[MS-GRVRDB]:
Groove RDB Commands 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	Editorial	Changed language and formatting in 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	2.05	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

This document specifies the Groove RDB Commands Protocol.

The Groove RDB Commands Protocol is an application-layer distributed protocol for specifying database operations. The protocol consists of encoded XML messages.

The Groove RDB Commands Protocol is used between clients and servers to synchronize the data in a shared space.

1.1 Glossary

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

Unicode

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

account
endpoint
engine
record definition
shared space
table
tool

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.

[IEEE754] Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE 754-1985, October 1985, http://ieeexplore.ieee.org/servlet/opac?punumber=2355

[MS-GRVDYNM] Microsoft Corporation, "Groove Dynamics Protocol Specification", June 2008.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006, http://www.ietf.org/rfc/rfc4648.txt

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary", June 2008.

[XML10] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Third Edition)", February 2004, http://www.w3.org/TR/REC-xml

1.3 Protocol Overview (Synopsis)

The Groove RDB Commands Protocol is used to distribute database operations among **endpoints (3)** in a **shared space**. A shared space consists of a set of zero or more **tools**. Each tool has zero or more **engines**, and each engine defines a set of operations, or commands. The record database (RDB) is one such engine. The messages defined by the protocol correspond to the commands executed in the RDB engine on each endpoint.

A typical example would be a shared space with a threaded discussion tool that enables multiple endpoints to contribute discussion topics and post replies. This tool could be built using RDB. RDB has a command set for manipulating records, which includes commands for adding and deleting records, and setting fields on existing records. Data consistency across all endpoints is achieved by using the Groove Dynamics Protocol [MS-GRVDYNM] to sequence the execution of the commands.

A simple RDB scenario starts with the user at an endpoint creating a new discussion topic. The RDB engine creates a command to add a new record. The command includes a new database record with the title and contents of the discussion topic, which are fields in the record. RDB encodes the command, including the new record, into an Add Record message as an XML [XML10] element, which is appended to a Groove Dynamics Protocol command element. RDB then instructs the Groove Dynamics Protocol to execute the command, using it as the transport to distribute the command to all other endpoints. Updates to existing records and deletions of records are handled in a similar fashion.

1.4 Relationship to Other Protocols

The Groove RDB Commands Protocol is dependent on the Groove Dynamics Protocol [MS-GRVDYNM] for transport of the command messages.

1.5 Prerequisites/Preconditions

The Groove RDB Commands Protocol operates within a shared space. It assumes that the shared space has already been created and that all endpoints in the shared space are running compatible implementations of the Groove RDB Commands Protocol.

1.6 Applicability Statement

This protocol can be used anytime that peer-to-peer synchronization of database operations is necessary. It does not define relational operations, so it is best suited for scenarios which require only relatively simple, straightforward database models.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

Groove RDB Commands Protocol messages MUST use the Groove Dynamics Protocol for transport.

2.2 Message Syntax

Messages outside the Groove RDB Commands Protocol MUST be ignored. The Groove RDB Commands Protocol uses XML to encode its messages. The following specifies how data types for RDB messages are encoded as XML attributes:

Type	Encoding
String	A **Unicode** string
Int	An Int attribute MUST be a decimal string representation of an integer in the range -2147483648 to 2147483647.
Double	A Double attribute MUST be a decimal string representation of a floating point number that is representable without loss of information as a double-format floating point number as specified in [IEEE754].
ID	An ID attribute represents a unique identifier for a Record, **record definition**, or Table Definition. An ID is encoded identically to a Double, with the additional constraint that it MUST NOT have a value of -1, or be of any form of infinite number, NaN, -0, or denormalized number as specified in [IEEE754].
Timestamp	A Timestamp represents the number of milliseconds elapsed between 12:00 Midnight January 1, 1970 GMT and the moment in time represented by the timestamp. A Timestamp attribute is encoded identically to a Double attribute.

Each message is XML that MUST consist of an element with the name "urn:groove.net:Cmd". This is the **command element** created by the Groove Dynamics Protocol [MS-GRVDYNM] section 2.2.1.4.4. This element has a series of attributes maintained by the Groove Dynamics Protocol. In addition, all RDB command elements MUST have the following attribute:

DBName (**String**): The name of the database being modified by the execution of the command.

Each RDB message SHOULD contain the following attribute on the command element:

TableDefID (Double): The identifier of a **table** in the database. This is the table being modified by the execution of the command. The value of the identifier follows the restrictions for the **ID** type specified in section 2.2, with the exception that -1 is a valid value <1>. If the value of this attribute is -1, the record MUST be applied to all tables in the repository.

Wherever messages encode fields as XML, the XML representation for each of the supported field data types is as follows. The fields are represented by XML attributes for all but the XML element type, which uses a content element. Fields within a record, and the fields described in a Set Field message, conform to one of the following field types:

Field Type	Encoding
String	Encoded as **String**.
Boolean	Encoded as **String**, with "0" indicating False and "1" indicating True.

Field Type	Encoding
Four Byte Signed Integer	Encoded as **Int**.
Double	Encoded as **Double**.
Binary	Encoded as **String**, with binary content Base64 encoded, as defined in [RFC4648].
Date/Time	Encoded as **Timestamp**.
XML element	Encoded as XML as follows: A content element MUST be appended to either a serialized record for Add Record and Add Records messages, or to the command element for Set Field messages. For Add Record and Add Records messages the name of the content element MUST match the name of the field.

2.2.1 Add Record

The Add Record message element MUST have the following attributes:

CMD (Int): The command to execute. For Add Record messages, the value MUST be 0.

EngineURL: An engine identifier as specified in [MS-GRVDYNM] section 2.2.1.4.4.

PurNot: A purge notification indicator as specified in [MS-GRVDYNM] section 2.2.1.4.4.

The Add Record message element MUST include a serialized representation of one record as a content element, as specified in section 2.2.1.1. There MUST NOT be any other content elements within the command element.

2.2.1.1 Serialized Record XML

The serialized record MUST be an XML element named "urn:groove.net:Record3". This element MUST include the following two attributes:

_RecordID (ID): The numeric identifier for the record.

RecDefID (ID): The numeric identifier of the record definition which is the schema of the record.

The serialized record SHOULD have additional attributes that represent client-defined fields in the record, as described in section 2.2. The names of any such attributes MUST be identical to the names given for the fields in the corresponding record definition.

2.2.2 Add Records

The Add Records message element MUST have the following attributes:

CMD (Int): The command to execute. For Add Records messages, the value MUST be 1.

EngineURL: An engine identifier as specified in [MS-GRVDYNM] section 2.2.1.4.4.

PurNot: A purge notification indicator as specified in [MS-GRVDYNM] section 2.2.1.4.4.

The Add Records message element MUST include serialized representations for each record being added. Each such record MUST be a content element of the command element, serialized as specified in section 2.2.1.1. The Add Records message element MUST NOT have any other content.

2.2.3 Delete Records

The Delete Records message element MUST have the following attributes:

CMD (Int): The command to execute. For Delete Records messages the value MUST be 3.

EngineURL: An engine identifier as specified in [MS-GRVDYNM] section 2.2.1.4.4.

PurNot: A purge notification indicator as specified in [MS-GRVDYNM] section 2.2.1.4.4.

NumIDs (Int): The number of records being deleted in the command.

The Delete Records message element MUST contain attributes equal in number to **NumIDs**, each of which identifies a record identifier for a record being deleted, as follows:

_N (ID): The 'N' MUST be replaced by a numeric value. The first number used MUST be '0', incrementing by one for each additional record being deleted. For example, if there are two records to be deleted, NumIDs is 2, and there are two of these attributes, named "_0" and "_1". The values of these attributes are the numeric record identifiers of the records being deleted.

2.2.4 Set Field

The Set Field message element MUST have the following attributes:

CMD (Int): The command to execute. For Set Field commands the value MUST be 6.

EngineURL: An engine identifier as specified in [MS-GRVDYNM] section 2.2.1.4.4.

PurNot: A purge notification indicator as specified in [MS-GRVDYNM] section 2.2.1.4.4.

_RecordID (ID): The numeric identifier of the record to modify.

Name (String): The name of the field to modify.

Type (Int): The data type of the field being modified. The value of this attribute MUST be set to one of the following values based on type:

Type	Value
String	1
Boolean	2
Four Byte Signed Integer	5
Double	7
Binary	8
Date/Time	9
XML element	10

_Modified (Timestamp): A timestamp indicating the time that the Set Field message was created.

The Set Field message element SHOULD<2> have the following attribute for field types other than XML Element:

Value (any of the preceding data types): The value of the field to apply to the record.

The field value is encoded in the message as described in section 2.2.

3 Protocol Details

3.1 Common Details

All endpoints in the Groove RDB Commands Protocol behave identically. There are no separate roles for clients and servers.

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 message protocol described in section 2.2 assumes an abstract data model in which a repository is a container of records and record definitions. A record is a container of fields, which hold the individual data values. Each record refers to a record definition, which describes the schema of a record type known to the system. Each record definition includes a list of permissible fields, the data types of those fields, and optionally a default value for each field. If no default value is specified for a field in a record definition, the default value for that field is inherited from the system default value for the field type. The system default values are as follows:

Field Type	System Default Value
String	Empty string
Boolean	False
Four Byte Signed Integer	0
Double	-1.0
Binary	Empty binary stream
Date/Time	-1.0
XML element	<urn:groove.net:Empty>

Record ID: Each record within a repository is identified by a numeric value, its **Record ID**, which is specified in the _RecordID field. The identifier MUST be unique among all records in the repository. All messages in the protocol use this identifier when referring to an instance of a record.

Record Definition ID: Each record definition within a repository is identified by a numeric value, its record definition identifier**.** The identifier MUST be unique among all record definitions contained in the repository.

Each record in a repository is associated with a record definition that defines the schema of that record. This is the value that is set in the **RecDefID** field of the record, as described in section 2.2.

When a field value is transmitted in a message using this protocol, and the field value is identical to the default value specified in the Record Definition for that field, the field SHOULD<3> be omitted from the message.

3.1.2 Timers

None.

3.1.3 Initialization

The protocol is initialized when the user logs into the **account** that contains the shared space. The implementation MUST be prepared to receive the messages described in section 2.2 at that time.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Record(s) added to repository

When a higher layer adds one or more records to a repository, the corresponding Add Record message (see section 2.2.1) or Add Records message (see section 2.2.2) MUST be sent to all other endpoints in the shared space, with the new record(s) included within the content of the message. The value of the _**RecordID** attribute in the serialized record element MUST be unique within the repository.

3.1.4.2 Record(s) deleted from repository

When a higher layer deletes one or more records from a repository, a corresponding Delete Records message (see section 2.2.3) MUST be sent to all other endpoints in the shared space. There is no need to serialize the record itself as content of the message.

3.1.4.3 Field updated on an existing record

When a higher layer updates a field value on an existing record in a repository, a Set Field message (see section 2.2.4) MUST be sent to all other endpoints in the shared space. The value of the field MUST be included in the command element, along with a RecordID attribute indicating the record to update.

3.1.5 Message Processing Events and Sequencing Rules

Implementations MUST at minimum process incoming messages as directed by the Groove Dynamics Protocol [MS-GRVDYNM], which is responsible for ordering the sequence of commands. Implementations SHOULD also provide services for generating messages.

When a message is received, the implementation MUST update the repository as directed by the message. Each message contains all of the necessary data to update the repository, either within the command element itself or the combination of the command element and an enclosed record element (see section 2.2). In addition, the implementation MUST maintain context about the previous state of records and fields to be able to undo a command if it is directed by the Groove Dynamics Protocol to do so (this can happen if the Groove Dynamics Protocol needs to roll back some commands to re-sequence a set of commands). An implementation SHOULD accomplish this by saving the previous state of the data object (be it a whole record or individual field) prior to processing a command. The specific data that is to be saved is an implementation detail that is dependent on how the implementation stores data objects.

3.1.5.1 Add Record

When an Add Record message is received, the serialized record within the command element is added to the repository, using the record identifier specified by the _RecordID attribute within the

record element as its identifier. If a record with the same _RecordID already exists in the repository, the message MUST be ignored.

3.1.5.2 Add Records

When an Add Records message is received, the serialized records within the command element are added to the repository, using the record identifiers specified by the _RecordID attributes within the record elements as their identifiers. For each record in the message, if a record with the same _RecordID already exists in the repository, the record MUST NOT be added to the repository.

3.1.5.3 Delete Records

When a Delete Records message is received, the set of records specified in the command element is deleted. For each record specified in the message, if a record with the specified _RecordID does not exist within the repository, that _RecordID MUST be ignored.

3.1.5.4 Set Field

When a Set Field message is received, the repository updates its version of the record identified by the _RecordID attribute on the command element. The field in the record to update is identified by the "Name" attribute on the command element, and the new value is in the "Value" attribute of the command element for all field types but XML element (see section 2.2). The value of the "Type" attribute on the command element indicates the type of field at the time that the message was created. If the field type in the corresponding record definition at the time of command execution does not match the field type in the message, the message MUST be ignored. If the field name does not exist in the corresponding record definition at the time of command execution, the message MUST be ignored.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

4 Protocol Examples

The following examples operate on a repository named "TestDatabase1". The repository contains a record definition with identifier 2885262406, with fields "TestString", "TestBool", "TestI4", "TestR8", "TestDateTime", and "TestBinary", with data types **String, Boolean, Four Byte Signed Integer, Double, Date/Time,** and **Binary**, respectively. The command elements in the examples also contain the attributes **EngineURL** and **PurNot**, which are Groove Dynamics Protocol attributes described in [MS-GRVDYNM] section 2.2.1.4.4. The engine URL in each of the examples is "ToolContainer/yrp57967myg94/RDBManager."

4.1 Add Record

This example adds a record to the repository, with record identifier -1.029366148152012E+070, and two client-defined fields: "TestString" the value of which is "aaa", and "TestI4", the value of which is 12345.

```
<urn:groove.net:Cmd CMD="0" DBName="TestDatabase1"
EngineURL="ToolContainer/yrp57967myg94/RDBManager" PurNot="" TableDefID="-1">
    <urn:groove.net:Record3 RecDefID="2885262406" TestString="aaa" TestI4="12345"
_RecordID="-1.029366148152012E+070"/>
</urn:groove.net:Cmd>
```

4.2 Add Records

This example adds two records, with identifiers -3.4889057889391039E-005 and -8.8460027594901169E+045, to the repository using a single Add Records message. Each record has two client defined fields, "TestString" and "TestI4". The values of the fields are "String 1", and 1 for the first record, and "String 2" and 2 for the second.

```
<urn:groove.net:Cmd CMD="1" DBName="TestDatabase1"
EngineURL="ToolContainer/yrp57967myg94/RDBManager" PurNot="" TableDefID="-1">
    <urn:groove.net:Record3 RecDefID="2885262406" TestString="String 1" TestI4="1"
_RecordID="-3.4889057889391039E-005"/>
    <urn:groove.net:Record3 RecDefID="2885262406" TestString="String 2" TestI4="2"
_RecordID="-8.8460027594901169E+045"/>
</urn:groove.net:Cmd>
```

4.3 Delete Records

This example deletes two records, with identifiers 9.6211695884421265E-017 and 4.5271350937905834E-142, from the repository using a single Delete Records message.

```
<urn:groove.net:Cmd CMD="3" DBName="TestDatabase1"
EngineURL="ToolContainer/yrp57967myg94/RDBManager" NumIDs="2" PurNot="" TableDefID="-1"
_0="9.6211695884421265E-017" _1="4.5271350937905834E-142"/>
```

4.4 Set Field

This example sets a **String** field named "TestString" to a value of "abc" on a record with identifier -4.8036800520483017E-096 in the repository.

```
<urn:groove.net:Cmd CMD="6" DBName="TestDatabase1"
EngineURL="ToolContainer/yrp57967myg94/RDBManager" Name="TestString" PurNot="" TableDefID="-
1" Type="1" Value="abc" _Modified="1202396046342" _RecordID="-4.8036800520483017E-096"/>
```

This example sets a **Boolean** field named "TestBool" to a value of True on a record with identifier -2.8414202314964928E-094.

```
<urn:groove.net:Cmd CMD="6" DBName="TestDatabase1"
EngineURL="ToolContainer/yrp57967myg94/RDBManager" Name="TestBool" PurNot="" TableDefID="-1"
Type="2" _Modified="1203108416062" _RecordID="-2.8414202314964928E-094"/>
```

This example sets a **Four Byte Signed Integer** field named "TestI4" to a value of 12345 on a record with identifier -2.8414202314964928E-094.

```
<urn:groove.net:Cmd CMD="6" DBName="TestDatabase1"
EngineURL="ToolContainer/yrp57967myg94/RDBManager" Name="TestI4" PurNot="" TableDefID="-1"
Type="5" Value="12345" _Modified="1203108416062" _RecordID="-2.8414202314964928E-094"/>
```

This example sets a **Double** field named "TestR8" to a value of 1.2344999999999999 on a record with identifier -2.8414202314964928E-094.

```
<urn:groove.net:Cmd CMD="6" DBName="TestDatabase1"
EngineURL="ToolContainer/yrp57967myg94/RDBManager" Name="TestR8" PurNot="" TableDefID="-1"
Type="7" Value="1.2344999999999999" _Modified="1203108416077" _RecordID="-
2.8414202314964928E-094"/>
```

This example sets a **Date/Time** field named "TestDateTime" to a value of Tuesday, March 04, 2008 1:14:49 PM GMT, with the value encoded in the message as 1203108416077 on a record with identifier -8.6465898231952427E+124.

```
<urn:groove.net:Cmd CMD="6" DBName="TestDatabase1"
EngineURL="ToolContainer/yrp57967myg94/RDBManager" Name="TestDateTime" PurNot=""
TableDefID="-1" Type="9" Value="1203108411124" _Modified="1203108416077" _RecordID="-
2.8414202314964928E-094"/>
```

This example sets a **Binary** field named "TestBinary" to a value of a 1024 byte binary stream in a record with identifier -1.4948713963691306E-005. The contents of the stream are the byte position within the stream plus the byte value of the character "A", that is the first byte in the stream is 'A' + 0, the second is 'A' + 1, and so on.

```
<urn:groove.net:Cmd CMD="6" DBName="TestDatabase1"
EngineURL="ToolContainer/yrp57967myg94/RDBManager" Name="TestBinary" PurNot="" TableDefID="-
1" Type="8"
Value="QUJDREVGR0hJSktMTU5PUFFSU1RVVldYWVpbXF1eX2BhYmNkZWZnaGlqa2xtbm9wcXJzdHV2d3h5ent8fX5/gI
GCg4SFhoeIiYqLjI2Oj5CRkpOUlZaXmJmam5ydnp+goaKjpKWmp6ipqqusra6vsLGys7S1tre4ubq7vL2+v8DBwsPExcb
HyMnKy8zNzs/Q0dLT1NXW19jZ2tvc3d7f4OHi4+Tl5ufo6err7O3u7/Dx8vP09fb3+Pn6+/z9/v8AAQIDBAUGBwgJCgsM
DQ4PEBESExQVFhcYGRobHB0eHyAhIiMkJSYnKCkqKywtLi8wMTIzNDU2Nzg5Ojs8PT4/QEFCQ0RFRkdISUpLTE1OT1BRU
lNUVVZXWFlaW1xdXl9gYWJjZGVmZ2hpamtsbW5vcHFyc3R1dnd4eXp7fH1+r+QkZKTlJWWl5iZmpucnZ6foKGio6Slpqe
oqaqrrK2ur7CxsrO0tba3uLm6u7y9vr/AwcLDxMXGx8jJysvMzc7P0NHS09TV1tfY2drb3N3e3+Dh4uPk5ebn6Onq6+zt
7u/w8fLz9PX29/j5+vv8/f7/AAECAwQFBgcICQoLDA0ODxAREhMUFRYXGBkaGxwdHh8gISIjJCUmJygpKissLS4vMDEyM
zQ1Njc4OTo7PD0+P0BBQkNERUZHSElKS0xNTk9QUVJTVFVWV1hZWltcXV5fYGFiY2RlZmdoaWprbG1ub3BxcnN0dXZ3eH
l6e3x9fn+AgYKDhIWGh4iJiouMjY6PkJGSk5SVlpeYmZqbnJ2en6ChoqOkpaanqKmqq6ytrq+wsbKztLW2t7i5uru8vb6
/wMHCw8TFxsfIycrLzM3Oz9DR0tPU1dbX2Nna29zd3t/g4eLj5OXm5+jp6uvs7e7v8PHy8/T19vf4+fr7/P3+/wABAgME
```

BQYHCAkKCwwNDg8QERITFBUWFxgZGhscHR4fICEiIyQlJicoKSorLC0uLzAxMjM0NTY3ODk6Ozw9Pj9AQUJDREVGR0hJS
ktMTU5PUFFSU1RVVldYWVpbXF1eX2BhYmNkZWZnaGlqa2xtbm9wcXJzdHV2d3h5ent8fX5/gIGCg4SFhoeIiYqLjI2Oj5
CRkpOUlZaXmJmam5ydnp+goaKjpKWmp6ipqqusra6vsLGys7S1tre4ubq7vL2+v8DBwsPExcbHyMnKy8zNzs/Q0dLT1NX
W19jZ2tvc3d7f4OHi4+Tl5ufo6err7O3u7/Dx8vP09fb3+Pn6+/z9/v8AAQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRob
HB0eHyAhIiMkJSYnKCkqKywtLi8wMTIzNDU2Nzg5Ojs8PT4/QA==" _Modified="1203339680345" _RecordID="-
1.4948713963691306E-005"/>

5 Security

5.1 Security Considerations for Implementers

The Groove RDB Commands Protocol relies on the Groove Dynamics Protocol for the security of messages. See [MS-GRVDYNM] section 5 for more information.

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 2010 suites
- Microsoft® Office Groove® 2007
- Microsoft® Office Groove® Server 2007
- Microsoft® Groove® Server 2010
- Microsoft® SharePoint® Workspace 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.

<1> Section 2.2: Office Groove 2007 and SharePoint Workspace 2010 always set the TableDefID attribute on the command elements, and always set it to a value of "-1".

<2> Section 2.2.4: For all data types but XML Element, Office Groove 2007 and SharePoint Workspace 2010 include the Value attribute in the Set Field command message, with the attribute value being the new field value, with two exceptions:

1. For all field types except the Boolean field type, where the value matches the system default value for the field, Office Groove 2007 and SharePoint Workspace 2010 do not include the Value attribute in the Set Field command message.
2. Office Groove 2007 and SharePoint Workspace 2010 treat Boolean type fields in the opposite manner. The system default value is "False", but for the Boolean field type, Office Groove 2007 and SharePoint Workspace 2010 do not include the Value attribute in the Set Field message if the value is "True". If the value is "False", which is the system default value, Office Groove 2007 and SharePoint Workspace 2010 do include the Value attribute in the Set Field message.

<3> Section 3.1.1: If the current value of a field is the default value specified in the Record definition for that field, Office Groove 2007 and SharePoint Workspace 2010 do not serialize the field value as part of the message sent for an Add Record or Add Records command message. Office Groove 2007 and SharePoint Workspace 2010 treat Boolean type fields in the opposite manner, as described in <2>.

7 Change Tracking

This section identifies changes that were made to the [MS-GRVRDB] protocol document between the November 2010 and December 2010 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- Changes made for template compliance.
- Removal of a document from the documentation set.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type "Editorially updated."

Some important terms used in revision type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change Type
2.2.1 Add Record	Added required attributes "EngineURL" and "PurNot".	Y	New content added.
2.2.2 Add Records	Added required attributes "EngineURL" and "PurNot".	Y	New content added.
2.2.3 Delete Records	Added required attributes "EngineURL" and "PurNot".	Y	New content added.
2.2.4 Set Field	Added required attributes "EngineURL" and "PurNot".	Y	New content added.
6 Appendix A: Product Behavior	Updated the list of applicable product versions.	N	Content updated.

8 Index

R

S

T

V