[MS-SSPSOS]:
SQL Configuration Object Stored Procedures Protocol Specification

---

Revision Summary

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

Table of Contents

1 Introduction

The SQL Configuration Object Stored Procedures Protocol specifies an interface for protocol clients to store and retrieve configuration settings on a protocol server, and to efficiently maintain a distributed cache of those settings by querying for settings that have changed on the protocol server.

1.1 Glossary

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

GUID
Security Support Provider Interface (SSPI)

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

back-end database server
configuration object
result set
return code
root element
stored procedure
Structured Query Language (SQL)
version stamp
XML fragment

The following terms are specific to this document:

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)

This protocol allows a protocol client to store and retrieve **configuration objects** on a protocol server and to maintain a cache of configuration objects that have changed on the protocol server, even if the change was not made by the protocol client.

1.4 Relationship to Other Protocols

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

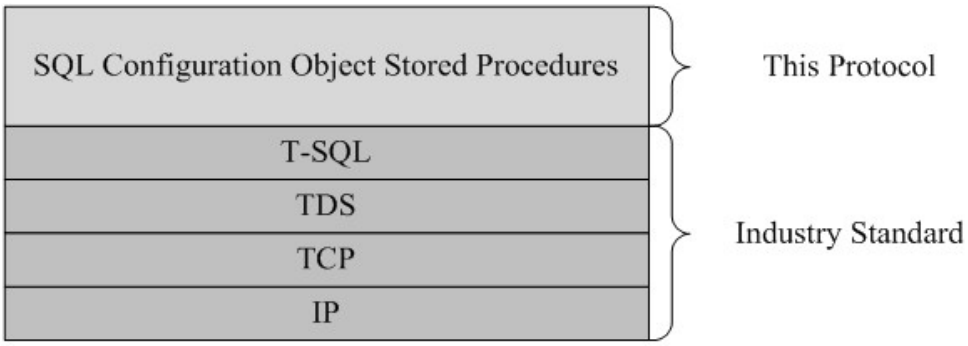

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

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

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

1.6 Applicability Statement

This protocol is designed to maintain configuration objects that are read frequently by many protocol clients, but are updated infrequently and typically by a single protocol client.

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

1.7 Versioning and Capability Negotiation

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

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

[MS-TDS] is the transport protocol used to call the stored procedures, query SQL tables, return **result sets** and **return codes**.

2.2 Common Data Types

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

2.2.1 Simple Data Types and Enumerations

2.2.2 Configuration Object XML

A Unicode string which MUST conform to the following XML Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema id="object" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="object">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="field" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

object: The **root element** of the **XML fragment**.

field: A name/value pair. The semantic meaning of the pair is defined by sub-protocols which implement this protocol.

Each **field** element MUST conform to one of the following XML schemas:

2.2.2.1 nullField

A **field** with no value.

```
<xs:complexType name="nullField">
  <xs:attribute name="name" type="xs:string" use="required" />
  <xs:attribute name="type" type="xs:string" use="required" fixed="null" />
</xs:complexType>
```

name: The name of the **field**.

type: The type of the **field**.

2.2.2.2 booleanField

A **field** that specifies a Boolean value.

```
<xs:complexType name="booleanField">
```

```
  <xs:simpleContent>
    <xs:extension base="xs:boolean" >
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="type" type="xs:string" use="required" fixed="boolean" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

name: The name of the **field**.

type: The type of the **field**.

2.2.2.3 intField

A **field** that specifies an integer value.

```
<xs:complexType name="intField">
  <xs:simpleContent>
    <xs:extension base="xs:int" >
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="type" type="xs:string" use="required" fixed="int" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

name: The name of the **field**.

type: The type of the **field**.

2.2.2.4 floatField

A **field** that specifies a float value.

```
<xs:complexType name="floatField">
  <xs:simpleContent>
    <xs:extension base="xs:float" >
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="type" type="xs:string" use="required" fixed="float" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

name: The name of the **field**.

type: The type of the **field**.

2.2.2.5 guidField

A **field** that specifies a **GUID** value.

```
<xs:simpleType name="guid">
  <xs:restriction base="xs:string">
    <xs:pattern value="\{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\}"/>
  </xs:restriction>
```

```
</xs:simpleType>

<xs:complexType name="guidField">
  <xs:simpleContent>
    <xs:extension base="guid" >
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="type" type="xs:string" use="required" fixed="guid" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

name: The name of the **field**.

type: The type of the **field**.

2.2.2.6 stringField

A **field** that specifies a string value.

```
<xs:complexType name="stringField">
  <xs:simpleContent>
    <xs:extension base="xs:string" >
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="type" type="xs:string" use="required" fixed="string" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

name: The name of the **field**.

type: The type of the **field**.

2.2.2.7 nullListField

A **field** that specifies a null list value.

```
<xs:complexType name="nullListField">
  <xs:attribute name="name" type="xs:string" use="required" />
  <xs:attribute name="type" type="xs:string" use="required" fixed="list" />
  <xs:attribute name="itemType" type="xs:string" use="required" fixed="null" />
</xs:complexType>
```

name: The name of the **field**.

type: The type of the **field**.

itemType: The type of items in the list.

2.2.2.8 booleanListField

A **field** that specifies a list of Boolean values.

```
<xs:complexType name="booleanListField">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
```

```
    <xs:element name="item" type="xs:boolean" />
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
  <xs:attribute name="type" type="xs:string" use="required" fixed="list" />
  <xs:attribute name="itemType" type="xs:string" use="required" fixed="boolean" />
</xs:complexType>
```

item: An item in the list.

name: The name of the **field**.

type: The type of the **field**.

itemType: The type of items in the list.

2.2.2.9 intListField

A **field** that specifies a list of integer values.

```
<xs:complexType name="intListField">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:element name="item" type="xs:int" />
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
  <xs:attribute name="type" type="xs:string" use="required" fixed="list" />
  <xs:attribute name="itemType" type="xs:string" use="required" fixed="int" />
</xs:complexType>
```

item: An item in the list.

name: The name of the **field**.

type: The type of the **field**.

itemType: The type of items in the list.

2.2.2.10 floatListField

A **field** that specifies a list of float values.

```
<xs:complexType name="floatListField">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:element name="item" type="xs:float" />
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
  <xs:attribute name="type" type="xs:string" use="required" fixed="list" />
  <xs:attribute name="itemType" type="xs:string" use="required" fixed="float" />
</xs:complexType>
```

item: An item in the list.

name: The name of the **field**.

type: The type of the **field**.

itemType: The type of items in the list.

2.2.2.11 guidListField

A **field** that specifies a list of GUID values.

```
<xs:simpleType name="guid">
  <xs:restriction base="xs:string">
    <xs:pattern value="\{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\}"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="guidListField">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:element name="item" type="guid" />
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
  <xs:attribute name="type" type="xs:string" use="required" fixed="list" />
  <xs:attribute name="itemType" type="xs:string" use="required" fixed="guid" />
</xs:complexType>
```

item: An item in the list.

name: The name of the **field**.

type: The type of the **field**.

itemType: The type of items in the list.

2.2.2.12 stringListField

A **field** that specifies a list of string values.

```
<xs:complexType name="stringListField">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element name="item" type="xs:string" />
      <xs:element name="null">
        <xs:complexType />
      </xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
  <xs:attribute name="type" type="xs:string" use="required" fixed="list" />
  <xs:attribute name="itemType" type="xs:string" use="required" fixed="string" />
</xs:complexType>
```

item: An item in the list.

name: The name of the **field**.

type: The type of the **field**.

itemType: The type of items in the list.

2.2.3 Configuration Object Status

The configuration object status represents status information about the configuration object returned by the protocol server. MUST be an integer value from the following list.

The semantic meaning of each value is determined by sub-protocols adhering to the protocol defined by this document. A possible meaning of each value is provided to facilitate the explanation of the protocol behavior.

Value	Description
0 (Default)	The configuration object is in the online state.
1	The configuration object is in the disabled state.
2	The configuration object is in the offline state.
3	The configuration object is transitioning to the disabled state.
4	The configuration object is transitioning to the online state.
5	The configuration object is in the upgrading state.

3 Protocol Details

3.1 Protocol Server Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that a protocol server 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.

This protocol requires the protocol server to maintain a list of configuration objects and a **version stamp**. The protocol server adds, changes, or deletes configuration objects from this list in response to messages from a protocol client.

For each configuration object in the list, the protocol server maintains a version stamp and a GUID. The GUID is used by the protocol client to uniquely identify items in the list.

3.1.2 Timers

None.

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 [MS-TDS].

The protocol server initializes the version stamp to the value 0 prior to processing the first message from a protocol client.

3.1.4 Message Processing Events and Sequencing Rules

This section describes the following stored procedures.

Procedure Name	Description
proc_MIP_PutObject	Adds or changes a configuration object.
proc_MIP_GetObject	Retrieves a configuration object.
proc_MIP_DropObject	Deletes a configuration object.
proc_MIP_GetVersion	Retrieves the current version stamp.
proc_MIP_GetObjectUpdates	Retrieves the list of configuration objects that have been changed.

The syntax for each stored procedure and result set and the variables they are composed of, is defined in the [MS-TDS] protocol. In the T-SQL (Transact-Structured Query Language) 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.

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**, as the ordinal position of any column with no defined name is expected by the protocol client.

3.1.4.1 proc_MIP_PutObject

The **proc_MIP_PutObject** stored procedure is called to add or change a configuration object.

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

```
PROCEDURE proc_MIP_PutObject (
    @ObjectId          uniqueidentifier,
    @Status            int,
    @Version           bigint,
    @Xml               ntext,
    @NewVersion        bigint OUTPUT
);
```

@ObjectId: The identifier of the configuration object which will be added or changed. The value MUST NOT be NULL.

@Status: The new status of the configuration object. The value MUST be a valid Configuration Object Status.

@Version: The version stamp of the configuration object which will be added or changed.

@Xml: The new data associated with the configuration object. The value MUST conform to the Configuration Object Xml schema.

@NewVersion: The new version stamp of the configuration object. The value MUST be ignored if the stored procedure fails.

If *@Version* is NULL, this stored procedure:

- MUST fail with a return code value of 3 if the configuration object identified by *@ObjectId* exists.
- MUST add a configuration object with the specified *@ObjectId*, *@Status*, and *@Xml* values.

If *@Version* is not NULL, this stored procedure:

- MUST fail with a return code value of 1 if the configuration object identified by *@ObjectId* does not exist.
- MUST fail with a return code value of 3 if the specified *@Version* does not equal the version stamp of the configuration object which is identified by *@ObjectId*.
- MUST change the configuration object which is identified by *@ObjectId* to the specified *@Status* and *@Xml* values.

If this stored procedure succeeds, it:

- MUST increment the protocol server version stamp.
- MUST set *@NewVersion* and the configuration object version stamp to the incremented protocol server version stamp.

- MUST return a return code value of 0.

If this stored procedure fails, it:

- MUST NOT change the configuration object on the protocol server.
- MUST NOT return a return code value of 0.

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

Value	Description
0	The configuration object was successfully added or changed.
1	The configuration object does not exist.
3	The version stamp of the configuration object does not match the specified *@Version*.
All other values	An error has occurred.

Result Sets: MUST NOT return any result sets.

3.1.4.2 proc_MIP_GetObject

The **proc_MIP_GetObject** stored procedure is called to retrieve a configuration object.

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

```
PROCEDURE proc_MIP_GetObject(
    @ObjectId         uniqueidentifier
);
```

@ObjectId: The identifier of the configuration object to be retrieved. The value MUST NOT be NULL.

This stored procedure MUST return a return code value of 0 if the configuration object with the specified *@ObjectId* does not exist.

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

Value	Description
0	The configuration object was successfully retrieved, or did not exist.
All other values	An error occurred.

Result Sets: MUST return the following result set:

3.1.4.2.1 Get Object Result Set

The Get Object Result Set contains information about a configuration object. The result set MUST contain one row if the configuration object exists; otherwise, the result set MUST be empty.

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

```
Status                int NOT NULL,
Version               bigint NOT NULL,
```

```
Xml                     ntext NULL;
```

Status: The status of the configuration object. The value MUST be a valid Configuration Object Status.

Version: The version stamp of the configuration object.

Xml: The data associated with the configuration object. The value MUST conform to the Configuration Object XML schema.

3.1.4.3 proc_MIP_DropObject

The **proc_MIP_DropObject** stored procedure is called to delete a configuration object.

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

```
PROCEDURE proc_MIP_DropObject(
    @ObjectId           uniqueidentifier
);
```

@ObjectId: The identifier of the configuration object to be deleted. The value MUST NOT be NULL.

This stored procedure:

- MUST delete the configuration object identified by *@ObjectId* if it exists.
- MUST succeed with a return code value of 0 if the configuration object identified by *@ObjectId* does not exist.
- MUST increase the value of the protocol server version stamp by an increment.

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

Value	Description
0	The configuration object was successfully deleted, or did not exist.
All other values	An error has occurred.

Result Sets: MUST NOT return any result sets.

3.1.4.4 proc_MIP_GetObjectVersion

The **proc_MIP_GetObjectVersion** stored procedure is called to retrieve the current protocol server version stamp.

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

```
PROCEDURE proc_MIP_GetObjectVersion(
    @CurrentVersion     bigint OUTPUT
);
```

@CurrentVersion: The current protocol server version stamp.

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

Value	Description
0	The protocol server version stamp was successfully retrieved.
All other values	An error has occurred.

Result Sets: MUST NOT return any result sets.

3.1.4.5 proc_MIP_GetObjectUpdates

The **proc_MIP_GetObjectUpdates** stored procedure is called to retrieve configuration objects that have been changed or deleted after a specified version stamp.

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

```
PROCEDURE proc_MIP_GetObjectUpdates(
    @Version           bigint,
    @CurrentVersion    bigint OUTPUT
);
```

@Version: The protocol client version stamp which identifies the changes to be returned.

@CurrentVersion: The current protocol server version stamp. The value MUST be ignored if the stored procedure fails.

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

Value	Description
0	The stored procedure execution was successful.
All other values	An error has occurred.

Result Sets:

If *@Version* matches the protocol server version stamp, this stored procedure

MUST NOT return a result set.

Otherwise, this stored procedure

MUST return two result sets in the following order:

3.1.4.5.1 Changed Objects Result Set

The Changed Objects Result Set contains information about configuration objects that have changed after the version stamp specified by *@Version*. Each row contains the current values of a changed configuration object. The result set contains 0 or more rows.

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

```
ObjectId        uniqueidentifier    NOT NULL,
Status          int                 NOT NULL,
```

```
Version         bigint              NOT NULL,
Xml             ntext               NULL;
```

ObjectId: The identifier of the configuration object. The value MUST NOT be NULL.

Status: The status of the configuration object. The value MUST be a valid Configuration Object Status.

Version: The version stamp of the configuration object.

Xml: The data associated with the configuration object. The value MUST conform to the Configuration Object Xml schema.

3.1.4.5.2 Deleted Objects Result Set

The Deleted Objects Result Set contains information about configuration objects that have been deleted after the version stamp specified by *@Version*. Each row contains the identifier of a deleted configuration object. The result set contains 0 or more rows.

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

```
ObjectId        uniqueidentifier    NOT NULL;
```

ObjectId: The identifier of the configuration object. The value MUST NOT be NULL.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

3.2 Protocol Client Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that a client 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 client sends messages to a protocol server to add, retrieve, change, and delete configuration objects on the protocol server. The protocol client maintains a version stamp and a cache of configuration objects retrieved from the protocol server. For each configuration object in the cache, the protocol client maintains a version stamp.

3.2.2 Timers

The protocol client maintains a timer which requests configuration objects that have been changed or deleted on the protocol server and need to be updated in the cache.

3.2.3 Initialization

The protocol client is initialized by requesting the current protocol server version stamp by calling the proc_MIP_GetObjectVersion stored procedure.

3.2.4 Message Processing Events and Sequencing Rules

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

This section describes the additional protocol client behavior when invoking some of the stored procedure listed in Section 3.1.4 on the protocol server:

3.2.4.1 proc_MIP_GetObjectUpdates

After calling the proc_MIP_GetObjectUpdates stored procedure:

For each row returned in the Changed Objects Result Set, the protocol client:

- MUST ignore the row if the configuration object specified by **ObjectId** does not exist in the cache.
- MUST update the cached configuration object with the values from the row if **Version** is greater than or equal to the version stamp of the cached configuration object, otherwise, MUST ignore the row.

For each row returned in the Deleted Objects Result Set, the protocol client:

- MUST delete the configuration object from the cache if the configuration object specified by ObjectId exists in the cache, otherwise, MUST ignore the row.

The protocol client MUST update the protocol client version stamp to *@CurrentVersion*.

3.2.5 Timer Events

The timer specified in section 3.2.2 triggers the following sequence of events:

- The protocol client calls the proc_MIP_GetObjectUpdates stored procedure on the protocol server with the *@Version* parameter set to the value of the protocol client version stamp.
- The protocol client processes the result set as specified in section 3.1.4.5.

3.2.6 Other Local Events

None.

4 Protocol Examples

4.1 Add a configuration object

In this example, a protocol client adds a configuration object to the protocol server. The configuration object contains a single setting that specifies the maximum duration of an operation in seconds.

The protocol client specifies the GUID 'AC41919C-98FD-4E81-ADA5-4EF2F2425EFA' for the configuration object identifier.

4.1.1 Adding the new configuration object

The protocol client calls the proc_MIP_PutObject stored procedure, setting the *@Status* parameter to 0 and the *@Version* parameter to null. The *@Xml* parameter is set to the following string value:

```
<object>
  <field name="maxSeconds" type="int">10</field>
</object>
```

maxSeconds is the name that the protocol client has chosen for the setting. The setting is of type **int** and the protocol client has chosen to initialize the value to 10.

The protocol server stores the configuration object, increments the protocol server version stamp, and returns the new configuration object version stamp in the *@NewVersion* output parameter**.**

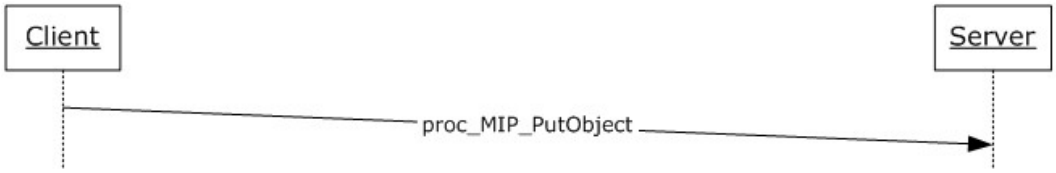

Figure 2: Adding the new configuration object

4.2 Change a configuration object

In this example, a protocol client changes a configuration object on a protocol server.

4.2.1 Getting the current configuration object values

The protocol client calls the proc_MIP_GetObject stored procedure, setting the *@ObjectId* parameter to 'AC41919C-98FD-4E81-ADA5-4EF2F2425EFA'.

The protocol server returns the Get Object Result Set for the configuration object specified by the *@ObjectId* parameter.

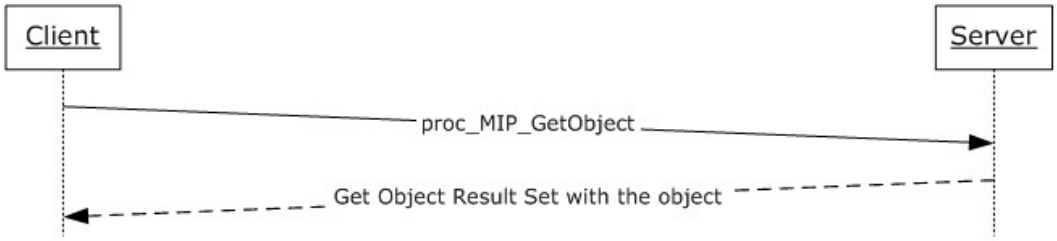

Figure 3: Getting the current configuration object values

4.2.2 Updating the configuration object values

The protocol client calls the proc_MIP_PutObject stored procedure, setting the *@Status* and *@Version* parameters to the **Status** and **Version** values retrieved in the Get Object Result Set, and the *@Xml* parameter to the following string value:

```
<object>
  <field name="maxSeconds" type="int">30</field>
</object>
```

maxSeconds is the name that the protocol client has chosen for the setting. The setting is of type **int** and the protocol client has chosen to set the value to 30.

The protocol server stores the configuration object, increments the protocol server version stamp, and returns the new configuration object version stamp in the *@NewVersion* output parameter.

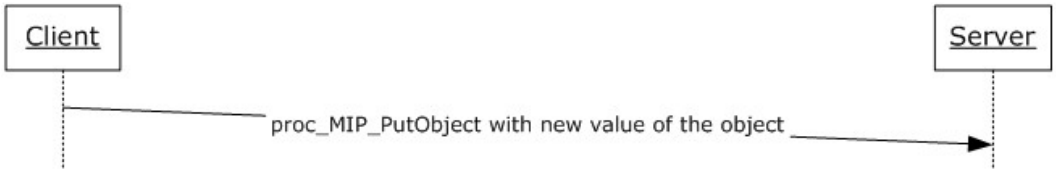

Figure 4: Updating the configuration object values

4.3 Maintain a cache on the protocol client

In this example, the protocol client maintains a cache of configuration objects. The protocol client adds configuration objects to a cache as they are requested and contacts the protocol server at periodic intervals to obtain configuration objects that changed after the previous interval and applies those changes to the cache.

4.3.1 Initializing the protocol client version stamp

The protocol client calls the proc_MIP_GetObjectVersion stored procedure and initializes the protocol client version stamp to the *@CurrentVersion* parameter.

Figure 5: Initializing the protocol client version stamp

4.3.2 Caching configuration objects

The protocol client requests the Get Object Result Set for a configuration object as needed from a cache. If the requested result set does not exist in the cache, the protocol client calls the proc_MIP_GetObject stored procedure, setting the *@ObjectId* parameter to the identifier of the requested configuration object, and then adds the returned Get Object Result Set to the cache.

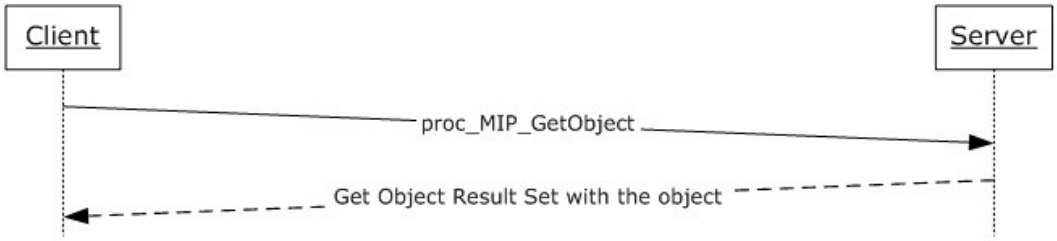

Figure 6: Caching configuration objects

4.3.3 Updating the cache

The protocol client periodically calls the proc_MIP_GetObjectUpdates stored procedure, setting the *@Version* parameter to the protocol client version stamp.

- If there are any changes specified in the Changed Objects Result Set, the protocol client applies the changes to the cache.
- If there are any deletions specified in the Deleted Objects Result Set, the protocol client removes the configuration objects with the specified **ObjectId** values from the cache.

Once all changes and deletions are applied to the cache, the protocol client updates the protocol client version stamp to the *@CurrentVersion* output parameter.

Figure 7: Updating the cache

5 Security

5.1 Security Considerations for Implementers

There are no additional security considerations for implementers. Security assumptions for this protocol are documented in section 1.5.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

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

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

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

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

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