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

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft 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.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Copyright © 2012 Microsoft Corporation.

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.
01/20/2012	2.5	Minor	Clarified the meaning of the technical content.
04/11/2012	2.5	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	2.5	No change	No changes to the meaning, language, or formatting of the technical content.

Copyright © 2012 Microsoft Corporation.

Table of Contents

1 Introduction 5
1.1 Glossary 5
1.2 References 5
1.2.1 Normative References 5
1.2.2 Informative References 6
1.3 Protocol Overview (Synopsis) 6
1.4 Relationship to Other Protocols 6
1.5 Prerequisites/Preconditions 6
1.6 Applicability Statement 6
1.7 Versioning and Capability Negotiation 7
1.8 Vendor-Extensible Fields 7
1.9 Standards Assignments 7

2 Messages 8
2.1 Transport 8
2.2 Common Data Types 8
2.2.1 Simple Data Types and Enumerations 8
2.2.2 Configuration Object XML 8
2.2.2.1 nullField 8
2.2.2.2 booleanField 8
2.2.2.3 intField 9
2.2.2.4 floatField 9
2.2.2.5 guidField 9
2.2.2.6 stringField 10
2.2.2.7 nullListField 10
2.2.2.8 booleanListField 10
2.2.2.9 intListField 11
2.2.2.10 floatListField 11
2.2.2.11 guidListField 12
2.2.2.12 stringListField 12
2.2.3 Configuration Object Status 13

3 Protocol Details 14
3.1 Protocol Server Details 14
3.1.1 Abstract Data Model 14
3.1.2 Timers 14
3.1.3 Initialization 14
3.1.4 Message Processing Events and Sequencing Rules 14
3.1.4.1 proc_MIP_PutObject 15
3.1.4.2 proc_MIP_GetObject 16
3.1.4.2.1 Get Object Result Set 16
3.1.4.3 proc_MIP_DropObject 17
3.1.4.4 proc_MIP_GetObjectVersion 17
3.1.4.5 proc_MIP_GetObjectUpdates 18
3.1.4.5.1 Changed Objects Result Set 18
3.1.4.5.2 Deleted Objects Result Set 19
3.1.5 Timer Events 19
3.1.6 Other Local Events 19
3.2 Protocol Client Details 19
3.2.1 Abstract Data Model 19

Copyright © 2012 Microsoft Corporation.

3.2.2 Timers 19
3.2.3 Initialization 20
3.2.4 Message Processing Events and Sequencing Rules 20
3.2.4.1 proc_MIP_GetObjectUpdates 20
3.2.5 Timer Events 20
3.2.6 Other Local Events 20

4 Protocol Examples 21
4.1 Add a configuration object 21
4.1.1 Adding the new configuration object 21
4.2 Change a configuration object 21
4.2.1 Getting the current configuration object values 21
4.2.2 Updating the configuration object values 22
4.3 Maintain a cache on the protocol client 22
4.3.1 Initializing the protocol client version stamp 22
4.3.2 Caching configuration objects 23
4.3.3 Updating the cache 23

5 Security 24
5.1 Security Considerations for Implementers 24
5.2 Index of Security Parameters 24

6 Appendix A: Product Behavior 25

7 Change Tracking 26

8 Index 27

Copyright © 2012 Microsoft Corporation.

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.

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]:

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)
Transact-Structured Query Language (T-SQL)
version stamp
XML fragment
XML schema

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

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical 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.

Copyright © 2012 Microsoft Corporation.

[Iseminger] 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://www.microsoft.com/mspress/books/5001.aspx

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

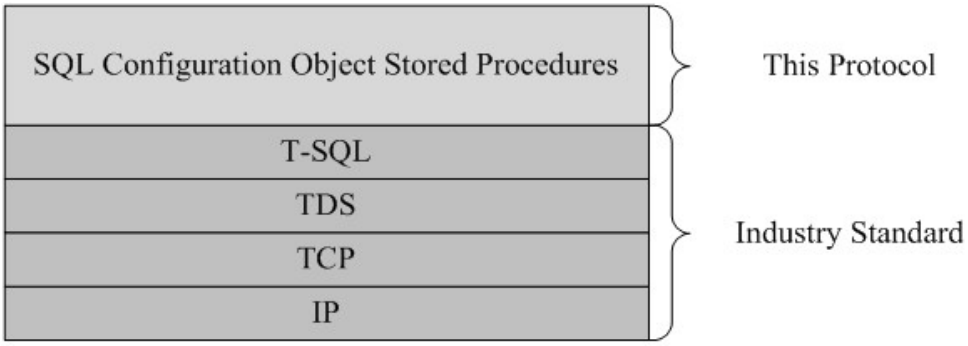

Figure 1: This protocol in relation to other protocols

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.

Copyright © 2012 Microsoft Corporation.

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.

Copyright © 2012 Microsoft Corporation.

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

The **Configuration Object** is 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 XML schemas in the following subsections.

2.2.2.1 nullField

The **nullField** is 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

The **booleanField** is a field that specifies a Boolean value.

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


Copyright © 2012 Microsoft Corporation.

```
  <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

The **intField** is 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

The **floatField** is 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

The **guidField** is 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>
```


Copyright © 2012 Microsoft Corporation.

```
</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

The **stringField** is 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

The **nullListField** is 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

The **booleanListField** is a field that specifies a list of Boolean values.

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


Copyright © 2012 Microsoft Corporation.

```
    <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

The **intListField** is 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

The **floatListField** is 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**.

Copyright © 2012 Microsoft Corporation.

itemType: The type of items in the list.

2.2.2.11 guidListField

The **guidListField** is 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

The **stringListField** is 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.

Copyright © 2012 Microsoft Corporation.

2.2.3 Configuration Object Status

The configuration object status represents status information about the configuration object returned by the protocol server. It 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.

Copyright © 2012 Microsoft Corporation.

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 **Transact-Structured Query Language (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.

Copyright © 2012 Microsoft Corporation.

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, as specified in section 2.2.3.

@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, as specified in section 2.2.2.

@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.

Copyright © 2012 Microsoft Corporation.

- 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,
```


Copyright © 2012 Microsoft Corporation.

```
Xml                     ntext NULL;
```

Status: The status of the configuration object. The value MUST be a valid configuration object status, as specified in section 2.2.3.

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, as specified in section 2.2.2.

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.

Copyright © 2012 Microsoft Corporation.

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 zero or more rows.

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

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


Copyright © 2012 Microsoft Corporation.

```
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, as specified in section 2.2.3.

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, as specified in section 2.2.2.

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 zero 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.

Copyright © 2012 Microsoft Corporation.

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.

Copyright © 2012 Microsoft Corporation.

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**.**

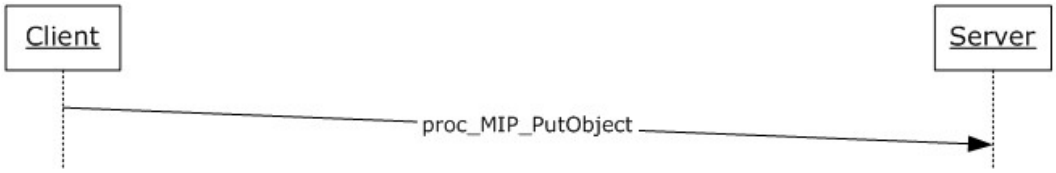

Figure 2: Adding the new configuration object

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.

Copyright © 2012 Microsoft Corporation.

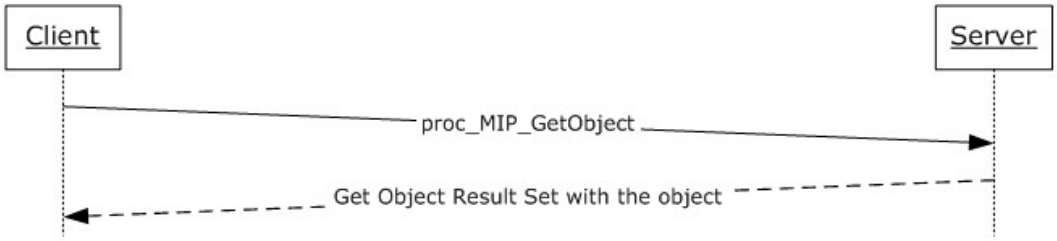

Figure 3: Getting the current configuration object values

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.

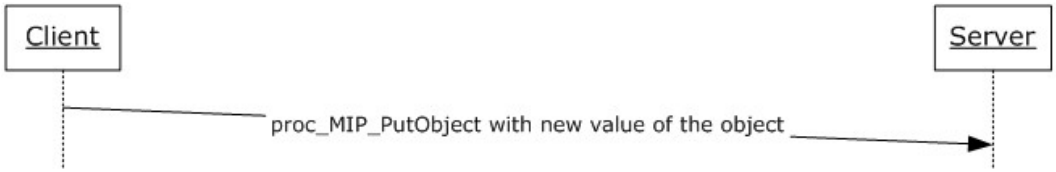

Figure 4: Updating the configuration object values

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.

Copyright © 2012 Microsoft Corporation.

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.

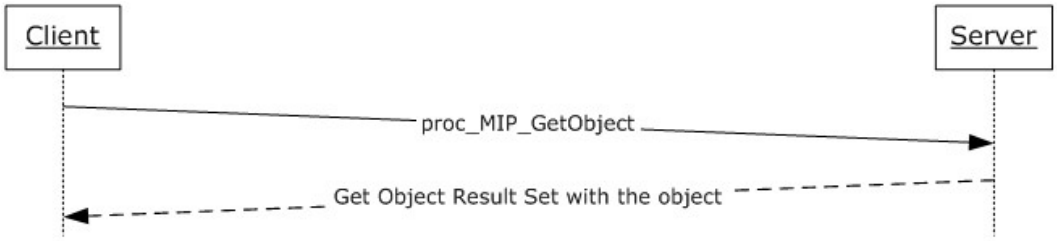

Figure 6: Caching configuration objects

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

Copyright © 2012 Microsoft Corporation.

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.

Copyright © 2012 Microsoft Corporation.

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.

Copyright © 2012 Microsoft Corporation.

7 Change Tracking

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

Copyright © 2012 Microsoft Corporation.

8 Index

A

Abstract data model
  client 19
  server 14
Add a configuration object example 21
Adding the new configuration object example 21
Applicability 6

B

booleanField simple type 8
booleanListField simple type 10

C

Caching configuration objects example 23
Capability negotiation 7
Change a configuration object example 21
Change tracking 26
Changed Objects result set 18
Client
  abstract data model 19
  initialization 20
  local events 20
  message processing 20
  proc_MIP_GetObjectUpdates method 20
  sequencing rules 20
  timer events 20
  timers 19
Common data types
  overview 8
Configuration Object Status simple type 13
Configuration Object XML simple type 8

D

Data model - abstract
  client 19
  server 14
Data types
  booleanField simple type 8
  booleanListField simple type 10
  common 8
  Configuration Object Status simple type 13
  Configuration Object XML simple type 8
  floatField simple type 9
  floatListField 11
  guidField 9
  guidListField 12
  intField simple type 9
  intListField simple type 11
  nullField simple type 8
  nullListField simple type 10
  stringField simple type 10
  stringListField simple type 12
Data types - simple
  booleanListField 10
  Configuration Object Status 13
  Configuration Object XML 8
  floatField 9
  floatListField 11
  guidField 9
  guidListField 12
  intField 9
  intListField 11
  nullListField 10
  stringField 10
  stringListField 12
Deleted Objects result set 19

E

Events
  local - client 20
  local - server 19
  timer - client 20
  timer - server 19
Examples
  Add a configuration object 21
  Adding the new configuration object 21
  Caching configuration objects 23
  Change a configuration object 21
  Getting the current configuration object values 21
  Initializing the protocol client version stamp 22
  Maintain a cache on the protocol client 22
  Updating the cache 23
  Updating the configuration object values 22

F

Fields - vendor-extensible 7
floatField simple type 9
floatListField simple type 11

G

Get Object result set 16
Getting the current configuration object values
  example 21
Glossary 5
guidField simple type 9
guidListField simple type 12

I

Implementer - security considerations 24
Index of security parameters 24
Informative references 6
Initialization
  client 20
  server 14
Initializing the protocol client version stamp
  example 22
intField simple type 9
intListField simple type 11
Introduction 5

Copyright © 2012 Microsoft Corporation.

L

Local events
client 20
server 19

M

Maintain a cache on the protocol client example 22
Message processing
client 20
server 14
Messages
common data types 8
transport 8
Methods
proc_MIP_DropObject 17
proc_MIP_GetObject 16
proc_MIP_GetObjectUpdates (section 3.1.4.5 18, section 3.2.4.1 20)
proc_MIP_GetObjectVersion 17
proc_MIP_PutObject 15

N

Normative references 5
nullField simple type 8
nullListField simple type 10

O

Overview (synopsis) 6

P

Parameters - security index 24
Preconditions 6
Prerequisites 6
proc_MIP_DropObject method 17
Get Object result set 16
proc_MIP_GetObject method 16
proc_MIP_GetObjectUpdates method (section 3.1.4.5 18, section 3.2.4.1 20)
Changed Objects result set 18
Deleted Objects result set 19
proc_MIP_GetObjectVersion method 17
proc_MIP_PutObject method 15
Product behavior 25

R

References 5
informative 6
normative 5
Relationship to other protocols 6
Result sets
Changed Objects 18
Deleted Objects 19
Get Object 16

S

Security
implementer considerations 24
parameter index 24
Sequencing rules
client 20
server 14
Server
abstract data model 14
initialization 14
local events 19
message processing 14
proc_MIP_DropObject method 17
proc_MIP_GetObject method 16
proc_MIP_GetObjectUpdates method 18
proc_MIP_GetObjectVersion method 17
proc_MIP_PutObject method 15
sequencing rules 14
timer events 19
timers 14
Simple data types
booleanField simple type 8
booleanListField 10
Configuration Object Status 13
floatField 9
floatListField 11
guidField 9
guidListField 12
intListField 11
nullField 8
nullListField 10
stringField 10
stringListField 12
Standards assignments 7
stringField simple type 10
stringListField simple type 12

T

Timer events
client 20
server 19
Timers
client 19
server 14
Tracking changes 26
Transport 8

U

Updating the cache example 23
Updating the configuration object values example 22

V

Vendor-extensible fields 7
Versioning 7

Copyright © 2012 Microsoft Corporation.