[MS-SPSTATE]:
Temporary State Service Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
07/13/2009	0.1	Major	Initial Availability
08/28/2009	0.2	Editorial	Revised and edited the technical content
11/06/2009	0.3	Editorial	Revised and edited the technical content
02/19/2010	1.0	Editorial	Revised and edited the technical content
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Editorial	Revised and edited the technical content
06/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	1.05	Minor	Clarified the meaning of the technical content.
09/27/2010	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	1.05	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies the Temporary State Service Protocol, which enables an interface for clients to store and retrieve binary data.

1.1 Glossary

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

Coordinated Universal Time (UTC)

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

back-end database server
session data

The following terms are specific to this document:

exclusive access: A condition where only one protocol client at a time is permitted to read or write an instance of binary data.

expiration time: A date-time value, in Coordinated Universal Time (UTC) format, that indicates when an instance of binary data is no longer valid.

lock age: The number of seconds after a lock was placed on an instance of session data. Protocol clients refer to the lock age to determine whether a lock is valid for a specific instance of session data.

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

virtual lock: A condition in which an instance of session data is reserved for exclusive access by a single protocol client. An implementation of a virtual lock does not require the use of physical lock semantics on a protocol server.

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-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)

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

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

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

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

Clients can modify the **expiration time** for an instance of binary data in the binary data store without modifying the binary data itself. Clients can also remove an instance of binary data or all instances of expired binary data from the binary data store.

1.4 Relationship to Other Protocols

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

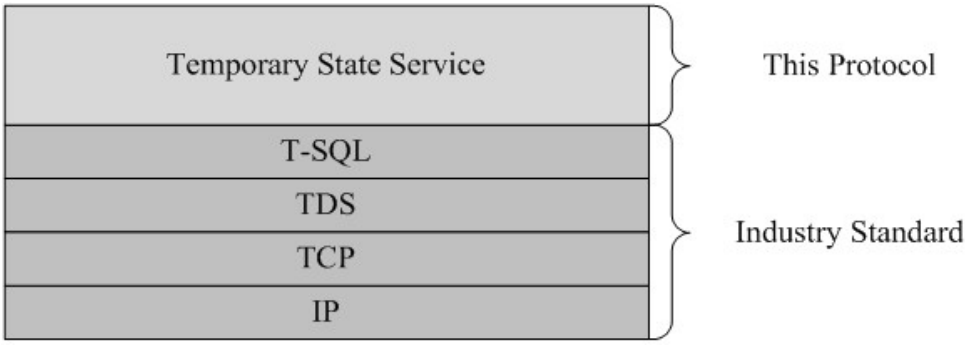

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

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

None.

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, query SQL views or SQL tables, and return result codes.

2.2 Common Data Types

The Temporary State Service Protocol uses T-SQL (Transact-Structured Query Language) standard data types as specified in [MSDN-TSQL-Ref].

2.2.1 Simple Data Types and Enumerations

No common simple data types or enumerations are defined in this protocol.

2.2.2 Bit Fields and Flag Structures

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

2.2.3 Binary Structures

No common binary structures are defined in this protocol.

2.2.4 Result Sets

No common Result Sets are defined in this protocol.

2.2.5 Tables and Views

No common table or view structures are defined in this protocol.

2.2.6 XML Structures

No common XML Structures are defined in this protocol.

2.2.6.1 Namespaces

None.

2.2.6.2 Simple Types

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

2.2.6.3 Complex Types

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

2.2.6.4 Elements

This specification does not define any common XML Schema element definitions.

2.2.6.5 Attributes

This specification does not define any common XML Schema attribute definitions.

2.2.6.6 Groups

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

2.2.6.7 Attribute Groups

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

3 Protocol Details

3.1 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 protocol server maintains mappings from binary data identifiers to the storage location of the binary data. The protocol server accepts requests to add, modify, query, and delete binary data in the binary data store using the identifiers as a primary key.

Each instance of binary data has an expiration time, which is computed by adding the time-out value associated with the instance of binary data to the current **Coordinated Universal Time (UTC)** of the protocol server. The expiration time associated with an instance of binary data is changed when specific stored procedures are called. An instance of binary data is considered expired if the expiration time has passed.

A client can request exclusive access to an instance of binary data. If the protocol server handles the request, it MUST place a virtual lock on that instance of binary data. If the protocol server does not handle the request, it MUST NOT change any 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 binary 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 binary data MUST use the lock cookie on subsequent communications with the protocol server to indicate that the client is allowed to modify an instance of binary 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 described in section 1.4 MUST be established before using this protocol as specified in [MS-TDS].

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

This section describes the following stored procedures.

Procedure name	Description
proc_AddItem	Inserts binary data in the binary data store and associates it with an identifier.
proc_DeleteExpiredItems	Deletes all expired instances of binary data from the binary data store.
proc_DeleteItem	Deletes an instance of binary data from the binary data store.
proc_GetItemWithoutLock	Retrieves binary data from the binary data store without placing a virtual lock on the binary data.
proc_GetItemWithLock	Retrieves binary data from the binary data store and attempts to place a virtual lock on the binary data for exclusive access.
proc_RefreshItemExpiration	Updates the expiration time of an instance of binary data in the binary data store.
proc_ReleaseItemLock	Removes a virtual lock from an instance of binary data
proc_UpdateItem	Updates an instance of binary data in the binary data store.

The T-SQL (Transact-Structured Query Language) syntax for each stored procedure and the variables they are composed of is defined in the [MSDN-TSQL-Ref] protocol. In the T-SQL syntax, the variable name is followed by the type of the variable which can optionally have a length value in brackets and can optionally have a default value indicated by an equals sign followed by the default value.

3.1.5.1 proc_AddItem

The proc_AddItem stored procedure inserts binary data into the binary data store and associates it with an identifier.

```
PROCEDURE proc_AddItem (
@id varchar(512),
@item varbinary(max),
@timeout int,

);
```

@id: The stored procedure MUST associate the binary data in the binary data store with the supplied identifier. @id MUST NOT be NULL. Binary data associated with the same @id MUST NOT already exist in the binary data store.

@item: The binary data that MUST be stored. The stored procedure MUST NOT place a virtual lock on the stored binary data.

@timeout: A positive integer value in minutes indicating the time-out of the binary data.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.2 proc_DeleteExpiredItems

The proc_DeleteExpiredItems stored procedure MUST remove batches of binary data which are expired from the binary data store until no expired binary data instances remain.

```
PROCEDURE proc_DeleteExpiredItems (

);
```

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.3 proc_DeleteItem

The proc_DeleteItem stored procedure MUST remove an instance of the binary data from the binary data store if there is binary data associated with the supplied identifier and the supplied lock cookie. Otherwise, the stored procedure MUST NOT make any changes.

```
PROCEDURE proc_DeleteItem (
@id varchar(512),
@lockCookie int,

);
```

@id: The identifier of binary data to remove.

@lockCookie: The lock cookie of the binary data to remove.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.4 proc_GetItemWithLock

The proc_GetItemWithLock stored procedure attempts to retrieve binary data associated with an identifier. The stored procedure MUST place a virtual lock on the binary data if it is not currently locked. The stored procedure MUST also update the expiration time of the binary data as specified in section 3.1.1.

```
PROCEDURE proc_GetItemWithLock (
@id varchar(512),
@item varbinary(max) OUTPUT,
@locked bit OUTPUT,
@lockAgeInSeconds int OUTPUT,
@lockCookie int OUTPUT,

);
```

@id: The stored procedure attempts to retrieve binary data associated with the supplied identifier.

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

If no binary data exists for the identifier supplied in the @id variable, a NULL value MUST be returned.

If the binary data identified by the @id variable does not have a virtual lock on it, the stored procedure MUST return the binary data in the @item output parameter. Otherwise a NULL value MUST be returned in the @item output parameter.

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

If no binary data exists for the identifier supplied in the @id variable, a NULL value MUST be returned.

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

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

If no binary data exists for the identifier supplied in the @id variable, a NULL value MUST be returned.

If the binary data identified by the @id variable does not have a virtual lock on it, the value returned in the @lockAgeInSeconds output parameter MUST be 0. Otherwise, the stored procedure MUST return the lock age in seconds of the binary data's virtual lock in the @lockAgeInSeconds output parameter which MUST NOT be NULL.

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

If no binary data exists for the identifier supplied in the @id variable, a NULL value MUST be returned. Otherwise, the stored procedure MUST return the lock cookie associated with the virtual lock in the @lockCookie output parameter which MUST NOT be NULL.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.5 proc_GetItemWithoutLock

The proc_GetItemWithoutLock stored procedure retrieves binary data associated with an identifier. The stored procedure MUST update the expiration time of the binary data as specified in section 3.1.1. This procedure MUST NOT place a virtual lock on the binary data.

```
PROCEDURE proc_GetItemWithoutLock (
@id varchar(512),
@item varbinary(max) OUTPUT,
@locked bit OUTPUT,
@lockAgeInSeconds int OUTPUT,
@lockCookie int OUTPUT,

);
```

@id: The stored procedure attempts to retrieve binary data associated with the supplied identifier.

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

If no binary data exists for the identifier supplied in the @id variable, a NULL value MUST be returned.

If the binary data identified by the @id variable does not have a virtual lock on it, the stored procedure MUST return the binary data in the @item output parameter. Otherwise a NULL value MUST be returned in the @item output parameter.

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

If no binary data exists for the identifier supplied in the @id variable, a NULL value MUST be returned.

If the binary 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.

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

If no binary data exists for the identifier supplied in the @id variable, a NULL value MUST be returned.

If the binary data identified by the @id variable does not have a virtual lock on it, the value returned in the @lockAgeInSeconds output parameter MUST be 0. Otherwise @lockAgeInSeconds MUST NOT be NULL, and the stored procedure MUST return the lock age in seconds of the binary data in the @lockAgeInSeconds output parameter.

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

If no binary data exists for the identifier supplied in the @id variable, a NULL value MUST be returned.

If the binary data identified by the @id variable does not have a virtual lock on it, the value returned in the @lockCookie output parameter 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 which MUST NOT be NULL.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.6 proc_RefreshItemExpiration

The proc_RefreshItemExpiration stored procedure updates the expiration time of an instance of binary data in the binary data store as specified in section 3.1.1.

```
PROCEDURE proc_RefreshItemExpiration (
@id varchar(512),

);
```

@id: The stored procedure updates the expiration time of the binary data associated with the supplied identifier. If no binary data exists for the identifier supplied in the @id variable, the stored procedure MUST NOT make any changes.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.7 proc_ReleaseItemLock

The proc_ReleaseItemLock stored procedure attempts to remove the virtual lock from a specific binary data instance.

```
PROCEDURE proc_ReleaseItemLock (
```

```
  @id varchar(512),
  @lockCookie int,

);
```

@id: The stored procedure removes the virtual lock from the binary data associated with the supplied identifier. If no binary data exists for the identifier supplied in the @id variable, the stored procedure MUST NOT make any changes.

@lockCookie: If the value supplied in this parameter equals the value of the lock cookie currently associated with the instance of binary data, the stored procedure MUST remove the virtual lock from the binary data, and it MUST update the expiration time of the binary data as specified in section 3.1.1. Otherwise, the stored procedure MUST NOT remove the virtual lock and MUST NOT update the expiration time.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.8 proc_UpdateItem

The proc_UpdateItem stored procedure attempts to update an instance of binary data in the binary data store.

```
  PROCEDURE proc_UpdateItem (
  @id varchar(512),
  @item varbinary(max),
  @timeout int,
  @lockCookie int,

);
```

@id: The stored procedure updates the binary data in the binary data store associated with the supplied identifier. If no binary data exists for the identifier supplied in the @id variable, the stored procedure MUST NOT make any changes.

@item: The new binary data that MUST be stored if the value of the lock cookie supplied in the @lockCookie parameter equals the value of the lock cookie currently associated with the instance of binary data. Additionally, the stored procedure MUST remove the virtual lock from the binary data.

@timeout: A value in minutes indicating the lifetime of the supplied binary data. If the value of the lock cookie supplied in the @lockCookie parameter equals the value of the lock cookie currently associated with the instance of binary data, the stored procedure MUST update the time-out value and the subsequent expiration time associated with the instance of binary data as specified in section 3.1.1.

@lockCookie: The stored procedure MUST only update the binary data in the binary data store if the value of the lock cookie supplied in the @lockCookie parameter equals the value of the lock cookie currently associated with the instance of binary data. If this parameter does not match the lock cookie currently associated with the instance of binary data, the stored procedure MUST NOT make any changes.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Client Details

None.

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

None.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

4.1 Creating Binary Data (proc_AddItem)

This example describes the request made by a protocol client to create binary data.

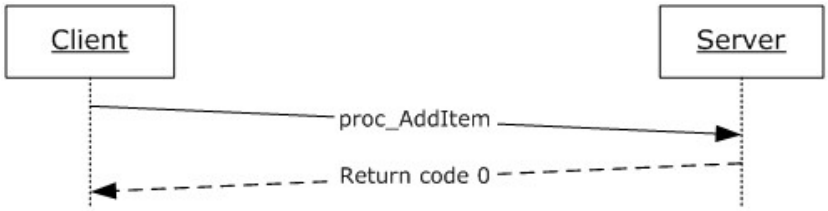

Figure 2: Binary data creation sequence

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

```
exec dbo.proc_AddItem

@id=N'bb513e2c367a494fbf68e63241a19509_zMFtomz0mwgoHSRng157WFwiSCXs6YcdLRhiY5ms+78='
,

@item=0x1400...truncated_for_brevity...0BFF,@timeout=20
```

The protocol server returns a 0 return code for successful execution.

4.2 Retrieving and then Updating Binary Data with Exclusive Access (proc_GetItemWithLock, proc_UpdateItem)

This example describes the requests made by a protocol client to retrieve and subsequently update binary data in the protocol server.

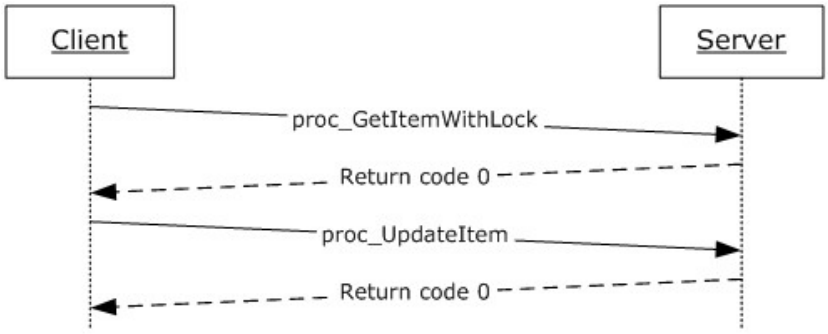

Figure 3: Binary data retrieval/update sequence

1. The protocol client calls proc_GetItemWithLock with an identifier to retrieve binary data from the protocol server with an exclusive lock.

   ```
   exec dbo.proc_GetItemWithLock
   ```

```
@id=N'bb513e2c367a494fbf68e63241a19509_zMFtomz0mwgoHSRng157WFwiSCXs6YcdLRhiY5ms+78=',@
item=@p2 output,
@locked=@p3 output,@lockAgeInSeconds=@p4 output,
@lockCookie=@p5 output
```

2. The protocol server responds with output parameters: the binary data in the @item parameter, the lock cookie in the @lockCookie output parameter, the lock age in the @lockAgeInSeconds output parameter, and the locked state of the binary data in the @locked output parameter.

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

```
exec dbo.proc_UpdateItem

@id=N'bb513e2c367a494fbf68e63241a19509_zMFtomz0mwgoHSRng157WFwiSCXs6YcdLRhiY5ms+78
=',

@item=0x1400...truncated_for_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 using SQL (Structured Query Language) 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:

- Microsoft® SharePoint® Server 2010

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

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

7 Change Tracking

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

8 Index

N

O

P

R

S

T

V

X