[MS-ASPSS]:
ASP.NET State Service Database Repository Communications Protocol

---

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.**The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names.** The example companies, organizations, products, domain names, email 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
04/08/2008	0.1		Initial Availability
06/20/2008	0.1.1	Editorial	Revised and edited the technical content.
07/25/2008	0.1.2	Editorial	Revised and edited the technical content.
08/29/2008	0.1.3	Editorial	Revised and edited the technical content.
10/24/2008	0.1.4	Editorial	Revised and edited the technical content.
12/05/2008	0.1.5	Editorial	Revised and edited the technical content.
01/16/2009	0.1.6	Editorial	Revised and edited the technical content.
02/27/2009	0.1.7	Editorial	Revised and edited the technical content.
04/10/2009	0.1.8	Editorial	Revised and edited the technical content.
05/22/2009	0.1.9	Editorial	Revised and edited the technical content.
07/02/2009	0.1.10	Editorial	Revised and edited the technical content.
08/14/2009	0.1.11	Editorial	Revised and edited the technical content.
09/25/2009	0.2	Minor	Updated the technical content.
11/06/2009	0.2.1	Editorial	Revised and edited the technical content.
12/18/2009	0.2.2	Editorial	Revised and edited the technical content.
01/29/2010	0.2.3	Editorial	Revised and edited the technical content.
03/12/2010	1.0	Major	Updated and revised the technical content.
04/23/2010	1.0.1	Editorial	Revised and edited the technical content.
06/04/2010	1.0.2	Editorial	Revised and edited the technical content.
07/16/2010	1.0.2	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	1.0.2	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	1.0.2	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	1.0.2	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	1.0.2	No change	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
02/11/2011	1.0.2	No change	No changes to the meaning, language, or formatting of the technical content.
03/25/2011	1.0.2	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	1.0.2	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	1.1	Minor	Clarified the meaning of the technical content.
09/23/2011	1.1	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	1.1	No change	No changes to the meaning, language, or formatting of the technical content.
03/30/2012	1.1	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	1.1	No change	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	1.1	No change	No changes to the meaning, language, or formatting of the technical content.
01/31/2013	1.1	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

The ASP .NET State Service Database Stored Procedure specifies an interface for clients to store and retrieve serialized session data.

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

1.1 Glossary

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

UTC (Coordinated Universal Time)

The following terms are specific to this document:

application identifier: A unique integer that identifies a protocol client application.

exclusive access: An instance of session data is considered to be reserved for **exclusive access** if stored procedures allow only a single client to read or write an instance of session data at a time.

expiration date: A UTC datetime indicating when an instance of session data is considered expired and thus no longer valid.

lock age: Indicates the age of a **virtual lock**.

lock cookie: An integer that uniquely identifies a **virtual lock** associated with an instance of session data.

time-out: An integer value, in seconds, indicating the lifetime of an instance of session data.

virtual lock: A programming approach for marking an instance of session data as reserved for **exclusive access** by a single client. Implementation of a **virtual lock** does not require use of a protocol server's physical lock semantics to establish **exclusive access** to an instance of session data.

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

1.2 References

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

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site,

http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MSDN-MS-TDS] Microsoft Corporation, "Tabular Data Stream Protocol Specification", http://msdn.microsoft.com/en-us/library/cc448435.aspx

[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".

1.3 Overview

Clients create instances of session data that are uniquely identified by a session identifier. Session data is stored in a session data store for future retrieval. When clients retrieve instances of session data from a session data store, clients can optionally request that a **virtual lock** be placed on the session data to ensure **exclusive access**.

The stored procedure in this protocol enables clients to add, modify, retrieve, and delete session data in a session data store. This protocol also enables clients to remove virtual locks from instances of session data.

If a client intends to modify session data, the client will request exclusive access for the session data. The client receives a **lock cookie** identifying the virtual lock that is placed on the session data. Once a client has completed working with an instance of session data, the client can store the updated session data in the session data store and release the virtual lock identified by the lock cookie, thus allowing other clients to access the session data. If no changes were made to an instance of session data, clients can release the virtual lock without making any modifications to the session data in the session data store.

In some client scenarios, a virtual lock held on a piece of session data will be considered stale. In these scenarios a client can forcibly remove the virtual lock from the session data.

Clients can modify the **expiration date** for an instance of session data in the session data store without modifying the session data itself. Clients can also remove an instance of session data from the session data store.

1.4 Relationship to Other Protocols

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

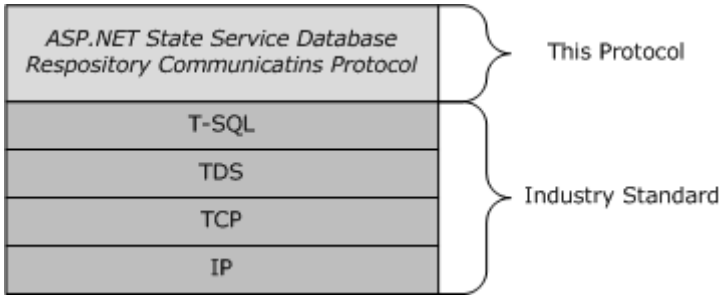

Figure 1: The ASP.NET State Service Database Repository Communications Protocol transport stack

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 on the back-end database server.

1.6 Applicability Statement

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

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the documentation of the following stored procedures: TempGetVersion and GetMajorVersion.

Security and Authentication Methods: This protocol supports the Security Support Provider Interface (SSPI) and SQL Authentication with the protocol server role specified in [MSDN-MS-TDS].

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

[MSDN-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 ASP .NET Session State Database Protocol uses standard data types as specified in the Transact-Structured Query Language (T-SQL).

2.2.1 Result Sets

2.2.1.1 Long Session Data Result Set

The Long Session Data result set is specified using T-SQL syntax, as follows.

```
SessionItemLong   image
```

SessionItemLong: Contains serialized session data.

2.2.2 XML Structures

None.

3 Protocol Details

The ASP .NET Session State Database Protocol allows protocol servers to perform implementation-specific localization of text in various messages. Except where specified, the localization of this text is an implementation-specific behavior of the protocol server and not significant for interoperability.

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 client is partially defined by an application name. The protocol server creates **application identifiers** based on application names and maintains a mapping of protocol client application names to application identifiers.

A protocol client creates session identifiers by including the protocol server generated application identifier as part of each session identifier.<1>

The protocol server maintains a mapping from the session identifier to the storage location of the serialized session data. A client MUST use the result of this mapping as part of the session identifiers passed into other stored procedures.

The protocol server accepts requests to add, modify, query, and delete session data in the session data store using the session identifiers as a primary key. Each instance of session data has an expiration date. An expiration date MUST be computed by adding the **time-out** value associated with an instance of session data to the current **UTC (Coordinated Universal Time)** time on the protocol server.

A protocol server MUST support storing session data using different storage mechanisms. A protocol server MUST have storage mechanisms that accommodate large quantities of session data. A protocol server MUST support storing session data using more compact storage mechanisms. A protocol server MUST support the ability to release storage previously allocated using either storage mechanism.

A client can request exclusive access to an instance of session data. If the protocol server handles the request it places a virtual lock on an instance of session data. Each virtual lock is associated with a lock cookie that uniquely identifies the virtual lock.

The **lock age** for a virtual lock placed on an instance of session data MUST be in seconds and MUST be computed by comparing the UTC date when the virtual lock was originally created on the protocol server against the current UTC date on the protocol server.

Clients with exclusive access to an instance of session data use the lock cookie on subsequent communications with the protocol server to indicate that the client is allowed to modify an instance of session data.

3.1.2 Timers

An execution time-out timer on the protocol server governs the execution time for any requests. The amount of time is specified by a time-out value that is configured on the protocol server for all connections.

3.1.3 Initialization

A connection that uses the underlying protocol layers that are specified in section 1.4 MUST be established before using this protocol, as specified in [MSDN-MS-TDS].

3.1.4 Message Processing Events and Sequencing Rules

This section describes the following stored procedures.

Procedure name	Description
TempGetVersion	Retrieves session data store schema version information
GetMajorVersion	Retrieves the version number of the protocol server
TempGetAppID	Retrieves the application identifier for a given application name
TempGetStateItem3	Retrieves session data from the session data store without placing a virtual lock on the session data
TempGetStateItemExclusive3	Retrieves session data from the session data store and attempts to place a virtual lock on the session data for exclusive access
TempReleaseStateItemExclusive	Removes a virtual lock from an instance of session data
TempRemoveStateItem	Deletes an instance of session data from the session data store
TempResetTimeout	Updates the expiration date of a specific instance of session data in the session data store
TempInsertStateItemShort	Inserts session data in the session data store and associates it with a session identifier
TempInsertStateItemLong	Inserts session data in the session data store and associates it with a session identifier
TempUpdateStateItemShort	Updates an instance of session data in the session data store
TempUpdateStateItemLong	Updates an instance of session data in the session data store
TempUpdateStateItemShortNullLong	Updates an instance of session data in the session data store
TempUpdateStateItemLongNullShort	Updates an instance of session data in the session data store

The T-SQL syntax for each stored procedure and result set, as well as the variables they are composed of, are defined in [MSDN-MS-TDS]. In the T-SQL syntax, the variable name is followed by the type of the variable, which MAY optionally have a length value in brackets and MAY optionally have a default value indicated by an equals sign followed by the default value and MAY optionally indicate that a value is returned to the calling client in the variable.

For definitional clarity, a name has been assigned to any columns in the result sets that do not have a defined name in their current implementation. This does not affect the operation of the result set, because the ordinal position of any column with no defined name is expected by the client.

3.1.4.1 TempGetVersion

The TempGetVersion stored procedure is called to retrieve a fixed value that indicates the existence of the correct schema in the session data store.

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

```
PROCEDURE TempGetVersion(
  [out] char @ver[10]
);
```

@ver: The value passed into the stored procedure in this parameter MUST be ignored.

Return Values: The stored procedure MUST return a character value of "2" in the *@ver* output variable. The stored procedure MUST return an integer return code, which MUST be 0.

Result sets: The stored procedure MUST NOT return a result set.

3.1.4.2 GetMajorVersion

The GetMajorVersion stored procedure is called to retrieve the major version number of the protocol server running the stored procedure.

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

```
PROCEDURE GetMajorVersion(
  [OUTPUT] int @@ver
);
```

@@ver: The value passed into the stored procedure in this parameter MUST be ignored.

Return Values: The stored procedure MUST return the major version number of the protocol server in the *@@ver* output variable. The version number MUST be greater than or equal to 8. The stored procedure MUST return an integer return code, which MUST be 0.

Result sets: The stored procedure MUST NOT return a result set.

3.1.4.3 TempGetAppID

The TempGetAppID stored procedure is called to retrieve an integer value that maps to the name of the application passed into the stored procedure.

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

```
PROCEDURE TempGetAppID(
  varchar @appName[280],
  [OUTPUT] int @appID
);
```

@appName: The name of the application calling the stored procedure.

@appID: The value passed into the stored procedure in this parameter MUST be ignored.

Return Values: The stored procedure MUST return an integer uniquely identifying the calling application in the *@appID* output variable. The same integer value MUST be returned for a given application name during subsequent calls to the stored procedure.

If no error occurs, the stored procedure MUST return an integer return code of 0.

If the stored procedure fails to generate a unique integer for the application name, it MUST raise a T-SQL error.

Result sets: The stored procedure MUST NOT return a result set.

3.1.4.4 TempGetStateItem3

The TempGetStateItem3 stored procedure retrieves session data and related session information associated with a session identifier. The stored procedure MUST always update the expiration date of the session data as a side effect of running the stored procedure.

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

```
PROCEDURE TempGetStateItem3(
  nvarchar @id[88],
  [OUTPUT] varbinary @itemShort[7000],
  [OUTPUT] bit @locked,
  [OUTPUT] int @lockAge,
  [OUTPUT] int @lockCookie,
  [OUTPUT] int @actionFlags
);
```

@id: The stored procedure attempts to retrieve session data and related session information associated with the supplied session identifier.

@itemShort: The value passed into the stored procedure in this parameter MUST be ignored.

@locked: The value passed into the stored procedure in this parameter MUST be ignored.

@lockAge: The value passed into the stored procedure in this parameter MUST be ignored.

@lockCookie: The value passed into the stored procedure in this parameter MUST be ignored.

@actionFlags: The value passed into the stored procedure in this parameter MUST be ignored.

Return Values: If no session data exists for the identifier supplied in the *@id* variable, a NULL value MUST be returned in the following output variables: *@itemShort*, *@locked*, *@lockAge*, *@lockCookie*, *@actionFlags*.

If the session data identified by the *@id* variable does not have a virtual lock on it, and the size of the session data in the session data store is less than or equal to 7,000 bytes, the stored procedure MUST return the session data in the *@itemShort* output parameter. Otherwise a NULL value MUST be returned in the *@itemShort* output parameter.

If the session data identified by the *@id* variable does not have a virtual lock on it, a 0 MUST be returned in the *@locked* output parameter. Otherwise a 1 MUST be returned in the *@locked* output parameter.

If the session data identified by the *@id* variable does not have a virtual lock on it, the value returned in the *@lockAge* output parameter MUST be 0. Otherwise the stored procedure MUST return the lock age of the session data's virtual lock in the *@lockAge* output parameter.

If the session data identified by the *@id* variable does not have a virtual lock on it, the value returned in the *@lockCookie* output parameter MUST be an implementation-specific integer value that MUST be ignored by the protocol client. Otherwise, the stored procedure MUST return the lock cookie currently associated with the virtual lock in the *@lockCookie* output parameter.

The stored procedure MUST return a value of 0 in the *@actionFlags* output parameter.

The stored procedure MUST return an integer return code, which MUST be 0.

Result sets: The stored procedure MUST return the result set specified in Long Session Data Result Set (section 2.2.1.1) if the session data does not have a virtual lock on it and the size of the session data in the session data store is greater than 7,000 bytes. If the size of the session data is less than or equal to 7,000 bytes or the session data has a virtual lock placed on it, the stored procedure MUST NOT return a result set.

3.1.4.4.1 Long Session Data Result Set

See Long Session Data Result Set (section 2.2.1.1).

3.1.4.5 TempGetStateItemExclusive3

The TempGetStateItemExclusive3 stored procedure attempts to retrieve session data and related session information associated with a session identifier. The stored procedure MUST also attempt to place a virtual lock on the session data if it is not currently locked. The stored procedure MUST always update the expiration date of the session data as a side effect of running the stored procedure.

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

```
PROCEDURE TempGetStateItemExclusive3(
  nvarchar @id[88],
  [OUTPUT] varbinary @itemShort[7000],
  [OUTPUT] bit @locked,
  [OUTPUT] int @lockAge,
  [OUTPUT] int @lockCookie,
  [OUTPUT] int @actionFlags
);
```

@id: The stored procedure attempts to retrieve session data and related session information associated with the supplied session identifier.

@itemShort: The value passed into the stored procedure in this parameter MUST be ignored.

@locked: The value passed into the stored procedure in this parameter MUST be ignored.

@lockAge: The value passed into the stored procedure in this parameter MUST be ignored.

@lockCookie: The value passed into the stored procedure in this parameter MUST be ignored.

@actionFlags: The value passed into the stored procedure in this parameter MUST be ignored.

Return Values: If no session data exists for the identifier supplied in the *@id* variable, a NULL value MUST be returned in the following output variables: *@itemShort*, *@locked*, *@lockAge*, *@lockCookie*, *@actionFlags*.

If the session data identified by the *@id* variable does not have a virtual lock on it, and the size of the session data in the session data store is less than or equal to 7,000 bytes, the stored procedure MUST return the session data in the *@itemShort* output parameter. Otherwise a NULL value MUST be returned in the *@itemShort* output parameter.

If the session data identified by the *@id* variable does not have a virtual lock on it, a 0 MUST be returned in the *@locked* output parameter and the stored procedure MUST place a virtual lock on the session data. Otherwise. a 1 MUST be returned in the *@locked* output parameter.

If the session data identified by the *@id* variable does not have a virtual lock on it, the value returned in the *@lockAge* output parameter MUST be 0. Otherwise, the stored procedure MUST return the lock age of the session data's virtual lock in the *@lockAge* output parameter.

If the session data identified by the *@id* variable does not have a virtual lock on it, the stored procedure MUST place a virtual lock on it. Furthermore, the stored procedure MUST return the lock cookie associated with the virtual lock in the *@lockCookie* output parameter.

If the session data identified by the *@id* variable already has a virtual lock on it, the stored procedure MUST return the lock cookie currently associated with the virtual lock in the *@lockCookie* output parameter.

The stored procedure MUST return a value of 0 in the *@actionFlags* output parameter.

The stored procedure MUST return an integer return code, which MUST be 0.

Result sets: The stored procedure MUST return the result set as specified in Long Session Data Result Set (section 2.2.1.1) if the session data does not have a virtual lock on it and the size of the session data in the session data store is greater than 7,000 bytes. If the size of the session data is less than or equal to 7,000 bytes or the session data has a virtual lock placed on it, the stored procedure MUST NOT return a result set.

3.1.4.5.1 Long Session Data Result Set

See Long Session Data Result Set (section 2.2.1.1)

3.1.4.6 TempReleaseStateItemExclusive

The TempReleaseStateItemExclusive stored procedure MUST remove a virtual lock from a specific session data instance if the state was previously locked.

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

```
PROCEDURE TempReleaseStateItemExclusive(
  nvarchar @id[88],
  int @lockCookie
);
```

@id: The stored procedure removes the virtual lock from the session data associated with the supplied session identifier. If no session data exists for the identifier supplied in the *@id* variable, the stored procedure MUST make no changes.

@lockCookie: The stored procedure MUST remove the virtual lock from the session data if the value supplied in this parameter matches the lock cookie currently associated with the instance of session data. Otherwise, the stored procedure MUST NOT remove the virtual lock. If the virtual lock was removed, the stored procedure MUST update the expiration date of the session data as well.

Return Values: The stored procedure MUST return an integer return code, which MUST be 0.

Result sets: The stored procedure MUST NOT return a result set.

3.1.4.7 TempRemoveStateItem

The TempRemoveStateItem stored procedure removes a specific session data instance from the session data store.

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

```
PROCEDURE TempRemoveStateItem(
  nvarchar @id[88],
  int @lockCookie
);
```

@id: The stored procedure removes the session data associated with the supplied session identifier from the session data store. If no session data exists for the identifier supplied in the *@id* variable, the stored procedure MUST make no changes.

@lockCookie: The stored procedure MUST remove the session data if the value supplied in this parameter matches the lock cookie currently associated with the instance of session data. Otherwise, the stored procedure MUST make no changes.

Return Values: The stored procedure MUST return an integer return code, which MUST be 0.

Result sets: The stored procedure MUST NOT return a result set.

3.1.4.8 TempResetTimeout

The TempResetTimeout stored procedure updates the expiration date of a specific instance of session data in the session data store.

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

```
PROCEDURE TempResetTimeout(
  nvarchar @id[88]
);
```

@id: The stored procedure updates the expiration date of the session data associated with the supplied session identifier. If no session data exists for the identifier supplied in the *@id* variable, the stored procedure MUST make no changes.

Return Values: The stored procedure MUST return an integer return code, which MUST be 0.

Result sets: The stored procedure MUST NOT return a result set.

3.1.4.9 TempInsertStateItemShort

The TempInsertStateItemShort stored procedure inserts session data into the session data store and associates it with a session identifier. This stored procedure indicates to the protocol server that the session data MUST be stored using a compact storage mechanism.

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

```
PROCEDURE TempInsertStateItemShort(
  nvarchar @id[88],
  varbinary @itemShort[7000],
  int @timeout
);
```

@id: The stored procedure associates the inserted session data in the session data store with the supplied session identifier.

@itemShort: The session data that MUST be stored. The stored procedure MUST store the session data without placing a virtual lock on it.

@timeout: An integer value in seconds indicating the lifetime of the supplied session data.

Return Values: The stored procedure MUST return an integer return code, which MUST be 0.

Result sets: The stored procedure MUST NOT return a result set.

If an attempt is made to insert two or more instances of session data associated with the same session identifier, the stored procedure MUST raise a SQL primary key violation error to the protocol client.

3.1.4.10 TempInsertStateItemLong

The TempInsertStateItemLong stored procedure inserts session data into the session data store and associates it with a session identifier. This stored procedure indicates to the protocol server that the session data MUST be stored using a storage mechanism that supports large data capacities.

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

```
PROCEDURE TempInsertStateItemLong(
  nvarchar @id[88],
  image @itemLong,
  int @timeout
);
```

@id: The stored procedure associates the inserted session data in the session data store with the supplied session identifier.

@itemLong: The session data that MUST be stored. The stored procedure MUST store the session data without placing a virtual lock on it.

@timeout: An integer value, in seconds, indicating the lifetime of the supplied session data.

Return Values: The stored procedure MUST return an integer return code, which MUST be 0.

Result sets: The stored procedure MUST NOT return a result set.

If an attempt is made to insert two or more instances of session data associated with the same session identifier, the stored procedure MUST raise a SQL primary key violation error to the protocol client.

3.1.4.11 TempUpdateStateItemShort

The TempUpdateStateItemShort stored procedure attempts to update an instance of session data in the session data store. This stored procedure indicates to the protocol server that the session data MUST be stored using a compact storage mechanism.

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

```
PROCEDURE TempUpdateStateItemShort(
  nvarchar @id[88],
  varbinary @itemShort[7000],
  int @timeout,
  int @lockCookie
);
```

@id: The stored procedure updates the session data in the session data store associated with the supplied session identifier.

@itemShort: The new session data that MUST be stored if the lock cookie supplied in the *@lockCookie* parameter matches the lock cookie currently associated with the instance of session data. The stored procedure MUST clear the virtual lock from the session data as a side effect of updating the session data.

@timeout: An integer value, in seconds, indicating the lifetime of the supplied session data. If the lock cookie supplied in the *@lockCookie* parameter matches the lock cookie currently associated with the instance of session data, the stored procedure MUST update the time-out associated with the instance of session data.

@lockCookie: The stored procedure MUST only update the session data in the session data store if the lock cookie supplied in the *@lockCookie* parameter matches the lock cookie currently associated with the instance of session data. If this parameter does not match the lock cookie currently associated with the instance of session data, the stored procedure MUST make no changes.

Return Values: The stored procedure MUST return an integer return code, which MUST be 0.

Result sets: The stored procedure MUST NOT return a result set.

3.1.4.12 TempUpdateStateItemShortNullLong

The TempUpdateStateItemShortNullLong stored procedure attempts to update an instance of session data in the session data store. This stored procedure indicates to the protocol server that the session data MUST be stored using a compact storage mechanism. Furthermore, the protocol server MUST release any unused storage currently previously allocated using larger storage capacity.

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

```
PROCEDURE TempUpdateStateItemShortNullLong(
  nvarchar @id[88],
  varbinary @itemShort[7000],
```

```
  int @timeout,
  int @lockCookie
);
```

@id: The stored procedure updates the session data in the session data store associated with the supplied session identifier.

@itemShort: The new session data that MUST be stored if the lock cookie supplied in the *@lockCookie* parameter matches the lock cookie currently associated with the instance of session data. The stored procedure MUST clear the virtual lock from the session data as a side effect of updating the session data.

@timeout: An integer value, in seconds, indicating the lifetime of the supplied session data. If the lock cookie supplied in the *@lockCookie* parameter matches the lock cookie currently associated with the instance of session data, the stored procedure MUST update the time-out associated with the instance of session data.

@lockCookie: The stored procedure MUST only update the session data in the session data store if the lock cookie supplied in the *@lockCookie* parameter matches the lock cookie currently associated with the instance of session data. If this parameter does not match the lock cookie currently associated with the instance of session data, the stored procedure MUST make no changes.

Return Values: The stored procedure MUST return an integer return code, which MUST be 0.

Result sets: The stored procedure MUST NOT return a result set.

3.1.4.13 TempUpdateStateItemLong

The TempUpdateStateItemLong stored procedure attempts to update an instance of session data in the session data store. This stored procedure indicates to the protocol server that the session data MUST be stored using a storage mechanism that supports large data capacities.

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

```
PROCEDURE TempUpdateStateItemLong(
  nvarchar @id[88],
  image @itemLong,
  int @timeout,
  int @lockCookie
);
```

@id: The stored procedure updates the session data in the session data store associated with the supplied session identifier.

@itemLong: The new session data that MUST be stored if the lock cookie supplied in the *@lockCookie* parameter matches the lock cookie currently associated with the instance of session data. The stored procedure MUST clear the virtual lock from the session data as a side effect of updating the session data.

@timeout: An integer value, in seconds, indicating the lifetime of the supplied session data. If the lock cookie supplied in the *@lockCookie* parameter matches the lock cookie currently associated with the instance of session data, the stored procedure MUST update the time-out associated with the instance of session data.

@lockCookie: The stored procedure MUST only update the session data in the session data store if the lock cookie supplied in the *@lockCookie* parameter matches the lock cookie currently associated with the instance of session data. If this parameter does not match the lock cookie currently associated with the instance of session data, the stored procedure MUST make no changes.

Return Values: The stored procedure MUST return an integer return code, which MUST be 0.

Result sets: The stored procedure MUST NOT return a result set.

3.1.4.14 TempUpdateStateItemLongNullShort

The TempUpdateStateItemLongNullShort stored procedure attempts to update an instance of session data in the session data store. This stored procedure indicates to the protocol server that the session data MUST be stored using a storage mechanism that supports large data capacities. Furthermore, the protocol server MUST release any unused storage previously allocated using more compact storage mechanisms.

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

```
PROCEDURE TempUpdateStateItemLongNullShort(
  nvarchar @id[88],
  image @itemLong,
  int @timeout,
  int @lockCookie
);
```

@id: The stored procedure updates the session data in the session data store associated with the supplied session identifier.

@itemLong: The new session data that MUST be stored if the lock cookie supplied in the *@lockCookie* parameter matches the lock cookie currently associated with the instance of session data. The stored procedure MUST clear the virtual lock from the session data as a side effect of updating the session data.

@timeout: An integer value, in seconds, indicating the lifetime of the supplied session data. If the lock cookie supplied in the *@lockCookie* parameter matches the lock cookie currently associated with the instance of session data, the stored procedure MUST update the time-out associated with the instance of session data.

@lockCookie: The stored procedure MUST only update the session data in the session data store if the lock cookie supplied in the *@lockCookie* parameter matches the lock cookie currently associated with the instance of session data. If this parameter does not match the lock cookie currently associated with the instance of session data, the stored procedure MUST make no changes.

Return Values: The stored procedure MUST return an integer return code, which MUST be 0.

Result sets: The stored procedure MUST NOT return a result set.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

4 Protocol Examples

4.1 Protocol Client Initialization (TempGetVersion, GetMajorVersion, TempGetAppID)

This example describes the requests made when a typical protocol client initializes its connection to the session data protocol server.

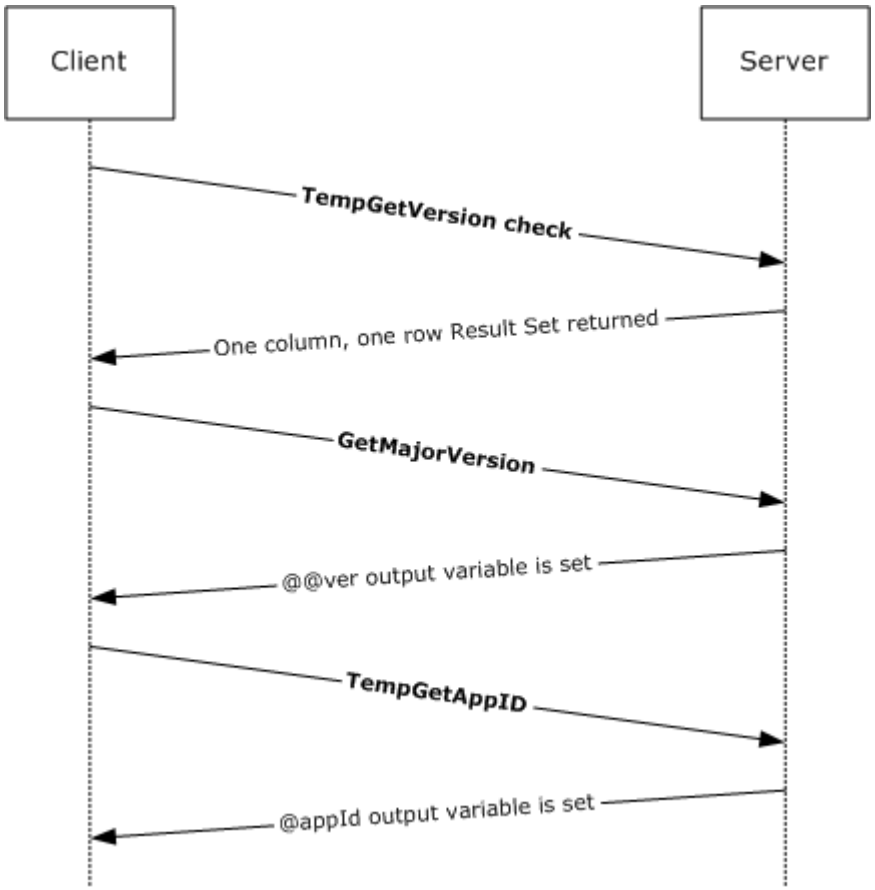

Figure 2: Protocol client initialization sequence

1. The protocol client checks to see if the TempGetVersion stored procedure exists.

```
Select name from sysobjects where type = 'P' and name =
'TempGetVersion'
```

2. If the protocol server can service session state requests, it returns a one-column, one-row result set whose value is TempGetVersion.

3. The protocol client then determines the version of the protocol server by executing the GetMajorVersion stored procedure.

```
exec dbo.GetMajorVersion @@ver=@p1 output
```

4. The protocol server returns the value 8 in the *@@ver* output parameter.

5. Because GetMajorVersion returns a value of 8 in the *@@ver* output parameter, the protocol client then calls TempGetAppID to retrieve its protocol server–generated unique identifier.

```
exec dbo.TempGetAppID
@appName='/LM/W3SVC/1/ROOT/SessionStateSerialization',@appID=@p2
output
```

6. The protocol server returns a protocol server–generated unique identifier in the *@appID* output parameter. The protocol client stores this value for later use when generating session identifiers.

4.2 Creating Session Data (TempInsertStateItemShort)

This example describes the request made by a protocol client in order to create session data in the session data store using a compact storage mechanism.

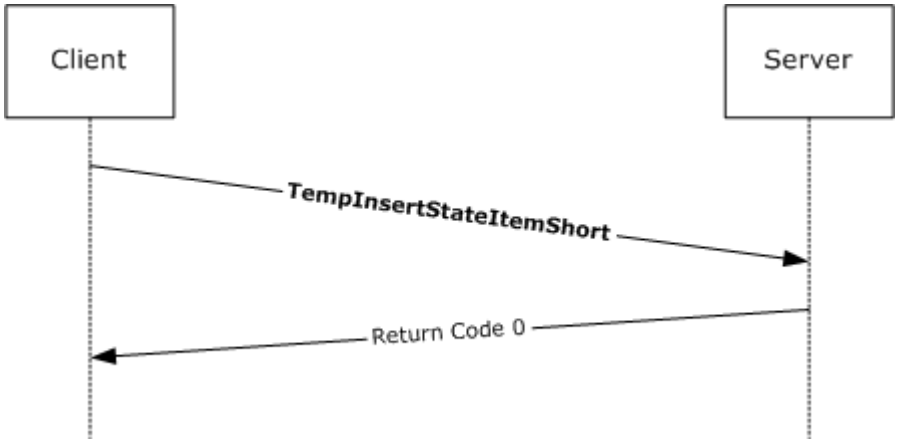

Figure 3: Session data creation sequence

1. The protocol client calls TempInsertStateItemShort with a protocol client–generated session identifier, session data, and time-out to create a new instance of session data on the protocol server.

```
exec dbo.TempInsertStateItemShort
@id=N'5ve0ag45ylticd3giq5a1bbhcd0903f9',@itemShort=0x1400...truncated_
for_brevity...0BFF,@timeout=20
```

2. The protocol server returns a 0 return code for successful execution, which is ignored by the client.

4.3 Retrieving and then Updating Session Data with Exclusive Access (TempGetStateItemExclusive3, TempUpdateStateItemLong)

This example describes the requests made by a protocol client to retrieve and subsequently update session data in the protocol server that supports large data capacities.

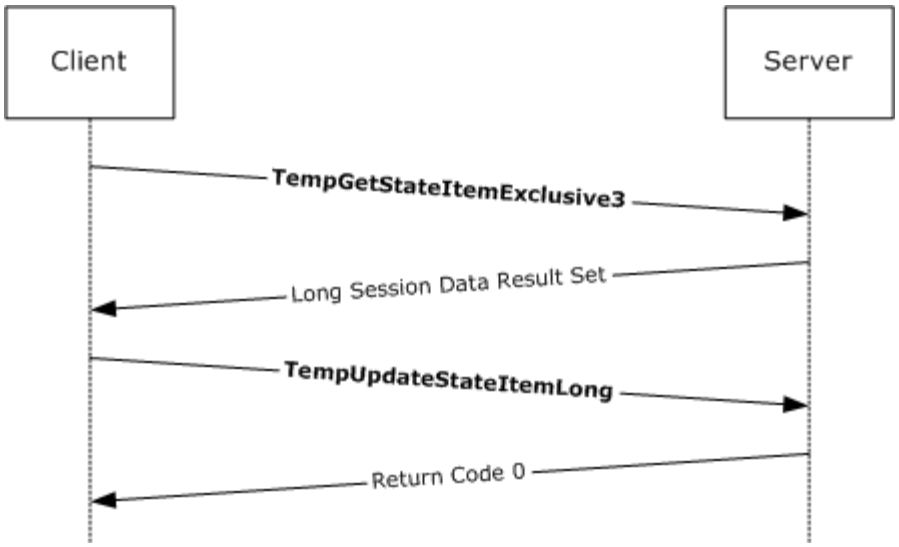

Figure 4: Session data retrieval/update sequence

1. The protocol client calls TempGetStateItemExclusive3 with a session identifier in order to retrieve session data from the protocol server with an exclusive lock.

```
exec dbo.TempGetStateItemExclusive3
@id=N'5ve0ag45ylticd3giq5a1bbhcd0903f9',@itemShort=@p2
output,@locked=@p3 output,@lockAge=@p4 output,@lockCookie=@p5
output,@actionFlags=@p6 output
```

2. The protocol server responds with output parameters: the lock cookie in the *@lockCookie* output parameter, the lock age in the *@lockAge* output parameter, the locked state of the session data in the *@locked* output parameter, and a value of 0 in the *@actionFlags* parameter. The protocol server also returns the session data in the result set specified in Long Session Data Result Set (section 2.2.1.1), which the protocol client uses an implementation specific manner.

3. If the session data has changed, the protocol client updates the session data in the protocol server by calling TempUpdateStateItemLong with a lock cookie and a protocol client–generated session identifier, as well as session data and a lock time-out.

```
exec dbo.TempUpdateStateItemLong
@id=N'5ve0ag45ylticd3giq5a1bbhcd0903f9',@itemLong=0x1400...truncated_f
or_brevity...0BFF,@timeout=20,@lockCookie=9
```

4. The protocol server returns a 0 return code for successful execution, which is ignored by the client.

5 Security

5.1 Security Considerations for Implementers

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

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

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

- Windows SharePoint Services
- Windows Vista operating system
- Windows Server 2003 operating system
- Windows Server 2008 operating system
- Windows 7 operating system
- Windows Server 2008 R2 operating system

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 3.1.1: The Microsoft convention for constructing a session identifier is to append an application identifier as a string of eight hexadecimal characters to the end of a globally unique client-generated string. However, stored procedures always treat the session identifier as an opaque string and assign no special behavior or significance to the characters in a session identifier.

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

V

X