[MS-SSP]:

Single Sign-On Protocol

Revision Summary

Date	Revision History	Revision Class	Comments
4/4/2008	0.1	New	Initial Availability
6/27/2008	1.0	Major	Revised and edited the technical content
12/12/2008	1.01	Editorial	Revised and edited the technical content
7/13/2009	1.02	Major	Changes made for template compliance
8/28/2009	1.03	Editorial	Revised and edited the technical content
11/6/2009	1.04	Editorial	Revised and edited the technical content
2/19/2010	2.0	Major	Updated and revised the technical content
3/31/2010	2.01	Editorial	Revised and edited the technical content
4/30/2010	2.02	Minor	Updated the technical content
6/7/2010	2.03	Editorial	Revised and edited the technical content
6/29/2010	2.04	Editorial	Changed language and formatting in the technical content.
7/23/2010	2.04	None	No changes to the meaning, language, or formatting of the technical content.
9/27/2010	2.04	None	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.04	None	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.04	None	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	2.04	None	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	2.04	None	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	2.04	None	No changes to the meaning, language, or formatting of the technical content.
4/11/2012	2.04	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	2.04	None	No changes to the meaning, language, or formatting of the technical content.
9/12/2012	2.04	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	2.5	Minor	Clarified the meaning of the technical content.
2/11/2013	2.5	None	No changes to the meaning, language, or formatting of the technical content.
7/30/2013	2.6	Minor	Clarified the meaning of the technical content.
11/18/2013	2.6	None	No changes to the meaning, language, or formatting of the

Date	Revision History	Revision Class	Comments
			technical content.
2/10/2014	2.6	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	2.7	Minor	Clarified the meaning of the technical content.
7/31/2014	2.7	None	No changes to the meaning, language, or formatting of the technical content.
10/30/2014	2.7	None	No changes to the meaning, language, or formatting of the technical content.
2/26/2016	3.0	Major	Significantly changed the technical content.
7/15/2016	3.0	None	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies the Single Sign-On Protocol, which protocol clients use to obtain the key that is used to symmetrically encrypt and decrypt credentials and single sign-on (SSO) tickets from the protocol server.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

dynamic endpoint: A network-specific server address that is requested and assigned at run time. For more information, see [C706].

Interface Definition Language (IDL): The International Standards Organization (ISO) standard language for specifying the interface for remote procedure calls. For more information, see [C706] section 4.

master secret: A key that is used to symmetrically encrypt and decrypt credentials and single sign-on (SSO) tickets.

master secret server: A protocol server that stores and can provide a master secret in response to a request from a protocol client.

opnum: An operation number or numeric identifier that is used to identify a specific **remote procedure call (RPC)** method or a method in an interface. For more information, see [C706] section 12.5.2.12 or [MS-RPCE].

remote procedure call (RPC): A context-dependent term commonly overloaded with three meanings. Note that much of the industry literature concerning RPC technologies uses this term interchangeably for any of the three meanings. Following are the three definitions: (*) The runtime environment providing remote procedure call facilities. The preferred usage for this meaning is "RPC runtime". (*) The pattern of request and response message exchange between two parties (typically, a client and a server). The preferred usage for this meaning is "RPC exchange". (*) A single message from an exchange as defined in the previous definition. The preferred usage for this term is "RPC message". For more information about RPC, see [C706].

RPC transport: The underlying network services used by the remote procedure call (RPC) runtime for communications between network nodes. For more information, see [C706] section 2.

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.

universally unique identifier (UUID): A 128-bit value. UUIDs can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects in cross-process communication such as client and server interfaces, manager entry-point vectors, and **RPC** objects. UUIDs are highly likely to be unique. UUIDs are also known as globally unique identifiers (GUIDs) and these terms are used interchangeably in the Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the UUID. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the UUID.

user object: An object of class user. A user object is a security principal object; the principal is a person or service entity running on the computer. The shared secret allows the person or service entity to authenticate itself, as described in ([MS-AUTHSOD] section 1.1.1.1).

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

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.

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, https://www2.opengroup.org/ogsys/catalog/c706

[MS-ERREF] Microsoft Corporation, "Windows Error Codes".

[MS-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions".

[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

None.

1.3 Overview

The Single Sign-On Service protocol is a **remote procedure call (RPC)**-based protocol used by protocol clients to obtain a **master secret** from a **master secret server**.

1.4 Relationship to Other Protocols

This protocol depends on remote procedure call (RPC) for its transport.

This protocol uses RPC over TCP/IP as described in section 2.1.

1.5 Prerequisites/Preconditions

The protocol is an RPC interface and therefore uses the common RPC interface prerequisites as described in [MS-RPCE].

The protocol requires the protocol client to obtain the name of the remote computer that supports this protocol before the protocol client calls methods that are associated with the remote computer. How a protocol client accomplishes this is outside of the scope of this specification.

1.6 Applicability Statement

This protocol is designed to be used only by **single sign-on (SSO) administrators**. The information associated with this protocol can contain highly sensitive information, so the protocol client is designed to be used in an environment that is appropriately secured.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

This protocol uses Win32 error codes, as specified in [MS-ERREF] section 2.2. Vendors SHOULD<1> reuse those values with their indicated meanings. Choosing any other value runs the risk of a collision in the future.

1.9 Standards Assignments

Parameter	**Value**	**Reference**
RPC interface **UUID**	9D07CA0D-8F02-4ed5-B727-ACF37FEA5BBC	[C706]

2 Messages

2.1 Transport

This protocol uses RPC **dynamic endpoints** as specified in [C706], part 4. This protocol allows any **user object** to establish a connection to the RPC server. The protocol server uses RPC as the underlying protocol to retrieve the identity of the caller that made the method call as specified in [MS-RPCE] section 3.3.3.4.3. The protocol server SHOULD<2> use this identity to perform method-specific permission checks as specified in section 3.

2.2 Common Data Types

This section specifies data types, in addition to RPC base types and definitions specified in [C706] and [MS-RPCE].

The following table summarizes the common data types used by this specification.

Data type	Description
MAX_MASTER_SECRET_BYTES (section 2.2.1)	The maximum number of bytes that the master secret can contain.
SizeOfMasterSecretInBytes (section 2.2.2)	The size in bytes of the master secret.
error_status_t (section 2.2.3)	The error code number reported by this protocol.

2.2.1 MAX_MASTER_SECRET_BYTES

The maximum number of bytes that the master secret can contain.

```
const unsigned long MAX_MASTER_SECRET_BYTES = 16;
```

2.2.2 SizeOfMasterSecretInBytes

The size of the master secret, specified in bytes. It MUST be in a range having 0 as the low allowed value and **MAX_MASTER_SECRET_BYTES** as the high allowed value.

```
typedef [range(MAX_MASTER_SECRET_BYTES,MAX_MASTER_SECRET_BYTES)] unsigned long
SizeOfMasterSecretInBytes;
```

2.2.3 error_status_t

The error code number reported by the protocol.

```
typedef unsigned long error_status_t;
```

This protocol MUST configure the RPC runtime to serialize this data type as specified in [MS-RPCE] section 2.1.

3 Protocol Details

The protocol client is simply a pass-through. That is, no additional timers or other state is required by the protocol client. Calls made by the higher-layer protocol or application are passed directly to the transport and the results returned by the transport are passed directly back to the higher-layer protocol or application.

The protocol method behaves in the same manner, regardless of whether the protocol server or the protocol client is running in a client or server version of the operating system. The protocol method returns the value "0x00000000" if it finishes processing successfully. If it fails, it sends a nonzero, implementation-specific error code. The protocol server uses nonzero Win32 error values to signify error conditions, as specified in section 1.8, unless the application requires other implementation-specific values.

The protocol client MUST NOT take action on the error codes it receives. It MUST send them to the calling application.

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.

A protocol server that implements this RPC interface MUST verify that the protocol client user object is an single sign-on (SSO) administrator and that the protocol server can retrieve the master secret.

3.1.2 Timers

None.

3.1.3 Initialization

Parameters necessary to initialize the RPC protocol are specified in section 2.1.

3.1.4 Message Processing Events and Sequencing Rules

The protocol server SHOULD<3> enforce security measures to verify that the user object was granted the permissions required to request the following routines.

This interface includes the following method:

Method	**Description**
RemoteGetMasterSecret	Returns the master secret. **opnum**: 0

3.1.4.1 RemoteGetMasterSecret (Opnum 0)

The **RemoteGetMasterSecret** method returns the master secret. A protocol server that implements this RPC interface MUST verify that the protocol client user object is a single sign-on (SSO) administrator before it retrieves the master secret. Calls associated with any other user object MUST fail.

```
error_status_t
RemoteGetMasterSecret(
[in] handle_t hBinding,
[in, out] SizeOfMasterSecretInBytes * pcbSecret,
[size_is(*pcbSecret), out] byte * pbSecret);
```

hBinding: Specifies the pointer to the protocol client remote procedure call (RPC) binding handle. The **handle_t** data type represents an explicit remote procedure call (RPC) binding handle, as specified in [C706].

pcbSecret: On input, this parameter points to a value of type **SizeOfMasterSecretInBytes** as specified in section 2.2.2. On output, this parameter MUST specify the actual number of bytes that the master secret contains.

pbSecret: A pointer to the master secret.

Return Code: An unsigned long integer whose values are specified in the following table.

Value	Description
ERROR_SUCCESS	Contains the value "0x00000000" and specifies that no error encountered.
SSO_E_ACCESSDENIED	Contains the value "0x80630005" and specifies that the user object is not a single sign-on (SSO) administrator.
E_OUTOFMEMORY	Contains the value "0x8007000E" and specifies that the input value of the *pcbSecret* parameter is less than the number of bytes needed for the master secret.
E_INVALIDARG	Contains the value "0x80070057" and specifies that the method parameters are NULL or the *pcbSecret* parameter points to a value that is less than or equal to 0.
E_UNEXPECTED	Contains the value "0x8000FFFF" and specifies that the protocol server that processes the call is not properly initialized.
RPC_S_SERVER_UNAVAILABLE	Contains the value "0xC0020017" and specifies that the protocol server that processes the call it’s shutting down.
SSO_E_EXCEPTION	Contains the value "0x80630428" and specifies that the protocol server that processes the call threw an exception.
SSO_E_MASTER_SECRET_NOT_EXIST	Contains the value "0x80630002" and specifies that the master secret does not exist.
SSO_E_SSO_NOT_CONFIGURED	Contains the value "0x8063064A" and specifies that the protocol server that processes the call is not configured.

Exceptions Thrown: No exceptions are thrown other than those thrown by the underlying protocol.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

There are no local events used on the protocol server other than the events maintained in the underlying **RPC transport**.

3.2 Protocol Client Details

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

The protocol client MUST create an RPC connection to the remote computer, using the information specified in section 2.1.

3.2.4 Message Processing Events and Sequencing Rules

This is a stateless protocol. No sequence of method calls is imposed on this protocol. When a method finishes processing, the values returned by the underlying protocol server MUST be returned unmodified to the upper layer by the protocol client.

The protocol client MUST ignore errors received from the protocol server and notify the calling application of the errors it received. It MUST also interact with the underlying RPC protocol as specified in [MS-RPCE] for message processing. Otherwise, the protocol client does not perform any special message processing.

3.2.5 Timer Events

None.

3.2.6 Other Local Events

None.

4 Protocol Examples

In this scenario, the protocol client calls the **RemoteGetMasterSecret** method on a protocol server whose URL contains "server.example.com". The protocol client originates the binding information that the protocol server uses to authenticate the protocol client. The protocol client allocates **MAX_MASTER_SECRET_BYTES** bytes for the *pbSecret* parameter before calling the remote method.

```
RemoteGetMasterSecret([in] handle_t hBinding,
[in, out] unsigned long * pcbSecret,
[out] BYTE * pbSecret);
```

On receiving this request the protocol server executes the method locally and returns:

```
Return value is: ERROR_SUCCESS(an error_status_t value)
RemoteGetMasterSecret([in] handle_t hBinding,
[in, out] unsigned long * pcbSecret,
[ out] BYTE * pbSecret);
```

5 Security

5.1 Security Considerations for Implementers

This protocol allows any user object to connect to the protocol server, as specified in section 2.1. The implementation enforces security on the method to reduce exploitable security issues. If the user object is not a single sign-on (SSO) administrator, the protocol fails as specified in section 3.1.4.1.

5.2 Index of Security Parameters

Security Parameter	Section
Authentication Protocol	2.1

6 Appendix A: Full IDL

For ease of implementation, the full **IDL** is provided in this section, where "ms-rpce.idl" refers to the IDL found in [MS-RPCE] Appendix A. The syntax uses the IDL syntax extensions defined in [MS-RPCE] Sections 2.2.4 and 3.1.1.5. For example, as noted in [MS-RPCE] section 2.2.4.9, a pointer_default declaration is not required and pointer_default(unique) is assumed.

```
import "ms-rpce.idl";
const unsigned long MAX_MASTER_SECRET_BYTES=16;
typedef [range(MAX_MASTER_SECRET_BYTES,MAX_MASTER_SECRET_BYTES)] unsigned long
SizeOfMasterSecretInBytes;
[
uuid(9D07CA0D-8F02-4ed5-B727-ACF37FEA5BBC),
version(1.0),
pointer_default(unique)
]
interface ISingleSignonRemoteMasterSecret
{
error_status_t RemoteGetMasterSecret(
[in] handle_t hBinding,
[in, out] SizeOfMasterSecretInBytes * pcbSecret,
                                                                         [size_is(*pcbSecret),
out] byte * pbSecret);
}
```

7 Appendix B: 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 Portal Server 2003
- Microsoft Office SharePoint Server 2007
- Microsoft SQL Server 2005
- Microsoft SQL Server 2008
- Microsoft SQL Server 2008 R2
- Microsoft SQL Server 2008 R2 Service Pack 1 (SP1)
- Microsoft SQL Server 2012
- Microsoft SharePoint Server 2010
- Microsoft SharePoint Server 2013
- Microsoft SharePoint Server 2016

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 1.8: Microsoft Products use only the values that are described in [MS-ERREF] section 2.2.

<2> Section 2.1: Microsoft Products use the identity of the caller to perform method specific access checks.

<3> Section 3.1.4: Office SharePoint Server 2007 uses the underlying Microsoft Windows security subsystem to determine the permissions for the caller.

8 Change Tracking

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

9 Index