[MS-SSDPS2]:
Secure Store Database
Version 2 Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release (beta) version of this technology. This Open Specification is final documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release (beta) versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Revision History	Revision Class	Comments
01/20/2012	0.1	New	Released new document.
04/11/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies the Secure Store Database Protocol Specification. This protocol specifies an interface for protocol clients to store and retrieve credential and related information typically used to authenticate to line-of-business (LOB) systems.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

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

checksum
credential
GUID
public key
salt

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

Advanced Encryption Standard (AES)
back-end database server
base64 encoding
claim
claim issuer
claim type
claim value
empty GUID
group target application
individual target application
line-of-business (LOB) system
master secret key
Secure Store Service (SSS)
Secure Store Service (SSS) action
Secure Store Service (SSS) store
Secure Store Service (SSS) ticket
Secure Store Service (SSS) user
security principal
session key
SHA-256
stored procedure
target application
target application field
Transact-Structured Query Language (T-SQL)
Uniform Resource Locator (URL)
XML namespace
XML schema definition (XSD)

The following terms are specific to this document:

Secure Store Service (SSS) audit entry: A record that stores information about a Secure Store Service (SSS) action, including when it was performed, whether it succeeded, why it

failed if it didn't succeed, the SSS user who performed it, and optionally the SSS user on whose behalf it was performed.

Secure Store Service (SSS) partition: A group of target applications and credentials that are identified by a GUID and are contained in a single Secure Store Service (SSS) store.

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

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

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.

[Iseminger] Microsoft Corporation, "SQL Server 2000 Architecture and XML/Internet Support", Volume 1 of Microsoft SQL Server 2000 Reference Library, Microsoft Press, 2001, ISBN 0-7356-1280-3, http://www.microsoft.com/mspress/books/5001.aspx

[MSDN-TSQL-Ref] Microsoft Corporation, "Transact-SQL Reference", http://msdn.microsoft.com/en-us/library/ms189826(SQL.90).aspx

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

[MS-NRTP] Microsoft Corporation, ".NET Remoting: Core Protocol Specification".

[MS-TDS] Microsoft Corporation, "Tabular Data Stream Protocol Specification".

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

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XMLSCHEMA1] Thompson, H.S., Ed., Beech, D., Ed., Maloney, M., Ed., and Mendelsohn, N., Ed., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.2.2 Informative References

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

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary".

1.3 Overview

Enterprises have a variety of data stored in various **line-of-business (LOB) systems**. Typically, each of these systems has its own security model where the same user is represented by a unique system-specific **security principal**. A set of **credentials** is required as input before a user is allowed to access to the LOB.

It is common for modern business applications to deliver functionality that requires data to be manipulated in more than a single software system concurrently. As a result, the user experience can be cumbersome, as each time a particular system is accessed, the user has to authenticate to it by providing his or her credentials for that particular system. It also burdens the user by requiring him or her to maintain different credentials for each system.

To improve the user experience and address the preceding issue, it is possible to store descriptions of LOB systems as **target applications** as well as the actual credentials for each user of each LOB system. Then an integrated application that spans multiple systems can programmatically obtain the credentials of the current **Secure Store Service (SSS) user** from the store and authenticate without prompting the SSS user for credentials each time a particular LOB system demands authentication. The SSS user never needs to authenticate more than once as long as the stored credentials remain valid with respect to the LOB.

This protocol allows multiple protocol clients sharing a single **SSS** configuration to communicate with a single protocol server.

This protocol allows protocol clients to create, read, update and delete target application definitions in a **back-end database server**. It allows for partitioning of the **SSS store** such that a client application can use the protocol client to store multiple target applications that are isolated from target applications of the other client applications, provided each client application is associated with a unique identifier that identifies a **SSS partition**. It also allows a protocol client to create, read, update and delete the credentials associated with each target application and to encrypt this information to keep it secure. Additionally, it allows a protocol client to create an **SSS ticket** that encapsulates the identity of a user of the protocol client into a token that may be later redeemed by a different user of the protocol client to retrieve credentials on behalf of the initial user. Finally it allows the maintenance of an audit trail of the operations performed by protocol clients.

The information handled and returned by the protocol client can contain highly sensitive information so consumers of the protocol client need to secure this data appropriately.

1.4 Relationship to Other Protocols

The following diagram shows the transport stack that the protocol uses:

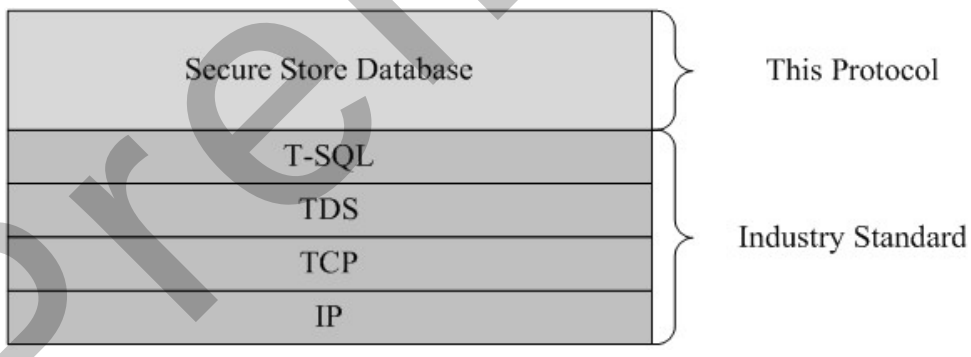


Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

The operations described by the protocol operate between a client and a back-end database server on which the databases are stored. The client is expected to know the location and connection information for the database.

This protocol requires that the protocol client has appropriate permissions to call the **stored procedures** stored on the back-end database server.

1.6 Applicability Statement

This protocol is intended for use by protocol client and protocol server that are both connected by high-bandwidth, low-latency network connections.

The information handled and returned by the protocol client can contain highly sensitive information, so the protocol client needs to be consumed in an environment that is appropriately secured.

1.7 Versioning and Capability Negotiation

Security and Authentication Methods: This protocol supports the SSPI and SQL Authentication with the protocol server role specified in [MS-TDS].

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The Tabular Data Stream Protocol, as specified in [MS-TDS], MUST be the transport protocol used to call the stored procedures, return codes and result sets.

2.2 Common Data Types

The following sections define the common data types that are used in this protocol.

2.2.1 Simple Data Types and Enumerations

2.2.1.1 SecureStoreCredentialType

This section specifies an enumeration, as specified in [MS-NRTP] section 2.2.5, passed between the protocol client and protocol server.

This enumeration is used to specify the type of a credential.

```
namespace Microsoft.BusinessData.Infrastructure.SecureStore
{
    enum SecureStoreCredentialType
    {
        UserName,
        Password,
        Pin,
        Key,
        Generic,
        WindowsUserName,
        WindowsPassword,
        Certificate,
        CertificatePassword
    }
}
```

UserName: A user name credential type.

Password: A password credential type.

Pin: A personal identification number credential type.

Key: An authentication key credential type.

Generic: Any generic credential type.

WindowsUserName: A Windows user name credential type.

WindowsPassword: A Windows password credential type.

Certificate: A certificate credential type.

CertificatePassword: A certificate password credential type.

2.2.2 Classes

This section specifies Classes, as specified in [MS-NRTP] section 2.2.5, passed between the protocol client and protocol server.

2.2.2.1 SecureStoreServiceClaim

This class contains **claim (2)** of an SSS user.

```
namespace Microsoft.Office.SecureStoreService.Server
{
   class SecureStoreServiceClaim
   {
      String claimType;
      String claimIssuer;
      String claimValue;
   }
}
```

claimType: Contains the **claim type**. The minimum length of this sting is one and the maximum length of this string is 2084.

claimIssuer: Contains the **claim issuer**. The minimum length of this sting is one and the maximum length of this string is 2084.

claimValue: Contains the **claim value**. The minimum length of this sting is one and the maximum length of this string is 2048.

2.2.2.2 SerializableSecureStoreCredential

This class contains the credential for an SSS user.

```
namespace Microsoft.Office.SecureStoreService.Server
{
   class SerializableSecureStoreCredential
   {
      SecureStoreCredentialType credentialType;
      byte[] credential;
   }
}
```

credentialType: A SecureStoreCredentialType representing the type of the credential.

credential: Contains the credential of an SSS user. The minimum length of this array is one and the maximum length of this array is 2084.

2.2.2.3 List<T>

This is a Generic Type, as specified in [MS-NRTP] section 1.1, class containing a collection of items.

```
namespace System.Collections.Generic
{
   class List<T>
   {
      private T[] _items;
```

```
        private int _size;
        private int _version;
    }
}
```

_items: Contains an array of the Generic Argument as specified in [MS-NRTP] section 1.1.

_size: Contains the number of items in the array _items**.**

_version: A version number for an instance of this Class.

2.2.2.4 SecureStoreTicket

This class represents an SSS ticket.

```
namespace Microsoft.Office.SecureStoreService.Server
{
    class SecureStoreTicket
    {
        byte[] ticket;
        Guid partitionId;
        Microsoft.Office.SecureStoreService.Server.SecureStoreServiceClaim identityClaim;
        System.Collections.Generic.IList<
Microsoft.Office.SecureStoreService.Server.SecureStoreServiceClaim> claims;
    }
}
```

ticket: A 32-byte random number representing an SSS ticket.

partitionId: The identifier for the SSS partition for which this SSS ticket is issued. This value MUST contain a valid **GUID**, as specified in [MS-DTYP] section 2.3.2.2.

identityClaim: The claim (2) that uniquely identifies an SSS user for whom this SSS ticket is issued.

claims: All the claims (2) of the SSS user for whom this SSS ticket is issued.

2.2.2.5 SecureStoreDbCredentials

This class contains the credentials for an SSS user with a set of claims (2).

```
namespace Microsoft.Office.SecureStoreService.Server
{
    class SecureStoreDbCredentials
    {

System.Collections.Generic.List<Microsoft.Office.SecureStoreService.Server.SecureStoreService
Claim> claims;

System.Collections.Generic.List<Microsoft.Office.SecureStoreService.Server.SerializableSecure
StoreCredential> credentials;
    }
}
```

claims: Contains the claims (2) of the SSS users that can retrieve the credentials.

credentials: The credentials for the SSS users.

2.2.3 Common Fields

2.2.3.1 ApplicationType

ApplicationType: an int that is NOT NULL of the type target application, and MUST be a value listed in the following table.

Value	Description
0x00	An **individual target application** that stores credentials for individual SSS users. The credentials are meant to be used by applications that perform no additional authorization against the data stored or retrieved from the system to which the credentials are used to authenticate.
0x01	A **group target application** that stores credentials for a group of SSS users. The credentials are meant to be used by applications that perform no additional authorization against the data stored or retrieved from the system to which the credentials are used to authenticate.
0x02	An individual target application that stores credentials for individual SSS users. The credentials are meant to be used by applications that perform no additional authorization against the data stored or retrieved from the system to which the credentials are used to authenticate. This target application also allows a user to create an SSS ticket. When using an SSS ticket, a user "A" can retrieve the credentials for another user "B", by using an SSS ticket created by user "B".
0x03	A group target application that stores credentials for a group of SSS users. The credentials are meant to be used by applications that perform no additional authorization against the data stored or retrieved from the system to which the credentials are used to authenticate. This target application also allows a user to create an SSS ticket. When using an SSS ticket, a user "A" can retrieve the credentials for another user "B", by using an SSS ticket created by user "B".
0x04	An individual target application that stores credentials for individual SSS users. The credentials are meant to be used by applications that perform additional, implementation specific, authorization against sensitive data stored or retrieved from the system to which the credentials are used to authenticate.
0x05	A group target application that stores credentials for a group of SSS users. The credentials are meant to be used by applications that perform additional, implementation specific, authorization against sensitive data stored or retrieved from the system to which the credentials are used to authenticate.

2.2.3.2 CredentialType

CredentialType: an int that is NOT NULL of the type of a credential that can be stored for a target application. This value is supplied by the user when creating a new target application. It MUST be a value listed in the following table.

Value	Description
0	A user name credential.
1	A password credential.
2	A personal identification number (PIN) credential.
3	An authentication key credential.
4	Any generic credential.
5	A Windows user name credential.
6	A Windows password credential.

2.2.3.3 StatusType

StatusType: an int that is NOT NULL that is a value utilized to track the status of implementation specific long running operations initiated by the protocol client. The value MUST be listed in the following table.

Value	Description
0	The implementation specific long running operation is either not started or is complete.
1	The implementation specific long running operation is executing.

2.2.3.4 PartitionId

PartitionId: unique identifier that is NOT NULL that is the identifier for the SSS partition.

2.2.3.5 ApplicationId

ApplicationId: unique identifier that is NOT NULL that is the identifier for a target application.

2.2.3.6 ActionType

ActionType: an int that is NOT NULL that is a value that denotes the action type of **SSS audit entry**. This value MUST be listed in the following table.

Value	Description
101	A target application has been created.
103	A target application has been updated.
105	A target application has been deleted.
107	The user claim (2) for an individual target application has been retrieved.
109	The group claims (2) for a group target application has been retrieved.
111	The claims (2) for the group of SSS users that are administrators for a target application have been retrieved.
113	The claims (2) for ticket redeemers for a target application have been retrieved.

Value	Description
115	The definition for a target application has been retrieved.
117	The fields for a target application have been retrieved.
119	The definitions for all target applications have been retrieved.
121	The credentials for an SSS user have been set.
123	The credentials for a group target application have been set.
125	The credentials for an SSS user for a target application have been deleted.
127	The credentials for an SSS user for all target applications have been deleted.
128	An SSS ticket was issued.
130	An SSS ticket was redeemed.
132	The credentials for an SSS user have been retrieved.
134	The restricted credentials for an SSS user have been retrieved.
136	A SSS user has set his/her own credentials in the SSS store.

2.2.3.7 PurgeAuditDays

PurgeAuditDays: an int that is NOT NULL that is a value that denotes how long an SSS audit entry is preserved in the SSS store, measured in days. It is part of the SSS configuration.

2.2.3.8 EnableAudit

EnableAudit: bit NOT NULL. A flag specifies whether the SSS audit entry is stored in SSS store when **proc_sss_InsertAudit** is called. The value MUST be in the following table.

Value	Description
0	The SSS audit entry is not stored in SSS store when **proc_sss_InsertAudit** is called.
1	The SSS audit entry is stored in SSS store when **proc_sss_InsertAudit** is called.

2.2.3.9 ClaimType

ClaimType: string NOT NULL. A string value that contains the claim type.

2.2.3.10 ClaimIssuer

ClaimIssuer: string NOT NULL. A string value that contains the claim issuer.

2.2.3.11 ClaimValue

ClaimValue: string NOT NULL. A string value that contains the claim value.

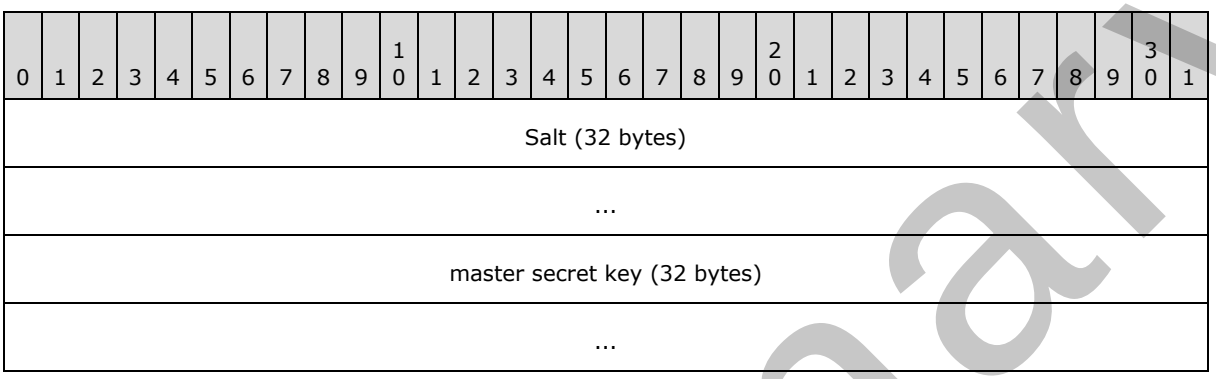
2.2.4 Bit Fields and Flag Structures

No common bit field or flag structures are defined in this protocol.

2.2.5 Binary Structures

2.2.5.1 Encryption Session Key Seed

The **Encryption Session Key Seed** is a sequence of bytes that will be hashed using the **SHA-256** algorithm to generate the session key that is used for encrypting the credentials and SSS ticket.

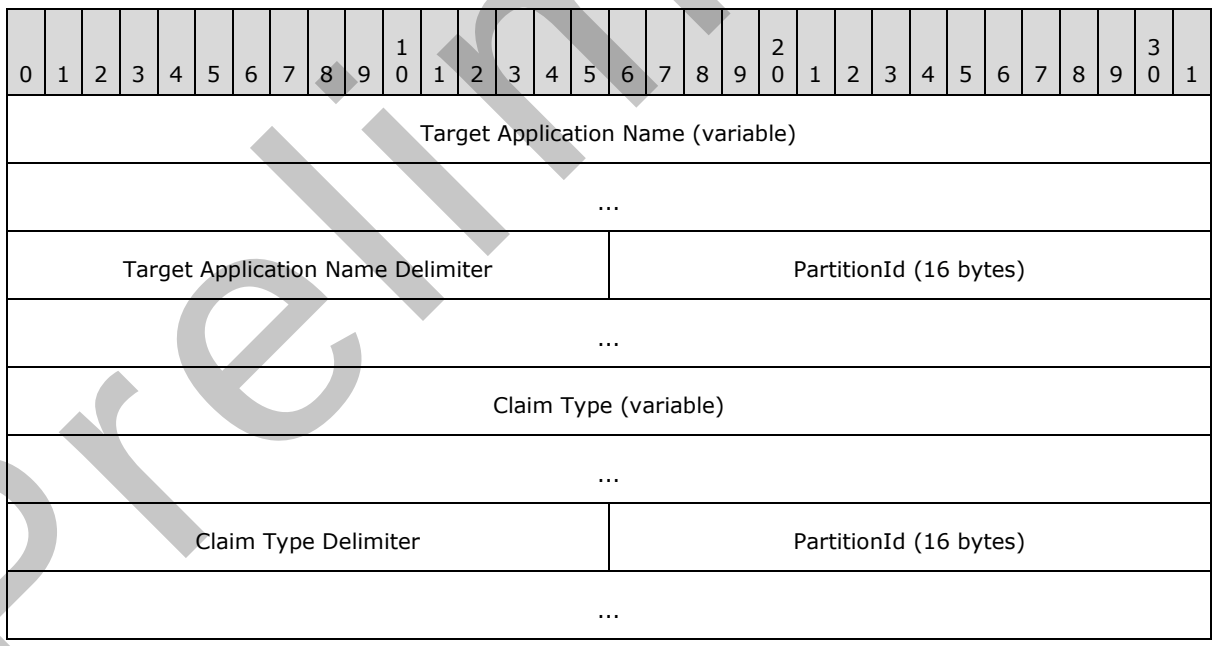


Salt (32 bytes): 32 byte **salt**.

master secret key (32 bytes): 32-byte **master secret key**.

2.2.5.2 Unencrypted claim

The sequence of bytes that make up a claim (2) prefixed with name of the target application and PartitionId.



Claim Value (variable)	
...	
Claim Value Delimiter	Claim Issuer (variable)
...	

Target Application Name (variable): The name of the target application.

Target Application Name Delimiter (2 bytes): The value of this field MUST be 0x003A.

PartitionId (16 bytes): The SSS partition for the specified target application.

Claim Type (variable): A string containing the claim type.

Claim Type Delimiter (2 bytes): The value of this field MUST be 0x001E.

Claim Value (variable): A string containing the claim value.

Claim Value Delimiter (2 bytes): The value of this field MUST be 0x001E.

Claim Issuer (variable): A string containing the claim issuer.

2.2.5.3 Unencrypted claim hash

The **Unencrypted claim hash** is a sequence of bytes that is obtained by the hashing the Unencrypted claim (2), and prefixing it with a salt.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Salt1 (32 bytes)																															
...																															
Salt2 (32 bytes)																															
...																															
Hash of Unencrypted claim (variable)																															
...																															

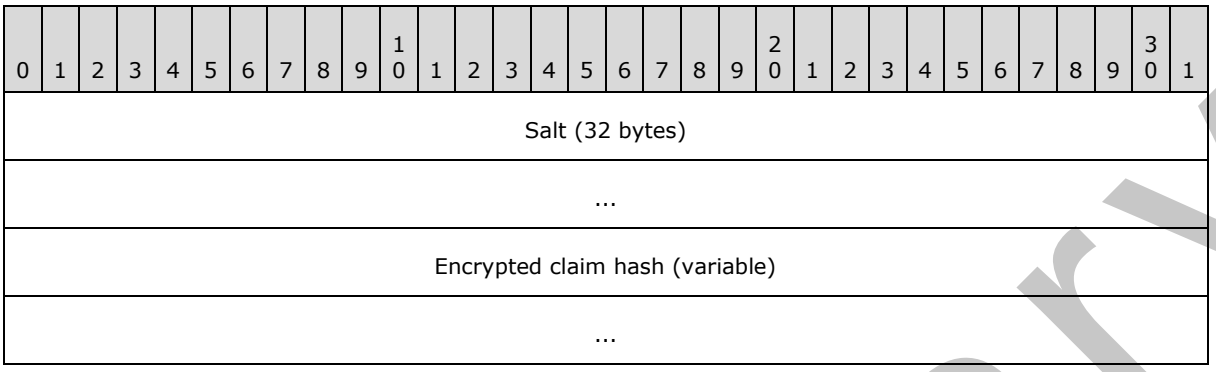
Salt1 (32 bytes): 32-byte salt.

Salt2 (32 bytes): 32-byte salt. This MUST be the same salt used to create the Encryption Session Key Seed.

Hash of Unencrypted claim (variable): The sequence of bytes obtained by hashing Unencrypted claim.

2.2.5.4 Encrypted claim hash

The **Encrypted claim hash** is a sequence of bytes obtained after encrypting Unencrypted claim (2) hash and prefixing with a salt.

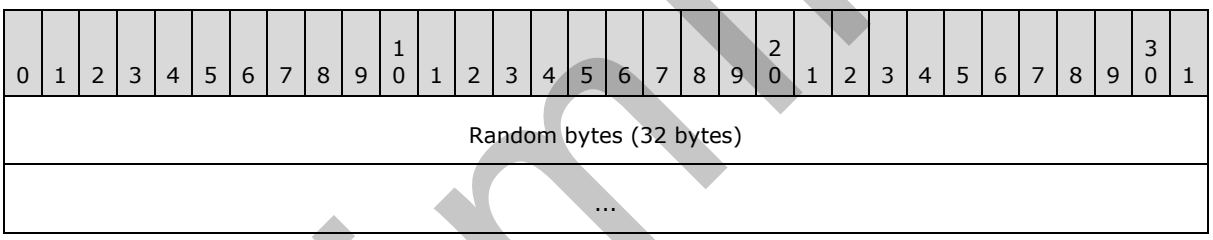


Salt (32 bytes): 32-byte salt. This MUST be the same salt used to create the Encryption Session Key Seed.

Encrypted claim hash (variable): The sequence of bytes obtained by encrypting Unencrypted claim hash.

2.2.5.5 Random Ticket

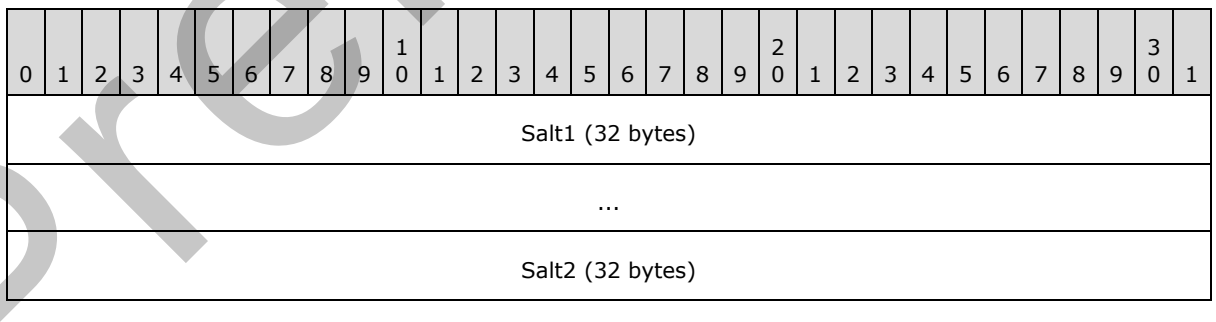
The **Random Ticket** is a sequence of 32 cryptographically random bytes.



Random bytes (32 bytes): 32 cryptographically random bytes.

2.2.5.6 Unencrypted Ticket

The **Unencrypted Ticket** is a sequence of bytes that make up the plain text contents of an SSS ticket.



...
Binary serialized SecureStoreTicket (variable)
...

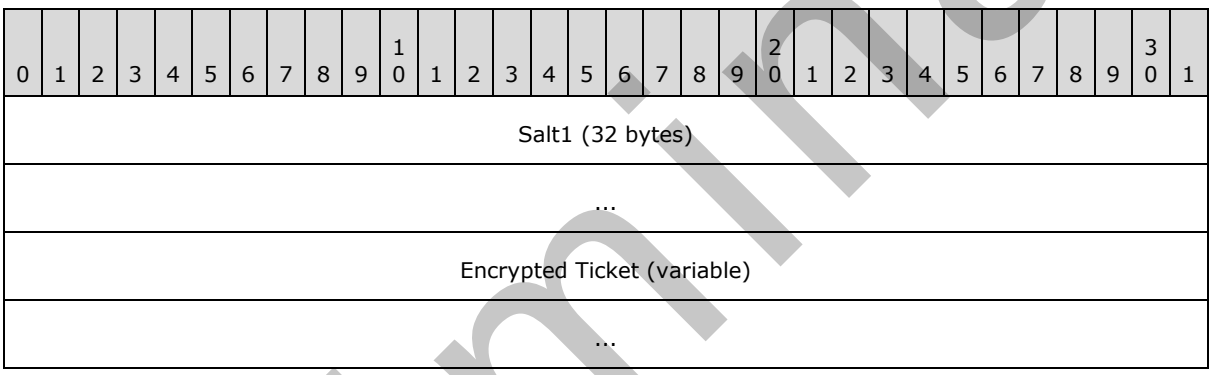
Salt1 (32 bytes): 32-byte salt.

Salt2 (32 bytes): 32-byte salt. This MUST be the same salt used to create the Encryption Session Key Seed.

Binary serialized SecureStoreTicket(variable): The binary serialized format of the SecureStoreTicket, as specified in [MS-NRTP] section 3.1.5.1.6.

2.2.5.7 Final SSS Ticket

The **Final SSS Ticket** is a sequence of bytes that make up the final SSS ticket for transmission to other structures which can later redeem it for credentials. This is obtained by **base64 encoding** after encrypting the Unencrypted Ticket and prefixing it with the salt used to create the Encryption Session Key Seed.

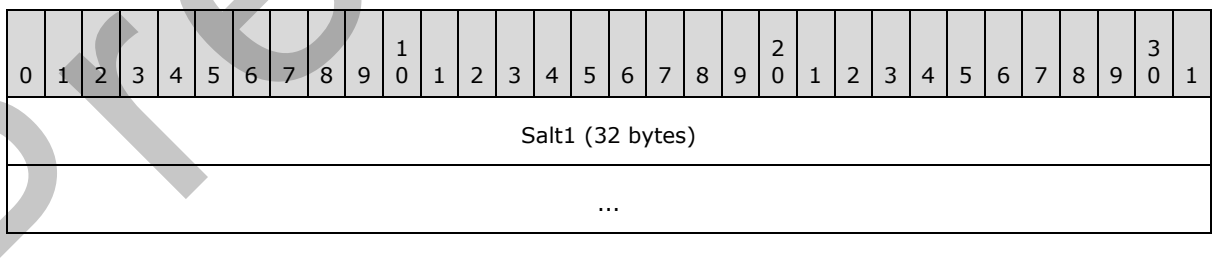


Salt (32 bytes): 32 byte salt. This MUST be the same salt used to create the Encryption Session Key Seed.

Encrypted Ticket (variable): The sequence of bytes obtained by encrypting an Unencrypted Ticket.

2.2.5.8 Unencrypted Credentials

The **Unencrypted Credentials** is a sequence of bytes that makes the Unencrypted Credentials.



Salt2 (32 bytes)
...
Binary serialized SecureStoreDbCredentials (variable)
...

Salt1 (32 bytes): 32-byte salt.

Salt2 (32 bytes): 32-byte salt. This MUST be the same salt used to create the Encryption Session Key Seed.

Binary serialized SecureStoreDbCredentials(variable): The binary serialized format of SecureStoreDbCredentials, as specified in [MS-NRTP] section 3.1.5.1.6.

2.2.5.9 Salted Encrypted Credentials

The **Salted Encrypted Credentials** is a sequence of bytes that makes the Salted Encrypted Credentials.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Salt (32 bytes)																															
...																															
Encrypted Credentials (variable)																															
...																															

Salt (32 bytes): 32-byte salt. This MUST be the same salt used to create the Encryption Session Key Seed.

Encrypted Credentials (variable): The sequence of bytes obtained by encrypting Unencrypted Credentials.

2.2.6 Result Sets

2.2.6.1 Paged Credentials Result Set

The **Paged Credentials Result Set** result set contains information about credentials stored in SSS store. Each row in the result set contains credentials for specific claim type, claim value and target application.

```
CredentialsId bigint,
ApplicationId uniqueidentifier,
IdentityClaimTypeId uniqueidentifier,
IdentityClaimValueHash varbinary(32),
```

```
Credentials varbinary(max),
```

CredentialsId: The identifier for each record in the SSSCredentials table. The value MUST NOT be NULL.

ApplicationId: The GUID for the target application to which the credentials belong. The value MUST NOT be NULL.

IdentityClaimTypeId: The identifier of the claim type. The value MUST NOT be NULL.

IdentityClaimValueHash: The SHA-256 hash of the claim value to which the credentials belong. This value MUST be set to 0x00 if the ApplicationType of the target application is equal to 0x01, 0x03 or 0x05. The value MUST NOT be NULL.

Credentials: Encrypted credentials stored in the SSS store. The value MUST NOT be NULL.

2.2.6.2 Application Administration Claims Result Set

The **Application Administration Claims Result Set** result set contains the set of claims (2) that specify the group of SSS users that are administrators of a target application. Each row in the result set contains one claim (2).

```
ClaimIssuer nvarchar(2084),
ClaimType nvarchar(2084),
ClaimValue nvarchar(2048),
```

ClaimIssuer: The claim issuer for the claim (2).

ClaimType: The claim type for the claim (2).

ClaimValue: The claim value for the claim (2).

2.2.6.3 Application Group Claims Result Set

The **Application Group Claims Result Set** result set contains the set of claims (2) that specify a group of related SSS users for a target application. Each row in the result set contains one claim (2). The group has a particular semantic associated with it, depending upon the context in which the result set is returned. For example, one semantic is the group of users that can reserve SSS tickets for a target application.

```
ClaimIssuer nvarchar(2084),
ClaimType nvarchar(2084),
ClaimValue nvarchar(2048),
ClaimValueHash varbinary(2048),
```

ClaimIssuer: The claim issuer for the claim (2).

ClaimType: The claim type for the claim (2).

ClaimValue: The claim value for the claim (2).

ClaimValueHash: The encrypted SHA-256 hash of the claim value. The value MUST be Encrypted claim hash.

2.2.6.4 Application Fields Result Set

The **Application Fields Result Set** result set contains the **target application field** information for a specific target application. Each row in the result set contains the information about a target application field for the target application. This result set MUST contain at least 1 row. The maximum number of rows in result set is 10

```
FieldId tinyint,
IsMasked bit,
CredentialType int,
FieldName nvarchar(256),
```

FieldId: An identifier for the target application field. This value MUST NOT be NULL.

IsMasked: A flag representing whether the target application field needs to be masked, when displayed in an implementation specific user interface. This value MUST NOT be NULL.

CredentialType: The type of the credential that this target application field will contain. This MUST be a CredentialType. This value MUST NOT be NULL.

FieldName: The name of the target application field. This value MUST NOT be NULL.

2.2.6.5 Application Information Result Set

The **Application Information Result Set** result set contains the information about a target application.

```
ApplicationId uniqueidentifier,
ApplicationName nvarchar(256),
FriendlyName nvarchar(256),
ApplicationType tinyint,
TicketTimeout int,
ContactEmail nvarchar(128),
CredentialManagementUrl nvarchar(2084),
```

ApplicationId: The GUID for the target application to which the credentials belong. The value MUST NOT be NULL.

ApplicationName: The name of the target application. The value MUST NOT be NULL.

FriendlyName: A descriptive name for the target application.

ApplicationType: The type of the target application. The value MUST be an ApplicationType.

TicketTimeout: The validity in minutes for SSS tickets for the target application.

ContactEmail: The e-mail address of an administrator who owns the administration responsibilities for the target application.

CredentialManagementUrl: The **URL** for a Web page where SSS users can set their credentials for the specified target application.

2.2.6.6 Configuration Result Set

The **Configuration Result Set** result set contains information about the protocol server configuration. The result set MUST have one row.

```
PurgeAuditDays int,
EnableAudit bit,
Version datetime,
```

PurgeAuditDays: An integer value that denotes how long an SSS audit entry is preserved in the SSS store, measured in days.

EnableAudit: A flag representing whether the SSS audit entry is stored in SSS store when **proc_sss_InsertAudit** is called. The value MUST be EnableAudit.

Version: The timestamp indicating the date and time at which the SSS configuration was last stored or updated in the SSS store.

2.2.6.7 Credentials Result Set

The **Credentials Result Set** result set contains information about credentials and target application.

```
ApplicationType int,
Credentials varbinary(max),
```

ApplicationType: The type of the target application. The value MUST be an ApplicationType.

Credentials: The encrypted credentials for the target application for the specific claim (2).

2.2.6.8 Paged Group Claims Result Set

The **Paged Group Claims Result Set** result set contains the claims (2) information for the users who can retrieve credentials for a group target application. Each row in the result set contains claims (2) information for a specific user and the identifier for the target application.

```
ClaimsId bigint,
ApplicationId uniqueidentifier,
ClaimTypeId uniqueidentifier,
ClaimValue nvarchar(2048),
ClaimValueHash varbinary(2048),
```

ClaimsId: The identifier for each record in the SSSApplicationGroupClaim table. The value MUST NOT be NULL.

ApplicationId: The GUID for the target application to which the credentials belong. The value MUST NOT be NULL.

ClaimTypeId: The identifier of the claim type. The value MUST NOT be NULL.

ClaimValue: The claim value of the specific user. The value MUST be ClaimValue.

ClaimValueHash: The encrypted SHA-256 hash of the claim value. The value MUST be Encrypted claim hash.

2.2.6.9 State Result Set

The **State Result Set** result set contains information about the state of a long running implementation specific operation, and the implementation specific owner of that operation. The result set MUST have exactly one row.

```
STATUS int,
OWNER uniqueidentifier,
ACTION int,
```

STATUS: The execution state of the implementation specific operation. The value MUST be StatusType.

OWNER: The identifier of the implementation specific owner of the implementation specific long running operation. The value MUST NOT be NULL.

ACTION: This MUST be 0.

2.2.6.10 Servers Key Exchange Result Set

The **Servers Key Exchange Result Set** result set contains the versioning information of the **public key** of the protocol clients. Each row in the result set contains information about a specific protocol client along with the encrypted master secret key and its salt.

```
SERVERID uniqueidentifier,
PUBLICKEY varbinary(2048),
SELFKEYVERSION int,
LATESTKEYVERSION int,
ENCRYPTEDKEY varbinary(256),
IV varbinary(48),
CHECKSUM varbinary(96),
```

SERVERID: Identifier of the protocol client. The value MUST NOT be NULL or **empty GUID**.

PUBLICKEY: The public key of the protocol client. The value MUST NOT be NULL.

SELFKEYVERSION: The version number of the public key of the protocol client.

LATESTKEYVERSION: The version number of the master secret key. The value MUST be 0 if the master secret key is not set.

ENCRYPTEDKEY: The 256-bit **Advanced Encryption Standard (AES)** encrypted master secret key.

IV: The salt associated with the master secret key. The value MUST NOT be NULL if LATESTKEYVERSION is not equal to 0.

CHECKSUM: The **checksum** of the master secret key. The value MUST NOT be NULL if LATESTKEYVERSION is not equal to 0.

2.2.6.11 Connection Settings Result Set

The **Connection Settings Result Set** contains information about a connection.

```
ConnectionSettingsName nvarchar(256),
```

```
Description nvarchar(1024),
Type nvarchar(256),
Target nvarchar(2084),
AuthenticationMode nvarchar(256),
SSOApplicationId nvarchar(1024),
SSOProviderImplId nvarchar(1024),
ProxyTarget nvarchar(2084),
ProxySSOApplicationId nvarchar(1024),
ParentName nvarchar(256),
Properties nvarchar(4000),
```

ConnectionSettingsName: Name of the connection. It MUST NOT be NULL.

Description: The description of the connection. It MUST NOT be NULL.

Type: The type of the connection. It MUST NOT be NULL.

Target: The URL of the connection.

AuthenticationMode: The authentication mode used by the connection.

SSOApplicationId: The target application of the connection.

SSOProviderImplId: The name of the custom implementation of the target application of the connection.

ProxyTarget: The URL of the proxy of the connection.

ProxySSOApplicationId: The target application of the proxy of the connection.

ParentName: The name of the parent connection. It MUST be NULL or MUST be a valid name of an existing connection.

Properties: Additional information of the connection.

2.2.7 Tables and Views

2.2.7.1 SSSCredentials

The **SSSCredentials** table contains information about encrypted credentials associated with a claim (2) for a target application. The **SSSCredentials** table MUST contain the following columns. The following syntax is specified in [MSDN-TSQL-Ref]:

```
CredentialsId bigint NOT NULL,
ApplicationId uniqueidentifier NOT NULL,
IdentityClaimTypeId uniqueidentifier NOT NULL,
IdentityClaimValueHash varbinary(32) NOT NULL,
Credentials varbinary(max) NOT NULL,
PartitionId uniqueidentifier NOT NULL,
```

CredentialsId: The identifier for each row in this table. It MUST be unique across all rows in the table.

ApplicationId: The identifier of the target application associated with the credentials stored in the row.

IdentityClaimTypeId: The identifier of the claim type. If the specific target application is a group target application, the value MUST be the identifier for the claim type equals "http://claimtype.securestoreservice.microsoft.com/group".

IdentityClaimValueHash: The SHA-256 hash of the claim value. The value MUST be 0x00 when the specific target application is group target application.

Credentials: Encrypted credentials associated with claim (2) for target application. The value MUST be Salted Encrypted Credentials.

PartitionId: The identifier for the specified SSS partition associated with the credentials.

2.2.7.2 SSSApplicationGroupClaim

The **SSSApplicationGroupClaim** table contains information about Claims (2) associated with a group target application. The claims (2) define the set of SSS users that can access the credentials for that group target application. The **SSSApplicationGroupClaim** table MUST contain the following columns, in **Transact-Structured Query Language (T-SQL)**.

```
ClaimsId bigint NOT NULL,
ApplicationId uniqueidentifier NOT NULL,
ClaimTypeId uniqueidentifier NOT NULL,
ClaimValue nvarchar(max) NOT NULL,
ClaimValueHash varbinary(2048) NOT NULL,
PartitionId uniqueidentifier NOT NULL,
```

ClaimsId: The identifier for each row in this table. It MUST be unique across all rows in the table.

ApplicationId: The identifier for target application.

ClaimTypeId: The identifier for claim type.

ClaimValue: The claim value.

ClaimValueHash: The encrypted SHA-256 hash of the claim value. The value MUST be Encrypted claim hash.

PartitionId: The identifier for the specified SSS partition.

2.2.7.3 SSSApplicationTicketRedeemerClaim

The **SSSApplicationTicketRedeemerClaim** table contains information about claims (2) representing the users who can redeem an SSS ticket for a target application. The **SSSApplicationTicketRedeemerClaim** table MUST contain the following columns. The following syntax is specified in [MSDN-TSQL-Ref]:

```
ClaimsId bigint NOT NULL,
ApplicationId uniqueidentifier NOT NULL,
ClaimTypeId uniqueidentifier NOT NULL,
ClaimValue nvarchar(max) NOT NULL,
ClaimValueHash varbinary(2048) NOT NULL,
PartitionId uniqueidentifier NOT NULL,
```

ClaimsId: The identifier for each row in this table. It MUST be unique across all rows in the table.

ApplicationId: The identifier for target application.

ClaimTypeId: The identifier for claim type.

ClaimValue: The claim value.

ClaimValueHash: The encrypted SHA-256 hash of the claim value. The value MUST be Encrypted claim hash.

PartitionId: The identifier for the specified SSS partition.

2.2.7.4 SSSApplicationGroupClaim_Secondary

The **SSSApplicationGroupClaim_Secondary** table contains information about claims (2) associated with a group target application. The **SSSApplicationGroupClaim_Secondary** table MUST contain the following columns. The following syntax is specified in [MSDN-TSQL-Ref]:

```
ClaimsId bigint NOT NULL,
ApplicationId uniqueidentifier NOT NULL,
ClaimTypeId uniqueidentifier NOT NULL,
ClaimValue nvarchar(max) NOT NULL,
ClaimValueHash varbinary(2048) NOT NULL,
PartitionId uniqueidentifier NOT NULL,
```

ClaimsId: The identifier for each row in this table. It MUST be unique across all rows in the table.

ApplicationId: The identifier for target application.

ClaimTypeId: The identifier for claim type.

ClaimValue: The claim value.

ClaimValueHash: The encrypted SHA-256 hash of the claim value. The value MUST be Encrypted claim hash.

PartitionId: The identifier for the specified SSS partition.

2.2.7.5 SSSApplicationTicketRedeemerClaim_Secondary

The **SSSApplicationTicketRedeemerClaim_Secondary** table contains information about Claims Information associated with a target application. The **SSSApplicationTicketRedeemerClaim_Secondary** table MUST contain the following columns. The following syntax is specified in [MSDN-TSQL-Ref]:

```
ClaimsId bigint NOT NULL,
ApplicationId uniqueidentifier NOT NULL,
ClaimTypeId uniqueidentifier NOT NULL,
ClaimValue nvarchar(max) NOT NULL,
ClaimValueHash varbinary(2048) NOT NULL,
PartitionId uniqueidentifier NOT NULL,
```

ClaimsId: The identifier for each row in this table. It MUST be unique across all rows in the table.

ApplicationId: The identifier for target application.

ClaimTypeId: The identifier for claim type.

ClaimValue: The claim value.

ClaimValueHash: The encrypted SHA-256 hash of the claim value. The value MUST be Encrypted claim hash.

PartitionId: The identifier for the specified SSS partition.

2.2.7.6 SSSCredentials_Secondary

The **SSSCredentials_Secondary** table contains information about encrypted credentials associated with a claim (2) for a target application. The **SSSCredentials_Secondary** table MUST contain the following columns. The following syntax is specified in [MSDN-TSQL-Ref]:

```
CredentialsId bigint NULL,
ApplicationId uniqueidentifier NOT NULL,
IdentityClaimTypeId uniqueidentifier NOT NULL,
IdentityClaimValue nvarchar(max) NOT NULL,
Credentials varbinary(max) NOT NULL,
PartitionId uniqueidentifier NOT NULL,
```

CredentialsId: The identifier for each row in this table. It MUST be unique across all rows in the table.

ApplicationId: The identifier of the target application.

IdentityClaimTypeId: The identifier of the claim type.

IdentityClaimValue: The SHA-256 hash of the claim value.

Credentials: Encrypted credentials associated with claim for target application. The value MUST be Salted Encrypted Credentials.

PartitionId: The identifier for the specified SSS partition.

2.2.8 XML Structures

The syntax of the definition in this section uses **XML schema definition (XSD)**, as specified in [XMLSCHEMA1] and [XMLSCHEMA2].

2.2.8.1 Namespaces

The specification defines and references various **XML namespaces** using the mechanisms specified in [XMLNS]. Although this specification associates a specific XML namespace prefix for each XML namespace that is used, the choice of any particular XML namespace prefix is implementation-specific and not significant for interoperability.

The following table shows the XML namespaces that are used by this protocol and the prefix for each namespace.

Prefix	Namespace URI	Reference
xs	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1]

2.2.8.2 Simple Types

This specification does not define any common XML schema simple type definitions.

2.2.8.3 Complex Types

This specification does not define any common XML schema complex type definitions.

2.2.8.4 Elements

This section summarizes the set of common XSD element definitions in this specification.

2.2.8.4.1 Fields Information

The following is an XML structure that stores data about target application fields.

```
<xs:element name="Fields">
 <xs:complexType>
  <xs:sequence>
   <xs:element minOccurs="1" maxOccurs="10" name="Field">
    <xs:complexType>
     <xs:attribute name="id" type="xs:short" use="required" />
     <xs:attribute name="ismasked" type="xs:byte" use="required" />
     <xs:attribute name="credentialtype" type="xs:int" use="required" />
     <xs:attribute name="name" type="xs:string" use="required" />
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Field: An element containing the properties for a target application field.

Field.id: An attribute numeric value containing the identifier for the field.

Field.ismasked: An attribute flag representing whether the field value needs to be masked, when displayed in an implementation specific user interface. The value MUST be zero or one. The value one indicates the field value needs to be masked. The value zero indicates the field value need not be masked.

Field.credentialtype: An attribute numeric value containing the type of the credential. It MUST be CredentialType.

Field.name: An attribute string containing the name of the field. The minimum length of the string is one and the maximum length of the string is 256.

2.2.8.4.2 Claims Information

The following is an XML structure that specifies a set of claims (2).

```
<xs:element name="Claims">
 <xs:complexType>
  <xs:sequence>
   <xs:element minOccurs="1" maxOccurs="unbounded" name="Claim">
    <xs:complexType>
     <xs:attribute name="claimType" type="xs:string" use="required" />
```

```
      <xs:attribute name="claimIssuer" type="xs:string" use="required" />
      <xs:attribute name="claimValue" type="xs:string" use="required" />
      <xs:attribute name="claimHash" type="xs:base64Binary" use="optional" />
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
```

Claim: An element containing a single claim (2) specifying one or more SSS users.

Claim.claimType: An attribute string that contains the claim type. The minimum length of this string is one and the maximum length of this string is 2084.

Claim.claimIssuer: An attribute string that contains the claim issuer. The minimum length of this string is one and the maximum length of this string is 2084.

Claim.claimValue: An attribute string that contains the claim value. The minimum length of this string is one and the maximum length of this string is 2048.

Claim.claimHash: An attribute containing the Encrypted claim hash from the values of the preceding claim type, claim value and claim issuer.

2.2.8.4.3 Key Exchange Information

The following is an XML structure that specifies the master secret key exchange information of one or more protocol clients.

```
<xs:element name="Parameter">
 <xs:complexType>
  <xs:sequence>
   <xs:element minOccurs="1" maxOccurs="unbounded" name="Parameters">
    <xs:complexType>
     <xs:attribute name="ServerId" type="xs:uniqueidentifier" use="required" />
     <xs:attribute name="EncryptedKey" type="xs:base64Binary" use="required" />
     <xs:attribute name="KeyVersion" type="xs:int" use="required" />
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Parameter: An element specifying information about a protocol client.

Parameter.ServerId: An attribute that specifies the identifier of the protocol client.

Parameter.EncryptedKey: An attribute containing the encrypted master secret key for the protocol client.

Parameter.KeyVersion: An attribute integer that specifies the version of the master secret key.

2.2.8.5 Attributes

This specification does not define any common XMS schema attribute definitions.

2.2.8.6 Groups

This specification does not define any common XML schema group definitions.

2.2.8.7 Attribute Groups

This specification does not define any common XML schema attribute group definitions.

3 Protocol Details

3.1 Common Details

None.

3.2 Server Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The protocol server maintains the following sets of data for this protocol within an SSS store. Data is maintained until updated or removed.

SSS configuration: A set of information that dictates the behavior of the protocol server and protocol clients. It includes information such as the number of days the SSS audit entries are preserved and a flag to indicate the auditing is enabled and timestamp indicating the version of the SSS configuration information set.

Target application definitions: A set of target applications that each consist of a unique identifier, programmatic name, descriptive name, e-mail contact, claims (2) representing the set of SSS users who can administer the target application, claims (2) representing the members of a group target application, claims (2) representing the users who can redeem an SSS ticket and a set of credential field labels and information about how they can each be displayed in a user interface.

Issued SSS tickets: A set of unexpired tokens that represent the SSS tickets issued with the date and time of issue.

Audit information: A record of what operations were executed, their results, by whom and when for auditing purposes.

Credentials: A set of credentials for a single SSS user or for a set of SSS users, for each target application. An identifier is associated with each set of credentials, the unique identifier of the owning target application and a claim (2) that specifies the security principal who owns the credentials, if the credentials are associated with an individual target application.

Master Secret Key: A secret key used by the protocol client to symmetrically encrypt and decrypt the credentials and SSS ticket to secure them, along with associated salt and implementation specific checksum. It is stored encrypted using implementation specific means.

Status Type: The owner and status of a single implementation specific long running operation initiated by the protocol client.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 proc_GetCredentialsPage

The **proc_GetCredentialsPage** stored procedure is called to retrieve the specified number of records of credentials stored in SSS store from the specified starting row number in the SSSCredentials table.

```
PROCEDURE proc_GetCredentialsPage (
@StartIndex int
,@PageSize int
);
```

@StartIndex: The starting row number from where the credentials are to be retrieved. To select which rows are to be returned, the implementation MUST assign virtual row numbers to the credentials stored in the SSSCredentials table. The first row MUST be numbered 0. Each subsequent row number MUST be the previous row's number incremented by 1. Ordering of rows MUST be by CredentialsId in the SSSCredentials table.

@PageSize: The maximum number of records of credentials to be retrieved from the specified starting row number.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a Paged Credentials Result Set

3.2.5.2 proc_sss_CreateApplication

The **proc_sss_CreateApplication** stored procedure is called to create a new target application in the SSS store. If the *@GroupClaims* is not NULL, a record for each claim (2) in *@GroupClaims* MUST be added to SSSApplicationGroupClaim table. If the *@TicketRedeemClaim* is not NULL, a record for each claim in *@TicketRedeemClaim* MUST be added to SSSApplicationTicketRedeemerClaim table.

```
PROCEDURE proc_sss_CreateApplication (
@ApplicationName nvarchar(256)
,@FriendlyName nvarchar(256)
,@PartitionId uniqueidentifier
,@ApplicationType int
,@TicketTimeout int
,@ContactEmail nvarchar(128)
,@CredentialManagementUrl nvarchar(2084)
,@FieldInfo xml
,@AdminClaims xml
,@GroupClaims xml
,@TicketRedeemClaims xml
,@CheckSum varbinary(96)
);
```

@ApplicationName: The name of the target application to be created. The value MUST NOT be NULL.

@FriendlyName: The descriptive name of the target application to be created. The value MUST NOT be NULL.

@PartitionId: The SSS partition for the target application to be created. The value MUST be a PartitionId.

@ApplicationType: The type of the target application. The value MUST be an ApplicationType.

@TicketTimeout: The validity in minutes for SSS tickets for this target application. The value MUST NOT be NULL, if the value of *@ApplicationType* is equal to 0x02 or0x03. The value MUST be set to NULL, if the value of *@ApplicationType* is not equal to 0x02 or 0x03.

@ContactEmail: The e-mail address of an administrator who owns the administration responsibilities for the specified target application.

@CredentialManagementUrl: The URL for a Web page where SSS users can set their credentials for the specified target application.

@FieldInfo: The Fields Information for the specified target application.

@AdminClaims: The Claims Information for group of SSS users who are administrators for the specified target application. The value MUST NOT be NULL. The claimHash in Claims Information MUST NOT be set.

@GroupClaims: The Claims Information of the members who has access to the credentials stored for the specified target application. The **claimHash** in Claims Information MUST be set if this value is not NULL.

The value MUST NOT be NULL if the value of *@ApplicationType* is equal to 0x01, 0x03 or 0x05. The value MUST be set to NULL if the value of *@ApplicationType* is not equal to 0x01, 0x03 or 0x05.

@TicketRedeemClaims: The Claims Information of members who has access to redeem SSS tickets for this target application. The claimHash in Claims Information MUST be set if this value is not NULL.

The value MUST NOT be NULL if the value of *@ApplicationType* is equal to 0x02 or 0x03. The value MUST be set to NULL if the value of *@ApplicationType* is not equal to 0x02 or, 0x03.

@CheckSum: The checksum of the master secret key. The value MUST NOT be Null..

Return Values: An integer which MUST be in the following table.

Value	Description
@@error	A T-SQL error code.
0x80630010	There are no Claim elements found in *@GroupClaims*.
0x80630011	There are no Claim elements found in *@TicketRedeemClaims*.
0x00000000	Successful execution. The value MUST be ignored by the protocol client.
0x806300b8	A target application with name equals *@ApplicationName* already exists in the specified *@PartitionId*.

Value	Description
0x8063000e	There are no Field elements found in *@FieldInfo*.
0x8063000f	There are no Claim elements found in *@AdminClaims*.

Result Sets: MUST NOT return any result sets.

3.2.5.3 proc_sss_DeleteAllUserCredentials

The **proc_sss_DeleteAllUserCredentials** stored procedure is called to delete all credentials for the specified SSS user for all target applications in the specified SSS partition from the SSS store. The user is uniquely identified by a claim (2) containing the specified claim type, claim issuer, claim value and SHA-256 hash of claim value. The rows that contain the credentials for the specified claim type, claim issuer, claim value for all target applications in the specified SSS partition MUST be removed from the SSSCredentials table.

```
PROCEDURE proc_sss_DeleteAllUserCredentials (
@PartitionId uniqueidentifier
,@IdentityClaimType nvarchar(2084)
,@IdentityClaimIssuer nvarchar(2084)
,@IdentityClaimValue nvarchar(2048)
,@IdentityClaimValueHash varbinary(32)
);
```

@PartitionId: The SSS partition for the credential to be deleted. The value MUST be a PartitionId.

@IdentityClaimType: The claim type for the credential to be deleted. The value MUST be a ClaimType.

@IdentityClaimIssuer: The claim issuer for the credential to be deleted. The value MUST be a ClaimIssuer.

@IdentityClaimValue: The claim value for the credential to be deleted. The value MUST be a ClaimValue.

@IdentityClaimValueHash: The SHA-256 hash of claim value. The value MUST be SHA-256 hash of *@IdentityClaimValue*.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80630012	No credentials found for the specified SSS partition (with the value specified in *@PartitionId*), claim type (with the value specified in *@IdentityClaimType*), claim issuer (with the value specified in *@IdentityClaimIssuer*), claim value (with the value specified in *@IdentityClaimValue*) and SHA-256 hash claim value (with the value specified in *@IdentityClaimValueHash*).
0x00000000	Successful execution. The value MUST be ignored by the protocol client.

Result Sets: MUST NOT return any result sets.

3.2.5.4 proc_sss_DeleteApplication

The **proc_sss_DeleteApplication** stored procedure is called to delete the specified target application from the SSS store. All the credentials associated with the specified target application MUST also be deleted from the SSSCredentials table. The rows that contain the claims (2) for this target application MUST be removed from SSSApplicationGroupClaim and SSSApplicationTicketRedeemerClaim.

```
PROCEDURE proc_sss_DeleteApplication (
@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@CurrentUserClaims xml
,@VerifyAdminClaims bit
);
```

@ApplicationName: The name of the target application to be deleted. The value MUST NOT be NULL.

@PartitionId: The SSS partition for the target application to be deleted. The value MUST be PartitionId.

@CurrentUserClaims: The claims (2) associated with user who is calling the stored procedure. The value MUST be ignored if *@VerifyAdminClaims* is not one. Otherwise, the value MUST satisfy the following conditions:

- The value MUST be Claims Information.
- The value MUST contain a claim (2) that uniquely identifies the caller. The **claimsHash** in Claims Information MUST be ignored by protocol server.

@VerifyAdminClaims: A flag which specifies to verify that *@CurrentUserClaims* is one of the administrators of the target application.

Value	Description
0	The stored procedure MUST ignore *@CurrentUserClaims*.
1	The stored procedure MUST verify that the **claimType**, **claimIssuer** and **claimValue** of any one of the claims (2) in *@CurrentUserClaims* MUST be equal to **claimType**, **claimIssuer** and **claimValue** respectively of any one of the claims (2) in the set of administrators' claims (2) associated with the specified target application.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80630490	The specified target application does not exist in the specified *@PartitionId*.
0x80630005	Access is denied because the caller is not an administrator of the specified target application. The value MUST NOT be returned when *@VerifyAdminClaims* is zero.
@@error	A T-SQL error code.
0x00000000	Successful execution. The value MUST be ignored by the protocol client.

Result Sets: MUST NOT return any result sets.

3.2.5.5 proc_sss_DeleteAuditRecords

The **proc_sss_DeleteAuditRecords** stored procedure is called to delete the audit entries from the SSS store when the difference between their creation and the current time, expressed in days is greater than or equal to the PurgeAuditDays in the SSS configuration.

```
PROCEDURE proc_sss_DeleteAuditRecords (
);
```

Return Values: An integer which MUST be in the following table.

Value	Description
0	Successful execution. The value MUST be ignored by the protocol client.
@@rowcount	The number of rows that were deleted.

Result Sets: MUST NOT return any result sets.

3.2.5.6 proc_sss_DeleteUserCredentials

The **proc_sss_DeleteUserCredentials** stored procedure is called to delete the credentials for the specified target application in the specified SSS partition, for the user identified by the specified claim type, claim issuer, claim value and SHA-256 hash of claim value from the SSS store. The rows that contains the credentials for the specified target application in the specified SSS partition, claim type, claim issuer, claim value MUST be removed from the SSSCredentials table.

```
PROCEDURE proc_sss_DeleteUserCredentials (
@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@IdentityClaimType nvarchar(2084)
,@IdentityClaimIssuer nvarchar(2084)
,@IdentityClaimValue nvarchar(2048)
,@IdentityClaimValueHash varbinary(32)
,@CurrentUserClaims xml
,@VerifyAdminClaims bit
);
```

@ApplicationName: The name of the target application.

@PartitionId: The SSS partition for the credentials to be deleted. The value MUST be a PartitionId.

@IdentityClaimType: The claim type for the credential to be deleted. The value MUST be a ClaimType.

@IdentityClaimIssuer: The claim issuer for the credential to be deleted. The value MUST be a ClaimIssuer.

@IdentityClaimValue: The claim value for the credential to be deleted. The value MUST be a ClaimValue.

@IdentityClaimValueHash: The SHA-256 hash of the claim value. The value MUST be SHA-256 hash of *@IdentityClaimValue*.

@CurrentUserClaims: The claims (2) associated with user who is calling the stored procedure. The value MUST be ignored if *@VerifyAdminClaims* is not one. Otherwise, the value MUST satisfy the following conditions:

- The value MUST be Claims Information.
- The value MUST contain a claim (2) that uniquely identifies the caller.
- The claimsHash in Claims Information MUST be ignored by protocol server.

@VerifyAdminClaims: A flag which specifies to verify that *@CurrentUserClaims* is one of the administrators of the target application. The value MUST be in the following table.

Value	Description
0	The stored procedure MUST ignore *@CurrentUserClaims*.
1	The stored procedure MUST verify that the **claimType**, **claimIssuer** and **claimValue** of one of the claims (2) in *@CurrentUserClaims* MUST be equal to **claimType**, **claimIssuer** and **claimValue** respectively of one of the claims (2) in the set of target application administrators.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80630490	The target application with the specified *@ApplicationName* was not found or specified target application is group target application.
0x80630005	The user claims (2) with the specified *@CurrentUserClaims* is not an administrator of the target application. The value MUST NOT be returned when *@VerifyAdminClaims* is zero.
0x80630001	No credentials found for the specified SSS partition (with the specified *@PartitionId*), claim type (with the specified *@IdentityClaimType*), claim issuer (with the specified *@IdentityClaimIssuer*), claim value (with the specified *@IdentityClaimValue*) and SHA-256 hash claim value (with the specified ***@****IdentityClaimValueHash*).
0x00000000	Successful execution. The value MUST be ignored by the protocol client.

Result Sets: MUST NOT return any result sets.

3.2.5.7 proc_sss_GetApplicationAdminClaims

The **proc_sss_GetApplicationAdminClaims** stored procedure is called to get the set of claims (2) that represent the group of SSS users that are administrators for the specified target application in the specified SSS partition.

```
PROCEDURE proc_sss_GetApplicationAdminClaims (
@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@CurrentUserClaims xml
,@VerifyAdminClaims bit
);
```

@ApplicationName: The name of the target application. The value MUST NOT be NULL.

@PartitionId: The SSS partition for the target application for which claims (2) are to be retrieved. The value MUST be a PartitionId.

@CurrentUserClaims: The claims (2) associated with user who is calling the stored procedure. The value MUST be ignored if *@VerifyAdminClaims* is not one. Otherwise, the value MUST satisfy the following conditions:

- The value MUST be Claims Information.
- The value MUST contain a claim (2) that uniquely identifies the caller.
- The claimsHash in Claims Information MUST be ignored by protocol server.

@VerifyAdminClaims: A flag which specifies to verify that *@CurrentUserClaims* is one of the administrators of the target application. The value MUST be in the following table.

Value	Description
0	The stored procedure MUST ignore *@CurrentUserClaims*.
1	The stored procedure MUST verify that the **claimType**, **claimIssuer** and **claimValue** of one of the claims (2) in *@CurrentUserClaims* MUST be equal to **claimType**, **claimIssuer** and **claimValue** respectively of one of the claims (2) in the set of target application administrators.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80630005	Access is denied because the caller is not an administrator of the specified target application. The value MUST NOT be returned when *@VerifyAdminClaims* is zero.
0x80630490	The specified target application does not exist in the specified *@PartitionId*.
0x00000000	Successful execution. The value MUST be ignored by the protocol client.

Result Sets:

If the return value from this stored procedure is not equal to 0x80630005, this stored procedure MUST return an Application Administration Claims Result Set

3.2.5.8 proc_sss_GetApplicationClaims

The **proc_sss_GetApplicationClaims** stored procedure is called to get the set of claims (2) that represent the SSS users that are administrators, group members or SSS users that can redeem an SSS ticket for the specified target application in the specified SSS partition.

Upon successful execution this stored procedure MUST return three result sets.

The first result set MUST contain the claims (2) that represent the group of SSS users who are administrators of the specified target application.

The second result set MUST contain the claims (2) information of the members of the group if the specified target application is a group target application. If the specified target application is not a group target application an empty result MUST be returned.

The third result set MUST contain the claims (2) information of who can redeem an SSS ticket for the specified target application. If the specified target application does not support issuing SSS ticket an empty result MUST be returned.

```
PROCEDURE proc_sss_GetApplicationClaims (
@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@CurrentUserClaims xml
,@VerifyAdminClaims bit
);
```

@ApplicationName: The name of the target application. The value MUST NOT be NULL.

@PartitionId: The SSS partition for the claims (2) information to be retrieved. The value MUST be a PartitionId.

@CurrentUserClaims: The claims (2) associated with user who is calling the stored procedure. The value MUST be ignored if *@VerifyAdminClaims* is not one. Otherwise, the value MUST satisfy the following conditions:

- The value MUST be Claims Information.
- The value MUST contain a claim (2) that uniquely identifies the caller.
- The claimsHash in Claims Information MUST be ignored by protocol server.

@VerifyAdminClaims: A flag which specifies to verify that *@CurrentUserClaims* is one of the administrators of the target application. The value MUST be in the following table.

Value	Description
0	The stored procedure MUST ignore *@CurrentUserClaims*.
1	The stored procedure MUST verify that the **claimType**, **claimIssuer** and **claimValue** of one of the claims (2) in *@CurrentUserClaims* MUST be equal to **claimType**, **claimIssuer** and **claimValue** respectively of one of the claims (2) in the set of target application administrators.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80630490	The specified target application does not exist in the specified *@PartitionId*.
0x80630005	Access is denied because the caller is not an administrator of the specified target application. The value MUST NOT be returned when *@VerifyAdminClaims* is zero.
0x00000000	Successful execution. The value MUST be ignored by the protocol client.

Result Sets: MUST NOT return any result sets.

3.2.5.9 proc_sss_GetApplicationFields

The proc_sss_GetApplicationFields stored procedure is called to get the target application fields for a specified target application.

```
PROCEDURE proc_sss_GetApplicationFields (
```

```
@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@CurrentUserClaims xml
,@UserApplication bit
,@VerifyAdminClaims bit
);
```

@ApplicationName: The name of the target application. The value MUST NOT be NULL.

@PartitionId: The SSS partition for the target application fields to be retrieved. The value MUST be a PartitionId.

@CurrentUserClaims: The claims (2) associated with user who is calling the stored procedure. The value MUST be ignored if *@VerifyAdminClaims* or *@UserApplication* is not one. Otherwise, the value MUST satisfy the following conditions:

- The value MUST be Claims Information.
- The value MUST contain a claim (2) that uniquely identifies the caller.
- The claimsHash in Claims Information MUST be ignored by protocol server.

@UserApplication: A flag to verify that the caller is a member who can retrieve the credentials for the target application, if the specified target application is a group target application. The value MUST be in the following table.

@VerifyAdminClaims: A flag to verify that the caller is one of the administrators of the specified target application. The value MUST be in the following table.

Value	Description
0	The stored procedure MUST ignore *@CurrentUserClaims*.
1	The stored procedure MUST verify that the **claimType**, **claimIssuer** and **claimValue** of one of the claims (2) in *@CurrentUserClaims* MUST be equal to **claimType**, **claimIssuer** and **claimValue** respectively of one of the claims (2) in the set of target application administrators.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80630005	Access is denied because the caller is neither an administrator of the specified target application nor a member who can retrieve the credentials for the target application if the target application is a group target application. The value MUST NOT be returned when *@VerifyAdminClaims* and *@UserApplication* is zero.
0x80630490	The specified target application does not exist in the specified *@PartitionId*.
0x00000000	Successful execution. The value MUST be ignored by the protocol client.

Result Sets:

This stored procedure MUST return a Application Fields Result Set

3.2.5.10 proc_sss_GetApplicationGroupClaims

The **proc_sss_GetApplicationGroupClaims** stored procedure is called to get the set of claims (2) that represent the group of SSS users that are group members for a specified group target application in the specified SSS partition.

```
PROCEDURE proc_sss_GetApplicationGroupClaims (
@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@CurrentUserClaims xml
,@VerifyAdminClaims bit
);
```

@ApplicationName: The name of the target application. The value MUST NOT be NULL.

@PartitionId: The SSS partition for the claims (2) information to be retrieved. The value MUST be a PartitionId.

@CurrentUserClaims: The claims (2) associated with user who is calling the stored procedure. The value MUST be ignored if *@VerifyAdminClaims* is not one. Otherwise, the value MUST satisfy the following conditions:

- The value MUST be Claims Information.
- The value MUST contain a claim (2) that uniquely identifies the caller.
- The claimsHash in Claims Information MUST be ignored by protocol server.

@VerifyAdminClaims: A flag which specifies to verify that *@CurrentUserClaims* is one of the administrators of the target application. The value MUST be in the following table.

Value	Description
0	The stored procedure MUST ignore *@CurrentUserClaims*.
1	The stored procedure MUST verify that the **claimType**, **claimIssuer** and **claimValue** of one of the claims (2) in *@CurrentUserClaims* MUST be equal to **claimType**, **claimIssuer** and **claimValue** respectively of one of the claims (2) in the set of target application administrators.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80630005	Access is denied because the caller is not an administrator of the specified target application. The value MUST NOT be returned when *@VerifyAdminClaims* is zero.
0x00000000	Successful execution. The value MUST be ignored by the protocol client.

Result Sets:

If the return value from this stored procedure is not equal to 0x80630005, this stored procedure MUST return an Application Group Claims Result Set

3.2.5.11 proc_sss_GetApplicationInfo

The proc_sss_GetApplicationInfo stored procedure is called to retrieve the information of a target application for the specified target application in the specified SSS partition.

```
PROCEDURE proc_sss_GetApplicationInfo (
@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@CurrentUserClaims xml
,@UserApplication bit
,@VerifyAdminClaims bit
);
```

@ApplicationName: The name of the target application to be retrieved. The value MUST NOT be NULL.

@PartitionId: The SSS partition for the target application information to be retrieved. The value MUST be PartitionId.

@CurrentUserClaims: The claims (2) associated with user who is calling the stored procedure. The value MUST be ignored if *@VerifyAdminClaims* or *@UserApplication* is not one. Otherwise, the value MUST satisfy the following conditions:

- The value MUST be Claims Information.
- The value MUST contain a claim (2) that uniquely identifies the caller.
- The claimsHash in Claims Information MUST be ignored by protocol server.

@UserApplication: A flag to verify that the caller is a member who can retrieve the credentials for the specified target application, if the specified target application is a group target application. The value MUST be in the following table.

Value	Description
0	The stored procedure MUST ignore *@CurrentUserClaims*.
1	The stored procedure MUST verify that one of the following conditions MUST be true: If the specified target application is a group target application, the **claimType**, **claimIssuer** and **claimValue** of one of the claims (2) in *@CurrentUserClaims* MUST be equal to **claimType**, **claimIssuer** and **claimValue** respectively of one of the claims (2) for the members who can retrieve the credentials for the group target application. OR The specified target application is not a group target application.

@VerifyAdminClaims: A flag to verify that the caller is one of the administrators of the specified target application. The value MUST be in the following table.

Value	Description
0	The stored procedure MUST ignore *@CurrentUserClaims*.
1	The stored procedure MUST verify that the **claimType**, **claimIssuer** and **claimValue** of one of the claims (2) in *@CurrentUserClaims* MUST be equal to **claimType**, **claimIssuer** and **claimValue** respectively of one of the claims (2) in the set of target application administrators.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80630005	Access is denied because the caller is neither an administrator of the specified target application nor a member who can retrieve the credentials for the target application if the target application is a group target application. The value MUST NOT be returned when *@VerifyAdminClaims* and *@UserApplication* is zero.
0x80630490	The specified target application does not exist in the specified *@PartitionId*.
0x00000000	Successful execution. The value MUST be ignored by the protocol client.

Result Sets:

If the return value from this stored procedure is not equal to 0x80630005, this stored procedure MUST return a Application Information Result Set

3.2.5.12 proc_sss_GetApplicationsInfoForPartition

The **proc_sss_GetApplicationsInfoForPartition** stored procedure is called to retrieve the information for all the target applications in the specified SSS partition. If the value of the *@VerifyAdminClaims* is equal to one, the retrieved information MUST contain only the target applications where the caller is an administrator of that target application. If the value of the *@VerifyAdminClaims* is not equal to one, the retrieved information MUST contain all target applications in the specified SSS partition.

```
PROCEDURE proc_sss_GetApplicationsInfoForPartition (
@PartitionId uniqueidentifier
,@CurrentUserClaims xml
,@VerifyAdminClaims bit
);
```

@PartitionId: The SSS partition for the target application information to be retrieved. The value MUST be a PartitionId.

@CurrentUserClaims: The claims (2) associated with user who is calling the stored procedure. The value MUST be ignored if *@VerifyAdminClaims* is not one. Otherwise, the value MUST satisfy the following conditions:

- The value MUST be Claims Information.
- The value MUST contain a claim (2) that uniquely identifies the caller.
- The claimsHash in Claims Information MUST be ignored by protocol server.

@VerifyAdminClaims: A flag which specifies to verify that *@CurrentUserClaims* is one of the administrators of the target application. The value MUST be in the following table.

Value	Description
0	The stored procedure MUST ignore *@CurrentUserClaims*.
1	The stored procedure MUST verify that the **claimType**, **claimIssuer** and **claimValue** of one of the claims (2) in *@CurrentUserClaims* MUST be equal to **claimType**, **claimIssuer** and **claimValue** respectively of one of the claims (2) in the set of target application administrators.

Return Values: An integer which MUST be zero.

Result Sets:

This stored procedure MUST return a Application Information Result Set

3.2.5.13 proc_sss_GetApplicationTicketClaims

The **proc_sss_GetApplicationTicketClaims** stored procedure is called to get the set of claims (2) that represent the group of SSS users that can redeem an SSS ticket for a specified target application in the specified SSS partition.

```
PROCEDURE proc_sss_GetApplicationTicketClaims (
@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@CurrentUserClaims xml
,@VerifyAdminClaims bit
);
```

@ApplicationName: The name of the target application. The value MUST NOT be NULL.

@PartitionId: The SSS partition for the claims (2) information about who can redeem an SSS ticket to be retrieved. The value MUST be a PartitionId.

@CurrentUserClaims: The claims (2) associated with user who is calling the stored procedure. The value MUST be ignored if *@VerifyAdminClaims* is not one. Otherwise, the value MUST satisfy the following conditions:

- The value MUST be Claims Information.
- The value MUST contain a claim (2) that uniquely identifies the caller.
- The claimsHash in Claims Information MUST be ignored by protocol server.

@VerifyAdminClaims: A flag which specifies to verify that *@CurrentUserClaims* is one of the administrators of the target application. The value MUST be in the following table.

Value	Description
0	The stored procedure MUST ignore *@CurrentUserClaims*.
1	The stored procedure MUST verify that the **claimType**, **claimIssuer** and **claimValue** of one of the claims (2) in *@CurrentUserClaims* MUST be equal to **claimType**, **claimIssuer** and **claimValue** respectively of one of the claims in the set of target application administrators.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80630005	Access is denied because the caller is not an administrator of the specified target application. The value MUST NOT be returned when *@VerifyAdminClaims* is zero.
0x00000000	Successful execution. The value MUST be ignored by the protocol client.

Result Sets:

If the return value from this stored procedure is not equal to 0x80630005, this stored procedure MUST return an Application Group Claims Result Set

3.2.5.14 proc_sss_GetConfig

The **proc_sss_GetConfig** stored procedure is called to get the information about the protocol server configuration.

```
PROCEDURE proc_sss_GetConfig (
);
```

Return Values: An integer which MUST be in the following table.

Value	Description
0x00000000	Successful execution. The value MUST be ignored by the protocol client.
0x8063000c	There is no configuration information available for this protocol server.

Result Sets:

This stored procedure MUST return a Configuration Result Set

3.2.5.15 proc_sss_GetCredentials

The **proc_sss_GetCredentials** stored procedure is called to get the credentials for the specified target application, SSS partition, claim type, claim issuer, claim value and SHA-256 hash of encrypted claim value.

If the ApplicationType of the specified target application is 0x04 or 0x05 the protocol server MUST set the return value to 0x80630490 and it MUST NOT return any credentials.

The row in the SSSCredentials table MUST be used to return the result set if a row with the specified claim type, claim issuer, claim value and SHA-256 hash of claim value is found in the SSSCredentials table for the specified target application in the specified SSS partition.

```
PROCEDURE proc_sss_GetCredentials (
@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@IdentityClaimType nvarchar(2084)
,@IdentityClaimIssuer nvarchar(2084)
,@IdentityClaimValue nvarchar(2048)
,@IdentityClaimValueHash varbinary(32)
,@Machine nvarchar(256)
,@CredentialManagementUrl nvarchar(2084) OUTPUT
);
```

@ApplicationName: The name of the target application.

@PartitionId: The SSS partition for the credentials to be retrieved. The value MUST be a PartitionId.

@IdentityClaimType: The claim type for the credential to be returned. The value MUST be a ClaimType. The value MUST be ignored by the protocol server if the specified target application is group target application.

@IdentityClaimIssuer: The claim issuer for the credential to be returned. The value MUST be a ClaimIssuer. The value MUST be ignored by the protocol server if the specified target application is group target application.

@IdentityClaimValue: The claim value for the credential to be returned. The value MUST be a ClaimValue. The value MUST be ignored by the protocol server if the specified target application is group target application.

@IdentityClaimValueHash: The SHA-256 hash of the claim value. The value MUST be SHA-256 hash of *@IdentityClaimValue*. The value MUST be ignored by the protocol server if the specified target application is group target application.

@Machine: The protocol client identifier which is making this stored procedure call. The value MUST NOT be NULL. This value can be used for auditing purposes.

@CredentialManagementUrl: The URL for a Web page where SSS users can set their credentials for the specified target application.

The protocol client MUST set the value to NULL. Upon completion of the stored procedure, the value MUST be set to the stored URL for the specified target application if the target application is an individual target application. If the specified target application is not an individual target application, the protocol server MUST set the value to NULL and the protocol client MUST ignore this value.

Return Values: An integer which MUST be in the following table.

Value	Description
0x00000000	Successful execution. The value MUST be ignored by the protocol client.
0x80630490	The target application with the specified *@ApplicationName* was not found or the ApplicationType of the specified target application is equal to 0x04 or 0x05.
0x80630001	If the specified target application is a group target application, the credentials were not found in the specified target application and SSS partition. If the specified target application is not a group target application, the credentials were not found in the specified target application (with the specified *@ApplicationName*), SSS partition (with the specified *@PartitionId*), claim type (with the specified *@IdentityClaimType*), claim issuer (with the specified *@IdentityClaimIssuer*), claim value (with the specified *@IdentityClaimValue*) and SHA-256 hash of claim value (with the specified *@IdentityClaimValueHash*).

Result Sets:

This stored procedure MUST return a Credentials Result Set

3.2.5.16 proc_sss_GetGroupClaimsPage

The **proc_sss_GetGroupClaimsPage** stored procedure is called to retrieve the specified number of records of claims (2) information. The records retrieved are claims (2) information for the members who can retrieve the credentials for each group target application stored in SSS store. This stored procedure will return the records starting from the specified starting row number in the SSSApplicationGroupClaim table.

```
PROCEDURE proc_sss_GetGroupClaimsPage (
@StartIndex int
,@PageSize int
```

```
);
```

@StartIndex: The starting row number from where the claims (2) are to be retrieved. To select which rows are to be returned, the implementation MUST assign virtual row numbers to the claims (2) stored in the SSSApplicationGroupClaim table. The first row MUST be numbered 0. Each subsequent row number MUST be the previous row's number incremented by 1. Ordering of rows MUST be by ClaimsId in the SSSApplicationGroupClaim table.

@PageSize: The maximum number of records of claims (2) information to be retrieved from the specified starting row number.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a Paged Group Claims Result Set

3.2.5.17 proc_sss_GetMasterSecretKey

The **proc_sss_GetMasterSecretKey** stored procedure is called to retrieve the encrypted master secret key, salt, checksum and version.

```
PROCEDURE proc_sss_GetMasterSecretKey (
@EncryptedKey varbinary(48) OUTPUT
,@IV varbinary(48) OUTPUT
,@CheckSum varbinary(96) OUTPUT
,@Version int OUTPUT
);
```

@EncryptedKey: The 256-bit Advanced Encryption Standard (AES) encrypted master secret key. The value MUST NOT be NULL.

@IV: The salt associated with the master secret key. The value MUST NOT be NULL.

@CheckSum: The checksum of the master secret key. The value MUST NOT be NULL.

@Version: The version of the master secret key. The value MUST NOT be NULL.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80631004	The master secret key was not found.
0x00000000	Successful execution. The value MUST be ignored by the protocol client.

Result Sets: MUST NOT return any result sets.

3.2.5.18 proc_sss_GetRestrictedCredentials

The **proc_sss_GetRestrictedCredentials** stored procedure is called to get the credential for the specified target application, SSS partition, claim type, claim issuer, claim value and SHA-256 hash of claim value. The stored procedure is called for target application with ApplicationType equal to 0x04 or 0x05.The row in the SSSCredentials table MUST be used to return the result set if a row

with the specified claim type, claim issuer, claim value and SHA-256 hash of claim value is found in the SSSCredentials table for the specified target application in the specified SSS partition.

```
PROCEDURE proc_sss_GetRestrictedCredentials (
@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@IdentityClaimType nvarchar(2084)
,@IdentityClaimIssuer nvarchar(2084)
,@IdentityClaimValue nvarchar(2048)
,@IdentityClaimValueHash varbinary(32)
,@Machine nvarchar(256)
,@CredentialManagementUrl nvarchar(2084) OUTPUT
);
```

@ApplicationName: The name of the target application.

@PartitionId: The SSS partition for the credentials to be retrieved. The value MUST be a PartitionId.

@IdentityClaimType: The claim type for the credential to be returned. The value MUST be a ClaimType.

@IdentityClaimIssuer: The claim issuer for the credential to be returned. The value MUST be a ClaimIssuer.

@IdentityClaimValue: The claim value for the credential to be returned. The value MUST be a ClaimValue.

@IdentityClaimValueHash: The SHA-256 hash of the claim value. The value MUST be SHA-256 hash of *@IdentityClaimValue*. If the specified target application is a group target application, this value MUST be set to NULL and it MUST be ignored by the protocol server.

@Machine: The name of the computer that the protocol client is running on. The value MUST NOT be NULL. This value can be used for auditing purposes.

@CredentialManagementUrl: The URL for a Web page where SSS users can set their credentials for the specified target application.

The protocol client MUST set the value to NULL. Upon completion of the stored procedure, the value MUST be set to the stored URL for the specified target application if the target application is an individual target application. If the specified target application is not an individual target application, the protocol server MUST set the value to NULL and the protocol client MUST ignore this value.

Return Values: An integer which MUST be in the following table.

Value	Description
0x00000000	Successful execution. The value MUST be ignored by the protocol client.
0x80630490	The target application with the specified *@ApplicationName* was not found.
0x80630001	If the specified target application is not group target application, the credentials were not found in the specified target application and SSS partition. If the specified target application is not a group target application, the credentials were not found in the specified target application (with the specified *@ApplicationName*), SSS partition (with the specified *@PartitionId*), claim type (with the specified *@IdentityClaimType*), claim issuer (with the specified *@IdentityClaimIssuer*), claim value (with the specified

Value	Description
	@IdentityClaimValue) and SHA-256 hash of claim value (with the specified *@IdentityClaimValueHash*).

Result Sets:

Upon successful execution of the stored procedure, this stored procedure MUST return a Credentials Result Set

3.2.5.19 proc_sss_GetState

The **proc_sss_GetState** stored procedure is called to retrieve the owner and state of an implementation specific long running operation persisted in the SSS store as specified in proc_sss_SetStatus.

```
PROCEDURE proc_sss_GetState (
);
```

Return Values: An integer which MUST be in the following table.

Value	Description
0x80631000	There is no information available for any client.
0x00000000	Successful execution. The value MUST be ignored by the protocol client.

Result Sets:

This stored procedure MUST return a State Result Set

3.2.5.20 proc_sss_GetTicketRedeemerClaimsPage

The **proc_sss_GetTicketRedeemerClaimsPage** stored procedure is called to retrieve the specified number of records of claims (2) information about the members who can redeem SSS ticket for target applications stored in SSS store. This stored procedure will return the records starting from the specified starting row number in the SSSApplicationTicketRedeemerClaim table.

```
PROCEDURE proc_sss_GetTicketRedeemerClaimsPage (
@StartIndex int
,@PageSize int
);
```

@StartIndex: The starting row number from where the claims (2) are to be retrieved. To select which rows are to be returned, the implementation MUST assign virtual row numbers to the claims (2) stored in the SSSApplicationTicketRedeemerClaim table. The first row MUST be numbered 0. Each subsequent row number MUST be the previous row's number incremented by 1. Ordering of rows MUST be by ClaimsId in the SSSApplicationTicketRedeemerClaim table.

@PageSize: The maximum number of records of claims (2) information to be retrieved from the specified starting row number.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a Paged Group Claims Result Set

3.2.5.21 proc_sss_GetUserApplications

The **proc_sss_GetUserApplications** stored procedure is called to retrieve the information for all the target applications that can be accessed by the user with the specified Claims Information in the specified SSS partition. The target applications that can be accessed by the user MUST include all the group target applications in the specified SSS partition where the user is a member of the group target application and all the individual target applications in the specified SSS partition.

```
PROCEDURE proc_sss_GetUserApplications (
@PartitionId uniqueidentifier
,@CurrentUserClaims xml
);
```

@PartitionId: The SSS partition for the target applications to be retrieved. The value MUST be a PartitionId.

@CurrentUserClaims: The claim (2) of the client protocol user who is calling the stored procedure. The value MUST be Claims Information. The claimHash in Claims Information MUST NOT be set.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a Application Information Result Set

3.2.5.22 proc_sss_InsertAudit

The **proc_sss_InsertAudit** stored procedure is called to add a SSS audit entry to the SSS store when an **SSS action** is performed.

```
PROCEDURE proc_sss_InsertAudit (
@UserIdentityClaimType nvarchar(1028)
,@UserIdentityClaimValue nvarchar(1028)
,@UserIdentityClaimIssuer nvarchar(1028)
,@ActionType int
,@ActionResultCode int
,@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@SubscriptionId uniqueidentifier
,@Info1 nvarchar(1028)
,@Info2 nvarchar(1028)
,@Info3 nvarchar(1028)
,@Info4 nvarchar(1028)
,@Info5 nvarchar(1028)
,@Machine nvarchar(256)
);
```

@UserIdentityClaimType: The claim type for the SSS audit entry to be added. The value MUST be a ClaimType.

@UserIdentityClaimValue: The claim value for the SSS audit entry to be added. The value MUST be a ClaimValue.

@UserIdentityClaimIssuer: The claim issuer for the SSS audit entry to be added. The value MUST be a ClaimIssuer.

@ActionType: The action type of the SSS audit entry to be added. The value MUST be an ActionType.

@ActionResultCode: An implementation-specific return code denoting the status of the attempted operation. The value MUST NOT be NULL.

@ApplicationName: The name of the target application.

@PartitionId: The SSS partition for the SSS audit entry to be added. The value MUST be a PartitionId.

@SubscriptionId: An implementation-specific identifier. The value MUST NOT be NULL.

@Info1: Additional information to be audited. The value MUST be NULL if protocol client adds a SSS audit entry for a successful action. The value MUST NOT be NULL, if the protocol client adds a SSS audit entry for a failed SSS action.

@Info2: Additional information to be audited. Unused. The value MUST be NULL.

@Info3: Additional information to be audited. Unused. The value MUST be NULL.

@Info4: Additional information to be audited. Unused. The value MUST be NULL.

@Info5: Additional information to be audited. Unused. The value MUST be NULL.

@Machine: The name of the computer that the protocol client is running on. The value MUST NOT be NULL. This value can be used for auditing purposes.

Return Values: An integer which MUST be in the following table.

Value	Description
0x00000000	Successful execution. The value MUST be ignored by the protocol client.
0x80630009	Inserting SSS audit entry failed.

Result Sets: MUST NOT return any result sets.

3.2.5.23 proc_sss_PrepareSecondaryTables

The **proc_sss_PrepareSecondaryTables** stored procedure is called to ensure that tables **SssCredentials_Secondary**, **SssApplicationTicketRedeemerClaim_Secondary** and **SssApplicationGroupClaim_Secondary** are empty. The stored procedure MUST signal an error condition using RAISERROR, as specified in [MSDN-TSQL-Ref], with the RAISERROR msg_str as specified in the following table.

```
PROCEDURE proc_sss_PrepareSecondaryTables (
);
```

Error code values:

Value	Description
can not prepare. ssscredentials_secondary is not empty.	**SssCredentials_Secondary** table is not empty.
can not prepare. sssapplicationticketredeemerclaim_secondary is not empty.	**SssApplicationTicketRedeemClaim_Secondary** table is not empty.
can not prepare. sssapplicationgroupclaim_secondary is not empty.	**SssApplicationGroupClaim_Secondary** table is not empty.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.24 proc_sss_PublishSecondaryTables

The **proc_sss_PublishSecondaryTables** stored procedure is called to update the rows in tables SSSCredentials, SSSApplicationGroupClaim and SSSApplicationTicketRedeemerClaim with re-encrypted credentials and claims stored in SSSCredentials_Secondary, SSSApplicationGroupClaim_Secondary, and SSSApplicationTicketRedeemerClaim_Secondary tables respectively.

The tables **SSSCredentials_Secondary**, **SSSApplicationGroupClaim_Secondary** and **SSSApplicationTicketRedeemerClaim_Secondary** are populated directly by the protocol client as specified in proc_sss_SetMasterSecretKey. The tables are populated when the protocol client re-encrypts the existing rows in **SSSCredentials**, **SSSApplicationGroupClaim** and **SSSApplicationTicketRedeemerClaim**.

The stored procedure MUST update rows in table SSSCredentials by copying the rows in the table SSSCredentials_Secondary for the corresponding CredentialsId.

The stored procedure MUST update rows in table SSSApplicationGroupClaim by copying rows in the table SSSApplicationGroupClaim_Secondary for the corresponding ClaimsId.

The stored procedure MUST update rows in table SSSApplicationTicketRedeemerClaim by copying the rows in the table SSSApplicationTicketRedeemerClaim_Secondary for the corresponding ClaimsId.

Upon execution of the stored procedure, the stored procedure MUST delete all rows from tables SSSCredentials_Secondary, SSSApplicationGroupClaim_Secondary and SSSApplicationTicketRedeemerClaim_Secondary**.**

```
PROCEDURE proc_sss_PublishSecondaryTables (
);
```

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.25 proc_sss_PurgeClaims

The **proc_sss_PurgeClaims** stored procedure is deprecated and MUST NOT be called.

```
PROCEDURE proc_sss_PurgeClaims (
);
```

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.26 proc_sss_PurgeTickets

The **proc_sss_PurgeTickets** stored procedure is called to delete SSS tickets when the difference between their creation and the current time, expressed in minutes, is greater than or equal to the validity of the SSS ticket in target application definition (validity of SSS ticket is specified in **proc_sss_CreateApplication** stored procedure by *@TicketTimeout*).

```
PROCEDURE proc_sss_PurgeTickets (
);
```

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.27 proc_sss_RedeemTicket

The **proc_sss_RedeemTicket** stored procedure is called to retrieve the set of claims (2) that specify the ticket redeemer group of SSS users that are associated with the specified target application in the specified SSS partition. In addition, it retrieves the credentials associated with the specified target application in the specified SSS partition for the specified SSS user.

```
PROCEDURE proc_sss_RedeemTicket (
@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@IdentityClaimType nvarchar(2084)
,@IdentityClaimIssuer nvarchar(2084)
,@IdentityClaimValue nvarchar(2048)
,@IdentityClaimValueHash varbinary(32)
,@UserTicket varbinary(300)
,@Machine nvarchar(256)
);
```

@ApplicationName: The name of the specified target application. The value MUST NOT be NULL.

@PartitionId: The SSS partition for the SSS ticket to be redeemed. The value MUST be PartitionId.

@IdentityClaimType: The claim type for the SSS ticket to be redeemed. The value MUST be a ClaimType. The value MUST be ignored if the target application with the specified name *@ApplicationName* in the SSS partition (with the value specified in *@PartitionId*) is of ApplicationType 0x01, 0x03 or 0x05.

@IdentityClaimIssuer: The claim issuer for the SSS ticket to be redeemed. The value MUST be a ClaimIssuer. The value MUST be ignored if the target application with the specified name *@ApplicationName* in the SSS partition (with the value specified in *@PartitionId*) is of ApplicationType 0x01, 0x03 or 0x05.

@IdentityClaimValue: The claim value for the SSS ticket to be redeemed. The value MUST be a ClaimValue. The value MUST be ignored if the target application with the specified name *@ApplicationName* in the SSS partition (with the value specified in *@PartitionId*) is of ApplicationType 0x01, 0x03 or 0x05.

@IdentityClaimValueHash: The SHA-256 hash of the claim value. The value MUST be SHA-256 hash of *@IdentityClaimValue*. The value MUST be ignored if the target application with the specified name *@ApplicationName* in the SSS partition (with the value specified in *@PartitionId*) is of ApplicationType 0x01, 0x03 or 0x05.

@UserTicket: The SSS ticket which was previously stored using **proc_sss_SetTicket** stored procedure. The value MUST NOT be NULL.

@Machine: The name of the protocol client computer which is making this stored procedure call. The value MUST NOT be NULL. This value can be used for auditing purposes.

Return Values: An integer which MUST be in the following table.

Value	Description
0x00000000	Successful execution. The value MUST be ignored by the protocol client.
0x80630490	The target application with the specified *@ApplicationName* was not found in the SSS partition (with the specified *@PartitionId*).
0x80630005	Either of the following two conditions are true: ▪ The target application with the specified *@ApplicationName* in the SSS partition (with the specified *@PartitionId*) is of ApplicationType 0x01, 0x03 or 0x05. ▪ The specified SSS ticket has expired and hence been deleted from the SSS store
0x80630008	The SSS ticket with the specified *@UserTicket* is not a valid SSS ticket or this SSS ticket can no longer be redeemed because it has expired.
0x80630001	The credentials in the specified target application (with the specified *@ApplicationName*), SSS partition (with the specified *@PartitionId*), and SSS ticket (with the specified *@UserTicket*) was not found.

Result Sets:

This stored procedure MUST return a Application Group Claims Result Set

This stored procedure MUST return a Credentials Result Set

3.2.5.28 proc_sss_SetChangeKeyStatus

The proc_sss_SetChangeKeyStatus stored procedure is called to store status information when changing the master secret key in SSS store.

```
PROCEDURE proc_sss_SetChangeKeyStatus (
@KeyId nvarchar(48)
,@State nvarchar(48)
,@Message nvarchar(512)
,@Details nvarchar(2084)
);
```

@KeyId: A token that was previously reserved for changing the master secret key by calling proc_sss_ReserveKeyChangeToken. The value MUST NOT be NULL.

@State: A string representing the current state of changing the master secret key. The value MUST be in the following table.

Value	Description
Info	This value is used to store the initial status of changing the master secret key.
Failed	This value is used when changing the master secret key failed.
Success	This value is used when changing the master secret key was successful.

@Message: A protocol client specific short message about the current master secret key change status. The value MUST NOT be NULL.

@Details: A protocol client specific detailed message about the current master secret key change status.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.29 proc_sss_SetConfig

The **proc_sss_SetConfig** stored procedure is called to persist information about the protocol server configuration in the SSS store.

```
PROCEDURE proc_sss_SetConfig (
@PurgeAuditDays int
,@EnableAudit bit
);
```

@PurgeAuditDays: An integer value that denotes how long an SSS audit entry is preserved in the SSS store, measured in days. The parameter MUST be PurgeAuditDays.

@EnableAudit: A flag specifies whether the SSS audit entry is stored in SSS store when **proc_sss_InsertAudit** is called. The value MUST be EnableAudit.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.30 proc_sss_SetCredentials

The **proc_sss_SetCredentials** stored procedure is called to set the credentials for the user identified by claim type, claim issuer, claim value and SHA-256 hash of claim value for the specified target application in the specified SSS partition

The row in the SSSCredentials table MUST be updated with the credentials if a row with the specified claim type, claim issuer, claim value and SHA-256 hash of claim value is found in the SSSCredentials table for the specified target application in the specified SSS partition. Otherwise a new row MUST be added to the SSSCredentials table with the specified target application, claims (2) and credentials information.

```
PROCEDURE proc_sss_SetCredentials (
@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@IdentityClaimType nvarchar(2084)
,@IdentityClaimIssuer nvarchar(2084)
,@IdentityClaimValue nvarchar(2048)
,@IdentityClaimValueHash varbinary(32)
,@Credentials varbinary(max)
,@GroupCredentials bit
,@CurrentUserClaims xml
,@VerifyAdminClaims bit
,@CheckSum varbinary(96)
);
```

@ApplicationName: The name of the target application.

@PartitionId: The SSS partition to set the credentials. The value MUST be a PartitionId.

@IdentityClaimType: The claim type for the credential to be set. The value MUST be a ClaimType. The value MUST be ignored by the protocol server if the specified target application is group target application.

@IdentityClaimIssuer: The claim issuer for the credential to be set. The value MUST be ClaimIssuer. The value MUST be ignored by the protocol server if the specified target application is group target application.

@IdentityClaimValue: The claim value for the credential to be set. The value MUST be ClaimValue. The value MUST be ignored by the protocol server.

@IdentityClaimValueHash: The SHA-256 hash of the claim (2) value. The value MUST be SHA-256 hash of *@IdentityClaimValue*. The value MUST be ignored by the protocol server if the specified target application is group target application.

@Credentials: The encrypted credential to be set. The value MUST NOT be NULL.

@GroupCredentials: A flag indicating whether the credential being set is for an individual user or group of users.

Value	Description
0	The credential is for an individual user.
1	The credential is for a group of users.

@CurrentUserClaims: The claims (2) associated with user who is calling the stored procedure. The value MUST be ignored if *@VerifyAdminClaims* is not one. Otherwise, the value MUST satisfy the following conditions: - The value MUST be Claims Information.- The value MUST contain a claim (2) that uniquely identifies the caller.- The claimsHash in Claims Information MUST be ignored by protocol server.

@VerifyAdminClaims: A flag which specifies to verify that *@CurrentUserClaims* is one of the administrators of the target application. The value MUST be in the following table.

Value	Description
0	The stored procedure MUST ignore *@CurrentUserClaims*.

Value	Description
1	The stored procedure MUST verify that the **claimType**, **claimIssuer** and **claimValue** of any one of the claims (2) in *@CurrentUserClaims* MUST be equal to **claimType**, **claimIssuer** and **claimValue** respectively of any one of the claims (2) in the set of administrators' claims (2) associated with the specified target application.

@CheckSum: The checksum of the master secret key. The value MUST NOT be NULL.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80631005	An implementation specific long running operation has temporarily blocked access to the credentials in the SSS store.
0x80630490	The target application with specified *@ApplicationName* was not found in the SSS partition (with the specified *@PartitionId*).
0x00000000	Successful execution. The value MUST be ignored by the protocol client.
0x80630005	Access is denied because the caller is not an administrator of the specified target application. The value MUST NOT be returned when *@VerifyAdminClaims* is zero.
0x8063000d	The value of *@Checksum* is not the correct checksum of the master secret key.

Result Sets: MUST NOT return any result sets.

3.2.5.31 proc_sss_SetMasterSecretKey

The **proc_sss_SetMasterSecretKey** stored procedure is called to store the encrypted master secret key, salt associated with master secret key and checksum for master secret key in the SSS store.

```
PROCEDURE proc_sss_SetMasterSecretKey (
@EncryptedKey varbinary(48)
,@IV varbinary(48)
,@CheckSum varbinary(96)
,@Version int OUTPUT
);
```

@EncryptedKey: The 256-bit Advanced Encryption Standard (AES) encrypted master secret key. The value MUST NOT be NULL.

@IV: The random salt associated with the master secret key. The value MUST NOT be NULL.

@CheckSum: The checksum of the master secret key. The value MUST NOT be NULL.

@Version: The version of the master secret key. The protocol client MUST set the value to NULL. Upon completion of the stored procedure, the value MUST be set to the version of the master secret key that is stored in the SSS store.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.32 proc_sss_SetStatus

The **proc_sss_SetStatus** stored procedure is called to save or change the state of an implementation specific long running operation in the SSS store. If an operation initiated by an owner that is not equal to *@Owner* is already saved in the SSS store with *@Status* equal to one, the requested save or change of state MUST NOT be persisted. Otherwise, the values of *@Status* and *@Owner* MUST be persisted in the SSS store, overwriting previous values, if any. For example, a protocol client calls the stored procedure before re-encrypting existing credentials in SSS store. If the stored procedure executes successfully the protocol client re-encrypts existing credentials. If the stored procedure does not execute successfully, the protocol client does not re-encrypt existing credentials and retries the operation later.

```
PROCEDURE proc_sss_SetStatus (
@Status int
,@Owner uniqueidentifier
);
```

@Status: The synchronization state of SSS store. The value MUST be StatusType.

@Owner: Identifier of the protocol client. The value MUST NOT be NULL or empty GUID.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80631000	An implementation specific corruption is detected in the state maintained by the SSS store. For example, *@Status* is neither zero nor one.
0x80631001	The specified protocol client cannot set the synchronization state of SSS store. Another protocol client has set the synchronization state of SSS store with *@Status* value one.
0x00000000	Successful execution. The value MUST be ignored by the protocol client.

Result Sets: MUST NOT return any result sets.

3.2.5.33 proc_sss_SetTicket

The **proc_sss_SetTicket** stored procedure is called to store the specified SSS ticket representing the SSS user identified by claim type, claim issuer, claim value and SHA-256 hash of claim value in the specified SSS partition along with the current date and time.

```
PROCEDURE proc_sss_SetTicket (
@UserTicket varbinary(300)
,@PartitionId uniqueidentifier
,@IdentityClaimType nvarchar(2084)
,@IdentityClaimIssuer nvarchar(2084)
,@IdentityClaimValue nvarchar(2048)
,@Machine nvarchar(256)
,@CheckSum varbinary(96)
);
```

@UserTicket: The SSS ticket to be stored in the SSS store. The value MUST be Random Ticket.

@PartitionId: The SSS partition for the SSS ticket. The value MUST be PartitionId.

@IdentityClaimType: The claim type for the SSS ticket. The value MUST be a ClaimType.

@IdentityClaimIssuer: The claim issuer for the SSS ticket. The value MUST be a ClaimIssuer.

@IdentityClaimValue: The claim value for the SSS ticket. The value MUST be a ClaimValue.

@Machine: The identifier of the protocol client computer which is making this stored procedure call. The value MUST NOT be NULL. This value can be used for auditing purposes.

@CheckSum: The checksum of the master secret key. The value MUST NOT be NULL.

Return Values: An integer which MUST be in the following table.

Value	Description
0x00000000	Successful execution. The value MUST be ignored by the protocol client.
@@error	A T-SQL error code.
0x8063000d	The value of *@Checksum* is not the correct checksum of the master secret key.

Result Sets: MUST NOT return any result sets.

3.2.5.34 proc_sss_UpdateApplication

The **proc_sss_UpdateApplication** stored procedure is called to update an existing target application.

```
PROCEDURE proc_sss_UpdateApplication (
@ApplicationName nvarchar(256)
,@FriendlyName nvarchar(256)
,@PartitionId uniqueidentifier
,@ApplicationType int
,@TicketTimeout int
,@ContactEmail nvarchar(128)
,@CredentialManagementUrl nvarchar(2084)
,@FieldInfo xml
,@AdminClaims xml
,@GroupClaims xml
,@TicketRedeemClaims xml
,@Credentials varbinary(max)
,@CurrentUserClaims xml
,@VerifyAdminClaims bit
,@CheckSum varbinary(96)
);
```

@ApplicationName: The name of the target application to be updated. The value MUST NOT be NULL.

@FriendlyName: The descriptive name of the target application to be updated. The value MUST NOT be NULL.

@PartitionId: The SSS partition for the target application to be updated. The value MUST NOT be NULL.

@ApplicationType: The type of the target application. The value MUST be an ApplicationType.

@TicketTimeout: The validity in minutes for the SSS ticket for the specified target application. This MUST NOT be NULL if the value of *@ApplicationType* is equal to 0x02 or 0x03. This value MUST be set to NULL if the value of *@ApplicationType* is not equal to 0x02 or 0x03.

@ContactEmail: The e-mail address of the administrator who owns the administration responsibilities for the specified target application.

@CredentialManagementUrl: The URL for a Web page where SSS users can set their credentials for the specified target application.

@FieldInfo: The Fields Information for the specified target application.

@AdminClaims: The claim (2) of administrator of the specified target application who will own the administration responsibilities for the specified target application. The value MUST NOT be NULL.

@GroupClaims: The claim (2) of the members who has access to the credentials stored for the specified target application. The value MUST NOT be NULL if the value of *@ApplicationType* is equal to 0x01, 0x03 or 0x05. The value MUST be set to NULL if the value of *@ApplicationType* is not equal to 0x01, 0x03 or 0x05.

@TicketRedeemClaims: The claim (2) of members who has access to redeem SSS tickets for the specified target application. The value MUST NOT be NULL if the value of *@ApplicationType* is equal to 0x02 or 0x03. The value MUST be set to NULL if the value of *@ApplicationType* is not equal to 0x02 or 0x03.

@Credentials: Re-encrypted credentials for the target application. The value MUST NOT be NULL if the value of *@ApplicationType* is equal to 0x01, 0x03 or 0x05. The value MUST be set to NULL if the value of *@ApplicationType* is not equal to 0x01, 0x03 or 0x05.

@CurrentUserClaims: The claims (2) associated with user who is calling the stored procedure. The value MUST be ignored if *@VerifyAdminClaims* is not one. Otherwise, the value MUST satisfy the following conditions:

- the value MUST be Claims Information.
- the value MUST contain a claim (2) that uniquely identifies the caller.
- the **claimsHash** in Claims Information MUST be ignored by protocol server.

@VerifyAdminClaims: A flag which specifies to verify that *@CurrentUserClaims* is one of the administrators of the target application. The value MUST be in the following table.

Value	Description
0	The stored procedure MUST ignore *@CurrentUserClaims*.
1	The stored procedure MUST verify that the **claimType**, **claimIssuer** and **claimValue** of one of the claims (2) in *@CurrentUserClaims* MUST be equal to **claimType**, **claimIssuer** and **claimValue** respectively of one of the claims (2) in the set of target application administrators.

@CheckSum: The checksum of the master secret key. The value MUST NOT be NULL.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80630490	The target application with the specified *@ApplicationName* was not found in the SSS

Value	Description
	partition (with the specified *@PartitionId*).
0x80630005	Access is denied because the caller is not an administrator of the specified target application. The value MUST NOT be returned when *@VerifyAdminClaims* is zero.
@@error	A T-SQL error code.
0x80630013	The number of Fields Information elements provided in *@FieldInfo* is not equal to the number of Field elements of existing target application.
0x80630014	One or more of the Fields Information elements provided in *@FieldInfo* is not equal to the Field elements of existing target application.
0x80630010	There are no Claims Information elements found in *@GroupClaims*.
0x80630011	There are no Claims Information elements found in *@TicketRedeemClaims*.
0x00000000	Successful execution. The value MUST be ignored by the protocol client.
0x8063000d	The value of *@Checksum* is not the correct checksum of the master secret key.
0x8063000b	The ApplicationType for the specified target application is equal to 0x04 or 0x05 and the specified target application cannot be updated.
0x8063000e	There are no Fields Information elements found in *@FieldInfo*.
0x8063000f	There are no Claims Information elements found in *@AdminClaims*.

Result Sets: MUST NOT return any result sets.

3.2.5.35 proc_sss_GetServersKeyState

The **proc_sss_GetServersKeyState** stored procedure is called to retrieve the information about protocol clients along with the associated encrypted master secret key and its salt.

```
PROCEDURE proc_sss_GetServersKeyState (
);
```

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a Servers Key Exchange Result Set

3.2.5.36 proc_sss_PublishPublicKey

The **proc_sss_PublishPublicKey** stored procedure is called to update the public key of a specified protocol client in the SSS store. If the public key of the specified protocol client does not exist in the SSS store, it MUST be added.

```
PROCEDURE proc_sss_PublishPublicKey (
@ServerId uniqueidentifier
,@PublicKey varbinary(2048)
);
```

@ServerId: The identifier of the protocol client. The value MUST NOT be NULL or empty GUID.

@PublicKey: The public key of the protocol client. The value MUST NOT be Null.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.37 proc_sss_PurgeKeyChangeToken

The **proc_sss_PurgeKeyChangeToken** stored procedure is called to delete the specified token from the SSS store.

```
PROCEDURE proc_sss_PurgeKeyChangeToken (
@Token nvarchar(48)
);
```

@Token: The token to be deleted from the SSS store. The value MUST NOT be NULL.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.38 proc_sss_ReserveKeyChangeToken

The **proc_sss_ReserveKeyChangeToken** stored procedure is called to store the specified token in the SSS store. The specified token MUST be valid only for period of one minute from when it is stored in the SSS store.

```
PROCEDURE proc_sss_ReserveKeyChangeToken (
@Token nvarchar(48)
);
```

@Token: The token to be stored in SSS store. It is a protocol client implementation specific string. The value MUST NOT be NULL.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80631008	The token with the specified *@Token* value already exist in SSS store.
0x80631006	A valid token already exists in the SSS store.
0x80631001	The token cannot be stored in the SSS store because the StatusType is equal to 1.
0	Successful execution. The value MUST be ignored by the protocol client.

Result Sets: MUST NOT return any result sets.

3.2.5.39 proc_sss_UpdateServersKeyState

The **proc_sss_UpdateServersKeyState** stored procedure is called to update the encrypted master secret key and the version of the master secret key for one or more protocol clients.

```
PROCEDURE proc_sss_UpdateServersKeyState (
@UpdateKeyExchangeStatusXml xml
);
```

@UpdateKeyExchangeStatusXml: The Key Exchange Information for one or more protocol clients to be updated in the SSS store. The value MUST NOT be NULL.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.40 proc_sss_ValidateKeyChangeToken

The **proc_sss_ValidateKeyChangeToken** stored procedure is called to validate the specified token in the SSS store. The specified token is a protocol client implementation specific string and it MUST be valid only for period of one minute from when it is stored in the SSS store. If the return value is 0, the specified token is valid.

```
PROCEDURE proc_sss_ValidateKeyChangeToken (
@Token nvarchar(48)
);
```

@Token: The token to be validated in SSS store. The value MUST NOT be NULL.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80631007	The token does not exist in the SSS store or the token has expired. The token is marked as expired, if it has been stored in the SSS store for more than one minute.
0	Successful execution.

Result Sets: MUST NOT return any result sets.

3.2.5.41 proc_sss_IsApplicationMember

The **proc_sss_IsApplicationMember** stored procedure is called to find out if a user represented by @CurrentUserClaims parameter is a member of an existing target application.

```
PROCEDURE proc_sss_IsApplicationMember (
@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@CurrentUserClaims xml
,@IsMember bit OUTPUT
);
```

@ApplicationName: The name of the target application. It MUST NOT be NULL.

@PartitionId: The SSS partition for the target application to be updated. The value MUST NOT be NULL.

@CurrentUserClaims: The claims (2) associated with user who is calling the stored procedure. The value MUST satisfy the following conditions:

- The value MUST be Claims Information.
- The value MUST contain a claim (2) that uniquely identifies the caller. The claimsHash in Claims Information MUST be ignored by protocol serverdescription.

@IsMember: A bit field output of the stored procedure.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80630490	The specified target application does not exist in the specified *@PartitionId*.
0	Successful execution. The value MUST be ignored by the protocol client.

Result Sets: MUST NOT return any result sets.

3.2.5.42 proc_sss_CreateConnectionSettings

The stored procedure **proc_sss_CreateConnectionSettings** is used to create a new connection.

```
PROCEDURE proc_sss_CreateConnectionSettings (
@PartitionId uniqueidentifier
,@ConnectionSettingsName nvarchar(256)
,@Description nvarchar(1024)
,@Type nvarchar(256)
,@Target nvarchar(2084)
,@AuthenticationMode nvarchar(256)
,@SSOApplicationId nvarchar(1024)
,@SSOProviderImplId nvarchar(1024)
,@ProxyTarget nvarchar(2084)
,@ProxySSOApplicationId nvarchar(1024)
,@ParentName nvarchar(256)
,@Properties nvarchar(4000)
);
```

@PartitionId: The SSS partition of the connection to be created. The value MUST NOT be NULL.

@ConnectionSettingsName: The name of the connection. It MUST NOT be NULL.

@Description: The description of the connection. It MUST NOT be NULL.

@Type: The type of the connection. Valid values for the type are implementation specific.

@Target: The URL of the connection.

@AuthenticationMode: The authentication mode of the connection.

@SSOApplicationId: The target application of the connection.

@SSOProviderImplId: The name of the custom implementation of the target application of the connection.

@ProxyTarget: The URL of the proxy of the connection.

@ProxySSOApplicationId: The target application of the proxy of the connection.

@ParentName: The name of the parent connection.

@Properties: The properties of the connection.

Return Values: An integer which MUST be in the following table.

Value	Description
80632000	A connection with name as specified in *@ConnectionSettingsName* already exists in the partition *@PartitionId*.
@@error	INSERT BOILERPLATE SQL REFERENCE
0	Successful execution. The value MUST be ignored by the protocol client.

Result Sets:

The stored procedure MUST NOT return any result sets.

3.2.5.43 proc_sss_DeleteConnectionSettings

The stored procedure is used to delete an existing connection.

```
PROCEDURE proc_sss_DeleteConnectionSettings (
@PartitionId uniqueidentifier
,@ConnectionSettingsName nvarchar(256)
);
```

@PartitionId: The SSS partition of the connection to be deleted. The value MUST NOT be NULL.

@ConnectionSettingsName: The name of the connection to be deleted.

Return Values: An integer which MUST be in the following table.

Value	Description
80632001	A connection with the specified name does not exist.
0	Successful execution. The value MUST be ignored by the protocol client.

Result Sets: MUST NOT return any result sets.

3.2.5.44 proc_sss_GetAllConnectionSettings

The stored procedure is used to get all connections.

```
PROCEDURE proc_sss_GetAllConnectionSettings (
@PartitionId uniqueidentifier
,@Type nvarchar(256)
);
```

@PartitionId: The SSS partition of the connections to be queried. The value MUST NOT be NULL.

@Type: The type of the connections to be queried.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a Connection Settings Result Set

3.2.5.45 proc_sss_GetChildConnectionSettings

The stored procedure is used to get all child connections of an existing connection.

```
PROCEDURE proc_sss_GetChildConnectionSettings (
@PartitionId uniqueidentifier
,@ParentName nvarchar(256)
);
```

@PartitionId: The SSS partition of the connection to be queried. The value MUST NOT be NULL.

@ParentName: The name of the parent connection. It MUST NOT be NULL.

Return Values: An integer which MUST be in the following table.

Value	Description
80632001	A connection with the name specified by *@ParentName* does not exist.
0	Successful execution. The value MUST be ignored by the protocol client.

Result Sets:

This stored procedure MUST return a Connection Settings Result Set

3.2.5.46 proc_sss_GetConnectionSettings

The stored procedure is used to get the connection with the specified name in the specified partition.

```
PROCEDURE proc_sss_GetConnectionSettings (
@PartitionId uniqueidentifier
,@ConnectionSettingsName nvarchar(256)
,@IncludeCredentials bit
,@IdentityClaimType nvarchar(2084)
,@IdentityClaimIssuer nvarchar(2084)
,@IdentityClaimValue nvarchar(2048)
,@IdentityClaimValueHash varbinary(32)
,@Machine nvarchar(256)
,@CredentialManagementUrl nvarchar(2084) OUTPUT
,@TargetCredentialsFound bit OUTPUT
,@ProxyCredentialsFound bit OUTPUT
);
```

@PartitionId: The SSS partition for the connection. The value MUST NOT be NULL.

@ConnectionSettingsName: The name of the connection. The value MUST NOT be NULL.

@IncludeCredentials: If the value is true, then result set contains credential information. Otherwise the result set MUST NOT contain credential information.

@IdentityClaimType: The claim type for the credential to be returned. The value MUST be a ClaimType. The value MUST be ignored by the protocol server if the specified target application is group target application.

@IdentityClaimIssuer: The claim issuer for the credential to be returned. The value MUST be a ClaimIssuer. The value MUST be ignored by the protocol server if the specified target application is group target application.

@IdentityClaimValue: The claim value for the credential to be returned. The value MUST be a ClaimValue. The value MUST be ignored by the protocol server if the specified target application is group target application.

@IdentityClaimValueHash: The SHA-256 hash of the claim value. The value MUST be SHA-256 hash of *@IdentityClaimValue*. The value MUST be ignored by the protocol server if the specified target application is group target application.

@Machine: The identifier of the protocol client computer which is making this stored procedure call. The value MUST NOT be NULL. This value can be used for auditing purposes.

@CredentialManagementUrl: The URL for a Web page where SSS users can set their credentials for the specified target application.

@TargetCredentialsFound: It MUST be true if a target application exists for the connection.

@ProxyCredentialsFound: It MUST be true if a target application exists for the proxy of the connection.

Return Values: An integer which MUST be in the following table.

Value	Description
0x80632001	A connection with the name specified by *@ConnectionSettingsName* does not exist.
0	Successful execution. The value MUST be ignored by the protocol client.

Result Sets:

This stored procedure MUST return a Connection Settings Result Set

3.2.5.47 proc_sss_IsApplicationAdmin

The **proc_sss_IsApplicationAdmin** stored procedure is called to find out if a user represented by @CurrentUserClaims parameter is a member of an existing target application in the specified partition.

```
PROCEDURE proc_sss_IsApplicationAdmin (
@ApplicationName nvarchar(256)
,@PartitionId uniqueidentifier
,@CurrentUserClaims xml
,@IsAdmin bit OUTPUT
);
```

@ApplicationName: The name of the target application. It MUST NOT be NULL.

@PartitionId: The SSS partition for the target application to be updated. The value MUST NOT be NULL.

@CurrentUserClaims: The claims (2) associated with user who is calling the stored procedure. The value MUST satisfy the following conditions:

- The value MUST be Claims Information.
- The value MUST contain a claim (2) that uniquely identifies the caller. The claimsHash in Claims Information MUST be ignored by protocol serverdescription.

@IsAdmin: Represents the output of the stored procedure.

Return Values: An integer which MUST be in the following table.

Value	Description
80630490	The specified target application does not exist in the specified *@PartitionId*.
0	Successful execution. The value MUST be ignored by the protocol client.

Result Sets: MUST NOT return any result sets.

3.2.5.48 proc_sss_UpdateConnectionSettings

The stored procedure **proc_sss_UpdateConnectionSettings** is used to update an existing connection in the specified partition.

```
PROCEDURE proc_sss_UpdateConnectionSettings (
@PartitionId uniqueidentifier
,@ConnectionSettingsName nvarchar(256)
,@Description nvarchar(1024)
,@Type nvarchar(256)
,@Target nvarchar(2084)
,@AuthenticationMode nvarchar(256)
,@SSOApplicationId nvarchar(1024)
,@SSOProviderImplId nvarchar(1024)
,@ProxyTarget nvarchar(2084)
,@ProxySSOApplicationId nvarchar(1024)
,@ParentName nvarchar(256)
,@Properties nvarchar(4000)
);
```

@PartitionId: The SSS partition for the connection to be updated. The value MUST NOT be NULL.

@ConnectionSettingsName: The name of the connection to be updated. The value MUST NOT be NULL.

@Description: New value of the description of the connection.

@Type: New value of the connection type.

@Target: New value of the URL of the connection.

@AuthenticationMode: New value of the authentication mode.

@SSOApplicationId: New value of the target application associated with the connection.

@SSOProviderImplId: New value of the implementation id.

@ProxyTarget: New value of the proxy URL associated with the connection.

@ProxySSOApplicationId: New value of the target application associated with the new proxy specified by @ProxyTaget.

@ParentName: New value specifying the name of the parent connection.

@Properties: New value for the connection properties.

Return Values: An integer which MUST be in the following table.

Value	Description
80632001	A connection with the specified name does not exist in the specified partition id.
@@error	INSERT BOILERPLATE SQL REFERENCE
0	Successful execution. The value MUST be ignored by the protocol client.

Result Sets: MUST NOT return any result sets.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Client Details

3.3.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The target applications, credentials, SSS tickets and SSS configuration stored in the SSS store can be maintained as object structures within the protocol client.

3.3.2 Timers

An SSS ticket expiration timer is used to periodically poll for tickets that have expired in the SSS store. The amount of time elapsed between checks for whether tickets have expired is an implementation-dependent decision.

An SSS audit entry purge timer is used to periodically poll for audit entries that MUST be purged from the SSS store. The amount of time elapsed between checks for whether entries have expired is an implementation-dependent decision.

3.3.3 Initialization

The protocol client MUST get the claims (2) of the users and validate the users making the request before calling the stored procedure.

The protocol client MUST generate a master secret key and store it encrypted in SSS store by calling the stored procedure before using any stored procedures using target application or credentials. The protocol client MUST also keep a local, cached copy of the master secret key for use in encryption or decryption operations.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

The protocol client handles each stored procedure with the same basic processing method of calling the stored procedure and waiting for the return code and any result sets that will be returned.

The following stored procedures additionally include an encryption or decryption step for input or output and / or a step for auditing:

3.3.5.1 proc_sss_CreateApplication

The stored procedure proc_sss_CreateApplication MUST be called to create a new target application in SSS store. Before calling the stored procedure, the protocol client MUST create an encrypted claim hash for each claim (2) to be set in the *@GroupClaims* and *@TicketRedeemClaims* parameter values as specified in the following steps.

1. Create an Unencrypted claim obtained from the claim (2) information, SSS partition information and name of the target application.
2. Generate a temporary **session key** used for encryption by performing the following steps:
    1. Generate a cryptographically secure random salt of 32 bytes.
    2. Create an Encryption Session Key Seed using the salt obtained in step 2.1 in conjunction with the master secret key.
    3. Hash the Encryption Session Key Seed using SHA-256 algorithm. This will yield a 32 byte hash value.
3. Generate a cryptographically secure random salt of 32 bytes.
4. Create an Unencrypted claim hash from the salt created in step 2.1, the salt created in step 3 and the hash of the Unencrypted claim created in step 1 using SHA-256 algorithm.
5. Create an Encrypted claim hash from the Unencrypted claim hash obtained in step 4 using 256-bit Advanced Encryption Standard (AES) encryption and the key and salt generated in step 2.

Upon execution of the stored procedure proc_sss_CreateApplication, the protocol client MUST call proc_sss_InsertAudit with *@ActionType* equal to 101, *@ActionResultCode* equal to the implementation-specific result value, claim (2) that uniquely identifies the caller, the name of the target application, the SSS partition and the name of the computer where the protocol client is running.

3.3.5.2 proc_sss_DeleteAllUserCredentials

Upon execution of the stored procedure proc_sss_DeleteAllUserCredentials, the protocol client MUST call proc_sss_InsertAudit with *@ActionType* equal to 127, *@ActionResultCode* equal to the implementation-specific result value, claim (2) that uniquely identifies the caller, the name of the

target application, the SSS partition and the name of the computer where the protocol client is running.

3.3.5.3 proc_sss_DeleteApplication

Upon execution of the stored procedure proc_sss_DeleteApplication, the protocol client MUST call proc_sss_InsertAudit with *@ActionType* equal to 105, *@ActionResultCode* equal to the implementation-specific result value, claim (2) that uniquely identifies the caller, the name of the target application, the SSS partition and the name of the computer where the protocol client is running.

3.3.5.4 proc_sss_DeleteUserCredentials

Upon execution of the stored procedure proc_sss_DeleteUserCredentials, the protocol client MUST call proc_sss_InsertAudit with *@ActionType* equal to 125, *@ActionResultCode* equal to the implementation-specific result value, claim (2) that uniquely identifies the caller, the name of the target application, the SSS partition and the name of the computer where the protocol client is running.

3.3.5.5 proc_sss_GetApplicationInfo

Upon execution of the stored procedure proc_sss_GetApplicationInfo, the protocol client MUST call proc_sss_InsertAudit with *@ActionType* equal to 115, *@ActionResultCode* equal to the implementation-specific result value, claim (2) that uniquely identifies the caller, the name of the target application, the SSS partition and the name of the computer where the protocol client is running.

3.3.5.6 proc_sss_GetCredentials

The stored procedure proc_sss_GetCredentials is called to obtain the encrypted credentials to be returned to the SSS user as plain text credential after decrypting it. The protocol client MUST first obtain the claims (2) of the SSS user who is making the call, using implementation specific means.

To obtain the plain text credentials to be returned to the SSS user, the protocol client MUST perform the following steps:

1. Split the Salted Encrypted Credentials obtained by calling the stored procedure proc_sss_GetCredentials into its constituent parts of salt and encrypted credentials.
2. Generate a temporary session key used for decryption by performing the following steps:
    1. Create an Encryption Session Key Seed using the salt obtained in step 1 in conjunction with the master secret key.
    2. Hash the Encryption Session Key Seed using SHA-256 algorithm. This will yield a 32 byte hash value.
3. Generate an Unencrypted Credentials by decrypting the encrypted credentials obtained in step 1 with the temporary session key obtained in step 2 using 256-bit Advanced Encryption Standard (AES).
4. Split the Unencrypted Credentials to its constituent parts of salt and Binary serialized SecureStoreDbCredentials.
5. Obtain the claims and credentials by de-serialize, as specified in [MS-NRTP] section 3.1.5.1.6, the Binary serialized SecureStoreDbCredentials in step 4.

6. Compare the claims obtained in step 5 with the claims (2) of the SSS user. If the value of the ApplicationType retrieved by calling proc_sss_GetCredentials is equal to 0x00 or 0x02 perform step 6.1 otherwise perform step 6.2.

    1. The claim type, claim issuer and the claim value in the claim (2) that uniquely identifies the caller MUST be equal to claim type, claim issuer and claim value respectively of one of the SecureStoreServiceClaim stored in SecureStoreDbCredentials. If there is no match, an implementation specific error condition MUST be signaled. Skip to step 7.

    2. The claim type, claim issuer and the claim value of one of the claims (2) of the caller MUST be equal to claim type, claim issuer and claim value respectively of one of the SecureStoreServiceClaim stored in SecureStoreDbCredentials. If there is no match, an implementation specific error condition MUST be signaled.

7. If the claims (2) comparison is successful in step 6, credentials in SecureStoreDbCredentials is returned to the user.

If an error occur in any of the preceding steps, the protocol client MUST call proc_sss_InsertAudit with *@ActionType* equal to 132, *@ActionResultCode* equal to the implementation-specific result value, claim (2) that uniquely identifies the caller, the name of the target application, the SSS partition and the name of the computer where the protocol client is running. If there is no error in any of the preceding steps, the protocol client MUST NOT call proc_sss_InsertAudit.

3.3.5.7 proc_sss_RedeemTicket

The stored procedure proc_sss_RedeemTicket can be called to obtain the encrypted credentials using a previously generated SSS ticket. The protocol client decrypts the encrypted credentials to get the plaintext credentials that are to be returned to the SSS user.

In this case the protocol client MUST obtain the claims (2) of the caller, using implementation specific means, to verify that the caller can redeem SSS ticket for the specified target application.

Before calling the stored procedure, the protocol client MUST obtain the claim (2) that uniquely identifies the SSS user who generated the SSS ticket and the Random Ticket that are stored in the SSS ticket. To obtain the claim (2) that uniquely identifies the SSS user who generated the SSS ticket and the Random Ticket perform the following steps.

1. Obtain the Final SSS Ticket from the caller.

2. Split the Final SSS Ticket into its constituent parts of salt and encrypted ticket.

3. Generate a temporary session key used for encryption by performing the following steps:

    1. Create an Encryption Session Key Seed using the salt obtained in step 2 in conjunction with the master secret key.

    2. Hash the Encryption Session Key Seed using SHA-256 algorithm. This will yield a 32 byte hash value.

4. Generate an Unencrypted Ticket by decrypting the encrypted ticket obtained in step 2 with the temporary session key obtained in step 3 using 256-bit Advanced Encryption Standard (AES).

5. Split the Unencrypted Ticket obtained in step 4 into salt and Binary serialized SecureStoreTicket.

6. Get the SecureStoreTicket by de-serializing, as specified in [MS-NRTP] section 3.1.5.1.6, the Binary serialized SecureStoreTicket obtained in step 5. The SecureStoreTicket contains the claim

(2) that uniquely identifies the SSS user who generated the SSS ticket and Random Ticket stored in as ticket.

The stored procedure proc_sss_RedeemTicket is called with the Random Ticket and claim (2) obtained in step 6 to get the Salted Encrypted Credentials along with the claims (2) information about who can redeem the SSS ticket for the specified target application.

The protocol client MUST make sure that the caller has at least one claim (2) that is equal to one of the claims (2) obtained in the result set Application Group Claims Result Set by calling the stored procedure proc_sss_RedeemTicket.

To obtain the plaintext credentials to be returned to the caller, the protocol client MUST subsequently perform the following steps in the following order:

1. Split the Salted Encrypted Credentials obtained by calling the stored procedure proc_sss_RedeemTicket into its constituent parts of salt and encrypted credentials.
2. Generate a temporary session key used for encryption by performing the following steps:
    1. Create an Encryption Session Key Seed using the salt obtained in step 7 in conjunction with the master secret key.
    2. Hash the Encryption Session Key Seed using SHA-256 algorithm. This will yield a 32 byte hash value.
3. Generate an Unencrypted Credentials by decrypting the encrypted credentials obtained in step 7 with the temporary session key obtained in step 8 using 256-bit Advanced Encryption Standard (AES).
4. Split the Unencrypted Credentials to its constituent parts of salt and Binary serialized SecureStoreDbCredentials.
5. Get the SecureStoreDbCredentials by de-serializing, as specified in [MS-NRTP] section 3.1.5.1.6, the Binary serialized SecureStoreDbCredentials obtained in step 10.
6. Compare the claims (2) stored in the SecureStoreDbCredentials, obtained in step 11, with the claims (2) in SecureStoreTicket obtained in step6. If the value of the ApplicationType retrieved by calling proc_sss_RedeemTicket is equal to 0x02 perform step 12.1 otherwise perform step 12.2.
    1. The claim type, claim issuer and the claim value in the claim (2) obtained in step 6 MUST be equal to claim type, claim issuer and claim value respectively of one of the SecureStoreServiceClaim stored in SecureStoreDbCredentials. If there is no match, an implementation specific error condition MUST be signaled. If there is match, skip to step 13.
    2. The claim type, claim issuer and the claim value of one of the claims in SecureStoreTicket obtained in step 6 MUST be equal to claim type, claim issuer and claim value respectively of one of the SecureStoreServiceClaim stored in SecureStoreDbCredentials. If there is no match, an implementation specific error condition MUST be signaled.
7. If the claims (2) comparison is successful in step 12, credentials in SecureStoreDbCredentials are returned to the SSS user.

If an error occurs in any of the preceding steps, the protocol client MUST call proc_sss_InsertAudit with *@ActionType* equal to 130, *@ActionResultCode* equal to the implementation-specific result value, claim (2) that uniquely identifies the caller, the name of the target application, the SSS partition and the name of the computer where the protocol client is running.

3.3.5.8 proc_sss_SetCredentials

The stored procedure proc_sss_SetCredentials MUST be called to insert or update the encrypted credentials provided by an SSS user for a specified target application.

To encrypt the credentials before calling the stored procedure, the protocol client MUST:

1. Get a List<T> of SerializableSecureStoreCredential (section 2.2.2.2) from the user.
2. Get the claim (2) that uniquely identifies the caller and create a List<T> of SerializableSecureStoreCredential with it, if the specified target application is an individual target application. If the specified target application is not an individual target application, the value of the claim (2) that uniquely identifies the caller MUST be set to Null Object as specified in [MS-NRTP] section 1.1.
3. Generate a temporary session key used for encryption by performing the following steps:
    1. Generate a cryptographically secure random salt of 32 bytes.
    2. Create an Encryption Session Key Seed using the salt obtained in step 3.1 in conjunction with the master secret key.
    3. Hash the Encryption Session Key Seed using SHA-256 algorithm. This will yield a 32 byte hash value.
4. Generate a cryptographically secure random salt of 32 bytes.
5. Create an Unencrypted Credentials using the List<T> of SerializableSecureStoreCredential and List<T> of SecureStoreServiceClaim obtained in step 1 and 2 respectively along with the salt generated in step 3.1 and the salt generated in step 4.
6. Generate a Salted Encrypted Credentials from the Unencrypted Credentials obtained in step 5 using 256-bit Advanced Encryption Standard (AES) encryption and the key generated in step 3.

Upon execution of the stored procedure, the protocol client MUST call proc_sss_InsertAudit with *@ActionType* equal to 136, *@ActionResultCode* equal to the implementation-specific result value, claim (2) that uniquely identifies the caller, the name of the target application, the SSS partition and the name of the computer where the protocol client is running.

3.3.5.9 proc_sss_SetMasterSecretKey

If the protocol server does not have a stored master secret key, or if the user has indicated that the protocol server's copy of the encrypted master secret key is to be changed to a new master secret key without re-encrypting the contents of the SSS store with the new master secret key, the client side of **proc_sss_SetMasterSecretKey** is merely a pass through to the protocol server. Otherwise, the protocol client MUST re-encrypt the contents of the SSS store when changing the master secret key in the following manner:

1. Consider the existing protocol client's local cached master secret key as M1. Acquire a new master secret key M2 from the caller.
2. Execute **proc_sss_SetStatus** (section 3.2.5.32) with *@Status* equal to one and *@Owner* equal to client protocol identifier.
3. Execute **proc_sss_PrepareSecondaryTable** (section 3.2.5.23) stored procedure.

4. Obtain an implementation specific number of rows from the **SSSCredentials** table (section 2.2.7.1) by calling **proc_GetCredentialsPage** (section 3.2.5.1).

5. Decrypt the credentials in each row of the result set Paged Credentials Result Set (section 2.2.6.1) that is retrieved by calling proc_GetCredentialsPage as follows:

    1. Split the Salted Encrypted Credentials (section 2.2.5.9) in credentials into its constituent parts of salt and encrypted credentials.

    2. Generate a temporary session key used for decryption by performing the following steps:

        1. Create an Encryption Session Key Seed (section 2.2.5.1) using the salt obtained in step 5.1 in conjunction with the old master secret key (M1).

        2. Hash the Encryption Session Key Seed using SHA-256 algorithm. This will yield a 32 byte hash value.

    3. Generate an Unencrypted Credentials (section 2.2.5.8) by decrypting the encrypted credentials obtained in step 1 with the temporary session key obtained in step 5.2 using 256-bit Advanced Encryption Standard (AES).

    4. Split the Unencrypted Credentials to its constituent parts of salt and the Binary serialized SecureStoreDbCredentials element.

6. Encrypt the plain text Binary serialized SecureStoreDbCredentials obtained in step 5.4 with the new master secret key (M2) obtained in step 2 as follows.

    1. Generate a temporary session key used for encryption by performing the following steps:

        1. Generate a cryptographically secure random salt of 32 bytes.

        2. Create an Encryption Session Key Seed using the salt obtained in step 6.1.1 in conjunction with the new master secret key (M2).

        3. Hash the Encryption Session Key Seed using SHA-256 algorithm. This will yield a 32 byte hash value.

    2. Generate a cryptographically secure random salt of 32 bytes.

    3. Create an Unencrypted Credentials using Binary serialized SecureStoreDbCredentials obtained in step 5.4 along with the salt generated in step 6.1.1 and the salt generated in step 6.2.

    4. Generate a Salted Encrypted Credentials from the Unencrypted Credentials obtained in step 6.2 using 256-bit Advanced Encryption Standard (AES) encryption and the key generated in step 6.1.

7. Insert the row with the re-encrypted credentials to the SSSCredentials_Secondary table (section 2.2.7.6).

8. Perform steps 5 through 7 until all the rows of credentials retrieved by calling proc_GetCredentialsPage are re-encrypted and stored in the SSSCredentials_Secondary table.

9. Perform steps 4 through 8 until all the credentials from the SSSCredentials table are retrieved by calling proc_GetCredentialsPage**.**

10. Obtain an implementation specific number of rows from the SSSApplicationGroupClaim table (section 2.2.7.2) by calling proc_sss_GetGroupClaimsPage (section 3.2.5.16).

11. Decrypt the **ClaimValueHash** in each row of the result set Paged Group Claims Result Set (section 2.2.6.8) that is retrieved from calling proc_sss_GetGroupClaimsPage as follows:

    1. Split the Encrypted claim hash (section 2.2.5.4) in **ClaimValueHash** into its constituent parts of salt and encrypted Unencrypted claim hash (section 2.2.5.3).

    2. Generate a temporary session key used for decryption by performing the following steps:

        1. Create an Encryption Session Key Seed using the salt obtained in step 11.1 in conjunction with the old master secret key (M1).

        2. Hash the Encryption Session Key Seed using SHA-256 algorithm. This will yield a 32 byte hash value.

    3. Generate an Unencrypted claim hash by decrypting the encrypted Unencrypted claim hash obtained in step 11.1 with the temporary session key obtained in step 11.2 using 256-bit Advanced Encryption Standard (AES).

12. Encrypt the plain text Unencrypted claim hash in step 11.3 with the new master secret key (M2) as follows:

    1. Generate a temporary session key used for encryption by performing the following steps:

        1. Generate a cryptographically secure random salt of 32 bytes.

        2. Create an Encryption Session Key Seed using the salt obtained in step 12.1.1 in conjunction with the master secret key.

        3. Hash the Encryption Session Key Seed using SHA-256 algorithm. This will yield a 32 byte hash value.

    2. Generate a cryptographically secure random salt of 32 bytes.

    3. Create an Unencrypted claim hash from the salt created in step 12.1.1, the salt created in step 12.2 and the Unencrypted claim **hash** obtained in step 11.3.

    4. Create an Encrypted claim hash from the Unencrypted claim hash obtained in step 12.3 using 256-bit Advanced Encryption Standard (AES) encryption and the key and salt generated in step 12.1.

13. Insert the row with the re-encrypted **ClaimValueHash** to the SSSApplicationGroupClaim_Secondary table (section 2.2.7.4).

14. Perform steps 11 through 13 until all the rows of claims retrieved are re-encrypted and stored in the SSSApplicationGroupClaim_Secondary table.

15. Perform steps 10 through 14 until all the claims from the proc_sss_GetGroupClaimsPage table are retrieved by calling proc_sss_GetGroupClaimsPage**.**

16. Obtain an implementation specific number of rows from the SSSApplicationTicketRedeemerClaim table (section 2.2.7.3) by calling proc_sss_GetTicketRedeemerClaimsPage (section 3.2.5.20).

17. Decrypt the **ClaimValueHash** in each row of the result set Paged Group Claims Result Set that is retrieved from calling proc_sss_GetTicketRedeemerClaimsPage as follows:

    1. Split the Encrypted claim hash in **ClaimValueHash** into its constituent parts of salt and encrypted Unencrypted claim hash.

2. Generate a temporary session key used for decryption by performing the following steps:

    1. Create an Encryption Session Key Seed using the salt obtained in step 17.1 in conjunction with the old master secret key (M1).

    2. Hash the Encryption Session Key Seed using SHA-256 algorithm. This will yield a 32 byte hash value.

3. Generate an Unencrypted claim hash by decrypting the encrypted Unencrypted claim hash obtained in step 17.1 with the temporary session key obtained in step 17.2 using 256-bit Advanced Encryption Standard (AES).

18. Encrypt the plain text Unencrypted claim hash in step 17.3 with the new master secret key (M2) as follows:

    1. Generate a temporary session key used for encryption by performing the following steps:

        1. Generate a cryptographically secure random salt of 32 bytes.

        2. Create an Encryption Session Key Seed using the salt obtained in step 18.1.1 in conjunction with the master secret key.

        3. Hash the Encryption Session Key Seed using SHA-256 algorithm. This will yield a 32 byte hash value.

    2. Generate a cryptographically secure random salt of 32 bytes.

    3. Create an Unencrypted claim hash from the salt created in step 18.1.1, the salt created in step 18.2 and the Unencrypted claim **hash** obtained in step 17.3.

    4. Create an Encrypted claim hash from the Unencrypted claim hash obtained in step 18.3 using 256-bit Advanced Encryption Standard (AES) encryption and the key and salt generated in step 18.1.

19. Insert the row with the re-encrypted **ClaimValueHash** to the SSSApplicationTicketRedeemerClaim_Secondary table (section 2.2.7.5).

20. Perform steps 17 through 19 until all the rows of claim retrieved are re-encrypted and stored in the SSSApplicationTicketRedeemerClaim_Secondary table.

21. Perform steps 16 through 20 until all the claims from the SSSApplicationTicketRedeemerClaim table are retrieved by calling proc_sss_GetTicketRedeemerClaimsPage**.**

22. Encrypt the new master secret key M2 using implementation specific means and execute proc_sss_SetMasterSecretKey to store the new key for distribution purposes.

23. Execute proc_sss_SetStatus with *@Status* equal to zero and *@Owner* equal to client protocol identifier.

3.3.5.10 proc_sss_SetTicket

Before calling the stored procedure proc_sss_SetTicket, the protocol client generates a Random Ticket to pass in as input. The protocol client MUST then perform the following steps to generate a Final SSS Ticket which can be used by the protocol client at a later stage when calling proc_sss_RedeemTicket.

1. Generate a temporary session key used for encryption by performing the following steps:

1. Generate a cryptographically secure random salt of 32 bytes.
2. Create an Encryption Session Key Seed using the salt obtained in step 1.1 in conjunction with the master secret key.
3. Hash the Encryption Session Key Seed using SHA-256 algorithm. This will yield a 32 byte hash value.

2. Create a Random Ticket.
3. Generate a cryptographically secure random salt of 32 bytes.
4. Create an Unencrypted Ticket using the Random Ticket obtained in step 2, the salt generated in step 1.1 and the salt generated in step 3.
5. Generate a Final SSS Ticket from the Unencrypted Ticket obtained in step 4 using 256-bit Advanced Encryption Standard (AES) encryption and the key generated in step 1.

If an error occurs in any of the preceding steps, the protocol client MUST call proc_sss_InsertAudit with *@ActionType* equal to 128, *@ActionResultCode* equal to the implementation-specific result value, claim (2) that uniquely identifies the caller, the name of the target application, the SSS partition and the name of the computer where the protocol client is running.

3.3.6 Timer Events

When the SSS ticket expiration timer timeout event is triggered, the timer event handler MUST call proc_sss_PurgeTickets.

When the SSS audit entry purge timer timeout event is triggered, the timer event handler MUST call proc_sss_DeleteAuditRecords.

3.3.7 Other Local Events

None.

4 Protocol Examples

4.1 Example 1: Create Target Application

This example describes the requests that are made to create a new target application in the specified partition. In this case, the target application will be an individual type with two fields: user name and password.

The requests that are made to create a target application use the stored procedure **proc_sss_CreateApplication**.

```
EXECUTE @RC = proc_sss_CreateApplication
   @ApplicationName = N'MyTargetApplication'
  ,@FriendlyName = N'My Secure Target Application'
  ,@PartitionId = '0C37852B-34D0-418E-91C6-2AC25AF4BE5B'
  ,@ApplicationType = 0
  ,@TicketTimeout = 0
  ,@ContactEmail = 'userAdmin@contoso.com'
  ,@CredentialManagementUrl = 'https://server.contoso.com/credentials.aspx/'
  ,@FieldInfo = '<Fields><Field id="0" ismasked="0" credentialtype="0" name="User
Name"/><Field id="1" ismasked="1" credentialtype="1" name="Password"/></Fields>'
  ,@AdminClaims = '<Claims><Claim
claimType="http://schemas.microsoft.com/sharepoint/2009/08/claims/userlogonname"
claimIssuer="localhost" claimValue="CONTOSO\john"/></Claims>'
  ,@GroupClaims = NULL
  ,@TicketRedeemClaims = NULL
  ,@CheckSum = [checksum of master secret key]
```

4.2 Example 2: Delete Target Application

This example describes the requests that are made to delete the target application with the ID "MyTargetApplication".

The requests that are made to delete a target application use the stored procedure **proc_sss_DeleteApplication**.

```
EXECUTE @RC = proc_sss_DeleteApplication
   @ApplicationName = N'MyTargetApplication'
  ,@PartitionId = '0C37852B-34D0-418E-91C6-2AC25AF4BE5B'
  ,@CurrentUserClaims = '<Claims><Claim
claimType="http://schemas.microsoft.com/sharepoint/2009/08/claims/userlogonname"
claimIssuer="localhost" claimValue="CONTOSO\john"/></Claims>'
  ,@VerifyAdminClaims = 0
```

4.3 Example 3: Set Credentials

This example describes the requests that are made to set the credentials for user "CONTOSO\john" in the target application with ID "MyTargetApplication".

The requests that are made to set the credentials for a user use the stored procedure **proc_sss_SetCredentials**.

```
EXECUTE @RC = proc_sss_SetCredentials
   @ApplicationName = N'MyTargetApplication'
```

```
  ,@PartitionId = '0C37852B-34D0-418E-91C6-2AC25AF4BE5B'
  ,@IdentityClaimType =
N'http://schemas.microsoft.com/sharepoint/2009/08/claims/userlogonname'
  ,@IdentityClaimIssuer = N'localhost'
  ,@IdentityClaimValue = N'CONTOSO\john'
  ,@IdentityClaimValueHash = [hash of the user identity]
  ,@Credentials = [encrypted credentials]
  ,@GroupCredentials = 0
  ,@CurrentUserClaims = '<Claims><Claim
claimType="http://schemas.microsoft.com/sharepoint/2009/08/claims/userlogonname"
claimIssuer="localhost" claimValue="CONTOSO\john"/></Claims>'
  ,@VerifyAdminClaims = 0
  ,@CheckSum = [checksum of this record]
```

4.4 Example 4: Get Credentials

This example describes the requests that are made to get the credentials stored in the target application with the ID "MyTargetApplication". The server protocol will respond with the credentials associated with the current user, in this case "CONTOSO\john".

The requests that are made to get the credentials for a user use the stored procedure **proc_sss_GetCredentials**.

```
EXECUTE @RC = proc_sss_GetCredentials
   @ApplicationName =  N'MyTargetApplication'
  ,@PartitionId = '0C37852B-34D0-418E-91C6-2AC25AF4BE5B'
  ,@IdentityClaimType =
N'http://schemas.microsoft.com/sharepoint/2009/08/claims/userlogonname'
  ,@IdentityClaimIssuer = N'localhost'
  ,@IdentityClaimValue = N'CONTOSO\john'
  ,@IdentityClaimValueHash = [hash of the user identity]
  ,@Machine = N'server1'
  ,@CredentialManagementUrl OUTPUT
```

The protocol server will return the ApplicationType and the encrypted Credentials.

4.5 Example 5: Update Target Application

This example describes the requests that are made to update the information for the target application with the ID "MyTargetApplication".

The requests that are made to update a target application use the stored procedure **proc_sss_UpdateApplication**.

```
EXECUTE @RC = proc_sss_UpdateApplication
   @ApplicationName = N'MyTargetApplication'
  ,@FriendlyName = N'My Secure Target Application'
  ,@PartitionId = '0C37852B-34D0-418E-91C6-2AC25AF4BE5B'
  ,@ApplicationType = 0
  ,@TicketTimeout = 0
  ,@ContactEmail = 'userAdmin@contoso.com'
  ,@CredentialManagementUrl = 'https://server.contoso.com/credentials.aspx/'
  ,@FieldInfo = '<Fields><Field id="0" ismasked="0" credentialtype="0" name="User
Name"/><Field id="1" ismasked="1" credentialtype="1" name="Password"/></Fields>'
```

```
  ,@AdminClaims = '<Claims><Claim
claimType="http://schemas.microsoft.com/sharepoint/2009/08/claims/userlogonname"
claimIssuer="localhost" claimValue="CONTOSO\john"/></Claims>'
  ,@GroupClaims = NULL
  ,@TicketRedeemClaims = NULL
  ,@Credentials = NULL
  ,@CurrentUserClaims = '<Claims><Claim
claimType="http://schemas.microsoft.com/sharepoint/2009/08/claims/userlogonname"
claimIssuer="localhost" claimValue="CONTOSO\john"/></Claims>'
  ,@VerifyAdminClaims = 0
  ,@CheckSum = [checksum of the master secret key]
```

5 Security

5.1 Security Considerations for Implementers

General security considerations pertaining to SHA-256 and 256-bit Advanced Encryption Standard (AES) cryptographic algorithms apply.

Interactions with SQL are susceptible to tampering and other forms of security risks. Implementers are advised to sanitize input parameters for stored procedures before invoking the stored procedure.

Protocol clients are advised to provide implementation specific authorization checks to determine the set of security principals that can call each stored procedure.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® SharePoint® Server 2013 Preview

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index

R

S

T

U

V

X