[MS-SSOSP]:
Single Sign-On Database 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's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). 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.

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability
06/27/2008	1.0	Major	Revised and edited the technical content
08/15/2008	1.01	Editorial	Revised and edited the technical content
12/12/2008	1.02	Editorial	Revised and edited the technical content
07/13/2009	1.03	Major	Changes made for template compliance
08/28/2009	1.04	Editorial	Revised and edited the technical content
11/06/2009	1.05	Editorial	Revised and edited the technical content
02/19/2010	2.0	Editorial	Revised and edited the technical content
03/31/2010	2.01	Editorial	Revised and edited the technical content
04/30/2010	2.02	Editorial	Revised and edited the technical content
06/07/2010	2.03	Editorial	Revised and edited the technical content
06/29/2010	2.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	2.04	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies the Single Sign-On Database Protocol. This protocol specifies an interface for protocol clients to store and retrieve authentication credentials and related information for line-of-business systems.

1.1 Glossary

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

base64
credential
salt
Security Support Provider Interface (SSPI)
Unicode

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

back-end database server
line-of-business (LOB) system
master secret
master secret server
result set
return code
security principal
session key
single sign-on (SSO) ticket
stored procedure
string SID
Transact-Structured Query Language (T-SQL)

The following terms are specific to this document:

credential field: An atomic unit of information, such as a login name, that is used to authenticate a principal with a system.

group credentials: A set of credentials that are associated with a security principal (2) that represents more than one single sign-on (SSO) user.

single sign-on (SSO) administrator: A security principal (2) who is authorized to change a single sign-on (SSO) configuration and to obtain master secrets from a master secret server.

SSO action: Any operation that is executed through the Single Sign-On (SSO) Database Protocol, such as retrieving credentials or creating an SSO application.

SSO application: A logical entity that represents a software system for which credentials are maintained, such as a line-of-business (LOB) system.

SSO application manager: A security principal (2) who is authorized to create, read, update, and delete single sign-on (SSO) applications.

SSO audit entry: A record that stores information about a single sign-on (SSO) action, including when it was performed, whether it succeeded, why it failed if it didn't succeed, the SSO user who performed it, and optionally the SSO user on whose behalf it was performed.

SSO store: A database that contains the stored procedures for and provides storage for SSO applications and credentials. It is stored on a back-end database server.

validation bytes: A fixed sequence of constant bytes that is used during validation operations.

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

1.2 References

1.2.1 Normative References

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

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

[MS-SQL] 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://msdn.microsoft.com/en-us/library/dd631854(v=SQL.10).aspx

[MS-SSP] Microsoft Corporation, "Single Sign-On 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

1.2.2 Informative References

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

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

1.3 Protocol Overview (Synopsis)

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 (2)**. A set of **credentials** is required as input before a user is allowed to access the LOB system.

It is common for modern business applications to deliver functionality that requires data to be manipulated in more than a single software system. 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 **SSO applications** as well as the actual credentials for each user for each LOB system. Then an integrated application that spans multiple systems can programmatically obtain the credentials of the current SSO user from the store and authenticate without prompting the SSO user

for credentials each time a particular LOB system demands authentication. The SSO user never needs to authenticate more than once.

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

This protocol allows protocol clients to create, read, update and delete SSO application definitions in a **back-end database server**. It additionally allows a protocol client to create, read, update and delete the credentials associated with each SSO application and to encrypt this information to keep it secure. 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

This protocol relies on [MS-SSP] to retrieve the **master secret** used as the symmetric cryptographic key for the encryption and decryption operations.

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

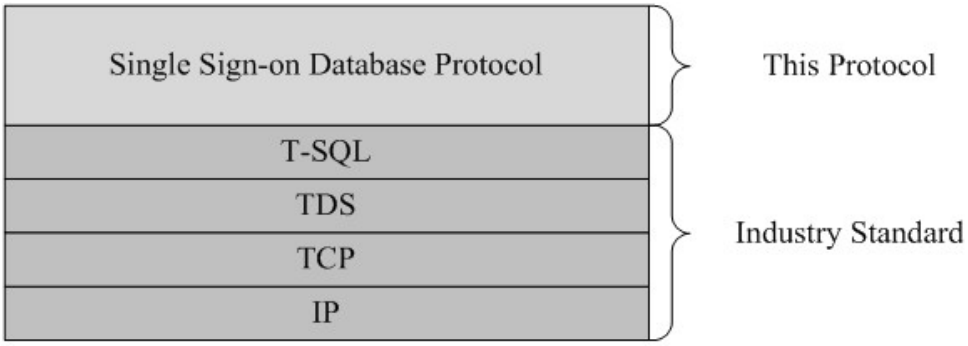


Figure 1: This protocol in relation to other protocols

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 databases.

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

This protocol requires that the protocol client has appropriate permissions to retrieve the master secret from the **master secret server**.

1.6 Applicability Statement

This protocol is intended for use by protocol clients and protocol servers 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 **Security Support Provider Interface (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

[MS-TDS] is the transport protocol used to call the stored procedures, return result codes, and return 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

None.

2.2.2 Common Fields

2.2.2.1 Type

Type: int NOT NULL. The type of an SSO application. It MUST be a value listed in the following table.

Value	Description
0	An SSO application that stores credentials for security principals (2) that represent individual SSO users. The credentials are used by users that do not perform their own authorization against the data retrieved from the system to which the credentials are used to authenticate.
1	An SSO application that stores credentials for security principals (2) that represent a group of SSO users. The credentials are used by users that do not perform their own authorization against the data stored or retrieved from the system to which the credentials are used to authenticate.
2	An SSO application that stores credentials for security principals (2) that represent individual SSO users. The credentials are used by users that do not perform their own authorization against the data stored or retrieved from the system to which the credentials are used to authenticate. The credentials stored MUST represent Windows credentials.
3	An SSO application that stores credentials for security principals (2) that represent a group of SSO users. The credentials are used by users that do not perform their own authorization against the data stored or retrieved from the system to which the credentials are used to authenticate. The credentials stored MUST represent Windows credentials.
4	An SSO application that stores credentials for security principals (2) that represent a group of SSO users. The credentials are used by users that perform their own authorization against the data stored or retrieved from the system to which the credentials are used to authenticate.
5	An SSO application that stores credentials for security principals (2) that represent a group of SSO users. The credentials are used by users that perform their own authorization against the data stored or retrieved from the system to which the credentials are used to authenticate. The credentials stored MUST represent Windows credentials.

2.2.2.2 MyVer

MyVer: datetime NOT NULL. A timestamp indicating the date and time at which the SSO configuration was last stored or updated in the **SSO store**.

2.2.2.3 TicketTimeoutMin

TicketTimeoutMin: int NOT NULL. An integer value that denotes how long an **SSO ticket** is deemed valid after creation, measured in minutes. It is part of the SSO configuration.

2.2.2.4 PurgeAuditDays

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

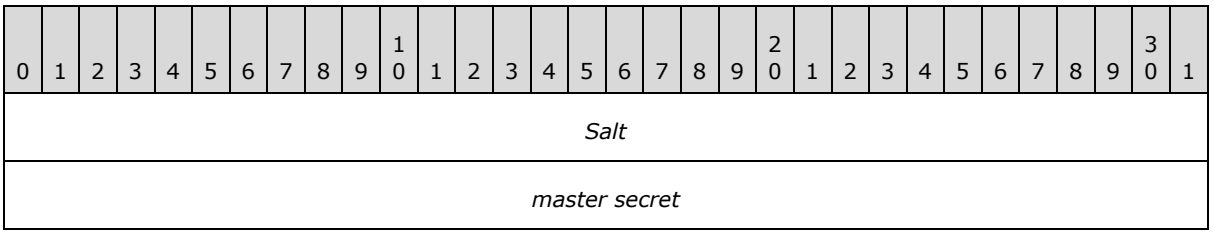
2.2.3 Bit Fields and Flag Structures

None.

2.2.4 Binary Structures

2.2.4.1 Ticket Encryption Session Key Seed

The sequence of bytes that will be hashed using the Secure Hash (SHA) algorithm to generate the **session key** used for encrypting the contents of an SSO ticket.

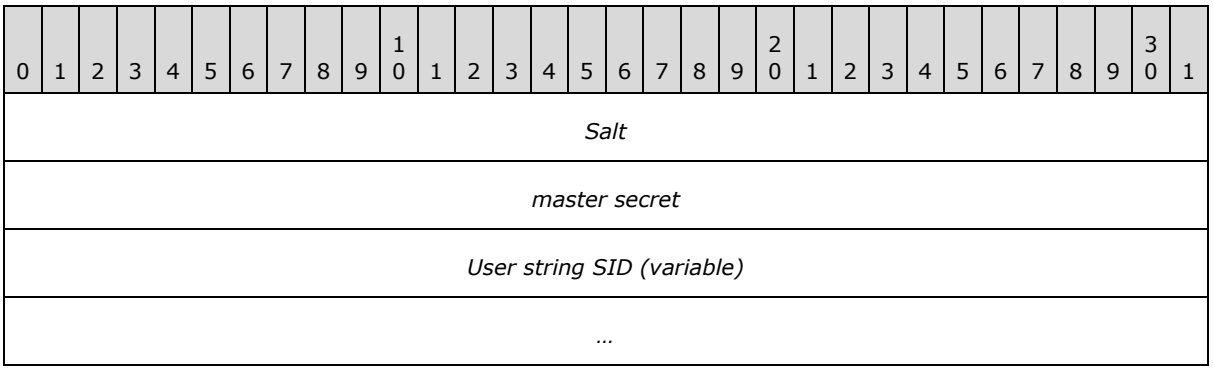


Salt: 16 byte **salt**.

master secret: 16 byte master secret.

2.2.4.2 Credential Encryption Session Key Seed

The sequence of bytes that will be hashed using the SHA algorithm to generate the session key used for encrypting a set of user credentials.



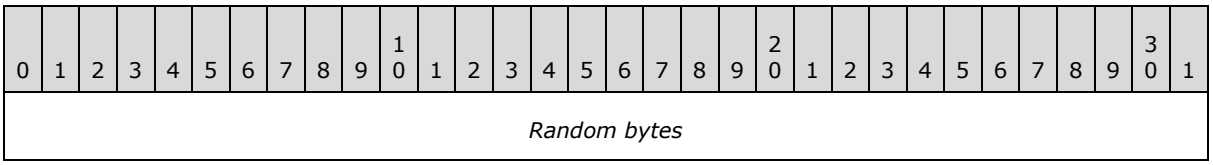
Salt: 4 byte salt.

master secret: 16 byte master secret.

User string SID: A **string SID** in the form of a WCHAR with the terminating NULL character.

2.2.4.3 Random Ticket

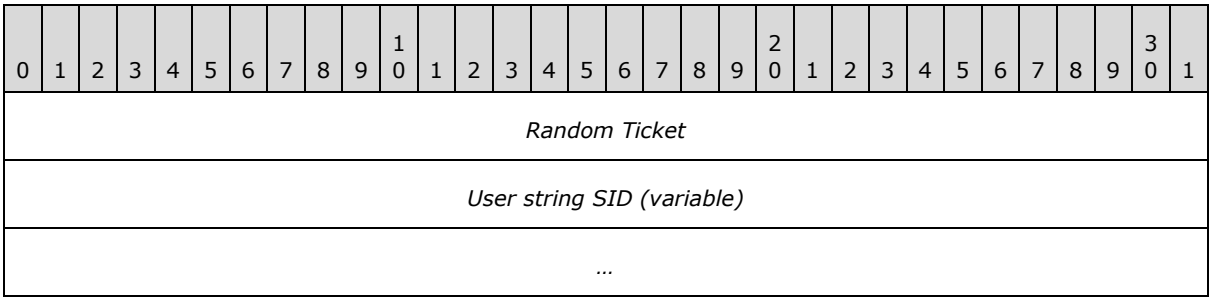
A sequence of 16 cryptographically random bytes.



Random bytes: 16 cryptographically random bytes.

2.2.4.4 Unencrypted Ticket

The sequence of bytes that make up the plain text contents of an SSO ticket.

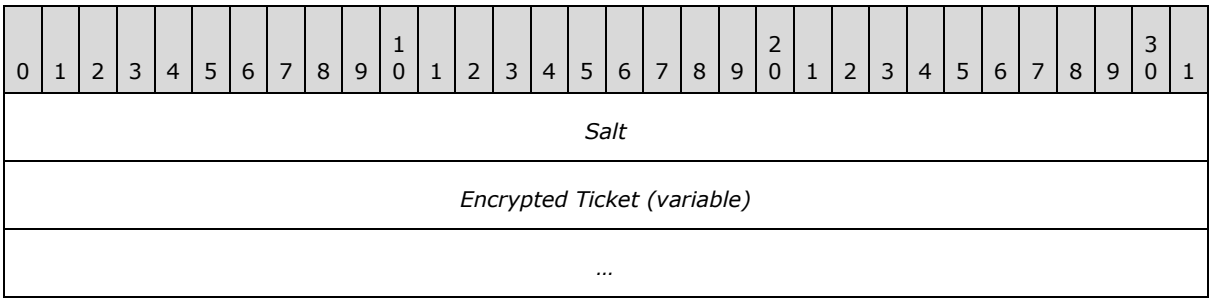


Random Ticket: A Random Ticket.

User string SID: The string SID of an SSO user in the form of a WCHAR without the terminating NULL character.

2.2.4.5 Salted Encrypted Ticket

The sequence of bytes obtained after encrypting an Unencrypted Ticket and prefixing it with a salt.

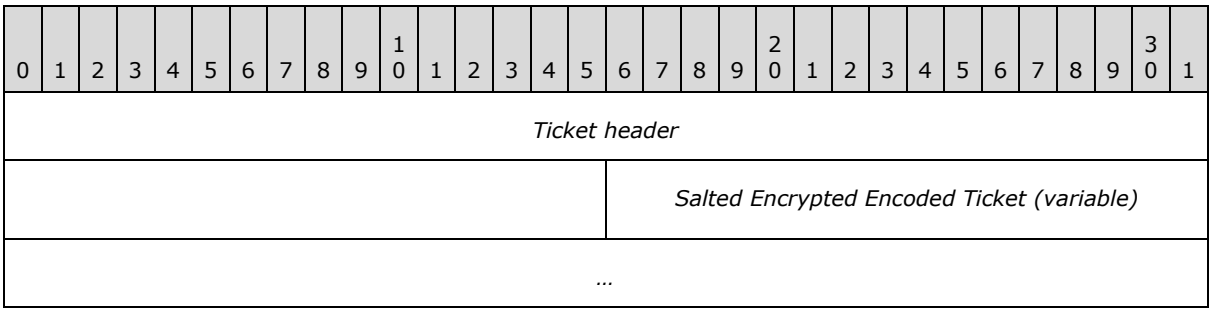


Salt: 16 byte salt. This MUST be the same salt used to create the Ticket Encryption Session Key Seed.

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

2.2.4.6 Final SSO Ticket

The sequence of bytes that make up the final SSO ticket for transmission to other components which can later redeem it for credentials.

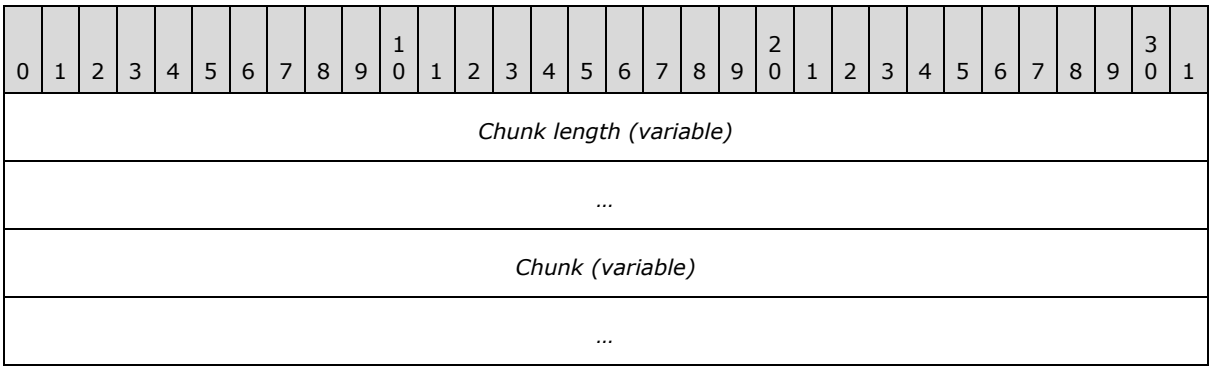


Ticket header: 6 header bytes which MUST be 0x33, 0x71, 0x65, 0x56, 0x70, 0x51.

Salted Encrypted Encoded Ticket: The bytes obtained by Base-64 encoding a Salted Encrypted Ticket.

2.2.4.7 Credential Chunk

The sequence of bytes that make up a single plain text field in credentials.

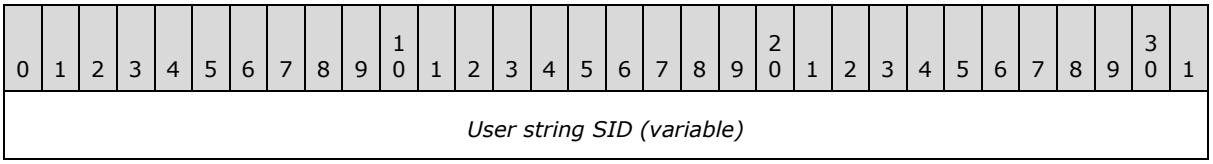


Chunk length: The length of the following chunk in bytes. The integer value is encoded as a WCHAR string that has a terminating NULL character. This MUST be zero if the **credential field** is unused.

Chunk: Plain text credential field. If the Chunk length is zero, the Chunk MUST be the two bytes 0x00, 0x00.

2.2.4.8 Unencrypted Credentials

The sequence of bytes that make up the plain text contents of credentials.



…		
…	*0x00*	*0x00*
Chunk count	*0x00*	*0x00*
Credential Chunks (variable)		
…		

User string SID: A string SID in the form of a WCHAR without the terminating NULL character.

Chunk count: An integer count of the number of Credential Chunks that follow in the form of a WCHAR without the terminating NULL character. The value MUST be 5.

Credential Chunks: A sequence of Credential Chunks.

2.2.4.9 Salted Encrypted Credentials

The sequence of bytes obtained after encrypting Unencrypted Credentials and prefixing it with a salt.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Salt																															
Encrypted Credentials (variable)																															
…																															

Salt: 4 byte salt. This MUST be the same salt used to create the Credential Encryption Session Key Seed.

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

2.2.4.10 SSO Administrator Encryption Session Key Seed

The sequence of bytes that will be hashed using the SHA algorithm to generate the session key used for encrypting the **SSO administrator** configuration setting.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Salt																															
																														Master Secret	

			0xa9
0xe0	*0xd0*	*0x3b*	*0xfb*
0xdb	*0xe3*	*0x52*	*0x70*

Salt: 7 byte salt.

Master Secret: 16 byte master secret.

2.2.4.11 Unencrypted SSO Administrator Configuration Setting

The sequence of bytes that make up the plain text SSO administrator string SID.

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
Configuration setting name																															
SSO administrator SID																															
…																															

Configuration setting name: 20 byte **Unicode** representation of the literal string ''AdminGroup''.

SSO administrator SID: The string SID of the SSO administrator encoded as a WCHAR string terminated by a NULL character.

2.2.4.12 Validated Salted Encrypted SSO Administrator Configuration Setting

The sequence of bytes obtained after encrypting Unencrypted SSO Administrator Configuration Setting and prefixing it with **validation bytes** and 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
Validation bytes																															
Salt																															
																									Encrypted SSO Administrator Configuration Setting (variable)							
…																															

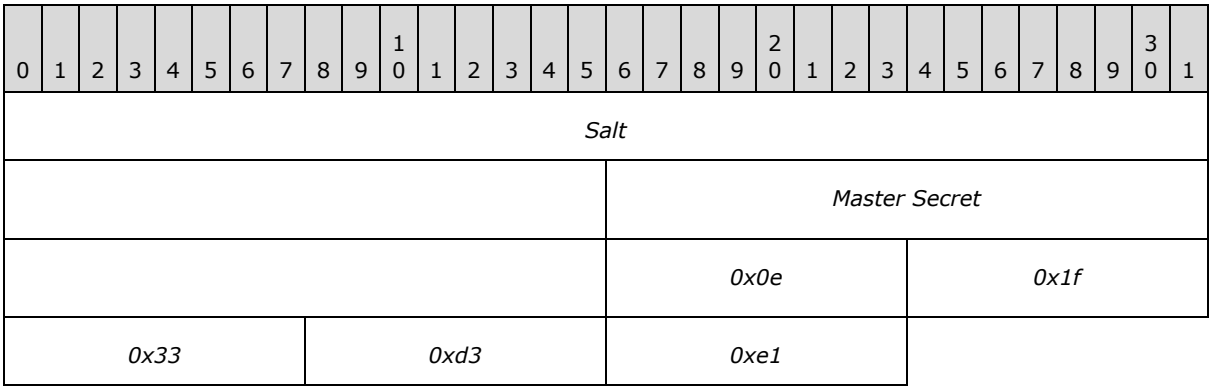
Validation bytes: validation bytes 0xca, 0x7e, 0x92, 0x5b.

Salt: 7 byte salt. This MUST be the same salt used to create the SSO Administrator Encryption Session Key Seed.

Encrypted SSO Administrator Configuration Setting: The sequence of bytes obtained by encrypting Unencrypted SSO Administrator Configuration Setting.

2.2.4.13 SSO Application Manager Encryption Session Key Seed

The sequence of bytes that will be hashed using the SHA algorithm to generate the session key used for encrypting the **SSO application manager** configuration setting.

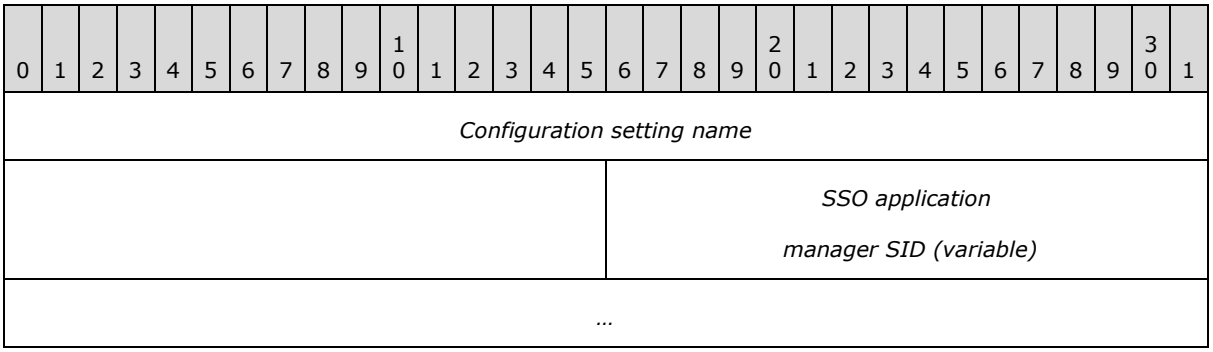


Salt: 10 byte salt.

Master Secret: 16 byte master secret.

2.2.4.14 Unencrypted SSO Application Manager Configuration Setting

The sequence of bytes that make up the plain text SSO application manager string SID.

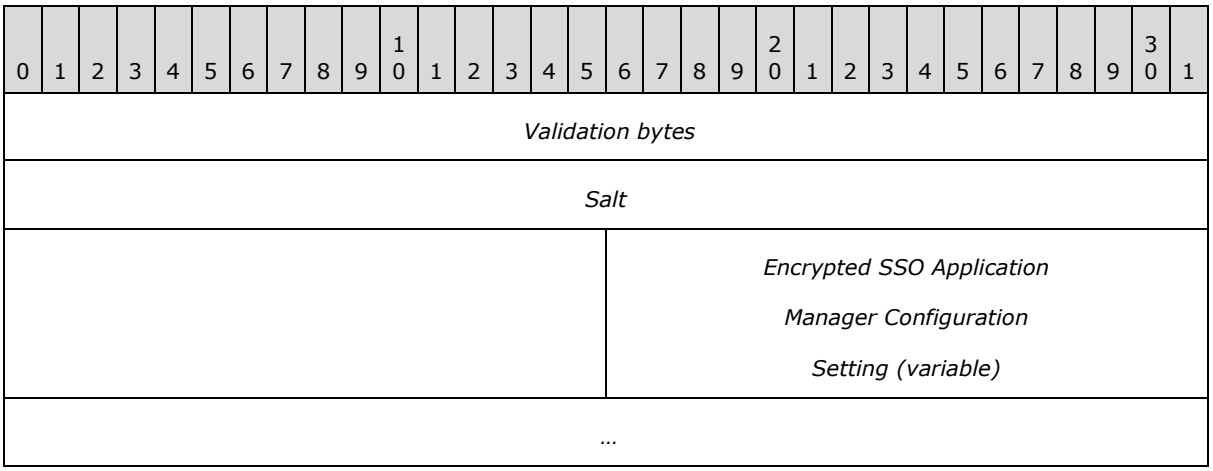


Configuration setting name: 46 byte Unicode representation of the literal string "ApplicationManagerGroup".

SSO application manager SID: The string SID of the SSO application manager encoded as a WCHAR string terminated by a NULL character.

2.2.4.15 Validated Salted Encrypted SSO Application Manager Configuration Setting

The sequence of bytes obtained after encrypting Unencrypted SSO Application Manager Configuration Setting and prefixing it with a salt.



Validation bytes: validation bytes 0x28, 0x40, 0xb6, 0xa8.

Salt: 10 byte salt. This MUST be the same salt used to create the SSO Application Manager Encryption Session Key Seed.

Encrypted SSO Application Manager Configuration Setting: The sequence of bytes obtained by encrypting Unencrypted SSO Application Manager Configuration Setting.

2.2.5 Result Sets

2.2.5.1 Credentials Result Set

The Credentials Result Set contains the credentials for the user for the requested SSO application along with the string SID of the SSO user or group of SSO users that owns the credentials.

The **T-SQL** syntax for the **result set** is as follows:

```
Credentials        image,
UserID             nvarchar(256);
```

Credentials: The encrypted credentials for the specified SSO application for the SSO user.

UserID: The string SID of the SSO user or group of SSO users that is associated with the retrieved credentials.

2.2.5.2 Null Result Set

The Null Result Set contains one column. The value of the fields in the column MUST be NULL.

The T-SQL syntax for the result set is as follows:

```
{Unused}          int;
```

{Unused}: An implementation specific column that the protocol client MUST ignore.

2.2.6 Tables and Views

None.

2.2.7 XML Structures

No common XML Structures are defined in this protocol.

3 Protocol Details

3.1 Protocol Server Details

3.1.1 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 back-end database server maintains the following sets of data for this protocol within an SSO store. Data is maintained until updated or removed.

SSO configuration: A set of information that dictates the behavior of the protocol client. It includes information such as the time in minutes when the SSO ticket is valid, the number of days the SSO audit entries are preserved, date and timestamp indicating the version of the SSO configuration information set, name of the master secret server, and the encrypted string SIDs of the SSO administrator and SSO application manager.

Application definitions: A set of SSO applications that each consist of a programmatic name, descriptive name, e-mail contact, and a set of credential field labels and information about how they can each be displayed in a user interface.

Issued SSO tickets: A set of unexpired tokens that represent the SSO tickets issued along 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 each SSO user for each SSO application.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 sso_DeleteAllUserCredentials

The **sso_DeleteAllUserCredentials** stored procedure is called to delete all credentials of the specified security principal (2) for all SSO applications.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE sso_DeleteAllUserCredentials (
    @UserID                nvarchar(256)
);
```

@UserID: The string SID of the security principal (2). This value MUST NOT be NULL.

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

Value	Description
0x00000000	No error encountered.
0x80630001	No credentials for security principal (2) with string SID equal to *@UserID* exist.

Result Sets: MUST NOT return any result sets.

3.1.5.2 sso_DeleteAnyApplication

The **sso_DeleteAnyApplication** stored procedure is called to delete the specified SSO application from the SSO store. All credentials associated with the specified SSO application MUST also be deleted.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE sso_DeleteAnyApplication (
    @Application               nvarchar(128)
);
```

@Application: The programmatic name of the SSO application. This value MUST NOT be NULL.

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

Value	Description
0x00000000	No error encountered.
0x806300C3	Unable to access credentials because of implementation-specific reasons.
0x80630490	No SSO application with programmatic name equal to *@Application* exists.

Result Sets: MUST NOT return any result sets.

3.1.5.3 sso_DeleteAuditRecords

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

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE sso_DeleteAuditRecords();
```

Return Code Values: An integer which MUST equal the number of SSO audit entries deleted.

Result Sets: MUST NOT return any result sets.

3.1.5.4 sso_DeleteExpiredTicketRecords

The **sso_DeleteExpiredTicketRecords** stored procedure is called to delete SSO tickets from the SSO store when the difference between their creation and the current time, expressed in minutes, is greater than or equal to the TicketTimeOutMin in the SSO configuration.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE sso_DeleteExpiredTicketRecords();
```

Return Code Values: An integer which MUST equal the number of SSO tickets deleted.

Result Sets: MUST NOT return any result sets.

3.1.5.5 sso_DeleteUserCredentials

The **sso_DeleteUserCredentials** stored procedure is called to delete all credentials from the SSO store for a specified security principal (2) and a specified SSO application.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE sso_DeleteUserCredentials (
    @UserID                 nvarchar(256),
    @Application            nvarchar(128)
);
```

@UserID: The string SID of the security principal (2) whose credentials are to be deleted. This value MUST NOT be NULL.

@Application: The programmatic name of the SSO application. This value MUST NOT be NULL.

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

Value	Description
0x00000000	No error encountered.
0x80630490	No SSO application with programmatic name equal to *@Application* exists.
0x80630001	No credentials for security principal (2) with string SID equal to *@UserID* and SSO application equal to *@Application* exist.

Result Sets: MUST NOT return any result sets.

3.1.5.6 sso_InsertAudit

The **sso_InsertAudit** stored procedure is called to add an SSO audit entry to the SSO store when an **SSO action** is performed.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE sso_InsertAudit (
    @UserAuthorityName        nvarchar(128),
    @ActionType               int,
    @ActionResultCode         int,
    @Application              nvarchar(128),
    @UserName                 nvarchar(128),
    @Info1                    nvarchar(256),
    @Info2                    nvarchar(256),
    @Info3                    nvarchar(256),
    @Machine                  nvarchar(128)
);
```

@UserAuthorityName: The name of the account which performed the operation being audited. If the SSO action audited is related to credentials and the action is being executed by a security principal (2) other than the security principal (2) that owns the credentials, the value MUST be the name of the account of the executing security principal, otherwise the value MUST be NULL.

@ActionType: The action type of the SSO audit entry. This value MUST be listed in the following table.

Value	Name
0	Store credentials.
1	Retrieve credentials for SSO applications of Type 0, 1, 2 or 3.
2	Store **group credentials**.
20	Delete credentials.
21	Add SSO application.
22	Retrieve SSO applications.
23	Delete SSO application.
24	Issue SSO ticket.
25	Retrieve credentials using SSO ticket.
26	Retrieve SSO application.
30	Retrieve credentials.
Other positive integers	An implementation-specific SSO action.

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

@Application: The programmatic name of the SSO application that the SSO action is associated with. If the SSO action is not associated with an SSO application, it MUST be NULL.

@UserName: The name of the security principal (2) that executed the SSO action unless the SSO action audited is related to credentials and the action is being executed by a security principal (2) other than the credentials' owner, in which case it MUST be the name of the security principal (2) whose credentials are accessed. This value MUST NOT be NULL.

@Info1: Additional information to be audited. Unused. The value MUST be NULL.

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

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

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

Return Code Values: An integer which the protocol client MUST ignore.

Result Sets: MUST NOT return any result sets.

3.1.5.7 sso_InsertCredentialTicket

The **sso_InsertCredentialTicket** stored procedure is called to store a representation of the SSO ticket that is issued for future use, as well as the issue time, in the SSO store.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE sso_InsertCredentialTicket (
    @UserTicket                varbinary(300)
);
```

@UserTicket: The Random Ticket to store. This value MUST NOT be NULL.

Return Code Values: An integer which the protocol client MUST ignore.

Result Sets: MUST NOT return any result sets.

3.1.5.8 sso_InsertMyTempCredentials

The **sso_InsertMyTempCredentials** stored procedure is called to store the given encrypted credentials for a given SSO application in a temporary area of the SSO store used during a bulk re-encryption of all credentials when the master secret has been changed.

The T-SQL syntax for this stored procedure is as follows:

```
PROCEDURE sso_InsertMyTempCredentials (
    @MyVer                      datetime,
    @UserID                     nvarchar(256),
    @Application                nvarchar(128),
    @Credentials                image,
    @Type                       int
);
```

@MyVer: The version of the SSO configuration stored in the SSO store. The value MUST be MyVer.

@UserID: The string SID of the security principal (2) performing the operation. This value MUST NOT be NULL.

@Application: The programmatic name of the SSO application for which the credentials are to be stored. This value MUST NOT be NULL.

@Credentials: The encrypted credentials. This value MUST NOT be NULL.

@Type: The type of the SSO application. It MUST be Type.

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

Value	Description
0x00000000	Execution finished; the credentials MAY<1> have been successfully inserted.
0x80630490	No SSO application with name equal to *@Application* and type equal to *@Type* exists.
0x8063000F	*@Type* is 1, 3, 4 or 5 and credentials already exist for an SSO application with name equal to *@Application* and an SSO user with string SID not equal to *@UserID*.
0x8063064B	The current version of the SSO configuration in the SSO store is different from *@MyVer*.

Result Sets: MUST NOT return any result sets.

3.1.5.9 sso_InsertUpdateApplication

The **sso_InsertUpdateApplication** stored procedure is called to create a new SSO application or update an existing SSO application in the SSO store.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE sso_InsertUpdateApplication (
@Application                    nvarchar(128),
@FriendlyName                   nvarchar(128),
@Type                           int,
@ContactEmail                   nvarchar(128),
@Field1Label                    nvarchar(128),
@MaskField1                     bit,
@Field2Label                    nvarchar(128),
@MaskField2                     bit,
@Field3Label                    nvarchar(128),
@MaskField3                     bit,
@Field4Label                    nvarchar(128),
@MaskField4                     bit,
@Field5Label                    nvarchar(128),
@MaskField5                     bit,
@CreationDisposition            int
);
```

@Application: The programmatic name of the SSO application to be created or updated. This value MUST NOT be NULL.

@FriendlyName: The descriptive name of the SSO application to be created or updated. This value MUST NOT be NULL.

@Type: The type of the SSO application. It MUST be Type. For update, the value of this parameter MUST be equal to the Type of the SSO application being updated.

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

@Field1Label: The label describing the first credential field of the SSO application. If this is NULL, *@Field2Label*, *@Field3Label*, *@Field4Label* and *@Field5Label* MUST also be NULL.

@MaskField1: Indicator for whether the first credential field prompt to be displayed is masked. If *@Field1Label* is NOT NULL, this MUST NOT be NULL.

@Field2Label: The label describing the second credential field of the SSO application. If this is NULL, *@Field3Label*, *@Field4Label* and *@Field5Label* MUST also be NULL.

@MaskField2: Indicator for whether the second credential field prompt to be displayed is masked. If *@Field2Label* is NOT NULL, this MUST NOT be NULL.

@Field3Label: The label describing the third credential field of the SSO application. If this is NULL, *@Field4Label* and *@Field5Label* MUST also be NULL.

@MaskField3: Indicator for whether the third credential field prompt to be displayed is masked. If *@Field3Label* is NOT NULL, this MUST NOT be NULL.

@Field4Label: The label describing the fourth credential field of the SSO application. If this is NULL, *@Field5Label* MUST also be NULL.

@MaskField4: Indicator for whether the fourth credential field prompt to be displayed is masked. If *@Field4Label* is NOT NULL, this MUST NOT be NULL.

@Field5Label: The label describing the fifth credential field of the SSO application.

@MaskField5: Indicator for whether the fifth credential field prompt is to be displayed masked. If *@Field5Label* is NOT NULL, this MUST NOT be NULL.

@CreationDisposition: Indicates whether the SSO application is to be created or updated. The value of this parameter MUST be one of the values listed in the following table.

Value	Description
0	New SSO application needs to be created.
1	An existing SSO application needs to be modified. A new SSO application is created if the SSO application does not exist.

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

Value	Description
0x00000000	No errors encountered.
0x806300B8	An SSO application with programmatic name equal to *@Name* already exists and *@CreationDisposition* is 0.
0x80630007	*@CreationDisposition* is 1 and the value of *@Type* does not match with the Type for the SSO application with programmatic name *@Application*. The Type of the SSO application is not an updatable field.
0x80630006	*@CreationDisposition* is not 0 or 1.

Result Sets: MUST NOT return any result sets.

3.1.5.10 sso_InsertUpdateMyCredentials

The **sso_InsertUpdateMyCredentials** stored procedure is called to store or update the given encrypted credentials for a given SSO application in the SSO store.

The T-SQL syntax for this stored procedure is as follows:

```
PROCEDURE sso_InsertUpdateMyCredentials (
    @MyVer                      datetime,
    @UserID                     nvarchar(256),
    @Application                nvarchar(128),
    @Credentials                image,
    @Type                       int
);
```

@MyVer: The version of the SSO configuration stored in the SSO store. This MUST be MyVer.

@UserID: The string SID of the security principal (2) who owns the credentials. This value MUST NOT be NULL.

@Application: The programmatic name of the SSO application for which the credentials are to be stored. This value MUST NOT be NULL.

@Credentials: The encrypted credentials. This value MUST NOT be NULL.

@Type: The type of the SSO application. It MUST be Type.

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

Value	Description
0x00000000	Execution finished; the credentials MAY<2> have been successfully inserted.
0x80630490	SSO application with name equal *to @Application* and Type equal to *@Type* does not exist.
0x8063064B	The current version of the SSO configuration in the SSO store is different from *@MyVer*.

Result Sets: MUST NOT return any result sets.

3.1.5.11 sso_InsertUpdateSSOConfig

The **sso_InsertUpdateSSOConfig** stored procedure is called to store or change the SSO configuration in the SSO store.

The T-SQL syntax used for this stored procedure is as follows:

```
PROCEDURE sso_InsertUpdateSSOConfig (
    @SecretServer               nvarchar(256),
    @SSOAdminGroup              varbinary(1000),
    @AffiliateAppMgrGroup       varbinary(1000),
    @TicketTimeOutMin           int,
    @PurgeAuditDays             int,
    @OutNewVer                  datetime output
);
```

@SecretServer: The name of the server which supplies the master secret. This parameter MUST NOT be NULL.

@SSOAdminGroup: The encrypted string SID of the SSO administrator. This parameter MUST NOT be NULL.

@AffiliateAppMgrGroup: The encrypted string SID of the SSO application manager. This parameter MUST NOT be NULL.

@TicketTimeOutMin: The validity in minutes for SSO tickets. This parameter MUST be TicketTimeoutMin.

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

@OutNewVer: Upon return from this stored procedure, it MUST be set to the time the SSO configuration was created or updated.

Return Code Values: An integer that the protocol client MUST ignore.

Result Sets: MUST NOT return any result sets.

3.1.5.12 sso_RetrieveAllCredentials

The **sso_RetrieveAllCredentials** stored procedure is called to get all the credentials stored in the SSO store.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE sso_RetrieveAllCredentials();
```

Return Code Values: An integer which the protocol client MUST ignore.

Result Sets: MUST return the following result set:

3.1.5.12.1 All Credentials Result Set

The **All Credentials Result Set** contains encrypted credentials along with the string SID of the SSO user or group of SSO users and the SSO application name associated with it. It MUST contain zero or more rows.

The T-SQL syntax for the result set is as follows:

```
Credentials             image,
UserID                  nvarchar(256),
Application             nvarchar(128);
```

Credentials: The encrypted credentials.

UserID: The string SID of the SSO user or group of SSO users that owns the credentials.

Application: The programmatic name of the SSO application associated with the retrieved credentials.

3.1.5.13 sso_RetrieveApplication

The **sso_RetrieveApplication** stored procedure is called to retrieve the information for a given SSO application.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE sso_RetrieveApplication (
    @Application                nvarchar(128)
);
```

@Application: The programmatic name of the SSO application.

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

Value	Description
0x00000000	Successful execution.
0x80630490	The specified application does not exist.

Result Sets: MUST return the following result set:

3.1.5.13.1 Application Result Set

The **Application Result Set** contains detailed information about an SSO application. The **Application Result Set** MUST contain exactly one row on successful execution of the call.

The T-SQL syntax for the result set is as follows:

```
    FriendlyName                nvarchar(128),
    Type                        int,
    ContactEmail                nvarchar(128);
```

FriendlyName: The descriptive name of the SSO application.

Type: The type of the SSO application. It MUST be Type.

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

3.1.5.14 sso_RetrieveApplicationCount

The **sso_RetrieveApplicationCount** stored procedure is called to get the count of the SSO applications present in the SSO store.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE sso_RetrieveApplicationCount (
    @count                    int output
);
```

@count: Upon return from this stored procedure, it MUST be set to the number of SSO applications present in the SSO store.

Return Code Values: An integer which the protocol client MUST ignore.

Result Sets: MUST NOT return any result sets.

3.1.5.15 sso_RetrieveApplicationFields

The **sso_RetrieveApplicationFields** stored procedure is called to get all the credential fields for the specified SSO application.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE sso_RetrieveApplicationFields (
    @Application                nvarchar(128)
);
```

@Application: The programmatic name of the SSO application. This parameter MUST NOT be NULL.

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

Value	Description
0x00000000	Successful execution.
0x80630490	The specified SSO application does not exist.

Result Sets: MUST return the following result set:

3.1.5.15.1 Credential Fields Result Set

The **Credential Fields Result Set** contains the information about the credential fields associated with the specified SSO application. This result set MUST contain exactly one row on successful execution.

The T-SQL syntax for this result set is as follows:

```
NumFields                   numeric (3, 0),
Field1Label                 nvarchar(128),
MaskField1                  bit,
Field2Label                 nvarchar(128),
MaskField2                  bit,
Field3Label                 nvarchar(128),
MaskField3                  bit,
Field4Label                 nvarchar(128),
MaskField4                  bit,
Field5Label                 nvarchar(128),
MaskField5                  bit;
```

NumFields: The number of credential fields in the specified SSO application.

Field1Label: The label describing the first credential field of the SSO application.

MaskField1: Indicator for whether the first credential field prompt to be displayed is masked.

Field2Label: The label describing the second credential field of the SSO application.

MaskField2: Indicator for whether the second credential field prompt to be displayed is masked.

Field3Label: The label describing the third credential field of the SSO application.

MaskField3: Indicator for whether the third credential field prompt to be displayed is masked.

Field4Label: The label describing the fourth credential field of the SSO application.

MaskField4: Indicator for whether the fourth credential field prompt to be displayed is masked.

Field5Label: The label describing the fifth credential field of the SSO application.

MaskField5: Indicator for whether the fifth credential field prompt to be displayed is masked.

3.1.5.16 sso_RetrieveApplications

The **sso_RetrieveApplications** stored procedure is called to retrieve all the SSO applications in the SSO store.

The T-SQL syntax for this stored procedure is as follows:

```
PROCEDURE sso_RetrieveApplications();
```

Return Code Values: An integer which the protocol client MUST ignore.

Result Sets: MUST return the following result set:

3.1.5.16.1 Application Result Set

The **Application Result Set** contains information about SSO applications. It contains zero or more rows.

The T-SQL syntax for this result set is as follows:

```
Application                nvarchar(128),
FriendlyName               nvarchar(128),
Type                       int,
ContactEmail               nvarchar(128);
```

Application: The programmatic name of the SSO application.

FriendlyName: The descriptive name of the SSO application.

Type: The type of the SSO application. It MUST be Type.

ContactEmail: The e-mail address of the administrator who owns the administration responsibilities for this SSO application definition.

3.1.5.17 sso_RetrieveGroupApplicationGroup

The **sso_RetrieveGroupApplicationGroup** stored procedure is called to get the string SID for the group of SSO users authorized to access an SSO application of Type 1,3,4 or 5.

The T-SQL syntax for this stored procedure is as follows:

```
PROCEDURE sso_RetrieveGroupApplicationGroup (
    @Application               nvarchar(128)
);
```

@Application: The programmatic name of the SSO application.

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

Value	Description
0x00000000	Successful execution.
0x80630490	The specified SSO application does not exist or is not of Type 1,3,4 or 5.

Result Sets: MUST return a single result set which the protocol client MUST ignore if the specified SSO application does not exist or is not of Type 1, 3, 4 or 5:

3.1.5.17.1 Domain Group Result Set

The **Domain Group Result Set** contains the group string SID for the given SSO application. The **Domain Group Result Set** contains exactly one row on successful execution of the call if the group credentials for the SSO application exist, and zero rows if the group credentials do not exist.

The T-SQL syntax for this result set is as follows:

```
UserID                nvarchar(256);
```

UserID: The group string SID associated with the SSO application.

3.1.5.18 sso_RetrieveMyCredentials

The **sso_RetrieveMyCredentials** stored procedure is called to get the credentials for an SSO user for the specified SSO application or to redeem an SSO ticket for credentials.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE sso_RetrieveMyCredentials(
    @MyVer                  datetime,
    @UserID                 nvarchar(256),
    @Application            nvarchar(128),
    @UserTicket             varbinary(300)
);
```

@MyVer: The current version of the SSO configuration. It MUST be MyVer.

@UserID: The string SID of the SSO user or group of SSO users.

@Application: The programmatic name of the SSO application. The SSO application Type MUST be 0, 1, 2 or 3.

@UserTicket: The Random Ticket when redeeming an SSO ticket to get the credentials. This parameter MUST be set to NULL if SSO tickets are not used to get the credentials.

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

Value	Description
0x00000000	Successful execution.
0x80630490	The specified SSO application does not exist or is of Type 4 or 5.
0x8063064B	Supplied version does not match the current SSO configuration version.
0x80630005	Access is denied because the SSO ticket cannot be found or has expired.
0x80630428	Invalid SSO application Type.
0x80630001	Credentials not found.

Result Sets: MUST return the following result sets in the following order:

1. The Credentials Result Set MUST be returned if the **return code** is 0x0 or 0x80630001. The result set MUST contain zero or one row.
2. The Null Result Set MUST be returned if the return code is not equal to 0x0. The result set MUST contain one row.

3.1.5.19 sso_RetrieveMySensitiveCredentials

The **sso_RetrieveMySensitiveCredentials** stored procedure is called to get the credentials for an SSO user for the specified SSO application.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE sso_RetrieveMySensitiveCredentials(
    @MyVer                          datetime,
    @UserID                         nvarchar(256),
    @Application                    nvarchar(128),
    @UserTicket                     varbinary(300)
);
```

@MyVer: The current version of the SSO configuration. This MUST be MyVer.

@UserID: The string SID of the SSO user or group of SSO users.

@Application: The programmatic name of the SSO application.

@UserTicket: MUST be NULL.

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

Value	Description
0x00000000	Successful execution.
0x8063064B	Supplied version does not match the current SSO configuration version.
0x80630490	The specified SSO application does not exist.
0x80630428	Invalid SSO application type.
0x80630001	Credentials not found.

Result Sets: MUST return the following result sets in the following order:

1. The Credentials Result Set MUST be returned if the return code is 0x0 or 0x80630001. The result set MUST contain zero or one row.

2. The Null Result Set MUST be returned if the return code is not equal to 0x0. The result set MUST contain one row.

3.1.5.20 sso_RetrieveSSOConfig

The **sso_RetrieveSSOConfig** stored procedure is called to retrieve SSO configuration information which determines the behavior of the protocol client.

The T-SQL syntax for this stored procedure is as follows:

```
PROCEDURE sso_RetrieveSSOConfig (
    @MyVer                          datetime,
    @SecretServer                   nvarchar(256) output,
    @SSOAdminGroup                  varbinary(1000) output,
    @AffiliateAppMgrGroup           varbinary(1000) output,
    @Configured                     tinyint output,
    @NewVer                         datetime output,
    @Refresh                        tinyint output,
    @TicketTimeOutMin               int output,
    @PurgeAuditDays                 int output
);
```

@MyVer: The last version known by the protocol client. This parameter MUST be set to 0 if a previous value is not available. Otherwise, the value MUST be MyVer.

@**SecretServer:** Upon return from this stored procedure, it MUST be set to the name of the server from which the master secret can be retrieved, if the value of *@Refresh* is 1 and if the return code is 0. Otherwise, the protocol client MUST ignore the value of the parameter.

@**SSOAdminGroup:** Upon return from this stored procedure, it MUST be set to the encrypted string SID of the SSO administrator, if the value of *@Refresh* is 1 and if the return code is 0. Otherwise, the protocol client MUST ignore the value of the parameter.

@**AffiliateAppMgrGroup:** Upon return from this stored procedure, it MUST be set to the encrypted string SID of the SSO application manager, if the value of *@Refresh* is 1 and if the return code is 0. If the return code is not 0, the protocol client MUST ignore the value of the parameter.

@Configured: Flag indicating whether the server is configured. Upon return from this stored procedure, it MUST be set to 0 if the server is not configured, the value of *@Refresh* is 1 and if the return code is 0. Otherwise, the protocol client MUST ignore the value of the parameter.

@NewVer: Upon return from this stored procedure, it MUST be set to the version information for the latest available SSO configuration in the SSO store, if the value of *@Refresh* is 1 and if the return code is 0. Otherwise, the protocol client MUST ignore the value of the parameter.

@Refresh: A flag indicating whether the protocol client's copy of the SSO configuration information needs to be refreshed. Upon return from this stored procedure, it MUST be set to 0 if @*MyVer* is the same as the currently stored version of the SSO configuration in the SSO store, or if the protocol client is not configured; otherwise this flag MUST be set to 1.

@TicketTimeOutMin: Upon return from this stored procedure, it MUST be set to the validity in minutes for SSO tickets, if the value of *@Refresh* is 1 and if the return code is 0. Otherwise, the protocol client MUST ignore the value of the parameter. This parameter MUST be TicketTimeoutMin.

@PurgeAuditDays: Upon return from this stored procedure, it MUST be set to the number of days that MUST elapse before SSO audit entries are purged from the SSO store, if the value of @Refresh is 1 and if the return code is 0. Otherwise, the protocol client MUST ignore the value of the parameter. This parameter MUST be PurgeAuditDays.

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

Value	Description
0x00000000	Successful execution.
0x8063064A	Protocol client is not configured.

Result Sets: MUST NOT return any result sets.

3.2 Protocol Client 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 SSO applications, credentials, SSO tickets and SSO configuration stored in the SSO store can be maintained as object structures within the protocol client.

3.2.2 Timers

A timer can be used to detect changes in the SSO configuration of the protocol server signaling required changes in behavior of the protocol client. The amount of time elapsed between checks for whether the SSO configuration has changed is an implementation-dependent decision.

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

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

3.2.3 Initialization

The protocol client MUST validate the user making the request before calling the stored procedures.

A single protocol client MUST call sso_InsertUpdateSSOConfig to set up a baseline configuration.

All protocol clients MUST call sso_RetrieveSSOConfig to retrieve and maintain an in-memory copy of the current SSO configuration information in the SSO store.

All protocol clients MUST obtain the master secret from the master secret server as specified in [MS-SSP].

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 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:

3.2.5.1 sso_DeleteAllUserCredentials

Upon execution of the stored procedure, the protocol client MUST call sso_InsertAudit with *@ActionType* equal to 20 and @*ActionResultCode* equal to the implementation-specific result value.

3.2.5.2 sso_DeleteAnyApplication

Upon execution of the stored procedure, the protocol client MUST call sso_InsertAudit with *@ActionType* equal to 23 and @*ActionResultCode* equal to the implementation-specific result value.

3.2.5.3 sso_DeleteUserCredentials

Upon execution of the stored procedure, the protocol client MUST call sso_InsertAudit with *@ActionType* equal to 20 and @*ActionResultCode* equal to the implementation-specific result value.

3.2.5.4 sso_InsertCredentialTicket

Before calling the stored procedure sso_InsertCredentialTicket, 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 SSO Ticket which can be utilized by the protocol client at a later stage when calling sso_RetrieveMyCredentials:

1. Generate a temporary session key used for encryption by performing the following steps:
    1. Generate a cryptographically secure random salt of 16 bytes.
    2. Create a Ticket Encryption Session Key Seed using the salt obtained in step a in conjunction with the master secret.
    3. Hash the Ticket Encryption Session Key Seed using the SHA algorithm. This will yield a 20 byte hash value.
    4. Form a 64-byte buffer by repeating the constant 0x36 64 times. XOR the result of step c into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.
    5. Form another 64-byte buffer by repeating the constant 0x5C 64 times. XOR the result of step c into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.
    6. Concatenate the result of step d with the first 12 bytes of step e, to form a 32-byte (256-bit) key.

2. Create a Random Ticket.

3. Create an Unencrypted Ticket using the Random Ticket obtained in step 2.

4. Generate a Salted Encrypted Ticket from the Unencrypted Ticket obtained in step 3 using 256 bit Advanced Encryption Standard (AES) encryption and the key generated in step 1.

5. Generate a Final SSO Ticket from the Salted Encrypted Ticket obtained in step 4.

Upon execution of the stored procedure, the protocol client MUST call sso_InsertAudit with *@ActionType* equal to 24 and @*ActionResultCode* equal to the implementation-specific result value.

3.2.5.5 sso_InsertUpdateApplication

Upon execution of the stored procedure, the protocol client MUST call sso_InsertAudit with *@ActionType* equal to 21 and @*ActionResultCode* equal to the implementation-specific result value.

3.2.5.6 sso_InsertUpdateMyCredentials

The stored procedure sso_InsertUpdateMyCredentials MUST be called to insert or update the encrypted credentials provided by an SSO user for a given SSO application.

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

1. Pack the given plain text credential fields into Credential Chunks.

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

    1. Generate a cryptographically secure random salt of 4 bytes.

    2. Create a Credential Encryption Session Key Seed using the salt obtained in step a in conjunction with the master secret and the string SID of the given SSO user in @UserID.

    3. Hash the Credential Encryption Session Key Seed using the SHA algorithm. This will yield a 20 byte hash value.

    4. Form a 64-byte buffer by repeating the constant 0x36 64 times. XOR the result of step c into the first 20 bytes of this value, and compute a SHA hash of the resulting 64-byte buffer.

    5. Form another 64-byte buffer by repeating the constant 0x5C 64 times. XOR the result of step c into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.

    6. Concatenate the result of step d with the first 12 bytes of step e, to form a 32-byte (256-bit) key.

3. Create an Unencrypted Credential using the Credential Chunks obtained in step 1 along with the string SID of the specified SSO user in @UserID.

4. Generate a Salted Encrypted Credential from the Unencrypted Credential obtained in step 3 using 256 bit AES encryption and the key generated in step 2.

The protocol client MUST then use the Salted Encrypted Credential as input to the stored procedure.

Upon execution of the stored procedure, the protocol client MUST call sso_InsertAudit with *@ActionType* equal to 0 and @ActionResultCode equal to the implementation-specific result value if the SSO application Type is 0 or 2 and with *@ActionType* equal to 2 if the SSO application Type is 1, 3, 4 or 5 and *@ActionResultCode* equal to the implementation-specific result value.

3.2.5.7 sso_InsertUpdateSSOConfig

The stored procedure sso_InsertUpdateSSOConfig is called to store or change the SSO configuration in the SSO store.

To encrypt the SSO administrator string SIDs before calling the stored procedure, the protocol client MUST:

1. Generate a temporary session key used for encryption by performing the following steps:
    1. Generate a cryptographically secure random salt of 7 bytes.
    2. Create a SSO Administrator Encryption Session Key Seed using the salt obtained in step a in conjunction with the master secret.
    3. Hash the SSO Administrator Encryption Session Key Seed using the SHA algorithm. This will yield a 20 byte hash value.
    4. Form a 64-byte buffer by repeating the constant 0x36 64 times. XOR the result of step c into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.
    5. Form another 64-byte buffer by repeating the constant 0x5C 64 times. XOR the result of step c into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.
    6. Concatenate the result of step d with the first 12 bytes of step e, to form a 32-byte (256-bit) key.
2. Create an Unencrypted SSO Administrator Configuration Setting with the string SID of the SSO administrator supplied by the caller.
3. Generate an Validated Salted Encrypted SSO Administrator Configuration Setting from the Unencrypted SSO Administrator Configuration Setting obtained in step 2 using 256 bit AES encryption and the key generated in step 1.

To encrypt the SSO application manager string SIDs before calling the stored procedure, the protocol client MUST:

1. Generate a temporary session key used for encryption by performing the following steps:
    1. Generate a cryptographically secure random salt of 10 bytes.
    2. Create a SSO Application Manager Encryption Session Key Seed using the salt obtained in step a in conjunction with the master secret.
    3. Hash the SSO Application Manager Encryption Session Key Seed using the SHA algorithm. This will yield a 20 byte hash value.
    4. Form a 64-byte buffer by repeating the constant 0x36 64 times. XOR the result of step c into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.
    5. Form another 64-byte buffer by repeating the constant 0x5C 64 times. XOR the result of step c into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.
    6. Concatenate the result of step d with the first 12 bytes of step e, to form a 32-byte (in other words, a 256-bit) key.

2. Create an Unencrypted SSO Application Manager Configuration Setting with the string SID of the SSO application manager supplied by the caller.

3. Generate an Validated Salted Encrypted SSO Application Manager Configuration Setting from the Unencrypted SSO Application Manager Configuration Setting obtained in step 5 using 256 bit AES encryption and the key generated in step 4.

The values obtained in step 3 and step 6 MUST be passed as the *@SSOAdminGroup* and *@AffiliateAppMgrGroup* parameters, respectively, of the sso_InsertUpdateSSOConfig stored procedure. Parameters *@SecretServer*, *@TicketTimeOutMin*, *@PurgeAuditDays* MUST be passed unchanged as received by the protocol client from the SSO user.

3.2.5.8 sso_RetrieveApplication

Upon execution of the stored procedure, the protocol client MUST call sso_InsertAudit with *@ActionType* equal to 26 and @*ActionResultCode* equal to the implementation-specific result value.

3.2.5.9 sso_RetrieveApplications

Upon execution of the stored procedure, the protocol client MUST call sso_InsertAudit with *@ActionType* equal to 22 and @*ActionResultCode* equal to the implementation-specific result value.

3.2.5.10 sso_RetrieveMyCredentials

The stored procedure sso_RetrieveMyCredentials can be called to obtain the encrypted credentials to be returned to the SSO user, using a previously generated SSO ticket. In this case, the protocol client MUST obtain the string SID of the SSO user who generated the ticket to supply to the stored procedure as follows:

1. Extract the Salted Encrypted Ticket from the given Final SSO Ticket by performing the following steps in order:

    1. Remove the 6 byte ticket header.

    2. Decode the remaining bytes using **base64** encoding to obtain the Salted Encrypted Ticket.

2. Split the Salted Encrypted Ticket into its constituent parts of salt and encrypted ticket.

3. Generate a temporary session key used for decryption by performing the following steps in order:

    1. Create a Ticket Encryption Session Key Seed using the salt obtained in step 2.

    2. Hash the Ticket Encryption Session Key Seed using the SHA algorithm. This will yield a 20 byte hash value.

    3. Form a 64-byte buffer by repeating the constant 0x36 64 times. XOR the result of step b into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.

    4. Form another 64-byte buffer by repeating the constant 0x5C 64 times. XOR the result of step b into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.

    5. Concatenate the result of step c with the first 12 bytes of step d, to form a 32-byte (256-bit) key.

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 AES.

5. Split the Unencrypted Ticket into the Random Ticket and string SID.

   The string SID of the SSO user along with the Random Ticket obtained in step 5 preceding MUST be passed as the *@UserID* and *@UserTicket* parameters respectively of the sso_RetrieveMyCredentials stored procedure, to obtain the Salted Encrypted Credentials of the user.

To obtain the plain text credentials to be returned to the SSO user, the protocol client MUST subsequently perform the following steps in the following order:

1. Split the Salted Encrypted Credentials obtained by calling sso_RetrieveMyCredentials into its constituent parts of salt and encrypted credentials.
2. Generate a temporary session key used for decryption by performing the following steps in order:
   1. Create a Credential Encryption Session Key Seed using the salt obtained in step 6 and the string SID obtained in step 5.
   2. Hash the Credential Encryption Session Key Seed using the SHA algorithm. This will yield a 20 byte hash value.
   3. Form a 64-byte buffer by repeating the constant 0x36 64 times. XOR the result of step b into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.
   4. Form another 64-byte buffer by repeating the constant 0x5C 64 times. XOR the result of step b into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.
   5. Concatenate the result of step c with the first 12 bytes of step d, to form a 32-byte (256-bit) key.
3. Generate an Unencrypted Credentials by decrypting the encrypted credentials obtained in step 6 with the temporary session key obtained in step 7 using 256 bit AES.
4. Compare the string SID of the current SSO user with the first sequence of bytes in the Unencrypted Credentials obtained in step 8. If there is no match, the protocol client MUST signal an error condition to the SSO user.
5. If there is a match, the matched portion is truncated leaving the count of Credential Chunks followed by the Credential Chunks which represent the plain text credentials.

Upon execution of the stored procedure, the protocol client MUST call sso_InsertAudit with *@ActionType* equal to 25 and @*ActionResultCode* equal to the implementation-specific result value if the SSO application Type is 0, 1, 2 or 3.

The same stored procedure can be used to obtain the encrypted credentials to be provided to an SSO user directly without using a SSO ticket by setting *@UserTicket* to NULL and *@UserID* to the string SID of the SSO user in the preceding procedure. To obtain the plain text credentials, the protocol client MUST subsequently perform steps 6 through 10 in the preceding manner, where the string SID of the SSO user MUST be used in step 7a instead of the string SID obtained in step 5.

In this case, upon execution of the stored procedure, the protocol client MUST call sso_InsertAudit with *@ActionType* equal to 1 and @*ActionResultCode* equal to the implementation-specific result value if the SSO application Type is 0, 1, 2 or 3.

3.2.5.11 sso_RetrieveMySensitiveCredentials

The stored procedure sso_RetrieveMySensitiveCredentials MUST be called to obtain the encrypted credentials to be returned to the SSO user.

The string SID of the SSO user MUST be passed as the *@UserID* parameter and NULL as the *@UserTicket* to the sso_RetrieveMySensitiveCredentials stored procedure, to obtain the Salted Encrypted Credentials.

To obtain the plain text credentials to be returned to the SSO user, the protocol client MUST subsequently perform the following steps in the following order:

1. Split the Salted Encrypted Credentials obtained by calling sso_RetrieveMySensitiveCredentials into its constituent parts of salt and encrypted credentials.
2. Generate a temporary session key used for decryption by performing the following steps in order:
    1. Create a Credential Encryption Session Key Seed using the salt obtained in step 1 and the string SID of the current SSO user.
    2. Hash the Credential Encryption Session Key Seed using the SHA algorithm. This will yield a 20 byte hash value.
    3. Form a 64-byte buffer by repeating the constant 0x36 64 times. XOR the result of step b into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.
    4. Form another 64-byte buffer by repeating the constant 0x5C 64 times. XOR the result of step b into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.
    5. Concatenate the result of step c with the first 12 bytes of step d, to form a 32-byte (256-bit) key.
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 AES.
4. Compare the string SID of the current SSO user with the first sequence of bytes in the Unencrypted Credentials obtained in step 3. If there is no match, an error condition MUST be signaled.
5. If there is a match, the matched portion is truncated leaving the count of Credential Chunks followed by the Credential Chunks which represent the plain text credentials.

Upon execution of the stored procedure, the protocol client MUST call sso_InsertAudit with *@ActionType* equal to 30 and @*ActionResultCode* equal to the implementation-specific result value.

3.2.5.12 sso_RetrieveSSOConfig

The stored procedure sso_RetrieveSSOConfig is called to retrieve the SSO configuration in the SSO store.

To obtain the plain text string SID for the SSO administrator security principal (2), the protocol client MUST perform the following steps in the following order:

1. Split the Validated Salted Encrypted SSO Administrator Configuration Setting obtained by calling sso_RetrieveSSOConfig into its constituent parts of validation bytes, salt and encrypted string SID of the SSO administrator.

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

    1. Create a SSO Administrator Encryption Session Key Seed using the salt obtained in step1.

    2. Hash the SSO Administrator Encryption Session Key Seed using the SHA algorithm. This will yield a 20 byte hash value.

    3. Form a 64-byte buffer by repeating the constant 0x36 64 times. XOR the result of step b into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.

    4. Form another 64-byte buffer by repeating the constant 0x5C 64 times. XOR the result of step b into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.

    5. Concatenate the result of step c with the first 12 bytes of step d, to form a 32-byte (256-bit) key.

3. Generate an Unencrypted SSO Administrator Configuration Setting by decrypting the encrypted string SID obtained in step 1 with the temporary session key obtained in step 2 using 256 bit AES.

4. Extract the plain text string SID that represents the SSO administrator from the Unencrypted SSO Administrator Configuration Setting obtained in step 3.

To obtain the plain text string SID for the SSO application manager security principal (2), the protocol client MUST perform the following steps in the following order:

1. Split the Validated Salted Encrypted SSO Application Manager Configuration Setting obtained by calling sso_RetrieveSSOConfig into its constituent parts of validation bytes, salt and encrypted string SID of the SSO application manager.

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

    1. Create a SSO Application Manager Encryption Session Key Seed using the salt obtained in step1.

    2. Hash the SSO Application Manager Encryption Session Key Seed using the SHA algorithm. This will yield a 20 byte hash value.

    3. Form a 64-byte buffer by repeating the constant 0x36 64 times. XOR the result of step b into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.

    4. Form another 64-byte buffer by repeating the constant 0x5C 64 times. XOR the result of step b into the first 20 bytes of this buffer, and compute a SHA hash of the resulting 64-byte buffer.

    5. Concatenate the result of step c with the first 12 bytes of step d, to form a 32-byte (256-bit) key.

3. Generate an Unencrypted SSO Application Manager Configuration Setting by decrypting the encrypted string SID obtained in step 5 with the temporary session key obtained in step 6 using 256 bit AES.

4. Extract the plain text string SID that represents the SSO application manager from the Unencrypted SSO Application Manager Configuration Setting obtained in step 7.

3.2.6 Timer Events

If a timer is used for propagation of SSO configuration information, when the timer runs out, the timer event handler MUST call sso_RetrieveSSOConfig to get the latest SSO configuration in the SSO store and compare it with what it got on a previous event. It MUST then update its behavior to comply with the information in the settings that have changed.

When the SSO ticket expiration timer timeout event is triggered, the timer event handler MUST call sso_DeleteExpiredTicketRecords.

When the SSO audit entry purge timer timeout event is triggered, the timer event handler MUST call sso_DeleteAuditRecords.

3.2.7 Other Local Events

None.

4 Protocol Examples

This section provides specific example scenarios for operations on stored credentials and SSO applications. These examples describe in detail the process of communication between the protocol server and protocol client. In conjunction with the detailed client and server protocol specification in section 3, this information is intended to provide a comprehensive view on how the protocol client operates with the protocol server when executing such an operation.

4.1 Creating an SSO Application

This example illustrates how an SSO user can create an SSO application in the SSO store.

The example assumes that:

1. The protocol client has been configured by the SSO user on '2008/01/01 10:10:00' and hence the MyVer of the SSO configuration is '2008/01/01 10:10:00'.
2. The master secret has been retrieved from the master secret server and cached by the protocol client.
3. The protocol client computer name is 'SsoServer'.
4. The string SID of the SSO user is 'S-1-5-323-5445'.

The following actions are carried out:

1. The SSO user requests the protocol client to create an SSO application with programmatic name 'AdventureWorks', descriptive name 'Adventure Works 2000', Type equal to 0, contact e-mail 'username@contoso.com' to store a username and password.
2. The protocol client calls the sso_InsertUpdateApplication stored procedure using [MS-TDS]:

```
exec @return_value = sso_InsertUpdateApplication
    @Application = 'AdventureWorks',
    @FriendlyName = 'Adventure Works 2000',
    @Type = 0,
    @ContactEmail = 'username@contoso.com',
    @Field1Label = 'Enter Username',
    @MaskField1 = 0,
    @Field2Label = 'Enter Password',
    @MaskField2 = 1,
    @Field3Label = NULL,
    @MaskField3 = NULL,
    @Field4Label = NULL,
    @MaskField4 = NULL,
    @Field5Label = NULL,
    @MaskField5 = NULL,
    @CreationDisposition = 0
```

3. The protocol server creates the SSO application in the SSO store.
4. The protocol server returns a return code of 0x0 that indicates the SSO application was created.
5. The protocol client calls the sso_InsertAudit stored procedure using [MS-TDS].

```
exec @return_value = sso_InsertAudit
```

```
    @UserAuthorityName    = NULL,
    @ActionType = 21,
    @ActionResultCode = 0,
    @Application = 'AdventureWorks',
    @UserName = 'S-1-5-323-5445',
    @Info1 = NULL,
    @Info2 = NULL,
    @Info3 = NULL,
    @Machine = 'SsoServer'
```

6. The protocol server inserts the SSO audit entry into the SSO store
7. The protocol server returns a return code that is ignored by the protocol client.
8. The protocol client returns the return code obtained in step 4 to the SSO user.
9. The SSO user inspects the return code to see if the creation was successful.

4.2 Storing Credentials

This example illustrates how an SSO user can save his or her username and password for the AdventureWorks LOB system as encrypted credentials securely in the SSO store.

The example assumes that the preceding example has been successfully executed.

The following actions are carried out:

1. The SSO user requests the protocol client to save his or her plain text credentials for the AdventureWorks LOB system.
2. The protocol client encrypts the credentials.
3. The protocol client calls the sso_InsertUpdateMyCredentials stored procedure using [MS-TDS]:

```
exec @return_value = sso_InsertUpdateMyCredentials
    @MyVer = '2008/01/01 10:10:00',
    @UserID = 'S-1-5-323-5445',
    @Application = 'AdventureWorks'
    @Credentials = [encrypted credentials]
    @Type = 0
```

4. The protocol server stores the encrypted credentials in the SSO store.
5. The protocol server returns a return code of 0x0 that indicates the credentials were inserted or that a system error occurred.
6. The protocol client calls the sso_InsertAudit stored procedure using [MS-TDS].

```
exec @return_value = sso_InsertAudit
    @UserAuthorityName    = NULL,
    @ActionType = 0,
    @ActionResultCode = 0,
    @Application = 'AdventureWorks',
    @UserName = 'S-1-5-323-5445',
    @Info1 = NULL,
    @Info2 = NULL,
```

```
    @Info3 = NULL,
    @Machine = 'SsoServer'
```

7. The protocol server inserts the SSO audit entry into the SSO store.
8. The protocol server returns a return code that is ignored by the protocol client.
9. The protocol client returns the return code obtained in step 5 to the SSO user.

4.3 Retrieving Credentials

This example illustrates how a SSO user can retrieve his or her username and password for the AdventureWorks LOB system as plain text credentials from the SSO store

The example assumes that the preceding example has been successfully executed.

The following actions are carried out:

1. The SSO user requests the protocol client to retrieve his or her plain text credentials for the AdventureWorks LOB system.
2. The protocol client calls the sso_RetrieveMyCredentials stored procedure using [MS-TDS]:

```
exec @return_value = sso_RetrieveMyCredentials
     @MyVer = '2008/01/01 10:10:00',
     @UserID = 'S-1-5-323-5445',
    @Application = 'AdventureWorks',
    @UserTicket = NULL
```

The protocol server retrieves the encrypted credentials for the SSO user with specified string SID for the specified SSO application from the SSO store.

1. The protocol server returns the encrypted credentials to the protocol client in the Credentials Result Set which contains one row.
2. The protocol server returns a return code of 0x0 that indicates the credentials were retrieved.
3. The protocol client decrypts the credentials returned in the Credentials Result Set.
4. The protocol client calls the sso_InsertAudit stored procedure using [MS-TDS].

```
exec @return_value = sso_InsertAudit
    @UserAuthorityName    = NULL,
    @ActionType = 1,
    @ActionResultCode = 0,
    @Application = 'AdventureWorks',
    @UserName = 'S-1-5-323-5445',
    @Info1 = NULL,
    @Info2 = NULL,
    @Info3 = NULL,
    @Machine = 'SsoServer'
```

5. The protocol server inserts the SSO audit entry into the SSO store.
6. The protocol server returns a return code that is ignored by the protocol client.

7. The protocol client returns the return code obtained in step 5 to the SSO user along with the plain text credentials obtained in step 6. The SSO user uses the credentials to authenticate with the AdventureWorks LOB system.

4.4 Issuing an SSO Ticket

This example illustrates how an SSO user can embed his or her identity in an encrypted SSO ticket which can be persisted, transferred or communicated for redemption in exchange for the plain text credentials at a later time.

The example assumes that the preceding example has been successfully executed.

The following actions are carried out:

1. The SSO user requests the protocol client to issue an SSO ticket with his or her identity encrypted in it.
2. The protocol client encrypts the string SID of the SSO user and generates an SSO ticket
3. The protocol client calls the sso_InsertCredentialTicket stored procedure using [MS-TDS]:

```
exec @return_value = sso_InsertCredentialTicket
    @UserTicket = [random ticket]
```

4. The protocol server stores a portion of the ticket along with the time it was issued in the SSO store.
5. The protocol server returns a return code that is ignored by the protocol client.
6. The protocol client calls the sso_InsertAudit stored procedure using [MS-TDS].

```
exec @return_value = sso_InsertAudit
    @UserAuthorityName    = NULL,
    @ActionType = 24,
    @ActionResultCode = 0,
    @Application = 'AdventureWorks',
    @UserName = 'S-1-5-323-5445',
    @Info1 = NULL,
    @Info2 = NULL,
    @Info3 = NULL,
    @Machine = 'SsoServer'
```

7. The protocol server inserts the SSO audit entry into the SSO store.
8. The protocol server returns a return code that is ignored by the protocol client.
9. The protocol client returns the SSO ticket obtained in step 2 to the SSO user. The SSO user transfers the ticket to another SSO user.

4.5 Redeeming an SSO Ticket

This example illustrates how an SSO user can redeem a ticket generated by a different SSO user for the plain text credentials of the latter.

This example assumes that:

1. The preceding example has been successfully carried out.

2. The string SID of the user is 'S-1-5-111-2222'.

3. SSO user with string SID 'S-1-5-323-5445' has issued an SSO ticket and transferred it to SSO user with string SID 'S-1-5-111-2222'.

The following actions are carried out:

1. The SSO user requests the protocol client to redeem an SSO ticket with the identity of another SSO user encrypted in it.

2. The protocol client decrypts the SSO ticket and obtains the string SID of the original SSO user to whom it was issued.

3. The protocol client calls the sso_RetrieveMyCredentials stored procedure using [MS-TDS]:

```
exec @return_value = sso_RetrieveMyCredentials
     @MyVer = '2008/01/01 10:10:00',
     @UserID = 'S-1-5-323-5445',
    @Application = 'AdventureWorks',
    @UserTicket = [random ticket]
```

   The protocol server compares the specified SSO ticket against the SSO tickets in the SSO store. If a match is found and the matched SSO ticket has not expired, the protocol server retrieves the encrypted credentials corresponding to the SSO user whose string SID was obtained in step 2.

4. The protocol server returns the encrypted credentials to the protocol client in the Credentials Result Set which contains one row.

5. The protocol server returns a return code of 0x0 that indicates the credentials were retrieved.

6. The protocol client decrypts the credentials returned in the Credentials Result Set.

7. The protocol client calls the sso_InsertAudit stored procedure using [MS-TDS].

```
exec @return_value = sso_InsertAudit
    @UserAuthorityName    = 'S-1-5-323-5445',
    @ActionType = 25,
    @ActionResultCode = 0,
    @Application = 'AdventureWorks',
    @UserName = 'S-1-5-111-2222',
    @Info1 = NULL,
    @Info2 = NULL,
    @Info3 = NULL,
    @Machine = 'SsoServer'
```

8. The protocol server inserts the SSO audit entry into the SSO store.

9. The protocol server returns a return code that is ignored by the protocol client.

10. The protocol client returns the return code obtained in step 6 to the SSO user along with the plain text credentials obtained in step 7. The SSO user uses the credentials to authenticate with the AdventureWorks LOB system.

5 Security

5.1 Security Considerations for Implementers

This protocol makes use of SHA hashing to generate symmetric session encryption keys.

Other than this, there are no additional security considerations for implementers. Security assumptions of this protocol are documented in section 1.5.

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® Office SharePoint® Server 2007
- Microsoft® SharePoint® Server 2010
- Microsoft® SQL Server® 2005
- Microsoft® SQL Server® 2008
- Microsoft® SQL Server® 2008 R2

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.

<1> Section 3.1.5.8: System specific internal errors may cause the credentials to not be inserted.

<2> Section 3.1.5.10: System specific internal errors may cause the credentials to not be inserted.

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

L

M

N

O

P

R

S

T

U

V

X