB-ERROR message with KDC_ERR_C_PRINCIPAL_UNKNOWN is returned.

If the account exists and Service 1 is in the TGS'sRealm, the TGS will reply with the service ticket in the KRB_TGS_REP. The sname and realm in the service ticket MUST be set to name and realm of Service 1. The cname and crealm shall be that of the userName and userRealm fields of the PA-FOR-USER data. The reply ticketKRB_TGS_REP MAY also contain authorization data of the user, as specified in [MS-PAC].

If Service 1 is not in the user's realm, the user's TGS will reply with a KRB_TGS_REP message containing a referral TGT to the next realm in the trust chain. The cname and crealm in the KRB_TGS_REP will be set to name and realm of Service 1. The referral TGT MAY contain authorization data of the user identified in the PA-FOR-USER structure based on the policy of the TGS.

Multiple intermediate realms may need to be transited. Service 1 will send a KRB_TGS_REQ with the S4U2self data in the PA-FOR-USER structure to each TGS in turn along the referral path (as specified in [Referrals]). Each TGS along this path will reply with a TGT to the next TGS in the referral path.

When the KRB_TGS_REP is finally sent from Service 1 to the TGS in Service 1's realm, the TGS will reply with a service ticket. The TGS in Service 1's realm MUST set the sname and realm in the service ticket to the name and realm of Service 2. The cname and crealm fields MUST be set to the name and realm of the user. The KRB_TGS_REP will contain the cumulative authorization based on the policy of each TGS transited.

If requested by the service in the KRB_TGS_REQ to the service's TGS and allowed by that TGS, the FORWARDABLE flag in the service ticket in the KRB_TGS_REP shall be set. The service MUST request a forwardable ticket if it wants to use the returned service ticket as the input for a later Service-for-User-to-Proxy (S4U2proxy) request.

3.1.4.1.2 Using the User's Certificate to Identify the User

If Service 1 has the user certificate it may present it to the domain controller (DC) to identify the user.<8> For the user account to be identified based on a certificate, the DC MUST support certificate mapping for that user in the accounts database, based on one or more of the certificate's fields. If no user can be located based on any of the supported mappings then the request MUST fail.

To locate the user account object if the user's name is not available, Service 1 MUST send a KRB_AS_REQ to its KDC with a KRB5_PADATA_FOR_X509_USER padata that contains the client's X509 certificate encoded in ASN.1, as specified in [RFC3280]. This X.509 certificate is used by the KDC to locate the account object when the userName hint is not supplied.

Note that in section 2.2.2, the corresponding data for the KRB5_PADATA_FOR_X509_USER padata, as sent in the TGS-REQ, contains a PA-S4U-X509-USER ASN.1 structure. This means the content of KRB5_PADATA_FOR_X509_USER in AS-REQ is different than that in TGS-REQ.

3.1.4.2 S4U2proxy Message Processing

As shown in section 3.1.4 Figure 3, Service 1 uses the Service-for-User-to-Proxy (S4U2proxy) extension to request a service ticket to Service 2. By using the S4U2proxy extension, the cname and crealm in the resulting service ticket and any authorization data is that of the user, not Service 1. To use this extension, Service 1 and Service 2 MUST belong to the same realm. A KRB_TGS_REQ sent to a ticket-granting service (TGS) in a different realm MUST return a kdc-error message with the error code set to KRB-ERR-BADOPTION.

Before using the S4U2proxy extension, Service 1 must have already authenticated to the Key Distributor Center (KDC) and have a ticket-granting ticket (TGT). Service 1 must also have a service ticket to itself from the user for whom the S4U2proxy request will be made. Service 1 requests a service ticket to Service 2 by sending a KRB_TGS_REQ with the S4U2proxy extensions to the TGS. This request MUST include the new cname-in-addl-tkt options flag in the **kdc-options** field. The user's service ticket to Service 1 MUST be placed in the **additional-tickets** field of the request. The **sname** and **realm** fields shall be set to the name and realm of Service 2. The service ticket being passed in the **additional-tickets** field MUST have the forwardable flag set. If the forwardable flag is not set in the service ticket in the additional-tickets field, a kdc-error message will be returned with the error code set to KRB-ERR-BADOPTION.

If the TGS supports the cname-in-addl-tkt option and the service ticket in the **additional-tickets** field is forwardable, the TGS will use local policy to determine if Service 2, identified in the **sname** and **srealm** fields of the KRB_TGS_REQ, is a service to which Service 1 is allowed to obtain an S4U2proxy service ticket. The signature of the privilege attribute certificate (PAC) is also checked to make sure it was created in this realm, as specified in [MS-PAC] section 2.2.8.

If the policy allows this request, the TGS creates a new service ticket using the cname and crealm from the service ticket in the **additional-tickets** field as the cname and crealm of the new service ticket. Service 2's name and realm are placed in the sname and realm fields of the reply. The TGS will examine the PAC of the service ticket in the **additional-tickets** field for a S4U_DELEGATION_INFO structure. If this structure exists, it will be used to create the new PAC. If it does not exist, the structure will be added in the new PAC. The name of Service 2 will be placed in the S4U2proxyTarget, and the name of Service 1 will be added to the S4UTransitedServices list. The TransitedListSize size will be set to 1 if the structure did not exist in the PAC of the **additional-tickets** field. If the structure did exist, the count will be incremented by 1. The service ticket MUST be marked as forwardable. The TGS returns the new service ticket in the KRB_TGS_REP to Service 1.

Service 1 now has a service ticket to Service 2 with the cname and crealm of the user and authorization data of the user, just as if the user had requested the service ticket. Note, however, the session key for authenticating to that ticket is owned by Service 1.<9>

3.1.5 Timer Events

There are no timer events for Service for User (S4U) extensions.

3.1.6 Other Local Events

There are no other local events.

4 Protocol Examples

The following sections describe several operations as used in common scenarios to illustrate the function of this protocol.

4.1 S4U2self Single Realm Example

The following diagram depicts the S4U2self KRB_TGS_REQ from the service to the Kerberos TGS. In this case, the user's account belongs to the same realm as the service.

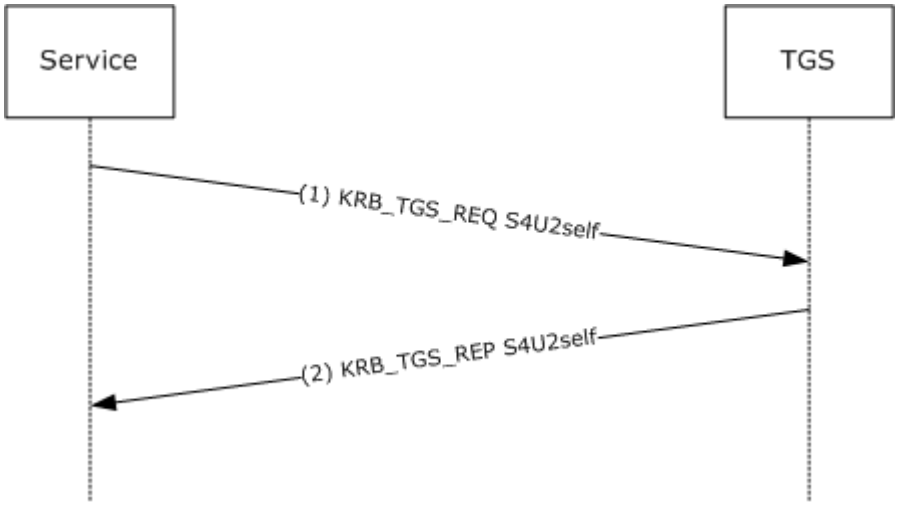

Figure 4: S4U2self KRB_TGS_REQ

The precondition to the diagram above is that the service has already authenticated to the KDC and has a TGT.

In step 1, the service uses the S4U2self extension to retrieve a service ticket to itself on behalf of the user. The service fills out the PA_FOR_USER data structure and sends the KRB_TGS_REQ to the TGS.

Assuming the TGS supports the PA_FOR_USER extension, the TGS returns the service ticket for the user in the KRB_TGS_REP in step 2. The privilege attribute certificate (PAC) returned in the service ticket contains the authorization data, as specified in [MS-PAC] section 3. If the service requested the forwardable option and the local policy of the TGS allows it, the TGS shall set the **ticket-flag** field to forwardable.

4.2 S4U2self Multiple Realm Example

The multi-realm scenario requires extra KRB_TGS_REQ/KRB_TGS_REP exchanges. The service must get a S4U2self service ticket for the user from the service's KDC. To do so, it must retrieve a TGT for the user to the service's TGS. This is accomplished by first obtaining a TGT to the user's TGS and then following referrals from the user's TGS to the service's TGS.

There are two preconditions to the diagram that follows. The first is that the service has already authenticated to its KDC and has a TGT to its TGS. The second precondition is that the service has identified the realm for the user account. One approach for finding the user's realm is through the use of KRB_AS_REQs and using the information returned from the KDC, as specified in [Referrals].

In the diagram that follows, TGS_A represents the TGS in the service's realm. TGS_B represents the TGS in the user's realm.

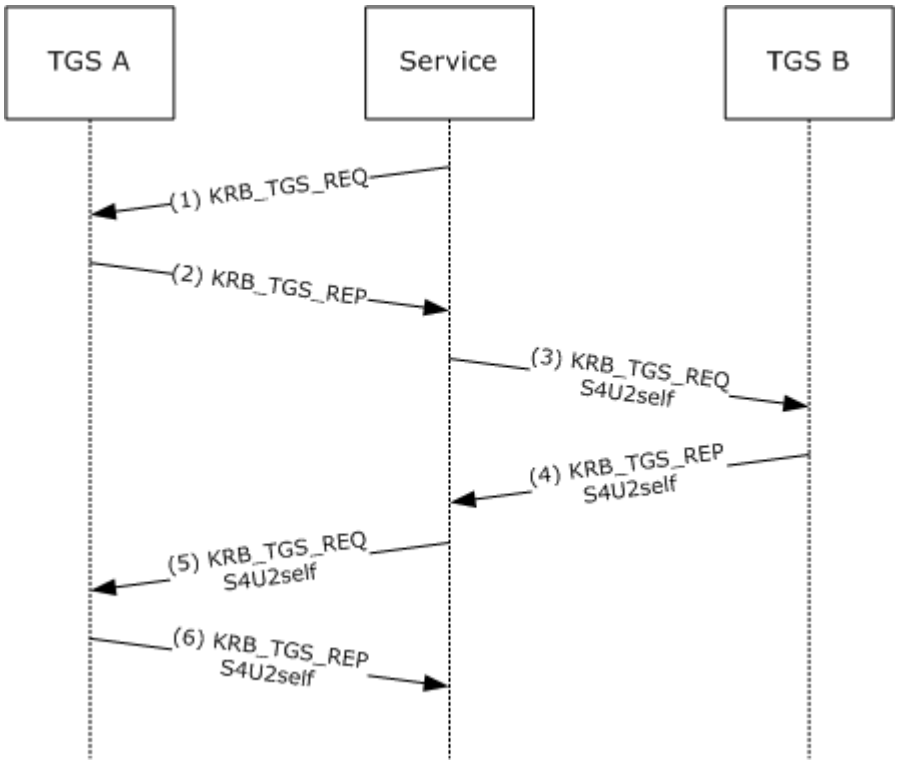

Figure 5: S4U2self Multiple Realm Example

In step 1, the service sends a request to its TGS, TGS_A, for a TGT to TGS_B. No S4U2self information is included in this request. TGS_A responds with the cross-realm TGT to TGS_B in step 2. If TGS_B was not the user's realm but was instead just a realm closer, then the service would send a KRB_TGS_REQ to TGS_B to get a TGT to the next realm.

The service now uses the TGT to TBS_B to make the S4U2self request in the KRB_TGS_REQ in step 3. The service uses the PA-FOR-USER padata-type in the request to indicate the user information in the S4U2self request.

TGS_B creates a PAC with the user's authorization information (as specified in [MS-PAC] section 3) and returns it in a TGT referral in the KRB_TGS_REP in step 4. TGS_B cannot create the service ticket because TGS_B does not possess the server's account information, as the server is part of the realm served by TGS_A.

If there are more TGSs involved in the referral chain, steps 3 and 4 will be repeated to follow the chain.

In step 5, the server uses the TGT from the referral from step 4 and uses the PA-FOR-USER padata-type to request the service ticket to itself on behalf of the user. TGS_A creates the service ticket for the user to the service and returns it in step 6. The PAC returned in this step will contain the appropriate combination of authorization data placed in the PAC by TGS_B in step 4 and the data from TGS_A in step 6, as specified in [MS-PAC] section 4.2.1.

4.3 S4U2proxy Example

The following diagram depicts a service obtaining a service ticket on behalf of a client to another service, a proxy service. The ticket-granting service (TGS) is the TGS for both service and proxy service. It is also assumed that service has already authenticated to the Key Distribution Center (KDC) and has obtained a ticket-granting ticket (TGT) to the TGS.

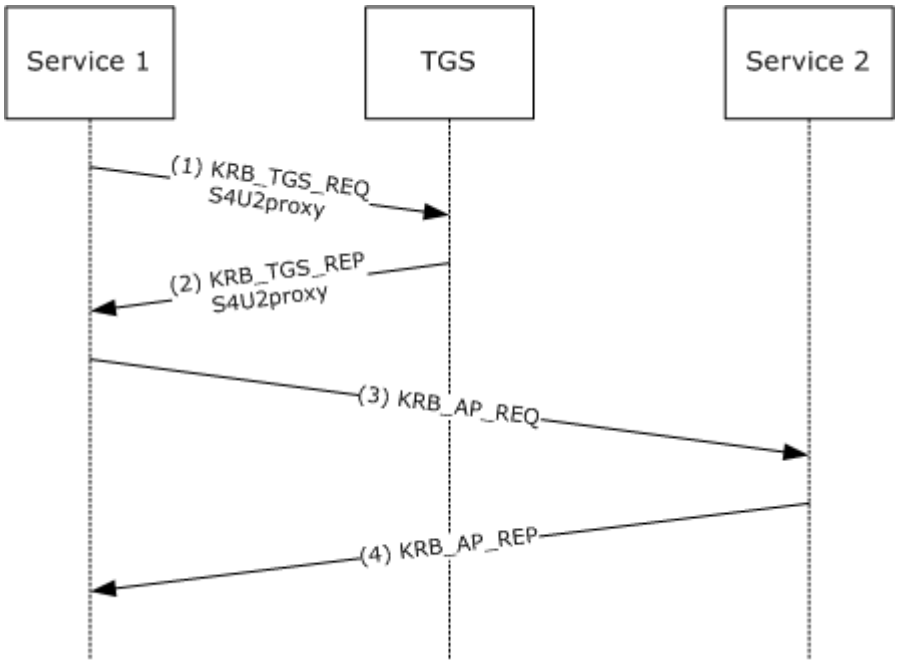

Figure 6: Service-for-User-to-Proxy (S4U2proxy) Example

In step 1, Service 1 is attempting to obtain a service ticket to Service 2 on behalf of the user. Service 1 sends the KRB_TGS_REQ with the user's service ticket to Service 1 as an additional-ticket in the request. Service 1 also sets the CNAME-IN-ADDL-TKT flag in the kdc-options in the request.

The TGS makes sure the forwardable flag is set in the additional-ticket and uses its local policy to determine if Service 1 is allowed to obtain a service ticket on behalf of a user to Service 2. If these conditions are met, the TGS crafts the KRB_TGS_REP to return a service ticket. This response will contain the cname of the user that is taken from the additional-ticket, instead of using the cname of Service 1. The forwardable flag will be set in the service ticket. The authorization data in the service ticket will be copied from the service ticket passed to the TGS in the additional-tickets field.

In step 3, Service 1 uses the service ticket from step 2 to contact Service 2. The service ticket will contain the user's name as the **cname** field. Step 4 shows the KRB_AS_REP from the KRB_AS_REQ.<10>

5 Security Considerations

The S4U2self extension allows a service to obtain a service ticket to itself on behalf of a user. This extension is used to obtain authorization data for the user to allow the service to make access control decisions on the local system. As such, the service should be certain to adequately authenticate the user before obtaining the service ticket.

The S4U2proxy extension allows a service to obtain a service ticket to a second service on behalf of a user. When combined with S4U2self, this allows the first service to impersonate any user principal while accessing the second service. This gives any service allowed access to the S4U2proxy extension a degree of power similar to that of the KDC itself. This implies that each of the services allowed to invoke this extension should be protected nearly as strongly as the KDC and should be limited to those that the customer knows to have correct behavior.

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows Server 2003
- Windows Vista
- Windows Server 2008

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Section 1.1: **Windows Key Distribution Centers (KDCs)** are integrated into the domain controller role of Windows 2000 Server and Windows Server 2003.

<2> Section 1.3.3: The KDC keeps for each service a list of the other services to which the first service is allowed to request an S4U2proxy ticket. In this way, Service 1 is not allowed to request a ticket to Service N. This is known as **constrained delegation**.

<3> Section 1.3.3: Service 2 can discover that the service ticket did not originate from the client by looking at the S4U_DELEGATION_INFO PAC buffer, as specified in section [MS-PAC] section 2.2.9.

<4> Section 1.3.3: The S4U protocol extensions are only supported on Windows Server 2003 and Windows Vista.

<5> Section 1.5: The PAC_CLIENT_INFO and S4U_DELGATION_INFO PAC buffers (as specified in [MS-PAC] sections 2.2.7 and 2.2.9 respectively) also contain information specific to these S4U extensions. Windows 2000 Server and Windows XP will simply ignore the S4U_DELEGATION_INFO PAC buffer if it is present. Windows 2000 Server and Windows XP can process the PAC_CLIENT_INFO. For more information, see section 3.1.4.1.

<6> Section 2.2.2: Windows Vista will send the PA-S4U-X509-USER padata type alone if the user's certificate is available. If the user's certificate is not available, it will send both the PA-S4U-X509-USER padata type and the PA-FOR-USER padata type. When the PA-S4U-X509-USER padata type is used without the user's certificate, the certificate field is not present.

Windows **domain** controllers starting with Windows Server 2008, will first look for the information in the PA-S4U-X509-USER padata type if present; if it is not present Windows Server 2008 will look at the PA-FOR-USER padata type.

<7> Section 3.1.4.1: The Forwardable flag is set by the service's TGS if the **Active Directory** account for this service has the Trusted-to-Authenticate-for-Delegation flag set by the domain administrators. The UserAccountControl flag for this feature is 0x1000000.

The S4U extension is only implemented in KDCs in Windows Server 2003. Therefore all the KDCs/TGSs involved in this process must be Windows Server 2003.

The following check is made to prevent possible misuse of the S4U2self referral TGT.

- Referral TGTs, as specified in [RFC4120] Section 1.2, returned as a result of a KRB_TGS_REQ contain a PAC_CLIENT_INFO structure, as specified in [MS-PAC] section 2.2.7, which contains a client name in the Name field. The client's realm is not included in this field.

- Referral TGTs returned as a result of a KRB_TGS_REQ with S4U2self extensions contain a PAC_CLIENT_INFO structure with a "client name@client realm" in the Name field.
- In all TGTs, the **KERB_VALIDATION_INFO** structure, as specified in section 2.2.5 of [MS-PAC], contains the client name in the **EffectiveName** field. The client's realm is not included in this field.

Application servers always verify that the **EffectiveName** in the **KERB_VALIDATION_INFO** structure matches the Name in the PAC_CLIENT_INFO buffer to make sure they match. If not, the KRB_AP_REQ is rejected. Mismatched names may indicate that Service 1 used the referral TGT to attempt to retrieve a service ticket to Service 2 with the user's privilege attribute certificate (PAC), but with the cname and crealm set to that of Service 1.

<8> Section 3.1.4.1.2: S4U2self requests using the user's certificate are supported only on Windows Vista and later. Windows Vista supports a number of mapping options:

- Based on the UPN contained in the **SubjectAltName** (SAN) field of the certificate
- Based on the issuer name and subject name combination
- Based on the subject name alone
- Based on subject name and serial number in the certificate
- Based on the Subject Key Identifier
- Based on the SHA1 hash of the public key
- Based on the user's e-mail name as defined in [RFC822]

The algorithm used to locate the user account is as follows:

- If the certificate contains SAN/UPN extension, KDC will use that to map the client. If the certificate contains an SAN/UPN extension and no user object is found based on the UPN, the authentication fails.
- If there is no UPN in the certificate, the KDC constructs the "X509:<I><S>" string (where "I" is the Issuer name and "S" is the subject name in the certificate) to look up.
- If there is no UPN in the certificate and no user object is located in the steps above, the client account is looked up based on the DN name of the subject; the KDC constructs the "X509:<S>" string (where "S" is the DN name of the subject in the certificate) to look up.
- If there is no UPN in the certificate and no user object is located in the steps above, the KDC uses the subject and serial number to construct the "X509:<S><SR>" string (where "S" is the subject name and "SR" is the serial number from the certificate) to look up.
- If there is no UPN in the certificate and no user object is located, and the client certificate contains a Subject Key Identifier, the KDC constructs the "X509:<SKI>" string (where "SKI" is the Subject Key Identifier) to look up.
- If there is no UPN in the certificate and no user object is located in the steps above, the KDC constructs the "X509:<SHA1-PUKEY>" string to look up.
- If there is no UPN in the certificate and no user object is located in the steps above, the client account is looked up based on the SAN/822name, the KDC constructs the "X509:<RFC822>" string to look up.

<9> Section 3.1.4.2: The local policy to determine if the service can obtain a service ticket to a proxy service on behalf of the user is the Allowed-to-Delegate-To information in the original service's account object in the Active Directory (AD) store. This multivalued field contains the list of services to which this service account can proxy (that is, obtain service tickets on behalf of the user).

<10> Section 4.3: The TGS checks the service's account in Active Directory (AD) for the Allowed-to-Authenticate-for-Delegation setting. The UserAccountControl flag for this feature is 0x1000000.

7 Index

V

W