[MS-UPSPROF2]:
User Profile Stored Procedures Version 2 Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
07/13/2009	0.1	Major	Initial Availability
08/28/2009	0.2	Editorial	Revised and edited the technical content
11/06/2009	0.3	Editorial	Revised and edited the technical content
02/19/2010	1.0	Major	Updated and revised the technical content
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Editorial	Revised and edited the technical content
06/29/2010	1.04	Minor	Clarified the meaning of the technical content.
07/23/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.05	Major	Significantly changed the technical content.
12/17/2010	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
04/11/2012	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	1.05	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document provides specific details of the User Profile Stored Procedures protocol. This protocol allows clients to perform create, read, update and delete operations on user information stored in a user profile store on a site.

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]:

access control list (ACL)
Coordinated Universal Time (UTC)
distinguished name (DN)
GUID
language code identifier (LCID)
security identifier (SID)
user principal name (UPN)

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

alternate account
audience
back-end database server
collation
collation sequence
colleague
content type
distribution list
e-mail address
language pack
mailto URI
membership
membership group
multivalue property
organization
owner
partition
personal site
profile subtype
Property
property identifier
quick link
record identifier
request identifier
result set
return code
Security Account Manager (SAM)
server-relative URL
Session Initiation Protocol (SIP)
Session Initiation Protocol (SIP) address

site
stored procedure
term set
Transact-Structured Query Language (T-SQL)
Uniform Resource Locator (URL)
user name
user profile
user profile privacy policy
user profile record identifier
user profile store
XML schema

The following terms are specific to this document:

organization profile: A collection of properties that pertain to a company, division, department, or other business unit within an organization.

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.

[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".

[MS-UPASP] Microsoft Corporation, "User Profile Admin Stored Procedures Protocol Specification".

[MS-UPSIMP] Microsoft Corporation, "User Profile Import 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

[RFC2251] Wahl, M., Howes, T., and Kille, S., "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997, http://www.ietf.org/rfc/rfc2251.txt

[RFC2368] Hoffman, P., Masinter, L., and Zawinski, J., "The mailto URL scheme", RFC 2368, July 1998, http://www.rfc-editor.org/rfc/rfc2368.txt

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

[XMLINFOSET] World Wide Web Consortium, "XML Information Set (Second Edition)", February 2004, http://www.w3.org/TR/2004/REC-xml-infoset-20040204

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XMLSCHEMA1] Thompson, H.S., Ed., Beech, D., Ed., Maloney, M., Ed., and Mendelsohn, N., Ed., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.2.2 Informative References

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

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

[MS-WSSFO2] Microsoft Corporation, "Windows SharePoint Services (WSS): File Operations Database Communications Version 2 Protocol Specification".

1.3 Protocol Overview (Synopsis)

In general, this protocol provides a way for a protocol client to interact with the **user profile store**. The user profile store holds **user profile** objects – which represent all the information about a particular user.

This protocol provides ways for the client to retrieve this information, write new information or update existing information for each user. More specifically, this protocol facilitates the protocol clients to add new user profile properties, update existing user profile properties as well as query the protocol server for a list of all user profile properties with metadata.

In addition to user profile properties, this protocol allows the protocol client to work with the **colleagues** associated with a specified user, and the quick link collection associated with a specified user. It provides ways for the protocol client to retrieve, add, edit or delete a user's colleagues and quick links.

This protocol also provides a way for the protocol client to retrieve each **membership** belonging to the specified user, in any relevant **distribution list** or **membership group**. The protocol also allows the protocol client to identify other users who have similar memberships as the current user. The protocol also allows to retrieve and update **organizations** that the user is a member of and the properties of those organizations.

This protocol also provides a way for protocol clients to control the **user profile privacy policy** [also referred to as privacy policy] associated with a user profile service. The user profile privacy policy describes when user profile service features such as colleagues on a **personal site** are enabled or disabled for all users on that user profile service. In addition to features, the user profile privacy policy also describes whether user profile properties are disabled, enabled and allowed to be null or enabled or mandatory. The user profile privacy policy describes the scope a user exists in to see a specified feature or user profile property. The protocol server facilitates this with any relevant **stored procedure** to update or delete the user profile privacy policy for a specified feature or user

profile property. The protocol server also offers protocol clients, stored procedures to retrieve these settings.

1.4 Relationship to Other Protocols

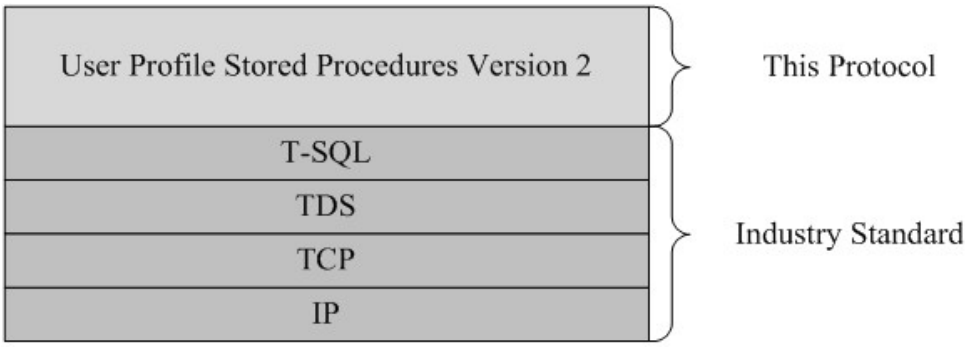

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

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

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

1.6 Applicability Statement

This protocol is designed to work well with up to 5 million user profiles. For each user profile, it works well with up to 100 user profile properties.

1.7 Versioning and Capability Negotiation

Versions of the data structures or stored procedures in the database need to be the same as expected by the front-end Web Server. If the stored procedures do not provide the calling parameters or return values as expected, the results of the call are indeterminate.

The version negotiation process for this protocol is identical to the process described in [MS-WSSFO2] section 1.7.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

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

2.2 Common Data Types

This section contains common definitions used by this protocol.

2.2.1 Simple Data Types and Enumerations

2.2.1.1 Group

A 1-byte integer that specifies the link group type. This value MUST be listed in the following table:

Value	Description
0	User Specified grouping.
1	Best Bet. User specified group to emphasize link in UI
2	General. Default grouping for links that are not Distribution lists or Sites
5	Users who share the same **Manager** property
7	Distribution list default grouping.
8	**Site (2)** default grouping.

2.2.1.2 Privacy

A 32-bit signed integer that specifies the set of users who are allowed to access a resource protected by a Privacy scope. The value MUST be listed in the following table:

Value	Description
1	All users are allowed to access the resource.
2	The only users allowed to access the resource are the owner of the resource and the owner's colleagues.
4	The only users allowed to access the resource are the owner of the resource and the owner's organization.
8	The only two users allowed to access the resource are the owner of the resource and the owner's **Manager** property.
16	The only user allowed to access the resource is the owner of the resource.

2.2.1.3 Short Group

A 1-byte unsigned integer that specifies a membership relationship grouping. The value MUST be listed in the following table:

Value	Description
0	User Specified grouping.
7	Distribution list default grouping.
8	Site (2) default grouping.

2.2.1.4 Short Link

A GUID that specifies the source of a membership group. This value MUST be listed in the following table:

Value	Description
A88B9DCB-5B82-41E4-8A19-17672F307B95	Specifies a membership group that was created based on a distribution list.
8BB1220F-DE8B-4771-AC3A-0551242CF2BD	Specifies a membership group that was created based on the user of a specific site (2).

2.2.1.5 Policy Link

A **GUID** that specifies the type of link. This value MUST be listed in the following table:

Value	Description
A88B9DCB-5B82-41E4-8A19-17672F307B95	Specifies a membership group that was created based on a distribution list.
8BB1220F-DE8B-4771-AC3A-0551242CF2BD	Specifies a membership group that was created based on the user of a specific site (2)
861D8FB6-7012-4CD9-A7A0-A615AED038B3	Specifies a quick link.
EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C	Specifies a colleague group.
E21FE63D-0BF6-42C0-85BC-AC8031552558	Specifies a user-defined link group.

2.2.1.6 Privacy Policy

A 32-bit signed integer that defines whether privacy protection is enforced for a protectable resource. The value MUST be in the following table:

Value	Description
1	Privacy protection is enabled on the definition of the resource. A user can NOT disable privacy protection on a specific instance of the resource.
2	Privacy protection is disabled on the definition of the resource. The user can enable privacy protection on a specific instance of the resource.
4	Privacy protection is enabled on the definition of the resource. The user can disable privacy

Value	Description
	protection on a specific instance of the resource.
8	Privacy protection is disabled on the definition of the resource. The user can NOT enable privacy protection on a specific instance of the resource.

2.2.1.7 Is MLS Enabled

A bit that indicates whether personal sites are allowed to be created in multiple languages. This value MUST be listed in the following table:

Value	Description
0	The personal sites associated with the specified user MUST be created in the language of the personal site host.
1	The language of the personal site host associated with the user MUST be one of the languages for which a **language pack** is installed on the personal site host.

2.2.1.8 Is Item Security Overridable

A bit specifying whether the system enables the user to override the default Privacy scope assigned to the user profile privacy policy. This value MUST be listed in the following table:

Value	Description
0	Each User is NOT permitted to override the value of the DefaultItemSecurity parameter for the privacy policy. The value of the DefaultItemSecurity parameter MUST be applied for each User.
1	Each User is permitted to override the value of the DefaultItemSecurity parameter for the privacy policy. If the User has NOT specified an override value, then the value of the DefaultItemSecurity parameter MUST be applied for the User.

2.2.1.9 Is User Created

A bit specifying if the membership group was created by a user or not. This value MUST be listed in the following table:

Value	Description
0	This membership group was not created by a user.
1	This membership group was created by a user.

2.2.1.10 Name Format

A 1-byte signed integer that identifies a format used to create personal sites. The value MUST be listed in the following table:

Value	Description
1	The personal sites MUST be created at 'user name' such as http://portal_site/location/user name. If a user attempts to create a new personal site called 'user name' and there is already a personal site with the same name, an error MUST occur and the new personal site MUST NOT be created.

Value	Description
2	The personal sites MUST be created at 'user name' such as http://portal_site/location/user name. If a user attempts to create a new personal site called 'user name' and there is already a personal site with the same name and the user is not an **alternate account** of the owner of that personal site, then 'domain_user name' MUST be used instead, unless user is the same user for whom the first site is created In that case, a new site won't be created.
3	The personal sites MUST be created at 'domain_user name' such as http://portal_site/location/domain_user name.

2.2.1.11 Property Choice

A 32-bit signed integer specifying the choice type of the property which MUST be a value listed in the following table.

Value	Description
0	Off – The property does not use a choice list.
1	None – The property uses a choice list, but users are not able to browse choice list values.
2	Open – The property uses a choice list, and users are able to browse choice list values. New choice list items can be added to the list.
3	Closed – The property uses a choice list with browsing. New choice list items cannot be added to the list except by the administration.

2.2.1.12 Property Data

A 4-byte signed integer that describes the data type of a property. **Length** indicates whether or not the protocol client is allowed to specify the length of the property value, and a maximum value for that length. This value MUST be listed in the following table:

Value	Description	Length
1	integer	MUST NOT be specified.
2	big integer	MUST NOT be specified.
3	date time	MUST NOT be specified.
4	float	MUST NOT be specified.
5	HTML	User-defined length.
6	string	User-defined length.
7	binary	User-defined length.
8	unique identifier	MUST NOT be specified.
9	**e-mail address**	MUST be specified and MUST be greater than or equal to 0 and MUST be less than or equal to 3600.
10	URL	MUST be specified and MUST be greater than or equal to 0 and MUST be less than or equal to 2048.

Value	Description	Length
11	Login name	MUST be specified and MUST be greater than or equal to 0 and MUST be less than or equal to 250.
12	date	MUST NOT be specified.
13	Boolean	MUST NOT be specified.
14	date no year	MUST NOT be specified.

2.2.1.13 Separator

A 1-byte unsigned integer specifying the multiple-value separator in the user interface. The value MUST be listed in the following table:

Value	Description
0	Comma
1	Semi-colon
2	New Line
255	Unknown

2.2.1.14 Visibility

A 4-byte, signed integer specifying whether a **user profile** has permission to view the result field value. This value MUST be listed in the following table:

Value	Description
0	Value is not visible.
1	Value is visible.
2	Value is visible.
4	Value is visible.
8	Value is visible.
16	Value is visible.

2.2.1.15 User Suggestion

A 1-byte, unsigned integer specifying the type of a user suggestion. The value MUST be listed in the following table:

Value	Description
1	Colleague
2	Keyword

2.2.1.16 User Suggestion Status

A 1-byte, unsigned integer specifying the status of a user suggestion. The value MUST be listed in the following table:

Value	Description
1	Suggested
2	Rejected

2.2.1.17 Property

A 1-byte, unsigned integer specifying the type of a Property. The value MUST be listed in the following table:

Value	Description
1	The property is a core Property
2	The property is a profile type Property
3	The property is a profile subtype Property

2.2.2 Bit Fields and Flag Structures

None.

2.2.3 Binary Structures

None.

2.2.4 Result Sets

2.2.4.1 ValidIdentifier

The **Valid Identifier** result set returns a set of membership group record identifiers that are both greater than or equal than the *@BeginId* input parameter and less than or equal than the *@EndId* input parameter for distribution list sourced membership groups that have e-mail address and site sourced membership groups. The Valid Identifier result set will always be returned and will contain a maximum of *@EndId* – (*@BeginId* - 1) rows. If the value of the preceding calculation is less than or equal to 0 then 0 rows MUST be returned. If either *@BeginId* or *@EndId* be NULL, 0 rows MUST be returned.

The **Valid Identifier** result set is defined using T-SQL syntax as follows:

```
Id bigint,
```

Id: A membership group record identifier.

2.2.4.2 SuggestedColleagues

The **SuggestedColleagues** result set returns user profile data for Colleague property. The **SuggestedColleagues** result set MUST NOT contain more than 75 rows. The

SuggestedColleagues result set MUST contain 0 rows when the RecordId input parameter is NULL or when the value does not specify a valid user profile.

The **SuggestedColleagues** result set is defined using T-SQL syntax as follows:

```
RecordId bigint,
UserID uniqueidentifier,
NTName nvarchar(400),
PreferredName nvarchar(256),
Email nvarchar(256),
SipAddress nvarchar(250),
ProfileSubtypeID int,
PictureUrl nvarchar(max),
PersonTitle nvarchar(150),
Weight numeric(15,255),
```

RecordId: A user profile record identifier.

UserID: A GUID for the user profile.

NTName: A **Security Account Manager (SAM)** user name for the entity specified by the user profile.

PreferredName: The name of the entity as defined in the user profile.

Email: An e-mail address for the entity the user profile specifies.

SipAddress: A **Session Initiation Protocol (SIP) address** for the entity the user profile specifies.

ProfileSubtypeID: A **profile subtype** identifier.

PictureUrl: URL to a picture for the entity the user profile specifies.

PersonTitle: The title user profile property value for the entity the user profile specifies**.**

Weight: A number indicating how likely the entity is to be a Colleague. The value ranges from 0.0328765519086464 to 1.79E+308. Larger values of Weigh indicate a greater likelihood of a relationship between the colleagues.

2.2.4.3 Count

The **Count** result set returns the number of the distribution list sourced membership groups that have an e-mail address and site sourced membership groups. The **Count** result set MUST be returned and MUST contain 1 row.

The **Count** result set is defined using T-SQL syntax as follows:

```
Count int,
```

Count: Contains the number of distribution list sourced membership groups that have an e-mail address and site sourced membership groups.

2.2.4.4 membership_getGroupMemberships.ResultSet0

The **Membership** result set returns data about all memberships in a particular membership group. The **Membership** result set MUST contain 1 row for each user profile that has a membership in the membership group specified by the *@Id* input parameter. The maximum number of rows is limited only by the number of memberships in the membership group. If the *@Id* parameter is NULL or if it refers to a non-existing membership group then the **Membership** result set MUST return 0 rows. Otherwise, each row will contain data about the user profile, the membership relationship and the membership group.

The **Membership** result set is defined using T-SQL syntax as follows:

```
Id bigint,
ItemSecurity int,
GroupType tinyint,
GroupTitle nvarchar(400),
PolicyId uniqueidentifier,
MemberGroupId bigint,
Id bigint,
SID varbinary(512),
DisplayName nvarchar(250),
MailNickName nvarchar(250),
Description nvarchar(1500),
Source uniqueidentifier,
SourceReference nvarchar(2048),
Url nvarchar(2048),
MemberCount bigint,
LastUpdate datetime,
DSGroupType bigint,
DataSource nvarchar(400),
RecordId bigint,
NTName nvarchar(400),
UserId uniqueidentifier,
PreferredName nvarchar(256),
Email nvarchar(256),
SipAddress nvarchar(250),
ProfileSubtypeID int,
PictureUrl nvarchar(max),
UserID uniqueidentifier,
```

Id: A GUID identifying the membership relationship.

ItemSecurity: MUST be a Privacy (section 2.2.1.2) Type value that defines security for the membership.

GroupType: MUST be a Short Group (section 2.2.1.3) Type value specifying the membership relation grouping.

GroupTitle: A name for the grouping of which the membership is a part.

PolicyId: This field MUST be ignored.

MemberGroupId: A membership group record identifier for the membership group.

Id: A membership group record identifier.

SID: This field MUST be ignored.

DisplayName: A descriptive name for the membership group.

MailNickName: An Active Directory attribute that contains an e-mail alias for the membership group.

Description: A description of the membership group.

Source: MUST be a Short Link (section 2.2.1.4) Type identifier specifying the source of the membership group.

SourceReference: A string used to distinguish the various membership groups within a particular source specified by @Source. The Unicode String Trim operation MUST have been performed on the input value. When the @Source input parameter specifies a distribution list sourced membership group the value MUST be the membership group's **distinguished name (DN)**. When the @Source input parameter specifies a site sourced membership group the value MUST be the site identifier.

Url: A URI for the membership group.

MemberCount: The count of members in this membership group.

LastUpdate: A **Coordinated Universal Time (UTC)** value specifying the last time membership_updateGroup (Section 3.1.5.8) was successfully called.

DSGroupType: A group type identifier from the data source (such as Active Directory, LDAP, and others).

DataSource: The **DN** of the group.

RecordId: A user profile record identifier.

NTName: A **Security Account Manager (SAM)** user name for the entity specified by the user profile.

UserId: An identifier for the user profile.

PreferredName: The name of the entity as defined in the user profile.

Email: An e-mail address for the entity specified by the user profile.

SipAddress: A SIP address for the entity the user profile specifies.

ProfileSubtypeID: A profile subtype identifier.

PictureUrl: The picture URI user profile property value for the entity the user profile specifies.

UserID: An identifier for the user profile.

2.2.4.5 TitlePagedMembership

Title Paged Membership result set returns data about memberships in a particular membership group. The Title Paged Membership result set MUST contain 1 row for each user profile that has a membership in the membership group specified by the Id input parameter up to a maximum of Count rows. If the Id parameter is NULL or if it refers to a non existing membership group then the Membership result set MUST return 0 rows. Each row will contain data about the member's user profile, the membership relationship and the membership group. The Title Paged Membership result set will be ordered on the Title field in the order specified by the SortDirection input parameter and then in ascending order on the RecordId field.

The Title Paged Membership result set is defined using T-SQL syntax as follows:

```
Title sql_variant,
```

Title: The occupation title user profile property value for the entity the user profile specifies.

Department: The department user profile property value for the entity the user profile specifies.

PreferredName: The name of the entity as defined in the user profile.

RecordID: A user profile record identifier.

NTName: A **Security Account Manager (SAM)** user name for the entity specified by the user profile.

Email: A e-mail address for the entity the user profile specifies.

SipAddress: A SIP address for the entity the user profile specifies.

UserId: An identifier for the user profile.

AboutMe: The about me user profile property value for the entity the user profile specifies.

PictureUrl: The picture URI user profile property value for the entity the user profile specifies.

These fields contain information about permissions:

IsAboutMeVisible: MUST be a Visibility (Section 2.2.1.14) Type value specifying if the user profile specified by the ViewerRecordId input parameter has permission to view the AboutMe result field value.

IsPictureUrlVisible: MUST be a Visibility (Section 2.2.1.14) Type value specifying if the user profile specified by the ViewerRecordId parameter has permission to view the PictureUrl result field.

These fields contain information about the membership relation:

Id: A unique identifier for the membership relationship.

RecordId: A user profile record identifier for the member.

MemberGroupId: A membership group record identifier for the membership group.

GroupType: MUST be a Short Group Type value (Section 2.2.1.3) specifying the membership relation grouping.

GroupTitle: A name for the grouping the membership is a part of.

SID: This field MUST be ignored.

PolicyId: This field MUST be ignored.

ItemSecurity: MUST be a Privacy (Section 2.2.1.2) Type value that defines security for the membership.

These fields contain information about the membership group:

Id: A membership group record identifier.

DisplayName: A descriptive name for the membership group.

MailNickName: A ms-Exch-Mail-Nickname Active Directory attribute for the membership group.

Description: A description of the membership group.

Source: MUST be a Short Link (Section 2.2.1.4) Type identifier specifying the source of the membership group.

SourceReference: A string used to distinguish the various membership groups within a particular source specified by @Source. The Unicode String Trim operation MUST have been performed on the input value. When the @Source input parameter specifies a distribution list sourced membership group the value MUST be the membership group's DN. When the @Source input parameter specifies a site sourced membership group the value MUST be the site identifier.

cs_SourceReference: This field MUST be ignored.

Url: A URI for the membership group. For distribution list sourced membership groups this is a mailto URI. For site sourced membership groups the value is a URI used to reach the root of the site.

MemberCount: The count of members in this membership group.

LastUpdate: A **UTC** Value specifying the last time membership_updateGroup (Section 3.1.5.8) was successfully called.

WebID: This field MUST be ignored.

Type: This field MUST be ignored.

UserCreated: This field MUST be ignored.

2.2.4.6 RelatedMemberGroup

The **RelatedMemberGroup** returns the set of membership groups whose ancestor is the membership group specified by the *@Id* input parameter. The parent-child relationship is specified by membership_addRecursiveGroup in [MS-UPSIMP]. The **RelatedMemberGroup** result set MUST contain 1 row per descendant of the input membership group if any descendants exist. The **RelatedMemberGroup** result set MUST contain 0 rows when the *@Id* input parameter is NULL or when *@Id* does not match a valid membership group.

The **RelatedMemberGroup** result set is defined using T-SQL syntax as follows:

```
Id bigint,
DisplayName nvarchar(250),
MailNickName nvarchar(250),
Description nvarchar(1500),
Source uniqueidentifier,
SourceReference nvarchar(2048),
cs_SourceReference int,
Url nvarchar(2048),
SID varbinary(512),
MemberCount bigint,
LastUpdate datetime,
WebID uniqueidentifier,
Type tinyint,
UserCreated bit,
DSGroupType bigint,
```

```
PartitionID uniqueidentifier,
```

Id: A membership group record identifier.

DisplayName: A descriptive name for the membership group.

MailNickName: An Active Directory attribute that contains a mail alias for the membership group.

Description: A description of the membership group.

Source: MUST be a Short Link (Section 2.2.1.4) Type identifier specifying the source of the membership group.

SourceReference: A string used to distinguish the various membership groups within a particular source specified by @Source. The Unicode String Trim operation MUST have been performed on the input value. When the @Source input parameter specifies a distribution list sourced membership group the value MUST be the membership group's DN. When the @Source input parameter specifies a site sourced membership group the value MUST be the site identifier.

cs_SourceReference: This field MUST be ignored.

Url: URI for the membership group.

SID: This field MUST be ignored.

MemberCount: A count of Members in this membership group.

LastUpdate: A UTC Value specifying the last time **membership_updateGroup** (Section 3.1.5.8) was successfully called.

WebID: This field MUST be ignored.

Type: This field MUST be ignored.

UserCreated: This field MUST be ignored.

DSGroupType: A group type identifier from the data source (such as Active Directory, LDAP, and others).

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

2.2.4.7 AllPrivacyPolicy

The **AllPrivacyPolicy** result set returns the properties associated with each privacy policy in the system. The **AllPrivacyPolicy** result set MUST contain 0 or more rows where each row defines the profile property of an existing privacy policy. The result set MUST include 1 row for each privacy policy that does not have an associated profile property. The result set MUST exclude privacy policies associated with a user profile property section.

The **AllPrivacyPolicy** result set is defined using T-SQL syntax as follows:

```
Id uniqueidentifier,
DisplayName nvarchar(256),
GroupName nvarchar(256),
Policy int,
DefaultItemSecurity int,
```

```
IsItemSecurityOverridable bit,
PropertyId bigint,
ProfileSubtypeId int,
IsPolicyOverridable bit,
FilterPrivacyItems bit,
PartitionID uniqueidentifier,
DisplayOrder int,
```

Id: The identifier assigned to the privacy policy described by this row.

DisplayName: The descriptive name of the privacy policy.

GroupName: The descriptive name of the collection of privacy policies to which the privacy policy belongs.

Policy: The enforcement level assigned to the privacy policy. The value MUST be a Privacy Policy (Section 2.2.1.6) Type.

DefaultItemSecurity: The default protection level assigned to the privacy policy. The value MUST be a Privacy (Section 2.2.1.2) type.

IsItemSecurityOverridable: MUST be an Is Item Security Overridable (Section 2.2.1.8) Type value.

PropertyId: The identifier assigned to the user profile property associated with the privacy policy. If the privacy policy is not associated with a user profile property, then the value MUST be set to NULL.

ProfileSubtypeId: A profile subtype identifier. The rows in the result set MUST be ordered in ascending order by the ProfileSubtypeId.

IsPolicyOverridable: This value MUST be ignored.

FilterPrivacyItems: This value MUST be ignored.

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

DisplayOrder: Rows in the result set with the same ProfileSupbTypeID MUST be ordered in ascending order by the DisplayOrder. For all privacy policies associated with a user profile property, the value will be the value of the DisplayOrder property defined for the user profile property. For all privacy policies NOT associated with a user profile property, the value MUST be set to -1.

2.2.4.8 FeaturePrivacyPolicy

The **FeaturePrivacyPolicy** result set returns the definition associated with each privacy policy that is not associated with a specific user profile property.

The **FeaturePrivacyPolicy** result set MUST contain 0 or more rows where each row defines the properties of an existing privacy policy which is not associated with a specific user profile property.

The **FeaturePrivacyPolicy** result set is defined using T-SQL syntax as follows:

```
Id uniqueidentifier,
DisplayName nvarchar(256),
GroupName nvarchar(256),
Policy int,
```

```
DefaultItemSecurity int,
IsItemSecurityOverridable bit,
PropertyId bigint,
ProfileSubtypeId int,
IsPolicyOverridable bit,
FilterPrivacyItems bit,
PartitionID uniqueidentifier,
```

Id: The GUID assigned to the Privacy Policy defined by this row.

DisplayName: The descriptive name of the privacy policy.

GroupName: The descriptive name held in common by any Privacy Policy.

Policy: The enforcement level assigned to the privacy policy. The value MUST be a Privacy Policy (Section 2.2.1.6) Type.

DefaultItemSecurity: The default protection level assigned to the privacy policy. The value MUST be a Privacy (Section 2.2.1.2) Type.

IsItemSecurityOverridable: MUST be an "Is Item Security Overridable" (Section 2.2.1.8) Type value.

PropertyId: MUST be NULL.

ProfileSubtypeId: A profile subtype identifier.

IsPolicyOverridable: This value MUST be ignored.

FilterPrivacyItems: This value MUST be set to 1.

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

2.2.4.9 profile_EnumUsers

The **profile_EnumUsers** result set MUST contain 0 or more rows. For a record to be included in the result set, it MUST NOT have a NULL login name.

The **profile_EnumUsers** result set is defined using T-SQL syntax as follows:

```
RecordID bigint,
UserID uniqueidentifier,
```

RecordID: The user profile record identifier.

UserID: The GUID of the user.

2.2.4.10 profile_GetCommonManager

The **profile_GetCommonManager** returns the common manager property between the 2 specified users. In the case that either GUID does not refer to an existing user this MUST returns 0 rows, else if successful it MUST return at least 1 row.

The **profile_GetCommonManager** result set is defined using T-SQL syntax as follows:

```
RecordId bigint,
UserID uniqueidentifier,
NTName nvarchar(400),
Email nvarchar(256),
SipAddress nvarchar(250),
PreferredName nvarchar(256),
ProfileSubtypeID int,
PictureUrl nvarchar(max),
Title nvarchar(150),
FirstCommon bit,
```

RecordId: Contains the record identifier of the manager property. This value MUST be returned and MUST NOT be NULL.

UserID: Contains the GUID identifying the manager property. This value MUST be returned and MUST NOT be NULL.

NTName: The login name of the manager property. The value MUST be returned and MUST NOT be NULL.

Email: The e-mail address of the manager property.

SipAddress: The **Session Initiation Protocol (SIP)** address of the manager.

PreferredName: The name of the manager property as defined in the user profile.

ProfileSubtypeID: A profile subtype identifier.

PictureUrl: The picture URL user profile property value for the entity the user profile specifies.

Title: The title of the manager property.

FirstCommon: This MUST be 1 if the manager property is the lowest-level manager property held in common between the 2 specified users. The value MUST be returned and MUST NOT be NULL.

2.2.4.11 UserProperties

The **UserProperties** result set returns the available properties for the user specified. The **UserProperties** result set MUST contain 0 or more rows of user properties.**UserProperties** is defined using T-SQL syntax as follows:

```
RecordId bigint,
ProfileSubtypeID int,
PropertyId bigint,
PropertyVal sql_variant,
Privacy int,
```

RecordId: The record identifier for the user. If *@AllowAlternateAccountName* is set to 1, then the record identifier MUST be the alternate record identifier of the specified user. RecordId MUST NOT be NULL.

ProfileSubtypeID: A profile subtype identifier.

PropertyId: The identifier of the property. MUST NOT be NULL.

PropertyVal: The value of the property that is specified by PropertyId.

Privacy: The privacy policy value, as defined in the Privacy Policy Section (2.2.1.6).

2.2.4.12 MultiLoginAccounts

The **MultiLoginAccounts** result set contains all of the login names for the user specified by *@RecordId*. If no login names match the specified *@RecordId*, the result set MUST contain 0 rows.

The **MultiLoginAccounts** result set is defined using T-SQL syntax as follows:

```
NTName nvarchar(400),
```

NTName: A **Security Account Manager (SAM)** user name for the entity specified by the user profile.

2.2.4.13 ProfilePersonalSite

The **ProfilePersonalSite** returns the personal site configuration properties of the specified server. The **ProfilePersonalSite** result set MUST return 1 row.

The **ProfilePersonalSite** result set is defined using T-SQL syntax as follows:

```
Inclusion nvarchar(500),
NameFormat smallint,
EnableMLS bit,
SiteReader ntext,
PartitionID uniqueidentifier,
```

Inclusion: The site under which user personal sites are created. This value MUST be returned.

NameFormat: MUST be a Name Format (Section 2.2.1.10) type value.

EnableMLS: MUST be an Is MLS Enabled (Section 2.2.1.7) Type value

SiteReader: A comma-delimited list of user display names that will be granted the permission to read content on new user personal sites. This value MUST NOT be NULL.

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

2.2.4.14 profile_GetProfileCount

The **profile_GetProfileCount** result set MUST contain 1 row.

The **profile_GetProfileCount** result set is defined using T-SQL syntax as follows:

```
CountTrack int,
```

CountTrack: The number of user profiles contained in the user profile store.

2.2.4.15 GetLocalizedProfileProperty

The **GetLocalizedProfileProperty** result set returns localized property information. The result set MUST contain 1 or more rows.The **GetLocalizedProfileProperty** result set is defined using T-SQL syntax as follows:

```
Id bigint,
PropertyId bigint,
PropertyField int,
Lcid int,
Text nvarchar(512),
PartitionID uniqueidentifier,
```

Id: This value MUST be ignored.

PropertyId: The identifier of the property.

PropertyField: This value indicates if the Text value is a user display name or description. The PropertyField MUST be a value listed in the following table:

Possible parameter values:

Value	Description
1	Indicates that the Text field contains the localized value of the user display name associated with the property.
2	Indicates that the Text field contains the localized value of the description associated with the property.

Lcid: The **language code identifier (LCID)** indicating what language the specified text is in.

Text: The localized value of the user display name or description associated with the property.

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

2.2.4.16 SharedList

The **SharedList** result set returns 0 or more rows of shared list items.

The **SharedList** result set is defined using T-SQL syntax as follows:

```
ListId uniqueidentifier,
ItemId bigint,
Title nvarchar(256),
Url nvarchar(2048),
TargetTo ntext,
Int1 bigint,
PartitionID uniqueidentifier,
Owner nvarchar(256),
```

ListId: Contains the GUID of the shared list. This MUST NOT be NULL.

ItemId: Contains the identifier of the specified list item from the shared list. This MUST NOT be NULL.

Title: Contains the user-friendly name of the specified list item from the shared list. This MUST NOT be NULL.

Url: Contains a URL link of the specified list item from the shared list. This MUST NOT be NULL.

TargetTo: Contains the **audience** information of the specified list item from the shared list. The field MUST either be empty or contain a triplet of comma-delimited audiences, comma-delimited distribution lists, and comma-delimited groups. The triplet separator MUST be ";;".

Int1: Contains the **content type** of the specified list item from the shared list.

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

Owner: This MUST contain either the NTName of the **owner** of the shared list item, or an empty string. This MUST NOT be NULL.

2.2.4.17 ProfileGetUserFormat

The **ProfileGetUserFormat** MUST return 1 row containing the user name format that is used by the system.

The **ProfileGetUserFormat** result set is defined using T-SQL syntax as follows:

```
PersonDBFormat int,
```

PersonDBFormat: The user name format that is used by the system. The field MUST be a value listed in the following table:

Possible parameter values:

Value	Description
1	Indicates that the system uses login names to identify users.
2	Indicates that the system uses distinguished names (DN) to identify users.
3	Indicates that the system uses a **user principal name (UPN)** to identify a specified user.
4	Indicates that the system uses a user display name to identify each user.
5	Indicates that the system uses GUIDs to identify users.

2.2.4.18 ViewerRights

The **ViewerRights** result set returns the viewer rights for the user specified. The **ViewerRights** result set MUST contain exactly 1 row.

The **ViewerRights** result set is defined using T-SQL syntax as follows:

```
ViewerRights int,
```

ViewerRights: Contains the viewer rights of the specified user. If the *@ViewerRights* parameter was set to PRIVACY_NOTSET, then this column MUST now contain the rights of the user requesting the information. If the *@ViewerRights* parameter was set to a value other than PRIVACY_NOTSET, this column MUST contain the value of *@ViewerRights* parameter. This column MUST NOT be NULL. This MUST be a Privacy (Section 2.2.1.2) type value.

2.2.4.19 UserProfile

The **UserProfile** result set returns the available properties for the user specified. The **UserProfile** result set MUST contain 0 or more rows of user properties.The **UserProfile** result set is defined using T-SQL syntax as follows:

```
RecordId bigint,
UserID uniqueidentifier,
NTName nvarchar(400),
PreferredName nvarchar(256),
Email nvarchar(256),
SipAddress nvarchar(250),
ProfileSubtypeID int,
PictureUrl nvarchar(max),
PersonTitle nvarchar(150),
```

RecordId: Contains the record identifier associated with the specified user. This MUST NOT be NULL.

UserID: Contains the GUID for the user. This MUST NOT be NULL.

NTName: A **Security Account Manager (SAM)** user name for the entity specified by the user profile. This MUST NOT be NULL.

PreferredName: The name of the entity as defined in the user profile. Contains the user display name.

Email: Contains the e-mail addresses for the user.

SipAddress: Contains the Session Initiation Protocol (SIP) address of the colleague.

ProfileSubtypeID: A profile subtype identifier.

PictureUrl: The picture URL user profile property value for the entity the user profile specifies.

PersonTitle: Contains the user title.

2.2.4.20 UpdateUserProfileBlobDataResult

The **UpdateUserProfileBlobData** MUST contain 1 row if the property specified by PropertyName is valid. If the property does not exist, a result set MUST NOT be returned.

The **UpdateUserProfileBlobData** result set is defined using T-SQL syntax as follows:

```
UserID uniqueidentifier,
RecordId bigint,
UpdateCount int,
```

UserID: The GUID of the user whose property was updated.

RecordId: The record identifier of the user.

UpdateCount: The count of the properties updated. This MUST be 1 if the property was successfully updated. If the property wasn't successfully updated, this MUST be 0.

2.2.4.21 UserLinks

The **UserLinks** contains a user's links data.

The **UserLinks** result set is defined using T-SQL syntax as follows:

```
Id bigint,
ItemSecurity int,
GroupType tinyint,
GroupTitle nvarchar(400),
PolicyId uniqueidentifier,
Title nvarchar(500),
Url nvarchar(2048),
ContentClass nvarchar(200),
```

Id: Contains the unique identifier for the link / user pair. This MUST NOT be NULL.

ItemSecurity: MUST be a Privacy (section 2.2.1.2) Type value that defines security for the user.

GroupType: MUST be a Group (section 2.2.1.1) Type value that specifies the link group type.

GroupTitle: Contains the group title for the link associated with the specified user..

PolicyId: Contains a Short Link (section 2.2.1.4) Type identifier specifying the link type.

Title: Contains the title for the user's link.

Url: Contains the URL for the user's link.

ContentClass: Specifies the Content type for the hyperlink specified in the URL for the user's link.

2.2.4.22 UserColleagues

The **UserColleagues** result set returns property data about colleagues.

The **UserColleagues** result set is defined using T-SQL syntax as follows:

```
Id bigint,
ItemSecurity int,
GroupType tinyint,
GroupTitle nvarchar(400),
IsWorkgroup bit,
ColleagueRecordId bigint,
```

Id: Contains the unique identifier for the colleague-user pair. This MUST NOT be NULL.

ItemSecurity: MUST be a Privacy (section 2.2.1.2). Type value that defines security for the user.

GroupType: MUST be a Group (section 2.2.1.1). Type value that specifies the group type of the Colleague.

GroupTitle: Contains the group title of the Colleague.

IsWorkgroup: Contains a flag specifying whether the record is a workgroup. A value of "0" indicates that the record is not a workgroup. A value of "1" indicates that the record is a workgroup.

ColleagueRecordId: Contains the record identifier of the user's Colleague.

2.2.4.23 QuickLinksRetrieveColleaguesOfColleagues.ResultSet0

The **QuickLinksRetrieveColleaguesOfColleagues** returns each user that is a colleague of any of the colleague properties of the specified user.

In order for a colleague's colleague to be returned in the result set, 1 of the following conditions MUST be met:

1. The privacy setting between the colleague and the specified colleague MUST be "2" (Contacts) and the user indicated by *@RecordId* MUST be a colleague of the colleague.
2. The privacy setting between the colleague and the specified colleague of the colleague MUST be "4" (Organization) and the user indicated by *@RecordId* MUST be flagged by the colleague as part of the specified workgroup of the colleague, by setting the **LinkUserIdIsWorkgroup** parameter to "1" when calling the **QuickLinksAdd** stored procedure to add the colleague.
3. The privacy setting between the colleague and the colleague property of the specified colleague MUST be "8" (Manager) and the user indicated by *@RecordId* MUST be the **Manager** property of the colleague.
4. The privacy setting between the colleague and the specified colleague property of the colleague MUST be "1" (Public).

Each row in the result set contains user profile identification about a different colleague of a colleague. Each row MUST reference a unique user profile.

The **QuickLinksRetrieveColleaguesOfColleagues** result set is defined using T-SQL syntax as follows:

```
ColleagueRecordId bigint,
```

ColleagueRecordId: The identifier of the user profile record.

2.2.4.24 QuickLinksRetrieveGroupList

The **QuickLinksRetrieveGroupList** stored procedure is called to retrieve the groups of a specified type for a group list.

The **QuickLinksRetrieveGroupList** result set is defined using T-SQL syntax as follows:

```
GroupTitle nvarchar(400),
```

GroupTitle: The name of each group returned by the stored procedure according to the specified input criteria.

2.2.4.25 membership_getGroupById.ResultSet0

The **MembershipGroupResultSet** returns the membership group specified by the *@Id* input parameter, in the **partition (1)** identified by *@partitionID*. If either *@Id* or *@partitionId* input parameters be NULL, 0 rows MUST be returned. The **MembershipGroupResultSet** MUST contain 1 row if *@Id* and *@partitionId* correspond to an existing membership group.

```
Id bigint,
SID varbinary(512),
DisplayName nvarchar(250),
MailNickName nvarchar(250),
Description nvarchar(1500),
Source uniqueidentifier,
SourceReference nvarchar(2048),
Url nvarchar(2048),
MemberCount bigint,
LastUpdate datetime,
DSGroupType bigint,
DataSource nvarchar(400),
AllWebsSynchID int,
Type tinyint,
UserCreated bit,
PartitionID uniqueidentifier,
```

Id: This MUST be a GUID identifying the membership group record.

SID: This field MUST be ignored.

DisplayName: A descriptive name for the membership group.

MailNickName: An Active Directory attribute that contains a mail alias for the membership group.

Description: A description of the membership group.

Source: This MUST be a Short Link (section 2.2.1.4) Type identifier specifying the source of the membership group.

SourceReference: A string used to distinguish the various membership groups within a particular source specified by @Source. The Unicode String Trim operation MUST have been performed on the input value. When the @Source input parameter specifies a distribution list sourced membership group the value MUST be the membership group's DN. When the @Source input parameter specifies a site sourced membership group the value MUST be the site identifier.

Url: A URI for the membership group.

MemberCount: A count of Members in this membership group.

LastUpdate: A UTC Value specifying the last time membership_updateGroup (Section 3.1.5.8) was successfully called.

DSGroupType: A group type identifier from the data source (such as Active Directory, LDAP, and others).

DataSource: The DN of the group.

AllWebsSynchID: The identifier of the web that the group belongs to.

Type: This value MUST be ignored.

UserCreated: A flag indicating if the group was created by a user.

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

2.2.4.26 ImmediateMembership

The **ImmediateMembership** result set returns data about all direct memberships in a particular membership group. The **ImmediateMembership** result set MUST contain 1 row for each user profile that has a direct membership in the membership group specified by the *@partitionID* and *@Id* input parameters. The maximum number of rows is limited only by the number of direct memberships in the membership group. If either *@Id* or *@partitionId* input parameters be NULL or refer to a non existing direct membership group, 0 rows MUST be returned. Otherwise, each row will contain data about the user profile, the membership relationship and the membership group.

The **ImmediateMembership** result set is defined using T-SQL syntax as follows:

```
Id bigint,
ItemSecurity int,
GroupType tinyint,
GroupTitle nvarchar(400),
PolicyId uniqueidentifier,
MemberGroupId bigint,
Id bigint,
SID varbinary(512),
DisplayName nvarchar(250),
MailNickName nvarchar(250),
Description nvarchar(1500),
Source uniqueidentifier,
SourceReference nvarchar(2048),
Url nvarchar(2048),
MemberCount bigint,
LastUpdate datetime,
DSGroupType bigint,
DataSource nvarchar(400),
RecordId bigint,
```

Id: This MUST be a record identifier for the membership relationship.

ItemSecurity: This MUST be a Privacy (Section 2.2.1.2) Type value that defines security for the membership. These fields contain information about the membership group.

GroupType: This MUST be a Short Group (Section 2.2.1.3) Type value specifying the membership relation grouping.

GroupTitle: A name for the grouping of which the membership is a part.

PolicyId: This field MUST be ignored.

MemberGroupId: This MUST be a membership group record identifier for the membership group.

Id: This MUST be a membership group record identifier for the membership group.

SID: This field MUST be ignored.

DisplayName: A descriptive name for the membership group.

MailNickName: An Active Directory attribute that contains a mail alias for the membership group.

Description: A description of the membership group.

Source: This MUST be a Short Link (Section 2.2.1.4) Type identifier specifying the source of the membership group.

SourceReference: A string used to distinguish the various membership groups within a particular source specified by @Source. The Unicode String Trim operation MUST have been performed on the input value. When the @Source input parameter specifies a distribution list sourced membership group the value MUST be the membership group's DN. When the @Source input parameter specifies a site sourced membership group the value MUST be the site identifier.

Url: A URI for the membership group.

MemberCount: This MUST be the count of members in this membership group.

LastUpdate: This MUST be a UTC value specifying the last time membership_updateGroup (Section 3.1.5.8) was successfully called.

DSGroupType: A group type identifier from the data source (such as Active Directory, LDAP, and others).

DataSource: The domain name of the group.

RecordId: This MUST be a GUID identifying the user profile record.

2.2.4.27 profile_GetColleagueAdders.ResultSet0

The **GetColleagueAdders** result set contains information about the users (colleagues) for whom a link of type Colleague has been created after a given time. All the records in the result set MUST represent links to the same user (the target user).

The **GetColleagueAdders** result set is defined using T-SQL syntax as follows:

```
RecordID bigint,
```

RecordID: The user profile record identifier for the colleague represented by this record.

2.2.4.28 CorePropertyInfoResultSet

The **CorePropertyInfoResultSet** result set contains information about a list of properties. If the *@PropertyURI* or *@PropertyName* parameters were specified, then the **CorePropertyInfoResultSet** result set MUST contain one row for each property matching either the specified *@PropertyURI* parameter or the specified *@PropertyName*; otherwise, it MUST contain one row for each property in the partition specified by the *@partitionId* parameter.

If no properties exist, or none match the specified *@PropertyURI* or *@PropertyName* parameters, both as further qualified by the partition (1) specified by the *@partitionId* parameter then the **CorePropertyInfoResultSet** result set must contain 0 rows.

The **CorePropertyInfoResultSet** result set is defined using T-SQL syntax as follows:

```
PropertyID bigint,
PropertyName nvarchar(250),
PropertyURI nvarchar(250),
DataTypeID int,
DataType nvarchar(50),
TermSetID uniqueidentifier,
Length int,
```

```
BlobType tinyint,
IsSection bit,
IsMultiValue bit,
IsAlias bit,
IsAuxiliary bit,
IsUpgrade bit,
IsUpgradePrivate bit,
IsSearchable bit,
Separator tinyint,
IsExpand bit,
PartitionID uniqueidentifier,
Name nvarchar(500),
FriendlyTypeName nvarchar(500),
IsEmail bit,
IsURL bit,
IsPerson bit,
IsHTML bit,
```

PropertyID: The identifier assigned to the property.

PropertyName: The name assigned to the property.

PropertyURI: The URI of the property.

DataTypeID: The identifier of the corresponding DataType for the property.

DataType: The name of the **T-SQL** type of the data that is stored for this property.

TermSetID: The identifier of the corresponding **term set** for this property.

Length: Contains the maximum length of values for the property.

BlobType: Contains a **TYPE:ProfilePropertyBlobType** (as specified in [MS-UPSIMP], section 2.2.6) enumeration value indicating how the binary large object (BLOB) is stored in the database.

IsSection: Contains a value that indicates whether the property is a section. This value MUST be 1 to indicate that this property is a section. This value MUST be 0 to indicate that this property is not a section.

IsMultiValue: Contains a value that indicates whether the property is a **multivalue property**. This value MUST be 1 to indicate that this property is a multi-value property. This value MUST be 0 to indicate that this property is not a multi-value property.

IsAlias: Contains a value that indicates whether the property serves as an e-mail alias of the user for user search purposes. This value MUST be 1 to indicate that this property serves as an alias of the user for user search purposes. This value MUST be 0 to indicate that this property does not serve as an alias of the user for user search purposes.

IsAuxiliary: Contains a value that indicates whether the property is an auxiliary property. This value MUST be 1 to indicate that this property is an auxiliary property. This value MUST be zero to indicate that this property is not an auxiliary property.

IsUpgrade: Contains a value that indicates whether the property exists in a previously upgraded installation. This value MUST be 1 to indicate that this property existed in a previously upgraded installation. This value MUST be 0 to indicate that this property did not exist in a previously upgraded installation.

IsUpgradePrivate: Contains a value that indicates whether the property was private in a previously upgraded installation. This value MUST be 1 to indicate that this property was private in a previously upgraded installation. This value MUST be 0 to indicate that this property was not private in a previously upgraded installation.

IsSearchable: Contains a value that indicates whether the property is indexed by the search server. This value MUST be 1 to indicate that this property is indexed by Search Server. This value MUST be 0 to indicate that this property is not indexed by Search Server.

Separator: MUST be a Separator (section 2.2.1.13) Type value.

IsExpand: This value MUST be ignored by the protocol client.

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

Name: Contains the unique name of the property's Data Type (see section 2.2.1.12).

FriendlyTypeName: Contains the human friendly description of the current Profile Data Type.

IsEmail: Contains a bit that if 1 indicates that all values for the property MUST be a valid e-mail address. This value MUST be 0 to indicate that the values for the property are not a valid e-mail address.

IsURL: Contains a bit that if 1 indicates that all values for the property MUST contain a value in URL format. If 1, the client MUST map the host name to match the request. This value MUST be 0 to indicate that the value for the property do not contain a value in URL format.

IsPerson: Contains a bit that if 1 indicates that all values for the property, if non-NULL, MUST be a valid Domain User Account that has been validated against the domain controller (DC) of the referenced Account Domain. This value MUST be 0 to indicate that values for the property are not a valid Domain User Account that has been validated against the domain controller of the referenced Account Domain.

IsHTML: Contains a bit that if 1 indicates that all values for the property are fully formatted HTML text data. This value MUST be 0 to indicate that the values for the property are not fully formatted HTML text data.

2.2.4.29 DeletedUserList

The **GetDeletedUserList** result set gives the list of user names for profiles that have been deleted.

The **GetDeletedUserList** result set is defined using T-SQL syntax as follows:

```
NTName nvarchar(400),
```

NTName: A Security Account Manager (SAM) user name for the deleted user profile.

2.2.4.30 ExtendedReports

The **ExtendedReports** data set gives a list of the employees who report to the same manager for the user specified by the parameter *@NTName*. If *@NTName* specifies a valid name, **ExtendedReports** MUST contain 1 or more rows. If *@NTName* is not a valid username, **ExtendedReports** MUST contain 0 rows. The **ExtendedReports** result set MUST be in ascending order by the values in the PreferredName column.

The **ExtendedReports** result set is defined using T-SQL syntax as follows:

```
RecordId bigint,
UserID uniqueidentifier,
NTName nvarchar(400),
PreferredName nvarchar(256),
Email nvarchar(256),
SipAddress nvarchar(250),
ProfileSubtypeID int,
PictureUrl nvarchar(max),
PersonTitle nvarchar(150),
```

RecordId: A user profile record identifier.

UserID: A GUID for the user profile.

NTName: A Security Account Manager (SAM) user name for the entity specified by the user profile.

PreferredName: The name of the entity as defined in the user profile.

Email: An e-mail address for the entity the user profile specifies.

SipAddress: A Session Initiation Protocol (SIP) address for the entity the user profile specifies.

ProfileSubtypeID: A profile subtype identifier.

PictureUrl: Contains the picture URI profile property value for the entity the profile specifies.

PersonTitle: The title user profile property value for the entity the user profile specifies.

2.2.4.31 GetProfileList

The **GetProfileList** result set contains the profile matching the *@ProfileName* parameter in the partition (1) identified by *@partitionID*, or all profiles under the specified partition (1) in the event that the *@ProfileName* parameter is NULL.

The **GetProfileList** result set MUST contain 1 row if *@PartitionName* and *@partitionId* correspond to an existing profile, and zero rows if no match is found.

The **GetProfileList** result set is defined using T-SQL syntax as follows:

```
ProfileSubtypeID int,
ProfileName nvarchar(250),
ProfileDisplayName nvarchar(400),
ProfileTypeID smallint,
```

ProfileSubtypeID: This MUST be a unique 32-bit integer identifying the profile.

ProfileName: This MUST be a unique name used to identify the profile.

ProfileDisplayName: This MUST be the display name for the profile used in the user interface.

ProfileTypeID: This value MUST be either 1 for user profiles or 2 for organization profiles.

2.2.4.32 GetProfileSubtypePropertyInfoResult

The **GetProfileSubtypePropertyInfoResult** result set contains property data for properties associated with the matching *@ProfileSubtypeID* parameter in the partition (1) identified by

@partitionID. If *@PropertyID* is defined and matches an existing Property ID, this result set MUST contain 1 row matching the Property ID. If *@PropertyID* is NULL, all properties for the specified *@ProfileSubtypeID* MUST be returned.

If no properties match on the *@ProfileSubtypeID* parameter in the partition (1) identified by *@partitionID*, or if the *@PropertyID* is defined but does not match an existing Property ID then this result set MUST contain 0 rows.

The **GetProfileSubtypePropertyInfoResult** result set is defined using T-SQL syntax as follows:

```
ProfileName nvarchar(250),
ProfileSubtypeID int,
PropertyID bigint,
DisplayOrder int,
IsEditable bit,
IsAdminEditOnly bit,
IsImport bit,
IsUpgrade bit,
IsUpgradePrivate bit,
PartitionID uniqueidentifier,
Policy int,
DefaultItemSecurity int,
IsItemSecurityOverridable bit,
IsPolicyOverridable bit,
IsSection bit,
```

ProfileName: This column MUST match the unique profile name of the specified profile subtype.

ProfileSubtypeID: This column MUST match the @ProfileSubtypeID parameter.

PropertyID: This column MUST match the unique Property ID for the property being described. If @PropertyID is not NULL, this column MUST match the @PropertyID parameter.

DisplayOrder: This column MUST contain an integer value used to specify the UI display order for the property being described.

IsEditable: This column MUST contain a Boolean value that specifies whether the property being described is editable by the user. This value MUST be 1 to indicate this property can be updated by the user. This value MUST be 0 to indicate this property cannot be updated by the user.

IsAdminEditOnly: This column MUST contain a Boolean value that specifies whether the property being described is editable by a site admin. This value MUST be 1 to indicate this property can only be edited by a site admin. This value MUST be 0 to indicate this property cannot only be edited by a site admin.

IsImport: This column MUST contain a Boolean value that specifies whether the property being described has a value that is imported from an external data source. This value MUST be 1 to indicate the property has a value that is imported from an external data source. This value MUST be 0 to indicate the property has a value that is not imported from an external data source.

IsUpgrade: This column MUST contain a Boolean value that specifies whether the property existed in a previously upgraded installation. This value MUST be 1 to indicate that this property existed in a previously upgraded installation. This value MUST be 0 to indicate that this property did not exist in a previously upgraded installation.

IsUpgradePrivate: This column MUST contain a Boolean value that specifies whether this property was private in a previously upgraded installation. This value MUST be 1 to indicate that this property

was private in a previously upgraded installation. This value MUST be 0 to indicate that this property was not private in a previously upgraded installation.

PartitionID: This column MUST contain the same value specified by the @partitionID parameter.

Policy: This column MUST specify the enforcement level assigned to the privacy policy. This value MUST be a Privacy Policy (Section 2.2.1.6) Type.

DefaultItemSecurity: This column MUST specify the default protection level assigned to the privacy policy. This value MUST be a Privacy (Section 2.2.1.2) type.

IsItemSecurityOverridable: This value MUST be an Is Item Security Overridable (Section 2.2.1.8) Type value.

IsPolicyOverridable: This value MUST be ignored.

IsSection: This column MUST contain a value that indicates whether the property being described is a section. This value MUST be 1 to indicate that this property is a section. This value MUST be 0 to indicate that this property is not a section.

2.2.4.33 GetProfileTypePropertyInfo

The **GetProfileTypePropertyInfo** result set MUST contain zero or more records.

The **GetProfileTypePropertyInfo** result set is defined using T-SQL syntax as follows:

```
ProfileTypeID smallint,
PropertyID bigint,
IsEventLog bit,
IsSystem bit,
IsUpgrade bit,
IsUpgradePrivate bit,
IsVisible bit,
IsVisibleOnViewer bit,
MaximumShown int,
Replicable bit,
PartitionID uniqueidentifier,
IsSection bit,
```

ProfileTypeID: Contains the Profile Type identifier.

PropertyID: Contains the Property record identifier.

IsEventLog: Contains a Boolean value to indicate whether or not changes to the property are returned for colleague's change tracking. This value MUST be 1 to indicate that changes to this property are displayed in a Colleague Tracker. This value MUST be 0 to indicate that changes to this property are not displayed in a Colleague Tracker.

IsSystem: Contains a Boolean value to indicate whether the property is a system reserved property or not. This value MUST be 1 to indicate that the property is a system reserved property. This value MUST be 0 to indicate that the property is not a system reserved property.

IsUpgrade: Contains a Boolean to indicate whether the property existed in a previously upgraded installation. This value MUST be 1 to indicate that this property exists in a previously upgraded installation. This value MUST be 0 to indicate that this property does not exist in a previously upgraded installation.

IsUpgradePrivate: Contains a Boolean to indicate whether the property was private in a previously upgraded installation. This value MUST be 1 to indicate that this property was private in a previously upgraded installation. This value MUST be 0 to indicate that this property was not private in a previously upgraded installation.

IsVisible: Contains a Boolean value to indicate whether the property is visible on Edit my profile page or not. This value MUST be 1 to indicate that this property is visible. This value MUST be 0 to indicate that this property is not visible.

IsVisibleOnViewer: Contains a Boolean value to indicate whether the property is visible on default profile viewer. This value MUST be 1 to indicate that this property is visible on the default profile viewer page. This value MUST be 0 to indicate that this property is not visible on the default profile viewer page.

MaximumShown: Contains a number to indicate the maximum number of values to be shown for a property.

Replicable: Contains a Boolean value to indicate whether the property will be replicated to sites or not. This value MUST be 1 to indicate that this property is Replicable. This value MUST be 0 to indicate that this property is not Replicable.

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

IsSection: This column MUST contain a value that indicates whether the property is a section. This value MUST be 1 to indicate that this property is a section. This value MUST be 0 to indicate that this property is not a section.

2.2.4.34 GetUserProfileName

The **GetUserProfileName** result set MUST contain 0 rows if the stored procedure finds no qualifying *@ProfileSubtypeID* or 1 row if such record is found.

The **GetUserProfileName** result set returned by profile_GetUserProfileName is defined using T-SQL syntax as follows:

```
ProfileSubtypeID int,
```

ProfileSubtypeID: A profile subtype identifier.

2.2.4.35 GetUsers

The **GetUsers** result set MUST return a result set of 0 or more rows. For a record to be included in the result set, it MUST be in the specified partition (1).

The **GetUsers** result set returned by profile_GetUsers is defined using T-SQL syntax as follows:

```
RecordID bigint,
UserID uniqueidentifier,
```

RecordID: A user profile record identifier. This MUST NOT be NULL.

UserID: A GUID for the user profile. This MUST NOT be NULL.

2.2.4.36 GetSuggestions

The **GetSuggestions** result set represents user suggestions.

The **GetSuggestions** result set is defined using T-SQL syntax as follows:

```
Id bigint,
RecordId bigint,
Suggestion nvarchar(256),
Type tinyint,
Status tinyint,
Weight numeric(15,255),
PartitionID uniqueidentifier,
Created datetime,
```

Id: Contains the identifier assigned to the user suggestion described by this row. The value MUST NOT be NULL.

RecordId: Contains the record identifier of the user profile which this suggestion is associated. The value MUST NOT be NULL.

Suggestion: Contains the text of the user suggestion. The value MUST NOT be NULL.

Type: Contains the type of the user suggestion. The value MUST be a User Suggestion Type (Section 2.2.1.15) and MUST NOT be NULL.

Status: Contains the status of the user suggestion. The value MUST be a User Suggestion Status (Section 2.2.1.16) and MUST NOT be NULL.

Weight: Contains the relative weight of this suggestion, a number indicating how likely the entity is to be a Colleague. The value ranges from 0.0328765519086464 to 1.79E+308. Larger values of Weigh indicate a greater likelihood of a relationship between the colleagues. The value MUST NOT be negative and MUST NOT be NULL.

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

Created: Contains the date and time when this suggestion was created. The value MUST NOT be NULL.

2.2.4.37 GetGroupBySourceAndSourceReference

The **GetGroupBySourceAndSourceReference** returns the membership group specified by *@Source* and *@SourceReference* input parameters, in the partition (1) identified by *@partitionID*. The **GetGroupBySourceAndSourceReference** result set MUST contain 1 row if *@partitionID*, *@Source* and *@SourceReference* correspond to an existing membership group.

The **GetGroupBySourceAndSourceReference** result set is defined using T-SQL syntax as follows:

```
Id bigint,
SID varbinary(512),
DisplayName nvarchar(250),
MailNickName nvarchar(250),
Description nvarchar(1500),
Source uniqueidentifier,
SourceReference nvarchar(2048),
Url nvarchar(2048),
```

```
MemberCount bigint,
LastUpdate datetime,
DSGroupType bigint,
DataSource nvarchar(400),
```

Id: This MUST be a GUID identifying the membership group record.

SID: This field MUST be ignored.

DisplayName: A descriptive name for the membership group.

MailNickName: An Active Directory attribute that contains a mail alias for the membership group.

Description: A description of the membership group.

Source: This MUST be a Short Link (section 2.2.1.4) Type identifier specifying the source of the membership group.

SourceReference: A string used to distinguish the various membership groups within a particular source specified by @Source. The Unicode String Trim operation MUST have been performed on the input value. When the @Source input parameter specifies a distribution list sourced membership group the value MUST be the membership group's DN. When the @Source input parameter specifies a site sourced membership group the value MUST be the site identifier.

Url: A URI for the membership group.

MemberCount: This MUST be a count of Members in this membership group.

LastUpdate: This MUST be a UTC Value specifying the last time membership_updateGroup (Section 3.1.5.8) was successfully called.

DSGroupType: A group type identifier from the data source (such as Active Directory, LDAP, and others).

DataSource: The domain name of the group.

2.2.4.38 DataTypeList

The **DataTypeList** returns the list of profile data types defined in the system. The **DataTypeList** is presented in ascending order by FriendlyTypeName column.

The **DataTypeList** result set is defined using T-SQL syntax as follows:

```
DataTypeID int,
DataTypeName nvarchar(100),
Name nvarchar(500),
FriendlyTypeName nvarchar(500),
MaxCharCount int,
IsFulltextIndexable bit,
AllowMultiValue bit,
BlobType tinyint,
IsEmail bit,
IsURL bit,
IsPerson bit,
IsHTML bit,
AllowTaxonomic bit,
```

```
PartitionID uniqueidentifier,
```

DataTypeID: The unique numeric identifier of the Profile Data Type.

DataTypeName: Contains the underlying SQL data for this Profile Data Type. Valid values are defined in the enumeration **TYPE:SQLDataType** as specified in [MS-UPSIMP], section 2.2.4.

Name: Contains the unique name of the Profile Data Type.

FriendlyTypeName: Contains the description as it appears in the user interface of the current Profile Data Type.

MaxCharCount: Contains the maximum input length for the value of a property associated with this Profile Data Type. If the value is not NULL, the client UI MUST limit input length to this value for properties associated with this Profile Data Type.

IsFulltextIndexable: Contains a bit value which if 1 indicates the data type is eligible for Full-Text Indexing. The server MUST ignore this value. This value MUST be 0 to indicate the data type is not eligible for Full-Text Indexing.

AllowMultiValue: Contains a bit that if 1 indicates that a profile property associated with this Profile Data Type supports multiple values. This value MUST be 0 to indicate that a profile property associated with this Profile Data Type does not support multiple values.

BlobType: Contains a **TYPE:ProfilePropertyBlobType** (as specified in [MS-UPSIMP], section 2.2.6) enumeration value indicating how the binary large object (BLOB) is stored in the database.

IsEmail: Contains a bit that if 1 indicates the value for the property associated with this Profile Data Type MUST be a valid e-mail address. This value MUST be 0 to indicate the value for the property associated with this Profile Data Type is not a valid e-mail address.

IsURL: Contains a bit that if 1, indicates the value for a profile property associated with this Profile Data Type MUST contain a value in URL format. If 1, the client MUST map the host name to match the request. This value MUST be 0 to indicate the value for a profile property associated with this Profile Data Type does not contain a value in URL format.

IsPerson: Contains a bit that if 1 indicates the value for the property associated with this **Profile Data Type**, if non-NULL, MUST be a valid Domain User Account that has been validated against the domain controller of the referenced Account Domain. This value MUST be 0 to indicate the value for the property associated with **Profile Data Type** is not a valid Domain User Account that has been validated against the domain controller of the referenced Account Domain.

IsHTML: Contains a bit that if 1 indicates the value this Profile Data Type stores is fully formatted HTML text data. This value MUST be 0 to indicate the value this Profile Data Type stores is not fully formatted HTML text data.

AllowTaxonomic: Contains a bit, that if 1 indicates that properties based on this data type can be used as a taxonomy-backed property and can be associated with a taxonomy term set. This value MUST be 0 to indicate that the properties based on this data type cannot be used as a taxonomy backed property and cannot be associated with a taxonomy term set.

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty. This MUST be equal to the **PartitionID** passed into profile_GetDataTypeList.

2.2.4.39 UpdateUserProfileDataResult

The **UpdateUserProfileDataResult** returns Error information in case of failure. In the case of a success, the result set contains the count of properties updated and, if a new user was created, it contains new user identification information for the new user that was created.

The **UpdateUserProfileDataResult** result set is defined using T-SQL syntax as follows:

```
ERROR int,
XMLUpdateUserErr int,
XMLUpdatePropertyErr int,
UpdatePropertyCount int,
NEWUSERGUID uniqueidentifier,
NEWRECORDID bigint,
```

ERROR: Contains the SQL process error not including the XML document error. This MUST be set if there is a SQL error. The output value MUST be set to 0 upon successful execution.

XMLUpdateUserErr: The count of the users that were not able to be updated by this procedure.

XMLUpdatePropertyErr: The count of properties that were not able to be updated by this procedure.

UpdatePropertyCount: The count of properties which have been successfully updated by this procedure.

NEWUSERGUID: If a new user is successfully created then this MUST be the new user GUID else this MUST be NULL.

NEWRECORDID: If a new user is successfully created then this MUST be the Record ID of the user else this MUST be NULL.

2.2.4.40 UpdateProfileDisplayResult

The **UpdateProfileDisplay** result set specifies if the properties and sections could be updated and the number of properties and sections that could not be updated. The **UpdateProfileDisplay** result set MUST return only 1 row.

The **UpdateProfileDisplay** result set is defined using T-SQL syntax as follows:

```
XMLProfileErr int,
UpdateItemCount int,
XMLUpdateItemErr int,
```

XMLProfileErr: This value MUST be set to 1 if the user profile (whose ProfileName value is set to "UserProfile") could not be opened for update; otherwise it MUST be set to 0

UpdateItemCount: This value MUST be ignored by the client.

XMLUpdateItemErr: Contains the number of properties and sections that could not be updated.

2.2.4.41 UpdatePropertyResult

The **UpdatePropertyResult** result set contains the count of properties that were successfully removed, added, or updated, as well as the count of errors related to removing, adding, or updating properties. The **UpdatePropertyResult** result set MUST contain only 1 row.

The **UpdatePropertyResult** result set is defined using T-SQL syntax for the stored procedure as follows:

```
ERROR int,
RemovedPropertyCount int,
XMLRemovePropertyErr int,
UpdatePropertyCount int,
XMLUpdatePropertyErr int,
```

ERROR: Contains the value of the first error encountered during processing which MUST be a value in the following table:

Value	Description
0	The properties were removed, updated or added successfully, and no error occurred.
1	The node of the *@PropertyURIPrefix* parameter was NULL, where the *@ PropertyURIPrefix* is a string prefix for the Property URI.
3	The value of the **PropertyType** attribute in the *@RemovePropertyList* XML is not a Property (Section 2.2.1.17) Type value.
4	The value of the **PropertyName** attribute in the *@RemovePropertyList* XML could not be found, or the specified property is a reserved system property.
50	The property could not be deleted because it is used in Audience.
22	The value of the **PropertyType** attribute in the *@ UpdatePropertyList* XML is not a Property (Section 2.2.1.17) Type value.
23	The value of the **PropertyName** attribute in the *@UpdatePropertyList* XML could not be found during an update operation. The value of the **DataTypeID** attribute was not specified in the *@UpdatePropertyList* XML during an add operation.
60	The value of the **PropertyName** attribute in the *@UpdatePropertyList* XML already exists as Profile property when adding a profile subtype property.
61	The value of the **PropertyName** attribute in the *@UpdatePropertyList* XML already exists as profile subtype property when adding a profile subtype.
65	Unable to add the profile subtype property from *@UpdatePropertyList* XML.
70	The value of the **PropertyName** attribute in the *@UpdatePropertyList* XML already exists as core property when adding a Profile property.
71	The value of the **PropertyName** attribute in the *@UpdatePropertyList* XML already exists as profile subtype property when adding a Profile property.
75	Unable to add the Profile property from *@UpdatePropertyList* XML.
81	The value of the **PropertyName** attribute in the *@UpdatePropertyList* XML already exists as core property when adding a core property or unable to add the core property from

Value	Description
	@UpdatePropertyList XML.
92	Unable to update the PrivacyPolicy, DefaultPrivacy, UserOverridePrivacy, or NewPropertyPolicy (Section 2.2.6.4.3.1.2) when updating a profile subtype property.
93	Unable to update the profile subtype property from *@UpdatePropertyList* XML.
96	Unable to update the core property from *@UpdatePropertyList* XML.
24	The value of the **PropertyName** attribute in the *@UpdatePropertyList* XML was not specified or already exists.
5001	The choice list values specified were not found.

RemovedPropertyCount: This value MUST be the number of properties removed.

XMLRemovePropertyErr: MUST be set to "0" if the property was successfully deleted; otherwise, XMLRemovePropertyERR MUST be set to "1".

UpdatePropertyCount: This value MUST be the number of properties updated or newly added.

XMLUpdatePropertyErr: MUST be set to "0" if the property was successfully added or updated; otherwise, XMLUpdatePropertyERR MUST be set to "1".

2.2.4.42 AudienceResult

The **Audience** result set contains the name of each **Audience** that is currently using the property specified for deletion.

The **Audience** result set is defined using T-SQL syntax for the stored procedure as follows:

```
OrgleName nvarchar(200),
```

OrgleName: Contains the name of the Audience.

2.2.4.43 OrganizationIds

The **OrganizationIds** result set contains a list of organization record identifiers.

The **OrganizationIds** result set is defined using T-SQL syntax for the stored procedure as follows:

```
RecordID bigint,
```

RecordID: Contains an organization record identifier.

2.2.4.44 OrganizationMemberships

The **OrganizationMemberships** result set contains a list of the members which belong immediately to a single organization. Members of descendant organizations are not included.

The **OrganizationMemberships** result set is defined using T-SQL syntax for the stored procedure as follows:

```
RecordId bigint,
NTName nvarchar(400),
MembershipType smallint,
UserID uniqueidentifier,
PreferredName nvarchar(256),
ProfileSubtypeID int,
PictureURL nvarchar(max),
Email nvarchar(256),
SipAddress nvarchar(250),
PhoneticDisplayName nvarchar(256),
DisplayOrder int,
PersonTitle nvarchar(150),
StatusNotes nvarchar(512),
StatusNotesMRU nvarchar(3200),
EditPermission int,
RowID bigint,
```

RecordId: The user profile record identifier. MUST NOT be NULL or empty.

NTName: A Security Account Manager (SAM) user name for the user associated with the user profile represented by the **RecordId**. MUST NOT be NULL or empty.

MembershipType: The membership type.

The value MUST exist in the following table:

Value	Description
1	Member membership type
2	Leader membership type

UserID: A GUID that represents the user profile.

PreferredName: The name of the entity as defined in the user profile.

ProfileSubtypeID: A profile subtype identifier. MUST NOT be NULL.

PictureURL: URL to a picture for the entity the user profile specifies.

Email: An e-mail address for the entity the user profile specifies.

SipAddress: A Session Initiation Protocol (SIP) address for the entity the user profile specifies.

PhoneticDisplayName: This column MUST be ignored.

DisplayOrder: This column MUST be ignored.

PersonTitle: Contains the title of the user specified by the current record.

StatusNotes: This column MUST be ignored.

StatusNotesMRU: This column MUST be ignored.

EditPermission: This column MUST be ignored.

RowID: This column MUST be ignored.

2.2.4.45 RootOrganization

The **RootOrganization** result set contains information about each root organization. Each result represents a single organization.

The **RootOrganization** result set is defined using T-SQL syntax for the stored procedure as follows:

```
OrganizationID bigint,
ProfileSubtypeID int,
OrganizationDisplayName nvarchar(400),
OrganizationGuid uniqueidentifier,
ParentType smallint,
ParentRecordID bigint,
ChildrenCount int,
```

OrganizationID: Contains the organization record identifier. This value MUST NOT be NULL.

ProfileSubtypeID: Contains the value identifier of the profile subtype.

OrganizationDisplayName: Contains the string display name for the organization.

OrganizationGuid: Contains the organization unique identifier. This value MUST NOT be NULL or empty.

ParentType: Contains the organization parent type. The value MUST be listed in the following table.

Value	Description
1	User
2	Organization
3	Group

ParentRecordID: Contains the record identifier of the parent.

ChildrenCount: Contains the count of direct child organizations.

2.2.4.46 ContactEntry

The ContactEntry result set contains information about the contact entry,

The ContactEntry result set is defined using T-SQL syntax for the stored procedure as follows:

```
SourceObjectDN nvarchar(2048),
ProfileID nvarchar(400),
PreferredName nvarchar(256),
```

SourceObjectDN: The LDAP standard DN [RFC2251] of the source object.

ProfileID: An identifier for the profile in the user profile store.

PreferredName: The display name for the profile.

2.2.4.47 EnumUsersFull

The **EnumUsersFull** result set MUST contain 0 or more rows. For a record to be included in the result set, it MUST NOT have a NULL login name

The **EnumUsersFull** result set is defined using T-SQL syntax for the stored procedure as follows:

```
RecordID bigint,
NTName nvarchar(400),
```

RecordID: Contains the record identifier associated with the specified user. This MUST NOT be NULL.

NTName: A Security Account Manager (SAM) user name for the entity specified by the user profile.

2.2.4.48 Siblings

The **Siblings** result set contains a list of organization record identifiers.

The **Siblings** result set is defined using T-SQL syntax for the stored procedure as follows:

```
RecordID bigint,
```

RecordID: Contains an organization record identifier.

2.2.4.49 Ancestors

The **Ancestors** result set contains a list of organization record identifiers.

The **Ancestors** result set is defined using T-SQL syntax for the stored procedure as follows:

```
RecordID bigint,
```

RecordID: Contains an organization record identifier.

2.2.4.50 OrganizationData

The **OrganizationData** result set contains information about an organization. This result set MUST contain exactly one row.

The **OrganizationData** result set is defined using T-SQL syntax for the stored procedure as follows:

```
RecordID bigint,
ProfileSubtypeID int,
ChildrenCount int,
```

RecordID: Contains an organization **record identifier**.

ProfileSubtypeID: Contains the value identifier of the profile subtype.

ChildrenCount: Contains the count of child organizations belonging to the organization whose record identifier is the value of **RecordID**.

2.2.4.51 OrganizationProperties

The **OrganizationProperties** result set contains details about each property of an organization. Each result represents exactly one property.

The **OrganizationProperties** result set is defined using T-SQL syntax for the stored procedure as follows:

```
RecordID bigint,
PropertyID bigint,
PropertyVal sql_variant,
SecondaryVal sql_variant,
Text nvarchar(max),
OrderRank int,
Privacy int,
PartitionID uniqueidentifier,
PropertyName nvarchar(250),
```

RecordID: This value MUST be ignored by the protocol client.

PropertyID: This value MUST be ignored by the protocol client.

PropertyVal: Contains the value of the property.

SecondaryVal: This value MUST be ignored by the protocol client.

Text: This value MUST be ignored by the protocol client.

OrderRank: This value MUST be ignored by the protocol client.

Privacy: This value MUST be ignored by the protocol client.

PartitionID: This value MUST be ignored by the protocol client.

PropertyName: Contains the name of the property.

2.2.4.52 BulkOrganizationMemberships

The **BulkOrganizationMemberships** result set contains a list of organization memberships. Each result MUST contain exactly one membership.

The **BulkOrganizationMemberships** result set is defined using T-SQL syntax for the stored procedure as follows:

```
RecordId bigint,
OrganizationGuid uniqueidentifier,
```

RecordId: Contains the user profile record identifier.

OrganizationGuid: Contains the organization unique identifier.

2.2.4.53 BulkOrganizationInformation

The **BulkOrganizationInformation** result set contains information about a list of organizations. Each result MUST contain information about exactly one organization.

The **BulkOrganizationInformation** result set is defined using T-SQL syntax for the stored procedure as follows:

```
ProfileSubtypeID int,
OrganizationDisplayName nvarchar(400),
OrganizationId bigint,
OrganizationGuid uniqueidentifier,
ParentType smallint,
ParentRecordId bigint,
ChildrenCount int,
```

ProfileSubtypeID: Contains the value identifier of the profile subtype.

OrganizationDisplayName: Contains the string name of the organization. This value MUST NOT be NULL.

OrganizationId: Contains the organization record identifier. This value MUST NOT be NULL or empty.

OrganizationGuid: Contains the organization unique identifier. This value MUST NOT be NULL or empty.

ParentType: Contains the organization parent type. The value MUST be listed in the following table.

Value	Description
1	User
2	Organization
3	Group

ParentRecordId: Contains the record identifier of the organization's parent object.

ChildrenCount: An integer which MUST be 0.

2.2.4.54 QueryMySiteDeletionSchedule

The **QueryMySiteDeletionSchedule** result set contains information about a list of notification status entries.

The **QueryMySiteDeletionSchedule** result set is defined using T-SQL syntax for the stored procedure as follows:

```
SiteID uniqueidentifier,
DisplayName nvarchar(400),
Email nvarchar(256),
NotificationStatus tinyint,
Created datetime,
PartitionID uniqueidentifier,
```

SiteID: A GUID used to specify the site. This value MUST NOT by null or empty.

DisplayName: A descriptive name for the site.

Email: The e-mail address of the site owner.

NotificationStatus: The notification status of the site.

Created: A UTC Value which represents the date when the deletion request was created.

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

2.2.4.55 profile_GetOrganizationMembershipForUser.ResultSet0

The **profile_GetOrganizationMembershipForUser.ResultSet0** result set contains a list of organization memberships for which a user is a direct member.

The **profile_GetOrganizationMembershipForUser.ResultSet0** result set is defined using T-SQL syntax as follows:

```
ProfileSubtypeID int,
OrganizationID bigint,
OrganizationDisplayName nvarchar(400),
OrganizationGuid uniqueidentifier,
ParentType smallint,
ParentRecordID bigint,
ChildrenCount int,
```

ProfileSubtypeID: Contains the value identifier of the profile subtype. MUST NOT be NULL.

OrganizationID: Contains the organization record identifier. MUST NOT be NULL.

OrganizationDisplayName: Contains the string display name for the organization.

OrganizationGuid: Contains the organization unique identifier. MUST NOT be NULL or empty.

ParentType: Contains the organization parent type. The value MUST be listed in the following table.

Value	Description
1	User
2	Organization
3	Group

ParentRecordID: Contains the record identifier of the parent.

ChildrenCount: Contains the count of direct child organizations. MUST NOT be NULL.

2.2.4.56 profile_GetOrganizationMembershipForUser.ResultSet1

The **profile_GetOrganizationMembershipForUser.ResultSet1** result set contains a list of organization memberships for which a user is a member, including organizations which are parents of organizations for which the user is a direct member.

The **profile_GetOrganizationMembershipForUser.ResultSet1** result set is defined using T-SQL syntax as follows:

```
ProfileSubtypeID int,
OrganizationID bigint,
OrganizationDisplayName nvarchar(400),
OrganizationGuid uniqueidentifier,
ParentType smallint,
ParentRecordID bigint,
ChildrenCount int,
```

ProfileSubtypeID: Contains the value identifier of the profile subtype. MUST NOT be NULL or empty.

OrganizationID: Contains the organization record identifier. MUST NOT be NULL or empty.

OrganizationDisplayName: Contains the string display name for the organization.

OrganizationGuid: Contains the organization unique identifier. MUST NOT be NULL or empty.

ParentType: Contains the organization parent type. The value MUST be listed in the following table.

Value	Description
1	User
2	Organization
3	Group

ParentRecordID: Contains the record identifier of the parent.

ChildrenCount: Contains the count of direct child organizations. MUST NOT be NULL or empty.

2.2.4.57 profile_GetOrganizationMembershipForUser.ResultSet2

The **profile_GetOrganizationMembershipForUser.ResultSet2** result set contains a list of organization memberships for a single user. This result set MUST be empty.

The **profile_GetOrganizationMembershipForUser.ResultSet2** result set is defined using T-SQL syntax as follows:

```
ProfileSubtypeID int,
OrganizationID bigint,
OrganizationDisplayName nvarchar(400),
OrganizationGuid uniqueidentifier,
ParentType smallint,
ParentRecordID bigint,
ChildrenCount int,
```

ProfileSubtypeID: Contains the value identifier of the profile subtype. MUST NOT be NULL or empty.

OrganizationID: Contains the organization record identifier. MUST NOT be NULL or empty.

OrganizationDisplayName: Contains the string display name for the organization.

OrganizationGuid: Contains the organization unique identifier. MUST NOT be NULL or empty.

ParentType: Contains the organization parent type. The value MUST be listed in the following table.

Value	Description
1	User
2	Organization
3	Group

ParentRecordID: Contains the record identifier of the parent.

ChildrenCount: Contains the count of direct child organizations. MUST NOT be NULL or empty.

2.2.4.58 profile_GetOrganizationMembershipForUser.ResultSet3

The **profile_GetOrganizationMembershipForUser.ResultSet3** result set contains a list of organization memberships for which a specified user is the organization leader.

The **profile_GetOrganizationMembershipForUser.ResultSet3** result set is defined using T-SQL syntax as follows:

```
OrganizationID bigint,
OrganizationDisplayName nvarchar(400),
OrganizationGuid uniqueidentifier,
ParentType smallint,
ParentRecordID bigint,
ChildrenCount int,
```

OrganizationID: Contains the organization record identifier. MUST NOT be NULL or empty.

OrganizationDisplayName: Contains the string display name for the organization.

OrganizationGuid: Contains the organization unique identifier. MUST NOT be NULL or empty.

ParentType: Contains the organization parent type. The value MUST be listed in the following table.

Value	Description
1	User
2	Organization
3	Group

ParentRecordID: Contains the record identifier of the parent.

ChildrenCount: Contains the count of direct child organizations. MUST NOT be NULL or empty.

2.2.4.59 GetTopOrganizationEvent

The **GetTopOrganizationEvent** result set MUST contain one result identifying the most recent organization change event prior to the specified **@MinEventTime** or **@MinEventId** parameter. If no events have occurred yet, then this result set MUST contain zero results.

The **GetTopOrganizationEvent** result set is defined using T-SQL syntax as follows:

```
EventTime datetime,
EventId bigint,
```

EventTime: Contains the UTC date and time when the organization change event occurred.

EventId: Contains the unique identifier for the organization change event.

2.2.4.60 GetOrganizationEvents

The **GetOrganizationEvents** result set returns the organization change events for all existing organizations up to a maximum of 1000 records.

The **GetOrganizationEvents** result set is defined using T-SQL syntax as follows:

```
EventId bigint,
RecordId bigint,
OrgID uniqueidentifier,
ChangeType int,
EventTime datetime,
OldValue sql_variant,
NewValueData sql_variant,
NewValueChecksum int,
SecondaryValue sql_variant,
ObjectType int,
ItemSecurity int,
ChangedPropertyId bigint,
ChangedUserId bigint,
ChangedSourceId uniqueidentifier,
PartitionID uniqueidentifier,
ProfileSubtypeID int,
OrganizationID bigint,
OrganizationDisplayName nvarchar(400),
OrganizationGuid uniqueidentifier,
ParentType smallint,
ParentRecordID bigint,
```

EventId: Contains the UTC date and time when the organization change event occurred.

RecordId: Contains the record identifier of the affected organization. This value MUST NOT be NULL.

OrgID: Contains the unique identifier of the affected organization.

ChangeType: Contains an identifier that represents the type of change. This value MUST be listed in the following table.

Value	Description
0x1	Add
0x2	Modify
Ox4	Delete

EventTime: Contains the UTC date and time when the organization change event occurred.

OldValue: This value MUST be ignored.

NewValueData: If the value of **ChangeType** is Delete and the value of **ObjectType** is Organization Membership, this value MUST contain the organization membership type. This value MUST be listed in the following table.

Value	Description
0x1	Member membership type
0x2	Leader membership type

If the value of **ChangeType** is not Delete, or the value of **ObjectType** is not Organization Membership, this value MUST be ignored.

NewValueChecksum: This value MUST be ignored.

SecondaryValue: This value MUST be ignored.

ObjectType: Contains an identifier that represents the type of object changed. This value MUST be listed in the following table.

Value	Description
0x800	Organization Profile
0x1000	Organization Membership

ItemSecurity: Contains an identifier that represents the privacy level of the change. The value MUST be listed in the following table.

Value	Description
0x1	Public
0x2	Contacts
0x4	Organization
0x8	Manager
0xF	Private
0x40000000	Not Set

ChangedPropertyId: This value MUST be ignored.

ChangedUserId: Contains the record identifier of the user profile associated with an organization membership change. If the value of **ObjectType** is not 0x1000, this value MUST be ignored.

ChangedSourceId: This value MUST be ignored.

PartitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

ProfileSubtypeID: This value MUST be ignored.

OrganizationID: Contains the record identifier of the affected organization.

OrganizationDisplayName: Contains the string display name of the organization.

OrganizationGuid: Contains the unique identifier of the affected organization.

ParentType: Contains the organization parent type. The value MUST be listed in the following table.

Value	Description
1	User
2	Organization
3	Group

ParentRecordID: Contains the record identifier of the affected organization parent or NULL if the organization does not have a parent.

2.2.4.61 GetOrganizationEventsForRecordId

The **GetOrganizationEventsForRecordId** result set contains the organization change events for the specified organization, up to a maximum of 1000 records.

The **GetOrganizationEventsForRecordId** result set is defined using T-SQL syntax as follows:

```
EventId bigint,
RecordId bigint,
OrgID uniqueidentifier,
ChangeType int,
EventTime datetime,
OldValue sql_variant,
NewValueData sql_variant,
NewValueChecksum int,
SecondaryValue sql_variant,
ObjectType int,
ItemSecurity int,
ChangedPropertyId bigint,
ChangedUserId bigint,
ChangedSourceId uniqueidentifier,
PartitionID uniqueidentifier,
```

EventId: Contains the UTC date and time when the organization change event occurred.

RecordId: Contains the record identifier of the affected organization. This value MUST NOT be NULL.

OrgID: Contains the unique identifier of the affected organization.

ChangeType: Contains an identifier that represents the type of change. This value MUST be listed in the following table.

Value	Description
0x1	Add

Value	Description
0x2	Modify
Ox4	Delete

EventTime: Contains the UTC date and time when the organization change event occurred.

OldValue: This value MUST be ignored.

NewValueData: If the value of **ChangeType** is Delete and the value of **ObjectType** is Organization Membership, this value MUST contain the organization membership type. This value MUST be listed in the following table.

Value	Description
0x1	Member membership type
0x2	Leader membership type

NewValueChecksum: This value MUST be ignored.

SecondaryValue: This value MUST be ignored.

ObjectType: Contains an identifier that represents the type of object changed. This value MUST be listed in the following table.

Value	Description
0x800	Organization Profile
0x1000	Organization Membership

ItemSecurity: Contains an identifier that represents the privacy level of the change. The value MUST be listed in the following table.

Value	Description
0x1	Public
0x2	Contacts
0x4	Organization
0x8	Manager
0xF	Private
0x40000000	Not Set

ChangedPropertyId: This value MUST be ignored.

ChangedUserId: Contains the record identifier of the user profile associated with an organization membership change. If the value of **ObjectType** is not 0x1000, this value MUST be ignored.

ChangedSourceId: This value MUST be ignored.

PartitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

2.2.4.62 UserMemberships

The **UserMemberships** result set returns a user's membership data.

The **UserMemberships** result set is defined using T-SQL syntax as follows:

```
ItemId bigint,
ItemSecurity int,
GroupType tinyint,
GroupTitle nvarchar(400),
PolicyId uniqueidentifier,
MemberGroupId bigint,
DisplayName nvarchar(250),
MailNickName nvarchar(250),
Description nvarchar(1500),
Source uniqueidentifier,
SourceReference nvarchar(2048),
Url nvarchar(2048),
MemberCount bigint,
DSGroupType bigint,
DataSource nvarchar(400),
LastUpdate datetime,
SID varbinary(512),
```

ItemId: Contains the unique identifier for the Membership-user pair.

ItemSecurity: MUST be a Privacy (section 2.2.1.2) Type value that defines security for the membership of the specified user.

GroupType: MUST be a Short Group (section 2.2.1.3) Type value specifying the user membership grouping.

GroupTitle: Contains the group title for the membership associated with the specified user.

PolicyId: Contains a Policy Link (section 2.2.1.5) Type identifier specifying the membership type.

MemberGroupId: Contains the membership group identifier for the user.

DisplayName: Contains the user display name for the Membership Group

MailNickName: Contains the mail nickname for the Membership Group.

Description: Contains the description for the Membership Group.

Source: Contains the Short Link (Section 2.2.1.4) Type identifier for the Membership Group.

SourceReference: Contains a string used to distinguish the various membership groups within a particular source specified by @Source.

Url: Contains the URL for the Membership Group.

MemberCount: Contains the members count for the Membership Group.

DSGroupType: A group type identifier from the data source (such as Active Directory, LDAP, and others).

DataSource: Contains the domain name of the group.

LastUpdate: A UTC Value specifying the last time the Membership Group was updated.

SID: Contains the **security identifier** for the Membership group.

2.2.4.63 GetLeaders

The GetLeaders result set contains leaders in the partition (1) identified by @partitionID.

The GetLeaders result set is defined using T-SQL syntax as follows:

```
UserName nvarchar(400),
ProfileUserName nvarchar(400),
ProfileManager nvarchar(400),
ReportCount int,
```

UserName: The Security Account Manager (SAM) user name for the leader. This MUST NOT be NULL.

ProfileUserName: The Security Account Manager (SAM) user name for the leader. This MUST NOT be NULL.

ProfileManager: The Security Account Manager (SAM) user name for the leader's manager.

ReportCount: The number of profiles with the leader as direct manager.

2.2.5 Tables and Views

2.2.5.1 DNLookup

This table contains the distinguished names (DN) for all users and groups.

```
DNId bigint NOT NULL,
DN nvarchar(2048) NOT NULL,
cs_DN int NOT NULL,
ObjectType nvarchar(20) NULL,
RecordId bigint NULL,
PartitionID uniqueidentifier NOT NULL,
```

DNId: A 64-bit record identifier for the DNLookup table.

DN: The LDAP standard DN [RFC2251] of the object.

cs_DN: An integer which specifies the checksum of the DN.

ObjectType: A string which specifies the type of object the DN identifies.

RecordId: The record identifier of the object. For users this is the user profile record identifier and for groups this is the member group identifier.

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

2.2.5.2 UserProfile_Full

The **UserProfile_Full** table is used to describe a set of intermediate data used by several stored procedures.

```
RecordID bigint NOT NULL,
DocID int NULL,
UserID uniqueidentifier NOT NULL,
NTName nvarchar(400) NOT NULL,
PreferredName nvarchar(256) NULL,
Email nvarchar(256) NULL,
SID varbinary(512) NULL,
Manager nvarchar(400) NULL,
SipAddress nvarchar(250) NULL,
LastUpdate datetime NOT NULL,
LastUserUpdate datetime NOT NULL,
LastImported datetime NULL,
bDeleted tinyint NOT NULL,
DataSource nvarchar(155) NULL,
MasterRecordID bigint NOT NULL,
ProfileSubtypeID int NOT NULL,
DSGuid uniqueidentifier NULL,
DNId bigint NULL,
PictureUrl nvarchar(max) NULL,
PartitionID uniqueidentifier NOT NULL,
```

RecordID: Record identifier of the profile.

DocID: Document identifier of the profile.

UserID: GUID identifier of the user.

NTName: Domain user name.

PreferredName: The name of the entity as defined in the profile. Contains the display name.

Email: E-mail address of the profile

SID: **Security identifier (SID)** of the profile.

Manager: Contains the value of the manager profile property.

SipAddress: Contains the SIP address for the profile.

LastUpdate: Contains the UTC value specifying the last time the profile was updated.

LastUserUpdate: Contains the UTC value specifying the last time the profile was updated by the user.

LastImported: Contains the UTC value specifying the last time the profile was imported.

bDeleted: If the profile is marked as deleted this MUST be 1, otherwise it MUST be 0.

DataSource: Contains the value specifying the source of the profile.

MasterRecordID: Contain the value identifier of the master account.

ProfileSubtypeID: Contains the value identifier of the profile subtype.

DSGuid: Contains the GUID identifier of the profile object in its original data source.

DNId: Identifier of the DN.

PictureUrl: Contains the picture URI profile property value for the entity the profile specifies.

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

2.2.5.3 MembershipNonRecursive

This table contains the membership groups that are members of another membership group. In this table the direct parent-child relationship of the membership groups is stored.

```
GroupId bigint NULL,
ParentGroupId bigint NOT NULL,
PartitionID uniqueidentifier NOT NULL,
```

GroupId: A 64-bit integer which identifies the membership group member.

ParentGroupId: A 64-bit integer which identifies the membership group that contains the member.

PartitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

2.2.5.4 UserProfileValue

This table contains the user profile property values.

```
RecordID bigint NOT NULL,
PropertyID bigint NOT NULL,
PropertyVal sql_variant NULL,
SecondaryVal sql_variant NULL,
Text ntext NULL,
OrderRank int NULL,
Privacy int NULL,
PartitionID uniqueidentifier NOT NULL,
```

RecordID: A 64-bit integer which specifies the record identifier of the user profile this property is associated with.

PropertyID: A 64-bit integer which specifies the identifier of the user profile property.

PropertyVal: A variant value which is the value of the user profile property.

SecondaryVal: A variant value which contains the taxonomy value of a user property. Only populated for taxonomy-backed properties.

Text: A string associated with the user profile property.

OrderRank: An integer which specifies the order of display of the user profile property in the user interface.

Privacy: A Privacy Type (section 2.2.1.6) value of the user profile property.

PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

2.2.5.5 MemberGroup

This table contains the membership groups.

```
Id bigint NOT NULL,
DisplayName nvarchar(250) NULL,
MailNickName nvarchar(250) NULL,
Description nvarchar(1500) NULL,
Source uniqueidentifier NOT NULL,
SourceReference nvarchar(2048) NULL,
cs_SourceReference int NOT NULL,
Url nvarchar(2048) NULL,
SID varbinary(512) NULL,
MemberCount bigint NOT NULL,
LastUpdate datetime NOT NULL,
AllWebsSynchID int NULL,
Type tinyint NOT NULL,
UserCreated bit NOT NULL,
DSGroupType bigint NULL,
DataSource nvarchar(400) NULL,
PartitionID uniqueidentifier NOT NULL,
```

Id: A 64-bit identifier of the membership group.

DisplayName: A string which specifies the display name of the membership group.

MailNickName: A string which specifies the e-mail name associated with the membership group.

Description: A string which contains the description of the membership group.

Source: A Short Link Type (section 2.2.1.4) which specifies the source type of membership group.

SourceReference: A string which specifies the LDAP standard DN [RFC2251] of the membership group.

cs_SourceReference: An integer which is the checksum of the Source column.

Url: A universal resource locator of a Web page associated with this membership group.

SID: A byte array containing the SID associated with the membership group.

MemberCount: The number of members of the group.

LastUpdate: The date and time of the last update to this membership group record.

AllWebsSynchID: The identifier of the web that the group belongs to.

Type: A Group Type (section 2.2.1.1) which identifies the type of membership group.

UserCreated: A bit which is zero if the membership group was created by importing the membership group or one if the membership group was created by a user.

DSGroupType: A group type identifier from the data source (such as Active Directory, LDAP, and others).

DataSource: The domain name of the group.

PartitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

2.2.5.6 MembershipRecursive

This table contains the membership groups which are members of another membership group.

```
GroupId bigint NOT NULL,
ParentGroupId bigint NOT NULL,
PartitionID uniqueidentifier NOT NULL,
```

GroupId: A 64-bit **integer** that identifies the membership group member.

ParentGroupId: A 64-bit **integer** that identifies the membership group that contains the member.

PartitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

2.2.5.7 Tenants

This table stores information about the partitions (1) in the user profile store.

```
PartitionID uniqueidentifier NOT NULL,
LastModifiedTime datetime NOT NULL,
CanonicalMySitePortalUrl nvarchar(2084) NOT NULL,
PreviousMySitePortalUrl nvarchar(2084) NOT NULL,
CanonicalSearchCenterUrl nvarchar(2084) NOT NULL,
PeopleResultsScope int NOT NULL,
DocumentResultsScope int NOT NULL,
DefaultRssFeed nvarchar(2084) NOT NULL,
MySiteEmailSenderName nvarchar(max) NULL,
SynchronizationOU nvarchar(max) NULL,
ProfileMasterCacheVersion int NOT NULL,
SerializedUserAcl nvarchar(max) NULL,
DataCacheVersion int NOT NULL,
```

PartitionID: A GUID identifying the partition (1) in the user profile store.

LastModifiedTime: The UTC date and time of the last modification made to the record.

CanonicalMySitePortalUrl: A string specifying the canonical **URL** for the partition's (1) my site portal.

PreviousMySitePortalUrl: A string specifying the canonical URL for the partition's (1) my site portal that personal sites are currently configured to use.

CanonicalSearchCenterUrl: A string specifying the canonical URL for the partition's (1) search center.

PeopleResultsScope: An integer value specifying the scope for the people search results.

DocumentResultsScope: An integer value specifying the scope for document search results.

DefaultRssFeed: A string specifying the default RSS feed URL.

MySiteEmailSenderName: A string specifying the sender e-mail address of my site e-mails.

SynchronizationOU: A string specifying the organizational unit of the partition (1) for synchronization.

ProfileMasterCacheVersion: An integer value specifying the version number of the internal profile master cache.

SerializedUserAcl: A string value representing an XML document specifying the serialized form of the user **access control list (ACL)** for the partition (1). This MUST be a valid XML document instance of **acl** type defined in [MS-UPASP] section 2.2.6.3.

DataCacheVersion: An integer value specifying the version number of the internal data cache.

2.2.6 XML Structures

The syntax of the definitions in this section uses XML Schema as defined in [XML10], [XMLNS], [XMLINFOSET], [XMLSCHEMA1], and [XMLSCHEMA2].

2.2.6.1 Namespaces

This specification does not define any namespaces.

2.2.6.2 Simple Types

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

2.2.6.3 Complex Types

The following table summarizes the set of common XML Schema complex type definitions defined by this specification. XML Schema complex type definitions that are specific to a particular operation are described with the operation.

Complex Type	Description
ItemsXml	Complex Type that contains shared list properties.

2.2.6.3.1 ItemsXml

The ItemsXml type contains data about shared list entries, as specified in the following example:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:element name="List" minOccurs="0">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Item" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:attribute name="ItemId" type="xs:long" />
                    <xs:attribute name="Title" type="xs:string" />
                    <xs:attribute name="Url" type="xs:string" />
                    <xs:attribute name="Owner" type="xs:string"/>
                    <xs:attribute name="TargetTo" type="xs:string" minOccurs="0"/>
                    <xs:attribute name="Int1" type="xs:long" minOccurs="0"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

ItemId: Contains the identifier of the shared list item. This MUST NOT be NULL.

Title: Contains the user-friendly name of the shared list item. This MUST NOT be NULL.

Url: Contains a URL link of the shared list item. This MUST NOT be NULL.

Owner: This MUST contain either the NTName of the owner of the shared list item, or an empty string. This MUST NOT be NULL.

TargetTo: Contains the audience information for the shared list item. The field MUST either be empty or contain a triplet of comma-delimited audiences, comma-delimited distribution Lists, and comma-delimited groups. The triplet separator MUST be ";;".

Int1: Contains the shared item Type information for the shared list item.

2.2.6.4 Elements

The following table summarizes the set of common XML Schema element definitions defined by this specification. XML Schema element definitions that are specific to a particular operation are described with the operation.

Element	Description
UpdateList	Stores an ordered list of properties and sections.
UpdatePropertyLoc	Stores user display names and descriptions of a property in multiple languages.
UpdateProperty	Stores property and property attributes.
UpdateUserProfileData	Stores user profiles and their attributes.

2.2.6.4.1 UpdateList Schema

The complex types, simple types, and elements that are specified in this schema are used in the **profile_UpdateProfileDisplay** stored procedure.

Usage Example:

```
<MSPROFILE>
   <PROFILE ProfileName="UserProfile">
     <PROPERTY  PropertyName="HomeAdd"  ProfileSubtypeID="1" DisplayOrder="1"/>
     <PROPERTY  PropertyName="HomeAdd2" ProfileSubtypeID="1" DisplayOrder="2"/>
     <PROPERTY  PropertyName="Zip code" ProfileSubtypeID="1" DisplayOrder="3" />   </PROFILE>
</MSPROFILE>
MSPROFILE: Complex Type that contains PROFILE.
PROFILE: Complex Type that contains ProfileName and Property.
PROPERTY: Complex Type that contains PropertyName, ProfileSubtypeID and Display Order.
```

2.2.6.4.1.1 MSPROFILE

```
<s:element name="MSPROFILE">
   <s:complexType>
      <s:sequence>
         <s:element type="s0:PROFILE"/>
      </s:sequence>
   </s:complexType>
</s:element>
```

PROFILE: Must be a PROFILE (Section 2.2.6.4.1.2) Type element. This element MUST be specified.

2.2.6.4.1.2 PROFILE

```
<s:element name="PROFILE">
   <s:complexType>
     <s:sequence>
       <s:element name="PROPERTY" type="s0:PROPERTY" minOccurs="0" maxOccurs="unbounded"/>
       </s:sequence>
     <s:attribute name="ProfileName" type="s:string" use="required">
   </s:complexType>
</s:element>
```

ProfileName: This attribute MUST be specified. This attribute MUST be equal to "UserProfile".

Property: Must be a Property (Section 2.2.6.4.1.3) Type element.

2.2.6.4.1.3 PROPERTY

```
<s:element name="PROPERTY">
   <s:complexType>
     <s:attribute name="PropertyName" type="s:string" use="required"/>
     <s:attribute name="ProfileSubtypeID" type="s:int" use="required"/>
     <s:attribute name="DisplayOrder" type="s:int" use="required"/>
   </s:complexType>
</s:element>
```

PropertyName: The name of the property or section.

ProfileSubtypeID: A profile subtype identifier.

DisplayOrder: The value specifying the order of the property or section when it is displayed. This value MUST be unique, starting at 1, and incrementing by 1 only for each property or section**.**

2.2.6.4.2 UpdatePropertyLoc Schema

The complex types, simple types, and elements that are described in this section are used in the **profile_UpdatePropertyLoc** stored procedure.

Usage Example:

```
<Loc>
   <Item Lcid="1033" Text="Value 1" />
   <Item Lcid="1025" Text="Value 2" />
   <Item Lcid="2052" Text="Value 3" />
</Loc>

Loc: contain Item.
Item: Complex Type that contains Lcid and Text.
```

2.2.6.4.2.1 Loc

```
<s:element name="Loc">
 <s:complexType>
```

```
  <s:sequence>
     <s:element minOccurs="0" maxOccurs="unbounded" name="Item" type="s0:Item"/>
  </s:sequence>
 </s:complexType>
</s:element>
```

Item: An **Item** (Section 2.2.6.4.2.2) element (from the UpdatePropertyLoc schema) that MUST be specified in the *@DisplayNamesXml* input parameter.

2.2.6.4.2.2 Item

```
<s:element name="Item">
   <s:complexType>
      <s:attribute name="Lcid" type="s:int" use="required"/>
      <s:attribute name="Text" type="s:string" use="required"/>
   </s:complexType>
</s:element>
```

Lcid: The LCID of the specified text.

Text: Descriptive value for the specified **Lcid**. For user display names, the value MUST NOT exceed 50 characters. For the description, the text SHOULD NOT exceed 256 characters but MAY be up to 512 characters. In the case that the value of Text includes more than 256 characters, the value will be truncated to 256 characters.

2.2.6.4.3 UpdateProperty Schema

The complex types, simple types, and elements that are specified in this section are used in the **profile_UpdateProperty** stored procedure.

MSPROFILE: Complex Type that contains Property.

2.2.6.4.3.1 MSPROFILE

The **MSPROFILE** element contains a nested **PROPERTY** element which is used to specify a property to be added, updated, or removed.

```
<s:element name="MSPROFILE">
   <s:complexType>
     <s:sequence>
         <s:element name="PROPERTY" type="s0:PROPERTY" />
     </s:sequence>
   </s:complexType>
</s:element>
```

Property: This value MUST be in 1 of the 3 following forms depending on whether the protocol client is using an UpdateProperty Schema value to add a property (Section 2.2.6.4.3.1.1), update a property (Section 2.2.6.4.3.1.2) or remove a property (Section 2.2.7.4.3.1.3).

2.2.6.4.3.1.1 PROPERTY Element for Add Operations

The PROPERTY element contains XML code for adding a property.

```
<s:element name="PROPERTY">
  <s:complexType>
    <s:attribute name="PropertyName" type="s:string" use="required" />
    <s:attribute name="PropertyType" type="s:int" use="required"/>
    <s:attribute name="ID" type="s:long" use="required"/>
    <s:attribute name="DataTypeId" type="s:int" use="required" />
    <s:attribute name="Length" type="s:int" />
    <s:attribute name="DefaultPrivacy" type="s:int" use="required" />
    <s:attribute name="UserOverridePrivacy" type="s:boolean" use="required" />
    <s:attribute name="Replicable" type="s:boolean" use="required" />
    <s:attribute name="PrivacyPolicy" type="s:int" use="required" />
    <s:attribute name="IsSection" type="s:boolean" use="required" />
    <s:attribute name="IsMultiValue" type="s:boolean" use="required" />
    <s:attribute name="TermSetID" type="s:string" use="required" />
    <s:attribute name="IsEditable" type="s:boolean" use="required" />
    <s:attribute name="IsAdminEditOnly" type="s:boolean" use="required" />
    <s:attribute name="IsEventLog" type="s:boolean" use="required" />
    <s:attribute name="IsUpgrade" type="s:boolean" use="required" />
    <s:attribute name="IsUpgradePrivate" type="s:boolean" use="required" />
    <s:attribute name="IsSearchable" type="s:boolean" use="required" />
    <s:attribute name="IsAlias" type="s:boolean" use="required" />
    <s:attribute name="IsVisible" type="s:boolean" use="required" />
    <s:attribute name="IsVisibleOnViewer" type="s:boolean" use="required" />
    <s:attribute name="IsExpand" type="s:boolean" use="required" />
    <s:attribute name="Separator" type="s:string" use="required" />
    <s:attribute name="MaximumShown" type="s:int" use="required" />
    <s:attribute name="bUpdate" type="s:boolean" use="required" />
  </s:complexType>
</s:element>
```

PropertyName: Contains the name of a property.

PropertyType: MUST contain a Property (section 2.2.1.17) Type value for the property.

ID: The value MUST be a Property ID if the PropertyType is a core property. The value MUST be a Profile Type ID if the PropertyType is profile type. The value MUST be a Profile Subtype ID if the PropertyType is profile subtype.

DataTypeId: MUST contain a Property Data (section 2.2.1.12) Type value for the property.

Length: Contains the maximum length of the property's value. If IsSection is 1, the Length attribute MUST NOT be specified. Otherwise, the length MUST be a value dictated by the Property Data (section 2.2.1.12) Type definition.

DefaultPrivacy: Contains a value indicating the default Privacy (section 2.2.1.2) Type of the property**.**

UserOverridePrivacy: MUST be an Is Item Security Overridable (section 2.2.1.8) Type value.

Replicable: Contains a value that indicates whether this property is Replicable. This value MUST be 1 to indicate that this property is Replicable.

PrivacyPolicy: MUST be a Privacy Policy (section 2.2.1.6) Type value.

IsSection: Contains a value that indicates whether this property is a section. This value MUST be 1 to indicate that this property is a section. This value MUST be 0 to indicate that this property is not a section.

IsMultiValue: Contains a value that indicates whether this property is a multi-value property. This value MUST be 1 to indicate that this property is a multi-value property. This value MUST be 0 to indicate that this property is not a multi-value property.

TermSetID: The identifier of the term set to associate with the property, or NULL to associate the property with no term set.

IsEditable: Contains a value that indicates whether this property is editable. This value MUST be 1 to indicate that this property is editable. This value MUST be 0 to indicate that this property is not editable.

IsAdminEditOnly: Contains a value that indicates whether this property is a property editable only by the administration. This value MUST be 1 to indicate that this property is a property editable only by the administration. This value MUST be 0 to indicate that this property is not a property editable only by the administration.

IsEventLog: Indicates whether changes to this property are displayed in a Colleague Tracker. This value MUST be 1 to indicate that changes to this property are displayed in a Colleague Tracker. This value MUST be 0 to indicate that changes to this property are not displayed in a Colleague Tracker.

IsUpgrade: Contains a value that indicates whether this property exists in a previously upgraded installation. This value MUST be 1 to indicate that this property exists in a previously upgraded installation. This value MUST be 0 to indicate that this property does not exist in a previously upgraded installation.

IsUpgradePrivate: Contains a value that indicates whether this property was private in a previously upgraded installation. This value MUST be 1 to indicate that this property was private in a previously upgraded installation. This value MUST be 0 to indicate that this property was not private in a previously upgraded installation.

IsSearchable: Contains a value that indicates whether this property is indexed by Search Server. This value MUST be 1 to indicate that this property is indexed by Search Server. This value MUST be 0 to indicate that this property is not indexed by Search Server.

IsAlias: Contains a value that indicates whether this property serves as an alias of the user for user search purposes. This value MUST be 1 to indicate that this property serves as an alias of the user for user search purposes. This value MUST be 0 to indicate that this property does not serve as an alias of the user for user search purposes.

IsVisible: Contains a value that indicates whether this property is visible. This value MUST be 1 to indicate that this property is visible. This value MUST be 0 to indicate that this property is not visible.

IsVisibleOnViewer: Contains a value that indicates whether this property is visible on the default profile viewer page. This value MUST be 1 to indicate that this property is visible on the default profile viewer page. This value MUST be 0 to indicate that this property is not visible on the default profile viewer page.

IsExpand: This value MUST be 1.

Separator: MUST be a Separator (section 2.2.1.13) Type value.

MaximumShown: Contains a value indicating the maximum number of multiple-value choice list entries to show for a property before displaying an ellipsis.

bUpdate: This value MUST be 0.

2.2.6.4.3.1.2 PROPERTY Element for Update Operations

The PROPERTY element contains XML code for updating a property.

```
<s:element name="PROPERTY">
  <s:complexType>
    <s:attribute name="PropertyName" type="s:string" use="required" />
    <s:attribute name="PropertyType" type="s:int" use="required"/>
    <s:attribute name="ID" type="s:long" use="required"/>
    <s:attribute name="DataTypeId" type="s:int" use="required" />
    <s:attribute name="Length" type="s:int" />
    <s:attribute name="DefaultPrivacy" type="s:int" use="required" />
    <s:attribute name="UserOverridePrivacy" type="s:boolean" use="required" />
    <s:attribute name="Replicable" type="s:boolean" use="required" />
    <s:attribute name="PrivacyPolicy" type="s:int" use="required" />
    <s:attribute name="IsSection" type="s:boolean" use="required" />
    <s:attribute name="IsMultiValue" type="s:boolean" use="required" />
    <s:attribute name="TermSetID" type="s:string" use="required" />
    <s:attribute name="IsEditable" type="s:boolean" use="required" />
    <s:attribute name="IsAdminEditOnly" type="s:boolean" use="required" />
    <s:attribute name="IsEventLog" type="s:boolean" use="required" />
    <s:attribute name="IsUpgrade" type="s:boolean" use="required" />
    <s:attribute name="IsUpgradePrivate" type="s:boolean" use="required" />
    <s:attribute name="IsSearchable" type="s:boolean" use="required" />
    <s:attribute name="IsAlias" type="s:boolean" use="required" />
    <s:attribute name="IsVisible" type="s:boolean" use="required" />
    <s:attribute name="IsVisibleOnViewer" type="s:boolean" use="required" />
    <s:attribute name="IsExpand" type="s:boolean" use="required" />
    <s:attribute name="Separator" type="s:string" use="required" />
    <s:attribute name="MaximumShown" type="s:int" use="required" />
    <s:attribute name="bUpdate" type="s:boolean" use="required" />
  </s:complexType>
</s:element>
```

PropertyName: Contains the name of a property, the value of PropertyName MUST NOT be changed.

```
PropertyType: MUST contain a Property (Section 2.2.1.17) Type value for the property.
```

ID: The value MUST be a Property ID if the PropertyType is a core property. The value MUST be a Profile Type ID if the PropertyType is profile type. The value MUST be a Profile Subtype ID if the PropertyType is profile subtype.

DataTypeId: MUST contain a Property Data (Section 2.2.1.12) Type value for the property, the value of DataTypeId MUST NOT be changed.

Length: Contains the maximum length of the property's value. If IsSection is 1, the Length attribute MUST NOT be specified. Otherwise, the length MUST be a value dictated by the Property Data (Section 2.2.1.12) Type definition, the value of Length MUST NOT be changed.

DefaultPrivacy: Contains a value indicating the default Privacy (Section 2.2.1.2) Type of the property**.** The value of this attribute MUST be 0 when updating a section.

UserOverridePrivacy: MUST be an Is Item Security Overridable (Section 2.2.1.8) Type value.

Replicable: Contains a value that indicates whether this property is Replicable. This value MUST be 1 to indicate that this property is Replicable. This value MUST be 0 to indicate that this property is not Replicable.

PrivacyPolicy: MUST be a Privacy Policy (Section 2.2.1.6) Type value. The value of this attribute MUST be 0 when updating a section.

IsSection: Contains a value that indicates whether this property is a section. This value MUST be 1 to indicate that this property is a section. This value MUST be 0 to indicate that this property is not a section.

IsMultiValue: Contains a value that indicates whether this property is a multi-value property. During an update operation, the value of IsMultiValue MUST NOT be changed. This value MUST be 1 to indicate that this property is a multi-value property. This value MUST be 0 to indicate that this property is not a multi-value property.

TermSetID: The identifier of the term set to associate with the property, or NULL to associate the property with no term set.

IsEditable: Contains a value that indicates whether this property is editable. This value MUST be 1 to indicate that this property is editable. This value MUST be 0 to indicate that this property is not editable.

IsAdminEditOnly: Contains a value that indicates whether this property is a property editable only by the administration. This value MUST be 1 to indicate that this property is a property editable only by the administration. This value MUST be 0 to indicate that this property is not a property editable only by the administration.

IsEventLog: Indicates whether changes to this property are displayed in a Colleague Tracker. This value MUST be 1 to indicate that changes to this property are displayed in a Colleague Tracker. This value MUST be 0 to indicate that changes to this property are not displayed in a Colleague Tracker.

IsUpgrade: Contains a value that indicates whether this property exists in a previously upgraded installation. This value MUST be 1 to indicate that this property exists in a previously upgraded installation. This value MUST be 0 to indicate that this property does not exist in a previously upgraded installation.

IsUpgradePrivate: Contains a value that indicates whether this property was private in a previously upgraded installation. This value MUST be 1 to indicate that this property was private in a previously upgraded installation. This value MUST be 0 to indicate that this property was not private in a previously upgraded installation.

IsSearchable: Contains a value that indicates whether this property is indexed by Search Server. This value MUST be 1 to indicate that this property is indexed by Search Server. This value MUST be 0 to indicate that this property is not indexed by Search Server.

IsAlias: Contains a value that indicates whether this property serves as an alias of the user for user search purposes. This value MUST be 1 to indicate that this property serves as an alias of the user for user search purposes. This value MUST be 0 to indicate that this property does not serve as an alias of the user for user search purposes.

IsVisible: Contains a value that indicates whether this property is visible. This value MUST be 1 to indicate that this property is visible. This value MUST be 0 to indicate that this property is not visible.

IsVisibleOnViewer: Contains a value that indicates whether this property is visible on the default profile viewer page. This value MUST be 1 to indicate that this property is visible on the default

profile viewer page. This value MUST be 0 to indicate that this property is not visible on the default profile viewer page.

IsExpand: Contains a value that MUST be listed in the following table:

Value	Description
0	This value MUST be specified when editing an existing property.
1	This value MUST be specified when editing an existing section.

Separator: MUST be a Separator (Section 2.2.1.13) Type value.

MaximumShown: Contains a value indicating the maximum number of multiple-value choice list entries to show for a property before displaying an ellipsis.

bUpdate: This value MUST be 1.

2.2.6.4.3.1.3 PROPERTY Element for Remove Operation

The PROPERTY element contains XML code for removing a property. This XML contains a PropertyName attribute describing the property to be removed.

```
<s:element name="PROPERTY"/>
  <s:complexType>
    <s:attribute name="ProfileName" type="s:string" use="required" />
    <s:attribute name="PropertyType" type="s:int" use="required"/>
    <s:attribute name="ID" type="s:long" use="required"/>
  </s:complexType>
</s:element>
```

PropertyName: Contains the name of a property.

PropertyType: MUST contain a Property (section 2.2.1.17) Type value for the property.

ID: If the Property Type is a core property the value MUST be a Property Identifier. If the Property Type is a profile type the value MUST be a Profile Type Identifier. If the Property Type is a profile subtype, the value MUST be a profile subtype identifier.

2.2.6.4.4 UpdateUserProfileData Schema

The complex types, simple types, and elements that are described in this section are used in the **profile_UpdateUserProfileData** stored procedure.

Usage Example:

```
<MSPROFILE >
   <PROFILE ProfileName ="UserProfile">
      <USER UserID="12345" NTAccount="redmond\user" NewUser="1">
        <PROPERTY  PropertyName="Home Address" PropertyValue="18530 Redmond"
                   Privacy="1"> </PROPERTY>
        <PROPERTY  PropertyName="Work Address" PropertyValue="18530 Redmond"
                   Privacy="1"> </PROPERTY>
        <PROPERTY  PropertyName="Work Address" RemoveFlag=1> </PROPERTY>
      </USER>
   </PROFILE>
```

```
</MSPROFILE>
```

MSPROFILE: Complex Type that contains PROFILE.

PROFILE: Complex Type that contains USER.

ArrayOfUser: contains UserID, NTAccount, NewUser and PROPERTY.

ArrayOfProperty: Complex Type that contains PropertyName, PropertyValue, Privacy and RemoveFlag.

2.2.6.4.4.1 MSPROFILE

```
<xs:element name="MSPROFILE" type="s0:Profile">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="PROFILE" type="s0:Profile">
  </xs:sequence>
 </xs:complexType>
</xs:element>

PROFILE: A Profile (Section 2.2.7.4.4.2) Type element.
```

2.2.6.4.4.2 PROFILE

```
<xs:element name="PROFILE" type="">
 <xs:complexType>
  <xs:sequence>
   <xs:element minOccurs="1" maxOccurs="unbounded" name="USER"
               type="s0:ArrayOfUser">
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

USER: An ArrayOfUser (Section 2.2.6.4.4.3) Type element.

2.2.6.4.4.3 ArrayOfUser

```
<xs:element minOccurs="1" maxOccurs="unbounded" name="USER">
 <xs:complexType>
   <xs:attribute name="UserID" type="xs:unsignedShort" use="required" />
   <xs:attribute name="NTAccount" type="xs:string" use="required" />
   <xs:attribute name="NewUser" type="xs: unsignedByte " use="required" />
   <s:sequence>
    <xs:element minOccurs="1" maxOccurs="unbounded" name="PROPERTY"
                type="s0:ArrayOfProperty">
   </s:sequence>
 </xs:complexType>
</xs:element>
```

UserID: The GUID of the user.

NTAccount: A **Security Account Manager (SAM)** user name for the entity specified by the user profile.

NewUser: A bit flag specifying if the user is a new user. A value of 1 indicates a new user and a value of 0 indicates an existing user.

PROPERTY: An ArrayOfProperty (Section 2.2.6.4.4.4) Type element.

2.2.6.4.4.4 ArrayOfProperty

```
<xs:element minOccurs="1" maxOccurs="unbounded" name="PROPERTY">
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="xs:string">
    <xs:attribute name="PropertyName" type="xs:string" use="required" />
    <xs:attribute name="PropertyValue" type="xs:string" use="optional" />
    <xs:attribute name="PropertySecondaryValue" type="xs:string" use="optional" />
    <xs:attribute name="Privacy" type="xs:unsignedByte" use="optional" />
    <xs:attribute name="RemoveFlag" type="xs:unsignedByte" use="optional" />
            </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

PropertyName: The name of the property.

PropertyValue: The value of the property.

Privacy: The privacy policy setting for the property.

RemoveFlag: A bit flag specifying that the property MUST be removed. If the property has multiple values, then all values MUST be removed.

If the PROPERTY element attribute name PropertyValue is empty, then the property value MUST be set to NULL.

If PROPERTY element attribute RemoveFlag is 1, then the property record MUST be removed; in a multiple values field, all values MUST be removed.

If USER element attribute UserID is empty, then a new user MUST be created with the specified property information.

If the property list contains multiple values, then the protocol server MUST compose the list of values in sequence and MUST NOT be interwoven with other properties.

2.2.6.4.5 UpdateOrganizationProfileData Schema

The complex types, simple types, and elements that are described in this section are used in the **profile_UpdateOrganizationProfileData** stored procedure.

Usage Example:

```
<MSPROFILE >
   <PROFILE ProfileName ="OrganizationProfile">
      <ORGANIZATION RecordID="1">
         <PROPERTY PropertyName="PreferredName" Privacy="1"
            PropertyValue="PreferredOrganizationName" />
         <PROPERTY PropertyName="SPS-Team-Site" Privacy="1"
            PropertyValue="http://path/to/site" />
         <PROPERTY PropertyName="SPS-AboutUs" Privacy="1" PropertyValue="About Us
```

```
            Description" />
        </ORGANIZATION>
    </PROFILE>
</MSPROFILE>
```

MSPROFILE: Complex Type that contains PROFILE.

PROFILE: Complex Type that contains ORGANIZATION.

ArrayOfOrganization: Complex Type that contains PROPERTY.

ArrayOfProperty: Complex Type that contains PropertyName, PropertyValue, PropertySecondaryValue, Privacy and RemoveFlag.

2.2.6.4.5.1 PROFILE

```
<xs:element name="PROFILE" type="">
 <xs:complexType>
  <xs:attribute name="ProfileName " type="xs:string" use="required" />
   <xs:element minOccurs="1" maxOccurs="unbounded" name="ORGANIZATION"
               type="s0:ArrayOfOrganization">
 </xs:complexType>
</xs:element>
```

ProfileName: This value MUST be "OrganizationProfile".

ORGANIZATION: An ArrayOfOrganization element.

2.2.6.4.5.2 ArrayOfOrganization

```
<xs:element name="ORGANIZATION" type="">
 <xs:complexType>
  <xs:attribute name="RecordID " type="xs:int" use="required" />
   <xs:element minOccurs="1" maxOccurs="unbounded" name="PROPERTY"
               type="s0:ArrayOfProperty">
 </xs:complexType>
</xs:element>
```

RecordId: The organization record identifier.

PROPERTY: An ArrayOfProperty element as specified in section 2.2.6.4.4.4.

2.2.6.5 Attributes

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

2.2.6.6 Groups

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

2.2.6.7 Attribute Groups

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

3 Protocol Details

3.1 Server Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

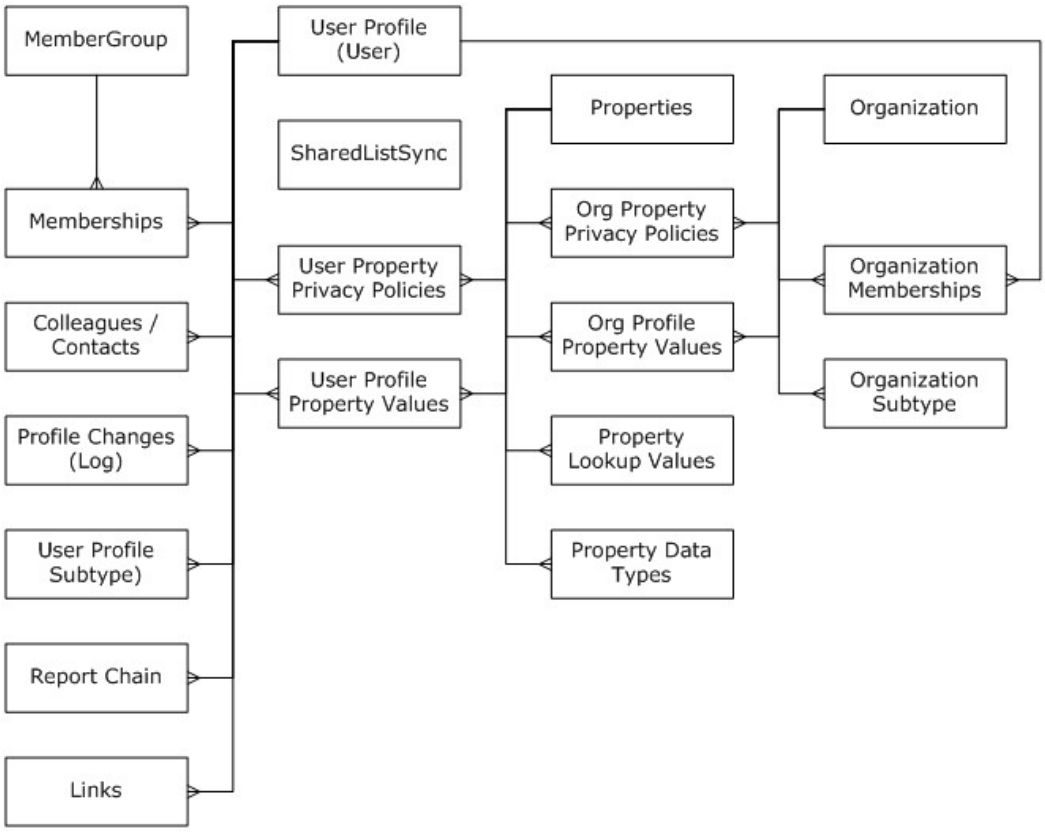

Figure 2: Types in the data model of the User Profile Stores Procedures Protocol

In the preceding diagram, each table specifies a type of entity in the model, and each arrow specifies that one type of entity always contains a reference to another.

- **Memberships:** a collection of entries containing information about the user membership data for a specified user.
- **Member Group:** a collection of entries containing information about the specified entity.
- **Organization:** a collection of properties and memberships. An organization can have up to one parent (there can exist only one root organization; and the root organization does not have

parents, while all non-root organizations must have a parent) and can have multiple child organizations.

- **Organization Membership:** is a property that contains information about the user, the organization in which the user has a membership and the nature of the membership. A user can have membership in multiple organizations, but cannot have multiple memberships in the same organization.
- **Organization Subtype:** a collection of properties that apply to a specific type of organization. Each organization has a subtype, and that subtype can be "default" which indicates the default set of properties for an organization.
- **User Profile**: a collection of entries containing information about a user.
- **User Property Privacy Policies**: a collection of entries containing information about the domain of visibility for a specific property.
- **User Profile Property Values**: a collection of entries containing information about the values associated with a pair of user profiles and a property.
- **User Profile Subtype**: a collection of properties that apply to a specific type of user. Each user has a subtype, and that subtype can be "default" which indicates the default set of properties for a user.
- **Properties** : a collection of entries containing information about properties for user profiles.
- **Property Data Types**: a collection of entries containing information about the possible data types of a property.
- **Property Lookup Value**: a collection of entries containing information about the possible values of a property value.
- **Links:** a collection of entries containing information about a link.
- **Colleagues / Contacts**: a collection of entries containing information about colleagues and contacts.
- **Profile Changes Log:** a collection of entries containing information about changes of user profile related data
- **SharedListSync:** a collection of entries containing information about shared list synchronization.
- **ReportChain:** a collection of entries containing information about the user hierarchy of the site.

3.1.2 Timers

An execution timeout timer on the protocol server governs the execution time for the client's requests. The amount of time is specified by a timeout value that is configured on the protocol server for all connections.

3.1.3 Initialization

A connection that uses the underlying protocol layers that are specified in Relationship to Other Protocols (section 1.4) MUST be established before using this protocol as specified in [MS-TDS].

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 membership_deleteGroup

The **membership_deleteGroup** stored procedure is called to remove a member group and its members. In the event of failure, **membership_deleteGroup** rolls back the transaction to restore data to the original state.**membership_deleteGroup** is defined using T-SQL syntax as follows:

```
PROCEDURE membership_deleteGroup (
@partitionID uniqueidentifier
,@Id bigint
,@SourceId uniqueidentifier = null
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@Id: A membership group record identifier. The protocol client MUST specify a valid ID.

@SourceId: The GUID identifying the Member Group Source of the Member Group to delete, as specified in [**MS-UPSIMP**]. The protocol client MUST specify a GUID with the value of 'A88B9DCB-5B82-41E4-8A19-17672F307B95'.

@correlationId: The optional **request identifier** for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.2 membership_enumerateGroups

The **membership_enumerateGroups** stored procedure retrieves a subset of the set of distribution list sourced membership groups that have e-mail address and site sourced membership groups.**membership_enumerateGroups** is defined using T-SQL syntax as follows:

```
PROCEDURE membership_enumerateGroups (
,@BeginId bigint
@partitionID uniqueidentifier
@EndId bigint
,@MINID bigint OUTPUT
,@MAXID bigint OUTPUT
,@IncludeAllDSGroups bit = 0
,@DeltaImport bit = 0
,@correlationId uniqueidentifier = null
);
```

@BeginId: The value of the inclusive minimum bound for the membership group record identifier when creating the subset of membership groups.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@EndId: The value of the inclusive maximum bound for the membership group record identifier when creating the subset of membership groups.

@MINID: Any input value MUST be ignored.

Value	Description
min	Any input value MUST be ignored. The output value MUST be set to the membership group record identifier of the distribution list sourced membership group that has an e-mail address or the site sourced membership group with the smallest membership group record identifier.

@MAXID: Any input value MUST be ignored.

Value	Description
ident_current	Any input value MUST be ignored. The output value MUST be set to a number that is greater than or equal to the maximum membership group record identifier.

@IncludeAllDSGroups: This value specifies whether to retrieve all membership groups or only the default ones. When set to a value of 0, only the default membership groups are retrieved. When set to a value of 1, all membership groups are retrieved.

@DeltaImport: This value specifies whether to retrieve membership groups if an import has not been performed. When set to a value of 0, all membership groups are retrieved. When set to 1, membership groups are retrieved only if an import has not been performed. When IncludeAllDSGroups is set to 0, this value MUST be ignored.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a ValidIdentifier

3.1.5.3 membership_getColleagueSuggestions

The **membership_getColleagueSuggestions** stored procedure retrieves a set of probable Colleague property for an entity specified by a user profile.

membership_getColleagueSuggestions is defined using T-SQL syntax as follows:

```
PROCEDURE membership_getColleagueSuggestions (
,@RecordId bigint
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@RecordId: A **user profile record identifier** for the specified user profile for whom the Colleague property value is sought.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a SuggestedColleagues

3.1.5.4 membership_getGroupCount

The **membership_getGroupCount** stored procedure retrieves the count of the distribution list sourced membership groups that have an e-mail address and site sourced membership groups.

membership_getGroupCount is defined using T-SQL syntax as follows:

```
PROCEDURE membership_getGroupCount (
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a Count

3.1.5.5 membership_getGroupMemberships

The **membership_getGroupMemberships** stored procedure retrieves all membership information about a particular membership group.

membership_getGroupMemberships is defined using T-SQL syntax as follows:

```
PROCEDURE membership_getGroupMemberships (
@partitionID uniqueidentifier
,@Id bigint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@Id: A membership group record identifier specifying the membership group to retrieve membership data for.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a membership_getGroupMemberships.ResultSet0

3.1.5.6 membership_getGroupMembershipsPaged

The membership_getGroupMembershipsPaged stored procedure is called to retrieve a subset of the membership information about a particular membership group.membership_getGroupMembershipsPaged is defined using T-SQL syntax as follows:

```
PROCEDURE membership_getGroupMembershipsPaged (
@partitionID uniqueidentifier
,@Id bigint
,@ViewerRecordId bigint
,@Count int
,@SortPropertyId bigint = 7
,@SortDirection bit = 0
,@ItemBeforeFirst nvarchar(1000)
,@RecordIdBeforeFirst bigint
,@Collation nvarchar(60)
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@Id: A membership group record identifier specifying the membership group to retrieve membership data for.

@ViewerRecordId: A user profile record identifier specifying the user profile of the individual requesting the data.

@Count: A value specifying the maximum number of records that MUST be returned by this query. This value MUST NOT be negative.

@SortPropertyId: A value specifying which field to sort the results on. This value MUST be listed in the following table:

Value	Description
13	The result set is sorted on the Title field.
14	The result set is sorted on the Department field.
7 or any other number	The result set is sorted on the PreferredName field.

@SortDirection: A value specifying the sort direction for the results. The value MUST be listed in the following table:

Value	Description
0	Ascending sort on the SortPropertyId field.
1	Descending sort on the SortPropertyId field.
NULL	Ascending sort on the SortPropertyId field.

@ItemBeforeFirst: A value used to determine the first membership information record to return. Records with values in the field indicated by SortPropertyId that are larger than the value of the ItemBeforeFirst input parameter MUST be returned if SortDirection input parameter specifies an ascending sort. Records with values in the field that are smaller than the value of the

ItemBeforeFirst input parameter MUST be returned if SortDirection input parameter specifies a descending sort. This value MUST be ignored if RecordIdBeforeFirst is NULL.

@RecordIdBeforeFirst: A value used to determine the first membership information record to return. Records with values in the SortPropertyId field that are equal to the value of the ItemBeforeFirst input parameter MUST be returned if the value of the user profile record identifier is smaller than the value of the RecordIdBeforeFirst input parameter. This value MUST be ignored if ItemBeforeFirst is NULL.

@Collation: This value MUST be ignored.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be "0".

Result Sets:

This stored procedure MUST return a TitlePagedMembership

3.1.5.7 membership_getRelatedGroups

The **membership_getRelatedGroups** stored procedure retrieves information about membership groups that are related to a particular membership group.

membership_getRelatedGroups is defined using T-SQL syntax as follows:

```
PROCEDURE membership_getRelatedGroups (
,@Id bigint
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@Id: A membership group record identifier specifying the membership group to return related membership groups for.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a RelatedMemberGroup

3.1.5.8 membership_updateGroup

The **membership_updateGroup** stored procedure is called to either create a new membership group or alter an existing membership group.**membership_updateGroup** is defined using T-SQL syntax as follows:

```
PROCEDURE membership_updateGroup (
@partitionID uniqueidentifier
,@Id bigint = null
,@Source uniqueidentifier
,@DisplayName nvarchar(250)
```

```
,@MailNickName nvarchar(250)
,@Description nvarchar(1500)
,@Url nvarchar(2048)
,@SourceReference nvarchar(2048)
,@DSGroupType bigint
,@DataSource nvarchar(400) = null
,@AllWebsSynchID int = null
,@UserCreated bit = 0
,@Type tinyint = 0
,@SID varbinary(512) = null
,@LastUpdate datetime OUTPUT
,@NewId bigint OUTPUT
,@Error int = null OUTPUT
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@Id: A membership group record identifier. The value MUST be NULL when creating a membership group. Id MUST NOT be NULL when updating a membership group.

@Source: MUST be a Short Link (section 2.2.1.4) Type identifier specifying the source of the membership group. This value MUST NOT be NULL.

@DisplayName: A descriptive name for the membership group. This value MUST NOT be NULL and MUST NOT contain leading or trailing space characters.

@MailNickName: An Active Directory attribute that contains a mail alias for the membership group. This value MUST NOT be NULL and MUST NOT contain leading or trailing space characters.

@Description: A description of the membership group. This value MUST NOT be NULL and MUST NOT contain leading or trailing space characters.

@Url: A URI for the membership group. Value MUST NOT be NULL and MUST NOT contain leading or trailing space characters. When the Source input parameter specifies a distribution list sourced membership group this MUST be a **mailto URI**. When the Source input parameter specifies a site sourced membership group the value MUST be a URI used to reach the root of the site.

@SourceReference: A string used to distinguish the various membership groups within a particular source specified by Source. The string MUST NOT contain leading or trailing space characters. When the Source input parameter specifies a distribution list sourced membership group the value MUST be the membership group's DN. When the Source input parameter specifies a site sourced membership group the value MUST be the site identifier.

@DSGroupType: A group type identifier from the data source (such as Active Directory, LDAP, and others).

@DataSource: The domain name of the group.

@AllWebsSynchID: identifier of the web that the group belongs to.

@UserCreated: MUST be an Is User Created (section 2.2.1.9) Type value.

@Type: Flag specifies the type of the membership group. When the Source input parameter specifies a site sourced membership group this value MUST be used. When the Source input parameter specifies a distribution list sourced membership group, the value MUST be listed in the following table:

Value	Description
0	The membership group MUST have an e-mail address. The Url input parameter MUST contain a mailto URI that contains a non empty to element as defined in [RFC2368].
1	The membership group MUST NOT have an e-mail address. The Url input parameter MUST contain the following text: "mailto:". The MailNickName input parameter MUST contain the text: "(null)".

@SID: Security identifier (SID) of the profile.

@LastUpdate: Any input value MUST be ignored. The output value MUST be set to the UTC value the membership group update operation was attempted.

@NewId: Any input value MUST be ignored. When Id is NULL (indicating that the caller intended to create a new membership group) the output value MUST be set to the membership group record identifier assigned to the newly created membership group. When Id is specified (indicating that the caller intended to update an existing membership group) the value MUST NOT be changed.

@Error: Any input value MUST be ignored. The output value MUST be set to 0 upon successful execution. The output value MUST be set to the error code of the operation upon failure.

@correlationId: The optional request identifier for the current request.

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

Value	Description
-1	Returned when Id is NULL (indicating that the caller intended to create a new membership group) and the Source and SourceReference pair were already in use by another membership group.
-2	Returned when Id is specified (indicating that the caller intended to update an existing membership group) , but no membership group was found matching the Id input parameter .

Result Sets: MUST NOT return any result sets.

3.1.5.9 privacy_deletePolicy

The **privacy_deletePolicy** stored procedure deletes a privacy policy which is associated with a specific user profile property.

privacy_deletePolicy is defined using T-SQL syntax as follows:

```
PROCEDURE privacy_deletePolicy (
,@Id uniqueidentifier
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@Id: The GUID identifying the privacy policy to be deleted.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.10 privacy_getAllPolicy

The **privacy_getAllPolicy** stored procedure returns the properties of all the privacy policies.

privacy_getAllPolicy is defined using T-SQL syntax as follows:

```
PROCEDURE privacy_getAllPolicy (
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a AllPrivacyPolicy

3.1.5.11 privacy_getFeaturePolicy

The **privacy_getFeaturePolicy** stored procedure returns the definition of all privacy policies which are not associated with a specific user profile property.

privacy_getFeaturePolicy is defined using T-SQL syntax as follows:

```
PROCEDURE privacy_getFeaturePolicy (
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a FeaturePrivacyPolicy

3.1.5.12 privacy_updatePolicy

The **privacy_updatePolicy** stored procedure updates the properties of a privacy policy or creates a new privacy policy.

privacy_updatePolicy is defined using T-SQL syntax as follows:

```
PROCEDURE privacy_updatePolicy (
@partitionID uniqueidentifier
,@Id uniqueidentifier = null
```

```
,@PropertyId bigint = null
,@ProfileSubtypeID int = null
,@DisplayName nvarchar(256)
,@GroupName nvarchar(256)
,@Policy int
,@DefaultItemSecurity int
,@IsItemSecurityOverridable bit
,@FilterPrivacyItems bit = 1
,@NewId uniqueidentifier OUTPUT
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@Id: The identifier assigned to the privacy policy to be updated. If this value is not NULL, then parameter PropertyId MUST be NULL and ProfileSubtypeID MUST be NULL.

@PropertyId: The identifier assigned to the user profile property whose privacy policy will be updated. If this value is not NULL then ProfileSubTypeID MUST NOT be NULL and parameter Id MUST be NULL.

@ProfileSubtypeID: A profile subtype identifier. If this value is not NULL then PropertyId MUST NOT be NULL and parameter Id MUST be NULL.

@DisplayName: The descriptive name of the privacy policy.

@GroupName: The descriptive name of the collection of privacy policies to which the privacy policy belongs.

@Policy: The enforcement level assigned to the privacy policy. The value MUST be a Privacy Policy (Section 2.2.1.6) type.

@DefaultItemSecurity: The default protection level assigned to the privacy policy. The value MUST be a Privacy (Section 2.2.1.2) Type.

@IsItemSecurityOverridable: MUST be an Is Item Security Overridable (Section 2.2.1.8) Type value.

@FilterPrivacyItems: This value MUST be set to 1.

@NewId: Any input value MUST be ignored.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.13 profile_EnumUsers

The **profile_EnumUsers** stored procedure returns the user profiles from the user profile store whose IDs are within the specified interval.

profile_EnumUsers is defined using T-SQL syntax as follows:

```
PROCEDURE profile_EnumUsers (
@partitionID uniqueidentifier
```

```
,@BeginID bigint
,@EndID bigint
,@MINID bigint OUTPUT
,@MAXID bigint OUTPUT
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@BeginID: The inclusive value where the Protocol server starts searching for existing user profile record identifiers. This parameter MUST be specified and it MUST NOT be NULL.

@EndID: The inclusive last value that the Protocol server searches for existing user profile record identifiers. This parameter MUST be specified and it MUST NOT be NULL.

@MINID: If the user profile store is empty the protocol server MUST ignore this parameter, otherwise the Protocol server MUST set this parameter to the smallest user profile record identifier that that is greater than the BeginId parameter.

@MAXID: If the user profile store is empty the server MUST ignore this parameter, otherwise the Protocol server MUST set this parameter to the last user profile record identifier that exists.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a profile_EnumUsers

3.1.5.14 profile_GetCommonManager

The **profile_GetCommonManager** stored procedure retrieves the common manager property between 2 users.

profile_GetCommonManager is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetCommonManager (
@partitionID uniqueidentifier
,@MyRecordId bigint
,@YourRecordId bigint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@MyRecordId: The record identifier for the first user. This MUST NOT be NULL.

@YourRecordId: The record identifier for the second user. This MUST NOT be NULL.

@correlationId: The optional request identifier for the current request.

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

Value	Description
0	Successful.
1	Encountered manager property loop.
2	Encountered the maximum manager property chain length of 40.
3	Encountered manager property loop and exceeded the maximum manager property chain length of 40.

Result Sets:

This stored procedure MUST return a profile_GetCommonManager

3.1.5.15 profile_GetDataTypeList

The **profile_GetDataTypeList** stored procedure returns the properties for Profile Data Types.

The **profile_GetDataTypeList** stored procedure is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetDataTypeList (
@partitionID uniqueidentifier
,@Collation nvarchar(60)
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@Collation: This parameter MUST be set to a valid SQL **collation** name. The server MUST use this collation name to sort the output by column FriendlyTypeName.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a DataTypeList

3.1.5.16 profile_GetMultiLoginAccounts

The **profile_GetMultiLoginAccounts** stored procedure finds all login names registered to the specified *@RecordId*. A single *@RecordId* can register 0 or more login names.**profile_GetMultiLoginAccounts** is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetMultiLoginAccounts (
@partitionID uniqueidentifier
,@RecordId bigint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@RecordId: A user profile record identifier. This parameter MUST be specified and it MUST NOT be NULL.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a MultiLoginAccounts

3.1.5.17 profile_GetNextUserProfileData

The **profile_GetNextUserProfileData** stored procedure retrieves the next user profile information: the record identifier and the GUID identifying the user.**profile_GetNextUserProfileData** is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetNextUserProfileData (
@partitionID uniqueidentifier
,@CurrentRecordID bigint
,@NextCurrentRecordID bigint OUTPUT
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@CurrentRecordID: The current user profile record identifier. This parameter MUST be specified and it MUST NOT be NULL.

@NextCurrentRecordID: The Protocol server MUST set this parameter to the smallest record identifier that is greater than the *@CurrentRecordID* value, or it MUST set it to -1 if no such record identifier exists.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.18 profile_GetPersonalSiteInfo

The **profile_GetPersonalSiteInfo** stored procedure retrieves the protocol server's personal site configuration properties.**profile_GetPersonalSiteInfo** is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetPersonalSiteInfo (
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a ProfilePersonalSite

3.1.5.19 profile_GetProfileCount

The **profile_GetProfileCount** stored procedure retrieves the number of user profiles contained in the user profile store.**profile_GetProfileCount** is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetProfileCount (
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a profile_GetProfileCount

3.1.5.20 profile_GetProfileCountWithProperty

The **profile_GetProfileCountWithProperty** stored procedure retrieves the number of user profiles contained in the user profile store that have a value defined for a specified property.**profile_GetProfileCountWithProperty** is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetProfileCountWithProperty (
,@PropertyName nvarchar(50)
@partitionID uniqueidentifier
@NoOfProfiles int OUTPUT
,@Error int OUTPUT
,@correlationId uniqueidentifier = null
);
```

@PropertyName: The name of a property. This parameter MUST be specified and MUST NOT be NULL.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@NoOfProfiles: MUST be the number of user profiles that have a non NULL value defined for the specified PropertyName property.

@Error: MUST be either 0 or -1.

Value	Description
-1	MUST be returned if *@PropertyName* is not a valid name of a property.

Value	Description
0	MUST be returned if *@PropertyName* is a valid property name.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.21 profile_GetProfilePropertyLoc

The **profile_GetProfilePropertyLoc** stored procedure retrieves all localized user display names and descriptions for each property.**profile_GetProfilePropertyLoc** is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetProfilePropertyLoc (
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a GetLocalizedProfileProperty

3.1.5.22 profile_GetSharedListSync

The **profile_GetSharedListSync** stored procedure retrieves a shared list object based on the specified *@ListId*.**profile_GetSharedListSync** is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetSharedListSync (
,@ListId uniqueidentifier
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@ListId: The GUID value that uniquely identifies the shared list object being retrieved.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a SharedList

3.1.5.23 profile_GetUserFormat

The **profile_GetUserFormat** stored procedure retrieves the **user name** format that is used by the Protocol server.**profile_GetUserFormat** is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetUserFormat (
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a ProfileGetUserFormat

3.1.5.24 profile_GetUserGUID

The **profile_GetUserGUID** stored procedure retrieves the GUID of a user's primary account.**profile_GetUserGUID** is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetUserGUID (
,@NTName nvarchar(120) = NULL
@partitionID uniqueidentifier
@SID varbinary(512) = NULL
,@GUID uniqueidentifier OUTPUT
,@RequireValues bit = 0
,@Debug bit = 0
,@correlationId uniqueidentifier = null
);
```

@NTName: A Security Account Manager (SAM) user name for the entity specified by the user profile.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@SID: The user's security identifier (SID).

@GUID: Output parameter that specifies the GUID of a user's primary account. If a valid user is not found, GUID MUST be NULL. This parameter MUST be specified.

@RequireValues: This parameter MUST be set to 0 or 1.

Value	Description
0	If set to 0, the procedure MUST look for a user Profile matching the specified parameters, including user Profiles which are corrupted.
1	If set to 1, the procedure MUST look for a user Profile matching the specified parameters, including only user Profiles which are not corrupted.

@Debug: This parameter MUST be ignored.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.25 profile_GetUserProfileData

The **profile_GetUserProfileData** stored procedure returns the user profile for the user specified by any of *@UserId*, *@NTName*, *@SID* or *@RecordId*.**profile_GetUserProfileData** is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetUserProfileData (
,@UserID uniqueidentifier
@partitionID uniqueidentifier
@NTName nvarchar(400) = NULL
,@SID varbinary(512) = NULL
,@RecordId bigint = NULL
,@ViewerRights int
,@ViewerNTName nvarchar(400) = NULL
,@AllowAlternateAccountName bit = 0
,@bQuickLoad bit = 0
,@bDebug bit = 0
,@correlationId uniqueidentifier = null
);
```

@UserID: Identifies the user requested.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@NTName: A **Security Account Manager (SAM)** user name for the entity specified by the user profile.

@SID: The SID for the user requested.

@RecordId: The matching integer that uniquely identifies the user requested.

@ViewerRights: The rights of the user requesting the information. This MUST be set to a Privacy (Section 2.2.1.2) type value or PRIVACY_NOTSET (0x40000000) if requesting the viewer rights.

@ViewerNTName: Security Account Manager (SAM) user name of user requesting rights. This MUST be NULL unless the ViewerRights is set to PRIVACY_NOTSET.

@AllowAlternateAccountName: A bit flag specifying whether the caller is requesting the alternate account instead of the master account. If the value of AllowAlternateAccountName is set to 1, the operation MUST be based on the alternate record identifier instead of the master account identifying the user.

@bQuickLoad: The value MUST be set to 0.

@bDebug: This value MUST be set to 0.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a ViewerRights

This stored procedure MUST return a UserProperties

3.1.5.26 profile_GetUserReportToData

The **profile_GetUserReportToData** stored procedure returns the manager property associated with a specific user, the colleague properties associated with a specific user, and the employees associated with a specific user. The protocol client MUST set 1 of the values: *@UserId*, *@NTName*, or *@SID*. If more than one value is set, the stored procedure will only use the first non-NULL value entered in *@UserId*, *@NTName*, or *@SID* respectively.

profile_GetUserReportToData is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetUserReportToData (
@partitionID uniqueidentifier
,@Collation nvarchar(60)
,@UserID uniqueidentifier
,@NTName nvarchar(400) = null
,@SID varbinary(512) = null
,@bDebug bit = 0
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@Collation: This is a **collation sequence** keyword that specifies how the results are sorted. This MUST NOT be NULL.

@UserID: The GUID that identifies a user.

@NTName: A **Security Account Manager (SAM)** user name for the entity specified by the user profile.

@SID: The user's security identifier (SID).

@bDebug: This parameter MUST be set 0 and MUST be ignored by the Protocol server.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

If the specified user exists, this stored procedure MUST return a UserProfile UserProfileresult set, as specified in section 2.2.4.19, listing the employees associated with the user.

If the specified user exists and has a manager property, this stored procedure MUST also return a second UserProfile result set, which MUST contain exactly one row detailing the manager associated with the user.

If the specified user exists and has a manager property, this stored procedure MUST also return a third UserProfile result set listing the colleagues associated with the user.

3.1.5.27 profile_GetViewerRights

The **profile_GetViewerRights** stored procedure returns the rights of the user specified by ViewerNTName on items owned by the user specified by *@RecordId*.

profile_GetViewerRights is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetViewerRights (
,@RecordId bigint
@partitionID uniqueidentifier
@ViewerNTName nvarchar(400)
,@ManagerNTName nvarchar(400) = null
,@correlationId uniqueidentifier = null
);
```

@RecordId: The RecordId of the user who owns the items. This MUST NOT be NULL.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@ViewerNTName: The login name of user requesting rights. This MUST NOT be NULL.

@ManagerNTName: The manager property of user specified by RecordId parameter.

@correlationId: The optional request identifier for the current request.

Return Values: profile_GetViewerRights returns a bitmask of the following permission values:

Value	Description
1	Public. This bit MUST always be set.
2	Contacts. This bit MUST be set if *@ViewerNTName* corresponds to a contact of the user indicated by *@RecordId.*
4	Workgroup. This bit MUST be set if the *@ViewerNTName* corresponds to a user in *@RecordId's* workgroup.
8	Manager. This bit MUST be set if *@ManagerNTName* matches *@ViewerNTName,* then the viewer is the manager of the user indicated by *@RecordId.*
16	Self. This bit and all other bits MUST be set if the *@RecordId* indicated belongs to the *@ViewerNTName.*

Result Sets: MUST NOT return any result sets.

3.1.5.28 profile_MigrateUserProfile

The **profile_MigrateUserProfile** stored procedure updates the security identifier (SID) and user name of an existing user profile. The stored procedure also updates audience, membership and user site information by replacing any occurrence of either the SID or the user name with the updated values.**profile_MigrateUserProfile** is defined using T-SQL syntax as follows:

```
PROCEDURE profile_MigrateUserProfile (
@partitionID uniqueidentifier
,@OldNTName nvarchar(400)
,@OldSID varbinary(512)
,@NewNTName nvarchar(400)
```

```
,@NewSID varbinary(512)
,@PersonalSiteReaders ntext
,@NewDN nvarchar(2048) = null
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@OldNTName: The old user name that identifies the user profile being updated. This value MUST be specified and MUST NOT be NULL.

@OldSID: The SID for the user profile being updated. This value MUST be specified and MUST NOT be NULL.

@NewNTName: The new user name of the profile being updated. If there is already a user profile under *@NewNTName*, profile_MigrateUserProfile it MUST be deleted avoid multiple profiles for a single user. This value MUST be specified and MUST NOT be NULL.

@NewSID: The new SID of the profile being migrated. This value MUST be specified and MUST NOT be NULL.

@PersonalSiteReaders: A comma-delimited list of user names. This list MUST first be populated by the profile_GetPersonalSiteInfo stored procedure and then all occurrences of the *@OldNTName* MUST be replaced with *@NewNTName* prior to the call to the profile_MigrateUserProfile stored procedure.

@NewDN: The new DN for the user profile being migrated.

@correlationId: The optional request identifier for the current request. To avoid duplicates, this parameter is used when deleting a pre-existing user profile with NTName = @NewNTName.

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

Value	Description
0	Successful execution.
Positive Integer Number	A SQL error code as defined in the SQL sysmessages table.

Result Sets: MUST NOT return any result sets.

3.1.5.29 profile_OnSqlRestore

The **profile_OnSqlRestore** stored procedure is called to apply database changes after a restore operation as used by the specified SQL server.

profile_OnSqlRestore is defined using T-SQL syntax as follows:

```
PROCEDURE profile_OnSqlRestore (
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.30 profile_RemoveUser

The **profile_RemoveUser** stored procedure is called to delete a user profile from the Protocol server.

profile_RemoveUser is defined using T-SQL syntax as follows:

```
PROCEDURE profile_RemoveUser (
@partitionID uniqueidentifier
,@UserID uniqueidentifier
,@NTName nvarchar(400) = null
,@SID varbinary(512) = null
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@UserID: Contains the GUID of the user to remove. If UserId is specified, NTName and SID MUST be NULL.

@NTName: Contains the login name of the user to remove. If NTName is specified, UserId and SID MUST be NULL.

@SID: Contains the SID of the user to remove. If SID is specified, NTName and UserId MUST be NULL.

@correlationId: The optional request identifier for the current request.

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

Value	Description
0	A user was deleted.
1	A user could not be found to delete.
Other	Operation not successful because of failures in stored procedure profile_sr_UpdateBucketInfo.

Result Sets: MUST NOT return any result sets.

3.1.5.31 profile_UpdateOrgColleagues

The **profile_UpdateOrgColleagues** stored procedure updates the colleagues organizational structure links for newly added users or users with updated managers in the user profile store.

profile_UpdateOrgColleagues is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdateOrgColleagues (
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
```

```
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request..

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

Value	Description
Integer number greater than zero	Success, the number represents the number of updated users.
0	Success, no users need to be updated.
-1	Failure, previous execution is still running.

Result Sets: MUST NOT return any result sets.

3.1.5.32 profile_UpdatePersonalSiteInfo

The **profile_UpdatePersonalSiteInfo** stored procedure updates the personal site configuration properties.

profile_UpdatePersonalSiteInfo is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdatePersonalSiteInfo (
@partitionID uniqueidentifier
,@CanonicalMySitePortalUrl nvarchar(2084)
,@Inclusion nvarchar(300)
,@NameFormat smallint
,@SiteReader ntext
,@EnableMLS bit
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@CanonicalMySitePortalUrl: A string specifying the canonical URL for the partition's (1) my site portal. This value MUST NOT be null or empty.

@Inclusion: The site under which user personal sites are created. The value MUST be a valid **server-relative URL**.

@NameFormat: MUST be a Name Format (Section 2.2.1.10) Type value

@SiteReader: A comma-delimited list of user names that will be granted the read permission level on new user personal sites. This parameter MUST be NULL or if it is not NULL it MUST be a list of user names with a comma character after each user name except the last user name in the list.

@EnableMLS: MUST be an Is MLS Enabled (Section 2.2.1.7) Type value.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.33 profile_UpdatePersonalSpace

The **profile_UpdatePersonalSpace** stored procedure updates a user's personal site URL.

Profile_UpdatePersonalSpace is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdatePersonalSpace (
,@UserGuid uniqueidentifier
@partitionID uniqueidentifier
@PersonalSpaceURL nvarchar(2048)
,@correlationId uniqueidentifier = null
);
```

@UserGuid: This MUST be the GUID for the user whose personal site is being updated.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@PersonalSpaceURL: The server-relative URL of the personal site that is to be updated. This parameter MUST be specified and MUST be a server-relative URL.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.34 profile_UpdateProfileDisplay

The **profile_UpdateProfileDisplay** stored procedure updates the display order of properties and sections.

Profile_UpdateProfileDisplay is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdateProfileDisplay (
@partitionID uniqueidentifier
,@UpdateList ntext
,@Debug bit = 0
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@UpdateList: XML coding which lists properties and sections. This parameter MUST be specified. This parameter MUST NOT be NULL and MUST adhere to the UpdateList (Section 2.2.6.4.1) schema.

@Debug: This parameter MUST be ignored.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

>

This stored procedure MUST return a UpdateProfileDisplayResult

3.1.5.35 profile_UpdateProperty

The **profile_UpdateProperty** stored procedure adds, updates and removes properties. It can also be used to update sections.

profile_UpdateProperty is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdateProperty (
,@RemovePropertyList ntext
@partitionID uniqueidentifier
@UpdatePropertyList ntext
,@Debug bit = 0
,@correlationId uniqueidentifier = null
);
```

@RemovePropertyList: XML code which lists the properties to remove. This parameter MUST be specified. If this parameter is not NULL, it MUST adhere to the UpdateProperty Schema (Section 2.2.6.4.3).

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@UpdatePropertyList: XML code which lists the properties to add or update. This parameter MUST be specified. It MUST adhere to the UpdateProperty Schema (Section 2.2.6.4.3).

@Debug: This parameter MUST be ignored.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST always return an UpdatePropertyResult

If *@RemovePropertyList* contains a property used in an Audience then this stored procedure MUST return an AudienceResult

3.1.5.36 profile_UpdatePropertyLoc

The **profile_UpdatePropertyLoc** stored procedure updates the user display names and descriptions of a property in multiple languages. Any existing user display names and descriptions for the property will be overwritten by the new user display names and descriptions specified in *@DisplayNamesXml* and *@DescriptionsXml*.

profile_UpdatePropertyLoc is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdatePropertyLoc (
,@PropertyName nvarchar(50)
@partitionID uniqueidentifier
@DisplayNamesXml ntext
,@DescriptionsXml ntext
,@correlationId uniqueidentifier = null
);
```

@PropertyName: The name of a property. This parameter MUST be specified and MUST NOT be NULL.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@DisplayNamesXml: XML code which lists user display names. This parameter MUST be specified and MUST adhere to the UpdatePropertyLoc Schema (Section 2.2.6.4.2).

@DescriptionsXml: XML code which lists descriptions. This parameter MUST be specified and MUST adhere to the UpdatePropertyLoc Schema (Section 2.2.6.4.2).

@correlationId: The optional request identifier for the current request.

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

Value	Description
-1	Property name does not exist
Positive Integer	A SQL error code as defined in the sysmessages table.
0	Successful Execution.

Result Sets: MUST NOT return any result sets.

3.1.5.37 profile_UpdateSharedListSync

The **profile_updateSharedListSync** stored procedure updates a shared list object.

profile_updateSharedListSync is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdateSharedListSync (
@partitionID uniqueidentifier
,@ListId uniqueidentifier
,@ItemsXml ntext
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@ListId: The GUID value that uniquely identifies the shared list object being updated. This value MUST be specified and MUST NOT be NULL

@ItemsXml: XML code that MUST conform to the ItemsXml (Section 2.2.6.3.1) **XML schema**. This value MUST be specified. Any shared list item not present in the *@ItemsXml* parameter MUST be deleted by the server. If *@ItemsXml* is NULL, the protocol server MUST delete all shared list items.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.38 profile_UpdateUserProfileBlobData

The **profile_UpdateUserProfileBlobData** stored procedure updates the user profile properties that cannot be updated through **profile_UpdateUserProfileData** (Section 3.1.5.39).

profile_UpdateUserProfileBlobData is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdateUserProfileBlobData (
@partitionID uniqueidentifier
,@UserID uniqueidentifier
,@NTName nvarchar(400)
,@PropertyName nvarchar(250)
,@PropertyValVarbinary varbinary(7500) = null
,@PropertyValText ntext = null
,@Debug bit = 0
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@UserID: Contains the GUID for the user. If this value is NULL then the user MUST be identified by NTName.

@NTName: A Security Account Manager (SAM) user name for the user specified by the user profile. This parameter MUST NOT be NULL if UserId is NULL.

@PropertyName: Contains the property name to be updated.

@PropertyValVarbinary: Contains the binary data value for the property to be updated.

@PropertyValText: Contains the text value of the property to be updated.

@Debug: This parameter MUST be ignored.

@correlationId: The optional request identifier for the current request.If both *@UserId* and *@NTName* are specified, *@UserId* is used by the protocol server. If *@UserId* and *@NTName* do not match an existing user, the protocol server creates a new user with the specified information. For a property to be updated, the value corresponding to the property's type MUST be specified in *@PropertyValbinary* or *@PropertyValText.*

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a UpdateUserProfileBlobDataResult

3.1.5.39 profile_UpdateUserProfileData

The **profile_UpdateUserProfileData** stored procedure is used to either create a new user profile or alter existing user profile data.**profile_UpdateUserProfileData** is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdateUserProfileData (
,@UpdatePropertyList ntext
@partitionID uniqueidentifier
@Debug bit = 0
,@correlationId uniqueidentifier = null
);
```

@UpdatePropertyList: Contains the list of properties that need to be updated. It MUST contain XML code. This parameter MUST be specified and MUST adhere to the UpdateUserProfileData schema (Section 2.2.6.4.4).

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@Debug: This parameter MUST be ignored.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a UpdateUserProfileDataResult

3.1.5.40 QuickLinksAdd

The **QuickLinksAdd** stored procedure is called to add a new link for a user in the user profile store. The link can be of type membership, colleague, or **quick link**.

QuickLinksAdd is defined using T-SQL syntax as follows:

```
PROCEDURE QuickLinksAdd (
,@RecordId bigint
@partitionID uniqueidentifier
@PageURL nvarchar(1250)
,@Title nvarchar(500)
,@ContentClass nvarchar(200)
,@Group nvarchar(400)
,@GroupType int
,@ItemSecurity int
,@PolicyId uniqueidentifier
,@ColleagueRecordId bigint
,@LinkUserIdIsWorkgroup bit
,@LinkMemberGroupId bigint
,@correlationId uniqueidentifier = null
);
```

@RecordId: The identifier of a user. This parameter MUST be specified and it MUST NOT be NULL.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@PageURL: The URL for a link. This parameter MUST be specified and MUST NOT be NULL if @LinkMemberGroupId and @ColleagueRecordId are NULL.

@Title: The text to display for the hyperlink specified in PageURL. This parameter MUST be specified and MUST NOT be NULL if @LinkMemberGroupId and @ColleagueRecordId are NULL.

@ContentClass: The content type for the hyperlink specified in PageURL. This parameter MUST be specified and MUST NOT be NULL if @LinkMemberGroupId and @ColleagueRecordId are NULL and @PolicyId is not a user defined link group Policy Link Type.

@Group: The group name for a link. This parameter MUST be specified and it MUST NOT be NULL. It MUST NOT be an empty string if GroupType is 0. It MUST be an empty string if GroupType is not 0.

@GroupType: MUST be a Group (Section 2.2.1.1) Type value that specifies the link group type. This parameter MUST be specified.

@ItemSecurity: MUST be a Privacy (Section 2.2.1.2) Type value that defines security for the link.

@PolicyId: MUST be a Policy Link (Section 2.2.1.5) Type value that specifies the link type. This parameter MUST be specified.

@ColleagueRecordId: The identifier of a colleague of @RecordId. If this parameter is not NULL, it specifies that the link data is a colleague link, and the parameter @LinkUserIdIsWorkgroup MUST NOT be NULL.

@LinkUserIdIsWorkgroup: A flag indicating whether @RecordId is a workgroup identifier. If this parameter is not NULL, it specifies that the link data is a colleague link, and the parameter @ColleagueRecordId MUST NOT be NULL.

@LinkMemberGroupId: The group identifier for a user. If this parameter is not NULL, it specifies that the link data is a Membership link.

@correlationId: The optional request identifier for the current request.

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

Value	Description
-1	Failure. Duplicate information exists. No data is added.
A positive number	Success. This is the identifier of the link that was added.

Result Sets: MUST NOT return any result sets.

3.1.5.41 QuickLinksDelete

The **QuickLinksDelete** stored procedure is called to delete the link for a user in the user profile store with the specified link type (specified by *@PolicyId*) and ID (specified by *@LinkID*).

QuickLinksDelete is defined using T-SQL syntax as follows:

```
PROCEDURE QuickLinksDelete (
,@RecordId bigint
@partitionID uniqueidentifier
@LinkID bigint
,@PolicyId uniqueidentifier
,@RemoveCount int OUTPUT
,@correlationId uniqueidentifier = null
);
```

@RecordId: The record identifier of a user. This parameter MUST be specified and it MUST NOT be NULL.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@LinkID: The identifier of a link. This parameter MUST be specified and it MUST NOT be NULL.

@PolicyId: MUST be a Policy Link (Section 2.2.1.5) Type value that specifies the link type. This parameter MUST be specified.

@RemoveCount: This is an output parameter that returns the number of data rows that have been deleted. This parameter MUST be set to the number of rows deleted.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.42 QuickLinksDeleteUser

The **QuickLinksDeleteUser** stored procedure deletes all link data for the specified user in the user profile store with the specified link type (specified by *@PolicyId*).

QuickLinksDeleteUser is defined using T-SQL syntax as follows:

```
PROCEDURE QuickLinksDeleteUser (
,@RecordId bigint
@partitionID uniqueidentifier
@PolicyId uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@RecordId: The identifier of a user. This parameter MUST be specified and it MUST NOT be NULL.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@PolicyId: MUST be a Policy Link (Section 2.2.1.5) Type value that specifies the link type.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.43 QuickLinksEdit

The **QuickLinksEdit** stored procedure modifies or adds a link for a user in the user profile store. The type of link can be a Group Membership Link, Colleague link, or user-defined hyperlink.

QuickLinksEdit is defined using T-SQL syntax as follows:

```
PROCEDURE QuickLinksEdit (
,@RecordId bigint
@partitionID uniqueidentifier
@LinkID bigint
,@PageURL nvarchar(1250)
,@Title nvarchar(500)
,@ContentClass nvarchar(200)
,@Group nvarchar(400)
,@GroupType int
,@ItemSecurity int
,@PolicyId uniqueidentifier
,@ColleagueRecordId bigint
,@LinkUserIdIsWorkgroup bit
,@LinkMemberGroupId bigint
,@correlationId uniqueidentifier = null
);
```

@RecordId: The identifier of a user. This parameter MUST be specified and it MUST NOT be NULL.

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@LinkID: The identifier of a link. This parameter MUST be specified and it MUST NOT be NULL. If LinkID does not reference an existing link ID, the protocol server MUST create the link.

@PageURL: The URL for a link. This parameter MUST be specified and MUST NOT be NULL if LinkMemberGroupId and ColleagueRecordId are NULL.

@Title: The text to display for the hyperlink specified in PageURL. This parameter MUST be specified and MUST NOT be NULL if LinkMemberGroupId and ColleagueRecordId are NULL.

@ContentClass: The Content type for the hyperlink specified in PageURL. This parameter MUST be specified and MUST NOT be NULL if LinkMemberGroupId and ColleagueRecordId are NULL and PolicyId is not a user-defined link group (E21FE63D-0BF6-42C0-85BC-AC8031552558).

@Group: The group name for a link. This parameter MUST be specified and it MUST NOT be NULL. It MUST NOT be an empty string if GroupType is 0. It MUST be an empty string if GroupType is not 0.

@GroupType: MUST be a Group (section 2.2.1.1) Type value that specifies the link group type. This parameter MUST be specified.

@ItemSecurity: MUST be a Privacy (section 2.2.1.2) Type value that defines security for the item.

@PolicyId: MUST be a Policy Link (section 2.2.1.5) Type value that specifies the link type. This parameter MUST be specified.

@ColleagueRecordId: The identifier of a colleague of the specified user. If this parameter is not NULL, it specifies that the link data is a colleague link, and the parameter LinkUserIdIsWorkgroup MUST NOT be NULL.

@LinkUserIdIsWorkgroup: A flag indicating whether RecordId is a Workgroup identifier. If this parameter is not NULL, it specifies that the link data is a colleague link, and the parameter ColleagueRecordId MUST NOT be NULL.

@LinkMemberGroupId: The group identifier for a user. If this parameter is not NULL, it specifies that the link data is a Membership link.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.44 QuickLinksRetrieveAllItems

The **QuickLinksRetrieveAllItems** stored procedure is called to retrieve link information for the following link types: Group Membership Links, Colleague links, or user-defined hyperlinks.

QuickLinksRetrieveAllItems is defined using T-SQL syntax as follows:

```
PROCEDURE QuickLinksRetrieveAllItems (
@partitionID uniqueidentifier
,@RecordId bigint
,@ViewerRecordId bigint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@RecordId: The identifier of a user. This parameter MUST be specified and it MUST NOT be NULL.

@ViewerRecordId: A GUID used to retrieve the records in the ViewerRights result set, based on the specified value.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a UserMemberships

This stored procedure MUST return a ViewerRights

This stored procedure MUST return a UserColleagues

This stored procedure MUST return a UserLinks

3.1.5.45 QuickLinksRetrieveColleaguesOfColleagues

The **QuickLinksRetrieveColleaguesOfColleagues** stored procedure retrieves the colleagues of the specified user's colleagues.

QuickLinksRetrieveColleaguesOfColleagues is defined using T-SQL syntax as follows:

```
PROCEDURE QuickLinksRetrieveColleaguesOfColleagues (
@partitionID uniqueidentifier
,@RecordId bigint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@RecordId: The user profile record identifier. This parameter identifies the user for which the colleague properties of colleague properties are returned. It MUST NOT be NULL.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a QuickLinksRetrieveColleaguesOfColleagues.ResultSet0

3.1.5.46 QuickLinksRetrieveGroupList

The **QuickLinksRetrieveGroupList** stored procedure retrieves the groups of a specified type for a user profile.

QuickLinksRetrieveGroupList is defined using T-SQL syntax as follows:

```
PROCEDURE QuickLinksRetrieveGroupList (
@partitionID uniqueidentifier
,@RecordId bigint
```

```
,@PolicyId uniqueidentifier = null
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@RecordId: The user profile record identifier for which the groups are retrieved. This parameter MUST be specified and it MUST NOT be NULL.

@PolicyId: **MUST** be a Policy Link (section 2.2.1.5) Type **value** that specifies the link type.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a QuickLinksRetrieveGroupList

3.1.5.47 membership_getGroupById

The **membership_getGroupById** stored procedure retrieves information about a particular membership group.

Membership_getGroupByID is defined using T-SQL syntax as follows:

```
PROCEDURE membership_getGroupById (
@partitionID uniqueidentifier
,@Id bigint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@Id: A membership group record identifier specifying the membership group to retrieve related membership groups for.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a membership_getGroupById.ResultSet0

3.1.5.48 membership_getGroupBySourceAndSourceReference

The **membership_getGroupBySourceAndSourceReference** stored procedure retrieves information about a particular membership group based on @*partitionID*, @*Source and* @*SourceReference*.

membership_getGroupBySourceAndSourceReference is defined using T-SQL syntax as follows:

```
PROCEDURE membership_getGroupBySourceAndSourceReference (
@partitionID uniqueidentifier
,@Source uniqueidentifier
,@SourceReference nvarchar(2048) = null
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@Source: MUST be a Short Link (section 2.2.1.4) Type identifier specifying the source of the membership group. Value MUST be ignored when parameter *@SourceReference* is NULL.

@SourceReference: A string used to distinguish the various membership groups within a particular source specified by @Source. The Unicode String Trim operation MUST have been performed on the input value. When the @Source input parameter specifies a distribution list sourced membership group the value MUST be the membership group's DN. When the @Source input parameter specifies a site sourced membership group the value MUST be the site identifier.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

For the following combination of parameters,

@partitionID: 0c37852b-34d0-418e-91c6-2ac25af4be5b

@Source: 8bb1220f-de8b-4771-ac3a-0551242cf2bd

@SourceReference: NOT NULL

@correlationId: NULL

This stored procedure MUST return a membership_getGroupById.ResultSet0

3.1.5.49 membership_getGroupImmediateMemberships

The **membership_getGroupImmediateMemberships** stored procedure retrieves information about a particular membership group based on @*partitionID* and @*Id*.**membership_getGroupImmediateMemberships** is defined using T-SQL syntax as follows:

```
PROCEDURE membership_getGroupImmediateMemberships (
@partitionID uniqueidentifier
,@Id bigint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@Id: A membership group record identifier specifying the membership group to retrieve related membership groups for.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a ImmediateMembership

3.1.5.50 membership_updateGroupMemberCount

The **membership_updateGroupMemberCount** stored procedure updates the group member count of a particular group membership based on ***@partitionID***, ***@sourceID***, *and* ***@ContentDBID***.

membership_updateGroupMemberCount is defined using T-SQL syntax as follows:

```
PROCEDURE membership_updateGroupMemberCount (
@partitionID uniqueidentifier
,@SourceId uniqueidentifier
,@ContentDBID uniqueidentifier = null
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@SourceId: The GUID identifying the Member Group Source of the Member Group to update, as specified in [**MS-UPSIMP**]. The protocol client MUST specify a GUID with the value of 'A88B9DCB-5B82-41E4-8A19-17672F307B95'.

@ContentDBID: The identifier of the content database.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.51 profile_AddProfile

The **profile_AddProfile** stored procedure creates a new profile subtype. Profile subtypes allow profiles to be further classified using unique names. For example, profile subtypes can be used to create subtypes of user profiles for employees, contractors, and part-time workers.

profile_AddProfile using T-SQL syntax as follows:

```
PROCEDURE profile_AddProfile (
@partitionID uniqueidentifier
,@ProfileName nvarchar(250)
,@ProfileDisplayName nvarchar(400)
,@ProfileTypeID smallint
,@Debug bit = 0
,@ProfileSubtypeID int OUTPUT
,@Error int OUTPUT
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@ProfileName: A unique name for this profile subtype. This name MUST be unique within the partition (1). Failure to specify a unique name MUST result in an error result of 2.

@ProfileDisplayName: A display name for this profile subtype as seen within the user interface.

@ProfileTypeID: A numeric value which indicates the profile type in which the subtype will be created under. This value MUST be 1, for "User" profiles or 2 for "Organization" profiles. Specifying any other value here MUST result in an error result of 1.

@Debug: This parameter MUST be ignored by the server.

@ProfileSubtypeID: This output parameter MUST be set to the unique identifier of the newly created profile subtype, or NULL in the event of any error.

Value	Description
null	This output parameter MUST be NULL only when an error occurred while creating the profile subtype, or invalid parameters were passed in to the stored procedure.
@@identity	An integer value returned MUST indicate a unique identifier generated for this profile subtype upon its successful creation.

@Error: This output parameter is used to return any error codes.

Value	Description
0	This return value indicates that no errors occurred.
1	This error code indicates a value other than 1 or 2 was specified as the @ProfileTypeID parameter. Subtypes can only be created on User or Organization profile types.
2	This error code indicates a profile subtype name already exists under the specified partition (1).
3	This error code indicates there was an internal SQL error while inserting data.
4	This error code indicates an error while initializing default properties for this profile subtype.
5	This error code indicates there was an internal SQL error while inserting data.
6	This error code indicates there was an internal SQL error while inserting data.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.52 profile_GetColleagueAdders

The **profile_GetColleagueAdders** stored procedure retrieves information about users for whom a link of type Colleague has been created in the time after *@MinEventTime*. All the links in the result set MUST be links to the user identified by the user profile record identifier *@PersonAddedId*.

profile_GetColleagueAdders is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetColleagueAdders (
```

```
@partitionID uniqueidentifier
,@PersonAddedId bigint
,@MinEventTime datetime
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@PersonAddedId: The value MUST be provided and MUST be a valid user profile record identifier.

@MinEventTime: The value MUST be provided and MUST represent 15 days prior to the time of invocation.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a profile_GetColleagueAdders.ResultSet0

For the following combination of parameters,

@partitionID: 0C37852B-34D0-418E-91C6-2AC25AF4BE5B

@PersonAddedId: 18

@MinEventTime: 2000-12-29 01:59:02.257

This stored procedure MUST return a profile_GetColleagueAdders.ResultSet0

3.1.5.53 profile_GetCorePropertyInfo

The **profile_GetCorePropertyInfo** stored procedure retrieves information about a core property. If either the *@PropertyURI* or *@PropertyName* parameters are not NULL, this stored procedure MUST return information for all properties matching the *@PropertyURI* or the *@PropertyName* as specified. If both the *@PropertyURI* and *@PropertyName* parameters are NULL, this stored procedure MUST return information for all properties.

profile_GetCorePropertyInfo is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetCorePropertyInfo (
@partitionID uniqueidentifier
,@PropertyURI nvarchar(250) = null
,@PropertyName nvarchar(50) = null
,@bDebug bit = 0
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@PropertyURI: The URI of the property.

@PropertyName: The name of the property.

@bDebug: This parameter MUST be ignored by the server..

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a CorePropertyInfoResultSet

3.1.5.54 Profile_GetDeletedUserList

The **Profile_GetDeletedUserList** stored procedure retrieves the list of user names for profiles that have been deleted.**Profile_GetDeletedUserList** is defined using T-SQL syntax as:

```
PROCEDURE Profile_GetDeletedUserList (
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a DeletedUserList

3.1.5.55 profile_GetExtendedReportsForUser

The **profile_GetExtendedReportsForUser** stored procedure returns all the employees for the specified user. The stored procedure MUST NOT return more than 200 rows.

Profile_GetExtendedReportsForUser is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetExtendedReportsForUser (
@partitionID uniqueidentifier
,@NTName nvarchar(400)
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@NTName: A **Security Account Manager (SAM)** user name for the entity specified by the user profile.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a ExtendedReports

3.1.5.56 profile_GetProfileList

The **profile_GetProfileList** stored procedure retrieves one or all profiles with a given **partitionID**.

profile_GetProfileList is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetProfileList (
@partitionID uniqueidentifier
,@ProfileName nvarchar(250) = null
,@Debug bit = 0
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@ProfileName: The unique name for the profile to return. If this parameter is NULL, then all profiles in the given partition (1) MUST be returned.

@Debug: This parameter MUST be ignored by the server.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a GetProfileList

3.1.5.57 profile_GetProfileSubtypePropertyInfo

The **profile_GetProfileSubtypePropertyInfo** stored procedure retrieves one or all profile subtype properties for the specified profile subtype. Subtype properties allow data to be associated with profiles of that subtype. For example, a subtype for "contractors" might have a property for "hourly pay."

profile_GetProfileSubtypePropertyInfo is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetProfileSubtypePropertyInfo (
@partitionID uniqueidentifier
,@PropertyID bigint = null
,@ProfileSubtypeID int = null
,@ReplicableSchemaVersion int = null OUTPUT
,@bDebug bit = 0
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@PropertyID: A 64-bit integer that specifies the individual property to be returned. This value MUST contain a **property identifier**, or contain NULL to return all properties associated with the specified @ProfileSubtypeID.

@ProfileSubtypeID: A 32-bit integer that species the profile subtype. This value MUST contain a valid profile subtype ID and cannot be NULL.

@ReplicableSchemaVersion: This output parameter MUST be set to the current schema version of the specified @ProfileSubtypeID and MUST be an integer greater than or equal to zero. Note the schema version MUST be incremented each time a replicable property is added on a subtype or an existing parent type becomes replicable.

@bDebug: This parameter MUST be ignored by the server.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a GetProfileSubtypePropertyInfoResult

3.1.5.58 profile_GetProfileTypePropertyInfo

The **profile_GetProfileTypePropertyInfo** stored procedure returns the profile type properties for the given profile type and property identifiers in a given partition (1).

profile_GetProfileTypePropertyInfo is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetProfileTypePropertyInfo (
@partitionID uniqueidentifier
,@PropertyID bigint = null
,@ProfileTypeID smallint = null
,@bDebug bit = 0
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@PropertyID: Specifies a property identifier assigned to a user profile property or an organization profile property. When this parameter is NULL the stored procedure MUST return all of the profile properties for the returned profiles and when this parameter is not NULL the stored procedure MUST return only records with property ID equals to the given value.

@ProfileTypeID: Specifies a profile type Id to look for. This parameter MUST be specified and the values MUST be either 1 for User Profile Type or 2 for Organization Profile Type.

@bDebug: This parameter MUST be ignored.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a GetProfileTypePropertyInfo

3.1.5.59 profile_GetUserProfileName

The **profile_GetUserProfileName** stored procedure retrieves the profile subtype ID for a given user profile.

profile_GetUserProfileName is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetUserProfileName (
@partitionID uniqueidentifier
,@UserID uniqueidentifier
```

```
,@NTName nvarchar(400) = null
,@SID varbinary(512) = null
,@RecordId bigint = null
,@bDebug bit = 0
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@UserID: A GUID used to identify the user profile.

@NTName: A Security Account Manager (SAM) user name used to identify the user.

@SID: A security identifier (SID) used to identify the user.

@RecordId: A user profile record identifier used to identify the user.

@bDebug: This parameter MUST be ignored.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a GetUserProfileName

3.1.5.60 profile_GetUserRecordId

The **profile_GetUserRecordId** stored procedure retrieves the record identifier for a given user profile.**profile_GetUserRecordId** is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetUserRecordId (
@partitionID uniqueidentifier
,@DSGuid uniqueidentifier = null
,@SID varbinary(512) = null
,@NTName nvarchar(400) = null
,@RecordId bigint OUTPUT
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@DSGuid: This value MUST be NULL and MUST be ignored.

@SID: This value MUST be NULL and MUST be ignored.

@NTName: The Security Account Manager (SAM) user name used to identify the user.

@RecordId: The record identifier for the retrieved user profile. This value MUST be the record identifier for the user profile or MUST be NULL when the user profile is not found for the provided search values.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.61 profile_GetUsers

The **profile_GetUsers** stored procedure returns a row for each record in that matches the *@correlationId* value. The result set MUST return 0 or more rows that contain the user info for each user in the specified partition (1). The result set MUST be sorted ascending by RecordID.

profile_GetUsers is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetUsers (
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a GetUsers

3.1.5.62 profile_MarkUserForDeletion

The **profile_MarkUserForDeletion** stored procedure marks a user profile for deletion.

profile_MarkUserForDeletion is defined using T-SQL syntax as follows:

```
PROCEDURE profile_MarkUserForDeletion (
@partitionID uniqueidentifier
,@RecordId bigint
,@DeleteValue tinyint = 1
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@RecordId: Specifies the ID of the user profile to mark for deletion. This value MUST be specified.

@DeleteValue: The optional flag for deleting the user profile. If **@DeleteValue** is 1, the user profile is deleted, else the user profile is unchanged for all other values. .

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.63 profile_RemoveProfile

The **profile_RemoveProfile** stored procedure removes an existing profile subtype.

profile_RemoveProfile is defined using T-SQL syntax as follows:

```
PROCEDURE profile_RemoveProfile (
@partitionID uniqueidentifier
,@ProfileName nvarchar(250)
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@ProfileName: The unique name that specifies the existing profile subtype to be removed.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.64 profile_suggestions_delete

The **profile_suggestions_delete** stored procedure deletes the user suggestion with the given identifier from the given partition (1).

profile_suggestions_delete is defined using T-SQL syntax as follows:

```
PROCEDURE profile_suggestions_delete (
@partitionID uniqueidentifier
,@Id bigint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@Id: The identifier of the user suggestion to be deleted.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.65 profile_suggestions_get

The **profile_suggestions_get** stored procedure retrieves all the suggestions in the given partition (1) for the user profile with the given record identifier.

profile_suggestions_get is defined using T-SQL syntax as follows:

```
PROCEDURE profile_suggestions_get (
@partitionID uniqueidentifier
,@RecordId bigint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@RecordId: A user profile record identifier.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a GetSuggestions

3.1.5.66 profile_suggestions_update

The **profile_suggestions_update** stored procedure creates new user suggestions and to update existing ones. If *@Id* is not NULL then **profile_suggestions_update** MUST find the user suggestion with the identifier *@Id* and the partition (1) identifier *@partitionID* and set its Status and Weight values to *@Status* and *@Weight*, respectively. If *@Id* is NULL the **profile_suggestions_update** MUST search for a user suggestion with the given user profile record identifier, suggestion, user suggestion type, and partition (1) identifier. If there is a matching record then **profile_suggestions_update** MUST set *@Id* to the identifier of the matching record and then proceed as if *@Id* had been non-NULL from the start. Otherwise, **profile_suggestions_update** MUST create a new user suggestion with the given partition (1) identifier, user profile record identifier, suggestion, user suggestion type, user suggestion status, and weight. On exit, *@Id* MUST be set to the identifier assigned to the new user suggestion record.

profile_suggestions_update is defined using T-SQL syntax as follows:

```
PROCEDURE profile_suggestions_update (
@partitionID uniqueidentifier
,@RecordId bigint
,@Id bigint = null OUTPUT
,@Suggestion nvarchar(256)
,@Type tinyint
,@Status tinyint
,@Weight float
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@RecordId: A user profile record identifier. The value MUST be provided and MUST NOT be NULL.

@Id: The user suggestion identifier. The value is optional. If the value is provided it MUST be either NULL or a valid user suggestion identifier. If the value is NULL or not provided then, on exit, it is set to the user suggestion identifier of the record that was affected by this invocation of **profile_suggestions_update**.

@Suggestion: The suggestion. The value MUST be provided and MUST NOT be NULL.

@Type: The user suggestion type. The value MUST be provided, MUST be a User Suggestion Type (Section 2.2.1.15), and MUST NOT be NULL.

@Status: The user suggestion status. The value MUST be provided, MUST be a User Suggestion Status (Section 2.2.1.16), and MUST NOT be NULL.

@Weight: The weight of the user suggestion, a number indicating how likely the entity is to be a Colleague. The value ranges from 0.0328765519086464 to 1.79E+308. Larger values of Weigh indicate a greater likelihood of a relationship between the colleagues. The value MUST be provided, MUST NOT be negative, and MUST NOT be NULL.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be zero.

Result Sets: MUST NOT return any result sets.

3.1.5.67 profile_UpdateProfileSubtype

The **profile_UpdateProfileSubtype** stored procedure updates the display name used for the profile of a user.**profile_UpdateProfileSubtype** is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdateProfileSubtype (
@partitionID uniqueidentifier
,@ProfileName nvarchar(250)
,@ProfileDisplayName nvarchar(400)
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@ProfileName: The Security Account Manager (SAM) user name of a user.

@ProfileDisplayName: The new value for the display name to be used for the profile of the user defined by *@ProfileName*.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.68 profile_UpdateTermSetIdForCoreProperty

The **profile_UpdateTermSetIdForCoreProperty** stored procedure changes the association between a **Property** and a term set.

profile_UpdateTermSetIdForCoreProperty is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdateTermSetIdForCoreProperty (
@partitionID uniqueidentifier
,@propertyName nvarchar(250)
,@termSetID uniqueidentifier
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@propertyName: The name of the Property for which to change the terminology set.

@termSetID: The identifier of the terminology set to associate with the property, or NULL to associate the property with no terminology set.

Return Values: An integer which MUST be zero.

Result Sets: MUST NOT return any result sets.

3.1.5.69 profile_GetOrganizationEvents

The **profile_GetOrganizationEvents** stored procedure returns organization change events.

profile_GetOrganizationEvents is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetOrganizationEvents (
@partitionID uniqueidentifier
,@RecordId bigint = null
,@ViewerRights int
,@MinEventId bigint = null
,@MinEventTime datetime = null
,@ChangeTypeMask int
,@ObjectTypeMask int
,@SortDescending int = 0
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

@RecordId: The record identifier of an organization or NULL. If this value is not NULL, the protocol server MUST return organization change events for the organization identified by the value. If the value is NULL, the protocol server MUST return all available organization change events for existing organizations.

@ViewerRights: An integer which MUST be 0.

@MinEventId: A unique identifier for organization change events. If the value of **@MinEventId** is not NULL then the protocol server MUST return events with identifiers greater than this value. If the value of **@MinEventId** is NULL then the value of **@MinEventTime** MUST NOT be NULL.

@MinEventTime: A value representing the UTC date and time for which all returned organization change events MUST be more recent. If this value is NULL then the protocol server MUST return all organization change events. If the value of **@MinEventId** is NULL then the value of **@MinEventTime** MUST NOT be NULL. If the value of **@MinEventId** is not NULL then the protocol server MUST ignore the value of **@MinEventId**.

@ChangeTypeMask: An integer that represents the types of changes that this stored procedure MUST return. This value MUST NOT be NULL. This value MUST be composed of zero or more values from the following table that have been combined using a bitwise or operation.

Value	Description
0x0	None
0x1	Add
0x2	Modify
Ox4	Delete
0xF	All

@ObjectTypeMask: An integer that represents the types of objects that this stored procedure MUST return changes about. This value MUST NOT be NULL. This value MUST be composed of zero or more values from the following table that have been combined using a bitwise or operation.

Value	Description
0x800	Organization Profile
0x1000	Organization Membership

@SortDescending: An integer that specifies the sort order of rows in the result sets returned by this stored procedure. If this value is 1 then the stored procedure MUST order results in descending order by the event identifier. For all input values other than 1, the stored procedure MUST order results in ascending order by event identifier.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

For the following combination of parameters,

@RecordId: NULL

@SortDescending: 1

This stored procedure MUST return a GetTopOrganizationEvent

This stored procedure MUST return a GetOrganizationEvents

For the following combination of parameters,

@RecordId: Non-NULL value.

@SortDescending: 1

This stored procedure MUST return a GetTopOrganizationEvent

This stored procedure MUST return a GetOrganizationEventsForRecordId

For the following combination of parameters,

@RecordId: NULL

@SortDescending: 0

This stored procedure MUST return a GetTopOrganizationEvent

This stored procedure MUST return a GetOrganizationEvents

For the following combination of parameters,

@RecordId: Non-NULL value.

@SortDescending: 0

This stored procedure MUST return a GetTopOrganizationEvent

This stored procedure MUST return a GetOrganizationEventsForRecordId

3.1.5.70 profile_GetOrganizationMembershipForUser

The **profile_GetOrganizationMembershipForUser** stored procedure retrieves a list of organization memberships for a given user. The user is represented by a Security Account Manager (SAM) user name and the partition (1) identifier.

profile_GetOrganizationMembershipForUser is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetOrganizationMembershipForUser (
@partitionID uniqueidentifier
,@NTName nvarchar(400)
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

@NTName: A Security Account Manager (SAM) user name representing the user.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a single profile_GetOrganizationMembershipForUser.ResultSet0

This stored procedure MUST return a single profile_GetOrganizationMembershipForUser.ResultSet1

This stored procedure MUST return a single profile_GetOrganizationMembershipForUser.ResultSet2

This stored procedure MUST return a single profile_GetOrganizationMembershipForUser.ResultSet3

3.1.5.71 profile_GetOrganizationMemberships

The **profile_GetOrganizationMemberships** stored procedure retrieves a list of memberships for an organization. The organization is represented by the organization record identifier and the partition identifier.

profile_GetOrganizationMemberships is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetOrganizationMemberships (
@partitionID uniqueidentifier
,@RecordID bigint
,@membershipType smallint = null
,@startRowIndex bigint = null
,@maxRows bigint = null
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

@RecordID: The organization record identifier. This value MUST NOT be NULL.

@membershipType: The membership type. If this value is specified, the protocol server MUST only return memberships with the specified membership type.

The value MUST exist in the following table:

Value	Description
1	Member membership type
2	Leader membership type

@startRowIndex: MUST be NULL..

@maxRows: MUST be NULL.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a OrganizationMemberships

3.1.5.72 profile_GetRootOrganization

The **profile_GetRootOrganization** stored procedure returns a list of root organizations. Root organizations are defined as those organizations whose parents are not defined or are users.

profile_GetRootOrganization is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetRootOrganization (
@partitionID uniqueidentifier
,@IncludeUserRooted bit = 0
,@Debug bit = 0
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

@IncludeUserRooted: If this value is set to 1, the protocol server MUST return organizations whose parent is a user in addition to the organization that does not have a parent. If this value is not set to 1, the protocol server MUST only return the organization that does not have a parent.

@Debug: This value MUST be zero.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a RootOrganization

3.1.5.73 profile_OrganizationMembersCount

The **profile_OrganizationMembersCount** stored procedure retrieves the count of members for an organization and optionally includes the members of its descendants in that count. The organization is represented by the organization record identifier and the partition (1) identifier.

profile_OrganizationMembersCount is defined using T-SQL syntax as follows:

```
PROCEDURE profile_OrganizationMembersCount (
@partitionID uniqueidentifier
,@OrganizationID bigint
,@membershipType smallint = null
,@isRecursive bit = null
,@MembersCount bigint OUTPUT
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

@OrganizationID: A 64-bit integer which is the identifier of the organization whose members should be counted. MUST NOT be NULL.

@membershipType: The type of members to include in the count. The count MUST only include members of the specified type if and only if @isRecursive is 0 and this parameter is not NULL. In any other case, members of both membership types MUST be included in the count.

The value MUST either be NULL or exist in the following table:

Value	Description
1	Member membership type
2	Leader membership type

@isRecursive: Whether to count the members of organizations which are descendants of the organization specified by @OrganizationID. Members of descendant organizations MUST be included in the count if this value is 1 or NULL. Members of descendant organizations MUST NOT be included if this value is 0.

@MembersCount: A 64-bit integer which MUST be equal to the number of members in the organization specified by @OrganizationID.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.74 profile_RemoveOrganization

The **profile_RemoveOrganization** removes all references to a single organization and its memberships on the protocol server.

profile_RemoveOrganization is defined using T-SQL syntax as follows:

```
PROCEDURE profile_RemoveOrganization (
@partitionID uniqueidentifier
,@RecordID bigint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

@RecordID: The organization record identifier. This value MUST not be NULL.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be in the following table. If the value is not equal to zero, the state of the protocol server MUST NOT be changed.

Value	Description
-1	The value of **@RecordID** does not represent an organization on the protocol server, or the protocol server was unable to retrieve the parent organization of the organization represented by the value of **@RecordID**.
0	The operation finished successfully.

Result Sets: MUST NOT return any result sets.

3.1.5.75 profile_RemoveUserFromOrganization

The **profile_RemoveUserFromOrganization** stored procedure removes a single membership to an organization for a user. If a membership to the organization specified for the user specified exists, the protocol server MUST delete the membership. If a membership to the organization specified for the user specified does not exist, the protocol server MUST do nothing.

Profile_RemoveUserFromOrganization is defined using T-SQL syntax as follows:

```
PROCEDURE profile_RemoveUserFromOrganization (
@partitionID uniqueidentifier
,@RecordID bigint
,@OrganizationID uniqueidentifier
,@UserRecordID bigint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

@RecordID: The organization record identifier. This value MUST NOT be NULL.

@OrganizationID: The organization unique identifier as returned from a call to the **profile_GetOrganizationalInfo**.

@UserRecordID: The user profile record identifier of the user to be removed. This value MUST NOT be NULL.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.76 profile_UpdateOrganizationProfileData

The **profile_UpdateOrganizationProfileData** stored procedure adds, updates, and removes one or more properties for an organization.

profile_UpdateUserProfileData is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdateOrganizationProfileData (
@partitionID uniqueidentifier
,@UpdatePropertyList ntext
,@Debug bit = 0
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

@UpdatePropertyList: Contains the list of properties that need to be added, updated, or removed. This value MUST NOT be null or empty, and MUST conform to the **UpdateOrganizationProfileData** XML schema as specified in section 2.2.6.4.5.

@Debug: An integer which MUST be 0.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.77 profile_GetContactEntry

The profile_GetContactEntry stored procedure retrieves the contact entry of the contact in a given partition (1) ID

```
PROCEDURE profile_GetContactEntry (
@partitionID uniqueidentifier
,@ContactID bigint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@ContactID: An identifier for the contact entry.

@correlationId: The optional request identifier for the current request. The value MUST be set to NULL and MUST be ignored by the server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a ContactEntry

3.1.5.78 profile_DeleteContactEntry

The **profile_DeleteContactEntry** stored procedure deletes the contact entry of the given DN from the given partition (1).

Profile_DeleteContactEntry is defined using T-SQL syntax as follows:

```
PROCEDURE profile_DeleteContactEntry (
@partitionID uniqueidentifier
,@DistinguishedName nvarchar(2048)
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@DistinguishedName: The LDAP standard DN [RFC2251] of the object. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.79 profile_EnumUsers_Full

The Profile_EnumUsers_Full stored procedure is called to retrieve the user profiles from the user profile store whose IDs are within the specified interval.

Profile_EnumUser_Full is defined using T-SQL syntax as follows:

```
PROCEDURE profile_EnumUsers_Full (
@partitionID uniqueidentifier
,@BeginID bigint
,@EndID bigint
,@MINID bigint OUTPUT
,@MAXID bigint OUTPUT
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@BeginID: The inclusive value where the Protocol server starts searching for existing user profile record identifiers. This parameter MUST be specified and it MUST NOT be NULL.

@EndID: The inclusive last value that the Protocol server searches for existing user profile record identifiers. This parameter MUST be specified and it MUST NOT be NULL.

@MINID: If the user profile store is empty the protocol server MUST ignore this parameter, otherwise the Protocol server MUST set this parameter to the smallest (first) user profile record identifier that exists.

@MAXID: If the user profile store is empty the server MUST ignore this parameter, otherwise the Protocol server MUST set this parameter to the biggest (last) user profile record identifier that exists.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a EnumUsersFull

3.1.5.80 profile_AddUserToOrganization

The **profile_AddUserToOrganization** stored procedure creates or updates a membership in an organization. If a membership in the organization for the specified user does not exist, the stored procedure MUST create a new membership for the user in the organization. If a membership in the organization for the specified user does exist, the stored procedure MUST update the membership type to the value in **@MembershipType**.

Profile_AddUserToOrganization is defined using T-SQL syntax as follows:

```
PROCEDURE profile_AddUserToOrganization (
@partitionID uniqueidentifier
,@RecordID bigint
,@OrganizationID uniqueidentifier
,@UserRecordID bigint
,@MembershipType smallint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

@RecordID: The organization record identifier. This value MUST NOT be NULL.

@OrganizationID: The organization unique identifier as returned from a call to the **profile_GetOrganizationalInfo**.

@UserRecordID: The user profile record identifier that represents the user. This value MUST NOT be NULL.

@MembershipType: The user membership type. The value MUST exist in the following table:

Value	Description
1	Member membership type
2	Leader membership type

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.81 profile_CreateOrganization

The **profile_CreateOrganization** stored procedure creates an organization with the specified name and partition (1) identifier and stores it on the protocol server.

profile_CreateOrganization is defined using T-SQL syntax as follows:

```
PROCEDURE profile_CreateOrganization (
@partitionID uniqueidentifier
,@ProfileName nvarchar(250)
,@RecordID bigint OUTPUT
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

@ProfileName: The string name of the organization. This value MUST not be NULL or empty, and MUST be less than 250 characters in length.

@RecordID: The organization record identifier. This value is assigned during the execution of this stored procedure, and MUST be stored on the protocol server. The organization record identifier assigned to each organization MUST be unique.

Value	Description
-1	The value that represents an invalid record identifier. The protocol server MUST return this value in the **@RecordID** parameter if a general error occurs and the organization is not created. If this value is returned this stored procedure MUST NOT make any changes to the state of the protocol server.
@@identity	The record identifier assigned to the organization by the protocol server.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.82 profile_GetOrganizationalInfo

The **profile_GetOrganizationalInfo** stored procedure MUST return three lists that comprise the organization's child, sibling, and ancestor organizations respectively. The lists of child and sibling organization MUST be ordered by organization display name. The list of ancestors MUST be ordered by the organization unique identifier in descending order.

profile_GetOrganizationalInfo is described using T-SQL syntax as follows:

```
PROCEDURE profile_GetOrganizationalInfo (
@partitionID uniqueidentifier
,@RecordID bigint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be "NULL" or empty.

@RecordID: The organization record identifier. This value MUST NOT be "NULL" or empty.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be "0".

Result Sets:

This stored procedure MUST return a OrganizationIds

This stored procedure MUST return a Siblings

This stored procedure MUST return a Ancestors

3.1.5.83 profile_GetOrganizationData

The **profile_GetOrganizationData** stored procedure returns the organization record identifier, the profile subtype identifier, the count of child organizations, and details on each property stored as part of the organization. For each organization property, this stored procedure MUST return the property name and property value.

profile_GetOrganizationData is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetOrganizationData (
@partitionID uniqueidentifier
,@OrgID uniqueidentifier
,@RecordID bigint = null
,@bDebug bit = 0
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID used to filter the current request. This value MUST NOT be NULL or empty.

@OrgID: The organization unique identifier. If this value is NULL or empty, the value of **@RecordID** MUST NOT be NULL or empty.

@RecordID: The organization record identifier. If **@OrgID** is NULL or empty, this value MUST NOT be NULL or empty.

@bDebug: An integer which MUST be zero.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be zero.

Result Sets:

This stored procedure MUST return a OrganizationData

This stored procedure MUST return a OrganizationProperties

3.1.5.84 proc_GetOrganizationMembershipsForUsers

The **proc_GetOrganizationMembershipsForUsers** stored procedure returns a set of organization memberships and information about the organizations associated to those memberships for a set of up to 200 users. The protocol client MUST specify each user profile record identifier using the parameters **@userRecordId1** through **@userRecordId200**. The protocol server MUST ignore any parameter in the set **@userRecordId1** through **@userRecordId200** with a value that is NULL or empty.

proc_GetOrganizationMembershipsForUsers is defined using T-SQL syntax as follows:

```
PROCEDURE proc_GetOrganizationMembershipsForUsers (
@partitionID uniqueidentifier
,@userRecordId1 int = null
,@userRecordId2 int = null
,@userRecordId3 int = null
,@userRecordId4 int = null
,@userRecordId5 int = null
,@userRecordId6 int = null
,@userRecordId7 int = null
,@userRecordId8 int = null
,@userRecordId9 int = null
,@userRecordId10 int = null
,@userRecordId11 int = null
,@userRecordId12 int = null
,@userRecordId13 int = null
,@userRecordId14 int = null
,@userRecordId15 int = null
,@userRecordId16 int = null
,@userRecordId17 int = null
,@userRecordId18 int = null
,@userRecordId19 int = null
,@userRecordId20 int = null
,@userRecordId21 int = null
,@userRecordId22 int = null
,@userRecordId23 int = null
,@userRecordId24 int = null
,@userRecordId25 int = null
,@userRecordId26 int = null
,@userRecordId27 int = null
,@userRecordId28 int = null
,@userRecordId29 int = null
,@userRecordId30 int = null
,@userRecordId31 int = null
,@userRecordId32 int = null
,@userRecordId33 int = null
,@userRecordId34 int = null
,@userRecordId35 int = null
,@userRecordId36 int = null
,@userRecordId37 int = null
,@userRecordId38 int = null
,@userRecordId39 int = null
,@userRecordId40 int = null
,@userRecordId41 int = null
,@userRecordId42 int = null
,@userRecordId43 int = null
,@userRecordId44 int = null
,@userRecordId45 int = null
```

```
  ,@userRecordId46 int = null
  ,@userRecordId47 int = null
  ,@userRecordId48 int = null
  ,@userRecordId49 int = null
  ,@userRecordId50 int = null
  ,@userRecordId51 int = null
  ,@userRecordId52 int = null
  ,@userRecordId53 int = null
  ,@userRecordId54 int = null
  ,@userRecordId55 int = null
  ,@userRecordId56 int = null
  ,@userRecordId57 int = null
  ,@userRecordId58 int = null
  ,@userRecordId59 int = null
  ,@userRecordId60 int = null
  ,@userRecordId61 int = null
  ,@userRecordId62 int = null
  ,@userRecordId63 int = null
  ,@userRecordId64 int = null
  ,@userRecordId65 int = null
  ,@userRecordId66 int = null
  ,@userRecordId67 int = null
  ,@userRecordId68 int = null
  ,@userRecordId69 int = null
  ,@userRecordId70 int = null
  ,@userRecordId71 int = null
  ,@userRecordId72 int = null
  ,@userRecordId73 int = null
  ,@userRecordId74 int = null
  ,@userRecordId75 int = null
  ,@userRecordId76 int = null
  ,@userRecordId77 int = null
  ,@userRecordId78 int = null
  ,@userRecordId79 int = null
  ,@userRecordId80 int = null
  ,@userRecordId81 int = null
  ,@userRecordId82 int = null
  ,@userRecordId83 int = null
  ,@userRecordId84 int = null
  ,@userRecordId85 int = null
  ,@userRecordId86 int = null
  ,@userRecordId87 int = null
  ,@userRecordId88 int = null
  ,@userRecordId89 int = null
  ,@userRecordId90 int = null
  ,@userRecordId91 int = null
  ,@userRecordId92 int = null
  ,@userRecordId93 int = null
  ,@userRecordId94 int = null
  ,@userRecordId95 int = null
  ,@userRecordId96 int = null
  ,@userRecordId97 int = null
  ,@userRecordId98 int = null
  ,@userRecordId99 int = null
  ,@userRecordId100 int = null
  ,@userRecordId101 int = null
  ,@userRecordId102 int = null
  ,@userRecordId103 int = null
  ,@userRecordId104 int = null
```

```
  ,@userRecordId105 int = null
  ,@userRecordId106 int = null
  ,@userRecordId107 int = null
  ,@userRecordId108 int = null
  ,@userRecordId109 int = null
  ,@userRecordId110 int = null
  ,@userRecordId111 int = null
  ,@userRecordId112 int = null
  ,@userRecordId113 int = null
  ,@userRecordId114 int = null
  ,@userRecordId115 int = null
  ,@userRecordId116 int = null
  ,@userRecordId117 int = null
  ,@userRecordId118 int = null
  ,@userRecordId119 int = null
  ,@userRecordId120 int = null
  ,@userRecordId121 int = null
  ,@userRecordId122 int = null
  ,@userRecordId123 int = null
  ,@userRecordId124 int = null
  ,@userRecordId125 int = null
  ,@userRecordId126 int = null
  ,@userRecordId127 int = null
  ,@userRecordId128 int = null
  ,@userRecordId129 int = null
  ,@userRecordId130 int = null
  ,@userRecordId131 int = null
  ,@userRecordId132 int = null
  ,@userRecordId133 int = null
  ,@userRecordId134 int = null
  ,@userRecordId135 int = null
  ,@userRecordId136 int = null
  ,@userRecordId137 int = null
  ,@userRecordId138 int = null
  ,@userRecordId139 int = null
  ,@userRecordId140 int = null
  ,@userRecordId141 int = null
  ,@userRecordId142 int = null
  ,@userRecordId143 int = null
  ,@userRecordId144 int = null
  ,@userRecordId145 int = null
  ,@userRecordId146 int = null
  ,@userRecordId147 int = null
  ,@userRecordId148 int = null
  ,@userRecordId149 int = null
  ,@userRecordId150 int = null
  ,@userRecordId151 int = null
  ,@userRecordId152 int = null
  ,@userRecordId153 int = null
  ,@userRecordId154 int = null
  ,@userRecordId155 int = null
  ,@userRecordId156 int = null
  ,@userRecordId157 int = null
  ,@userRecordId158 int = null
  ,@userRecordId159 int = null
  ,@userRecordId160 int = null
  ,@userRecordId161 int = null
  ,@userRecordId162 int = null
  ,@userRecordId163 int = null
```

```
  ,@userRecordId164 int = null
  ,@userRecordId165 int = null
  ,@userRecordId166 int = null
  ,@userRecordId167 int = null
  ,@userRecordId168 int = null
  ,@userRecordId169 int = null
  ,@userRecordId170 int = null
  ,@userRecordId171 int = null
  ,@userRecordId172 int = null
  ,@userRecordId173 int = null
  ,@userRecordId174 int = null
  ,@userRecordId175 int = null
  ,@userRecordId176 int = null
  ,@userRecordId177 int = null
  ,@userRecordId178 int = null
  ,@userRecordId179 int = null
  ,@userRecordId180 int = null
  ,@userRecordId181 int = null
  ,@userRecordId182 int = null
  ,@userRecordId183 int = null
  ,@userRecordId184 int = null
  ,@userRecordId185 int = null
  ,@userRecordId186 int = null
  ,@userRecordId187 int = null
  ,@userRecordId188 int = null
  ,@userRecordId189 int = null
  ,@userRecordId190 int = null
  ,@userRecordId191 int = null
  ,@userRecordId192 int = null
  ,@userRecordId193 int = null
  ,@userRecordId194 int = null
  ,@userRecordId195 int = null
  ,@userRecordId196 int = null
  ,@userRecordId197 int = null
  ,@userRecordId198 int = null
  ,@userRecordId199 int = null
  ,@userRecordId200 int = null
  ,@correlationGuid uniqueidentifier = null
  );
```

@partitionID: GUID used to filter the current request. This value MUST NOT be NULL or empty.

@userRecordId1: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId2: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId3: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId4: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId5: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId6: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId7: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId8: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId9: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId10: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId11: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId12: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId13: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId14: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId15: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId16: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId17: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId18: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId19: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId20: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId21: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId22: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId23: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId24: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId25: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId26: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId27: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId28: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId29: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId30: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId31: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId32: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId33: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId34: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId35: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId36: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId37: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId38: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId39: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId40: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId41: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId42: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId43: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId44: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId45: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId46: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId47: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId48: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId49: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId50: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId51: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId52: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId53: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId54: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId55: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId56: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId57: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId58: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId59: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId60: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId61: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId62: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId63: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId64: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId65: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId66: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId67: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId68: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId69: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId70: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId71: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId72: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId73: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId74: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId75: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId76: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId77: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId78: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId79: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId80: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId81: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId82: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId83: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId84: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId85: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId86: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId87: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId88: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId89: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId90: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId91: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId92: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId93: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId94: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId95: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId96: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId97: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId98: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId99: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId100: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId101: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId102: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId103: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId104: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId105: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId106: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId107: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId108: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId109: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId110: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId111: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId112: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId113: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId114: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId115: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId116: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId117: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId118: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId119: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId120: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId121: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId122: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId123: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId124: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId125: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId126: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId127: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId128: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId129: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId130: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId131: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId132: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId133: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId134: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId135: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId136: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId137: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId138: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId139: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId140: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId141: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId142: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId143: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId144: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId145: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId146: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId147: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId148: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId149: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId150: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId151: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId152: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId153: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId154: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId155: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId156: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId157: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId158: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId159: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId160: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId161: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId162: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId163: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId164: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId165: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId166: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId167: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId168: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId169: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId170: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId171: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId172: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId173: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId174: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId175: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId176: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId177: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId178: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId179: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId180: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId181: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId182: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId183: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId184: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId185: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId186: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId187: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId188: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId189: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId190: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId191: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId192: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId193: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId194: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId195: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId196: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId197: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId198: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId199: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@userRecordId200: Contains a user profile record identifier. The protocol server MUST ignore this value if it is NULL.

@correlationGuid: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a BulkOrganizationMemberships

This stored procedure MUST return a BulkOrganizationInformation

3.1.5.85 profile_UpdateMySiteDeletionSchedule

This **profile_UpdateMySiteDeletionSchedule** stored procedure updates the notification status of a site, specified by the **@SiteID**, to a new status specified by **@Status.**

```
PROCEDURE profile_UpdateMySiteDeletionSchedule (
@PartitionID uniqueidentifier
,@SiteID uniqueidentifier
,@Status tinyint
);
```

@PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@SiteID: A GUID used to specify the site. This value MUST NOT by null or empty.

@Status: The notification status of which the site needs to be updated to. The value MUST be 1 for first email sent, the value MUST be 2 for reminder of email sent.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.86 profile_RemoveMySiteDeletionSchedule

This **profile_RemoveMySiteDeletionSchedule** stored procedure removed the deletion status entry of a site, specified by **@SiteID.**

```
PROCEDURE profile_RemoveMySiteDeletionSchedule (
@PartitionID uniqueidentifier
,@SiteID uniqueidentifier
```

```
);
```

@PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@SiteID: A GUID used to specify the site. This value MUST NOT be null or empty.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.87 profile_ScheduleMySiteForDeletion

The profile_ScheduleMySiteForDeletion stores data associated with an orphaned MySite.

```
PROCEDURE profile_ScheduleMySiteForDeletion (
@PartitionID uniqueidentifier
,@SiteID uniqueidentifier
,@DisplayName nvarchar(400)
,@Email nvarchar(256)
);
```

@PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@SiteID: A GUID used to specify the site. This value MUST NOT be null or empty.

@DisplayName: The name of the original owner of the site.

@Email: The e-mail address of the new owner.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.88 profile_QueryMySiteDeletionSchedule

The **profile_QueryMySiteDeletionSchedule** stored procedure returns a set of notification status entries. It MUST return 0 or more rows that match the input parameters.

```
PROCEDURE profile_QueryMySiteDeletionSchedule (
@PartitionID uniqueidentifier
,@Status tinyint
,@CreatedBefore datetime
);
```

@PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@Status: A notification status used to filter the current request. The value MUST be 1 for first email sent, the value MUST be 2 for reminder of email sent.

@CreatedBefore: A UTC value used to filter the current request by the date the deletion request was created.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a QueryMySiteDeletionSchedule

3.1.5.89 profile_Admin_GetProfileStatistics

The **profile_Admin_GetProfileStatistics** stored procedure is called to get the count of partitions (1), user profiles and **organization profiles** contained in the user profile store.

```
PROCEDURE profile_Admin_GetProfileStatistics (
@tenantCount int OUTPUT
,@userProfileCount int OUTPUT
,@orgProfileCount int OUTPUT
,@correlationId uniqueidentifier = null
);
```

@tenantCount: An integer output value that will receive the count of partitions (1) contained in the user profile store. Any input value MUST be ignored.

@userProfileCount: An integer output value that will receive the count of user profiles contained in the user profile store. Any input value MUST be ignored.

@orgProfileCount: An integer output value that will receive the count of organization profiles contained in the user profile store. Any input value MUST be ignored.

@correlationId: The optional request identifier for the current request.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.90 profile_GetLeaders

The profile_GetLeaders stored procedure retrieves all the leaders in the given partition (1).

profile_GetLeaders is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetLeaders (
@PartitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a GetLeaders result set.

3.1.5.91 profile_AddLeader

The profile_AddLeader stored procedure designates a profile as a leader in the given partition (1).

profile_AddLeader using T-SQL syntax as follows:

```
PROCEDURE profile_AddLeader (
@PartitionID uniqueidentifier
,@NTName nvarchar(400)
,@correlationId uniqueidentifier = null
);
```

@PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@NTName: The Security Account Manager (SAM) user name of the profile to designate as a leader.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

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

Value	Description
1	The specified leader already exists in the protocol store.
2	The specified *@NTName* does not exist as a profile in the protocol store. Unable to add as a leader.
3	The specified leader has a manager in the protocol store. Unable to add as a leader.
0	The specified leader was successfully added to the protocol store.

Result Sets: MUST NOT return any result sets.

3.1.5.92 profile_RemoveLeader

The profile_RemoveLeader stored procedure removes the leader designation of a profile with the given Security Account Manager (SAM) user name from the given partition (1).

profile_RemoveLeader is defined using T-SQL syntax as follows:

```
PROCEDURE profile_RemoveLeader (
@PartitionID uniqueidentifier
,@NTName nvarchar(400)
,@correlationId uniqueidentifier = null
);
```

@PartitionID: A GUID used to filter the current request. This value MUST NOT be null or empty.

@NTName: The Security Account Manager (SAM) user name of the profile to remove as a leader.

@correlationId: The optional request identifier for the current request. The value MUST be ignored by the protocol server.

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

Value	Description
0	The specified leader was successfully removed from the protocol store.

Value	Description
1	The specified leader was not already a leader in the protocol store.

Result Sets: MUST NOT return any result sets.

3.1.6 Timer Events

No timer events impact the operation of this protocol.

3.1.7 Other Local Events

No other local events impact the operation of this protocol.

3.2 Client Details

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

None.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

This section provides specific example scenarios for managing user profiles, managing user profile properties, enumerating types of user data (links, colleagues, memberships), and managing service-level settings pertaining to policy. In conjunction with the detailed protocol documentation described in the reference documents, this information is intended to provide a comprehensive view of these topics.

4.1 Creating and Updating a User Profile Property

The following examples show how to create and update a user profile property in the user profile store. Then the example shows how to update the user display name and description in different languages associated with a property.

4.1.1 Creating a Property

To create a property, the protocol client uses the **profile_UpdateProperty** stored procedure by setting the *@RemovePropertyList* parameter to NULL and the *@UpdatePropertyList* to valid XML code that adheres to the appropriate schema. The T-SQL syntax is:

```
exec dbo.profile_UpdateProperty
@partitionID='0C37852B-34D0-418E-91C6-2AC25AF4BE5B',
@correlationId='F20ED392-AAE4-4845-9D50-F5BBAAB75E08',
@RemovePropertyList=NULL,
@UpdatePropertyList=N
'<?xml version="1.0" encoding="utf-16"?>
<MSPROFILE>
<PROPERTY PropertyName="TestProperty" bUpdate="0" PropertyType="3" ID="1" DefaultPrivacy="16"
UserOverridePrivacy="0" PrivacyPolicy="1" IsEditable="1" IsAdminEditOnly="1" IsUpgrade="0"
IsUpgradePrivate="0" />
</MSPROFILE>'
```

4.1.2 Localizing a User Display Name and Description

The protocol server allows for the User Display Names and descriptions to be localized in different languages. The **profile_GetProfilePropertyLoc** stored procedure can be used to get all localized user display names and descriptions for all properties and sections. The T-SQL syntax is:

```
exec dbo.profile_GetProfilePropertyLoc
```

The **profile_UpdatePropertyLoc** stored procedure is called to add new localized User Display Names or Descriptions. The T-SQL syntax is:

```
exec dbo.profile_UpdatePropertyLoc
@PropertyName=N'TestProperty',
@DisplayNamesXml=N
'<?xml version="1.0" encoding="utf-16"?>
<Loc>
  <Item Lcid="1033" Text="Test Property Display Name" />
  <Item Lcid="1036" Text="Test Property Display Name in French" />
  <Item Lcid="1031" Text="Test Property Display Name in German" />
</Loc>',
@DescriptionsXml=N
'<?xml version="1.0" encoding="utf-16"?>
```

```
<Loc>
  <Item Lcid="1033" Text="Description in English" />
  <Item Lcid="1036" Text="Description in French" />
  <Item Lcid="1031" Text="Description in German" />
</Loc>'
```

4.2 Enumerating Users

A protocol client can enumerate the users with Paging.

To do that the protocol client would call the **profile_EnumUsers** stored procedure, passing the following parameters:

@BeginId= 0

@EndID= desired page size

Each call to this method will return to parameters *@MINID* and *@MAXID* that can be used to determine when the end of the users list is reached. The protocol client will need to keep calling this stored procedure, incrementing the *@BeginId* and @EndID parameters to get the next page of users:

@BeginId= *@EndID* + 1

@EndID= *@BeginId* + desired page size

The **profile_EnumUsers** stored procedure returns the GUID identifying the user and the user profile record identifier for each user in the specified page.

There is a chance that the protocol client will get fewer rows in the result that the page size, if there are gaps in user profile record identifiers. This can happen if some user profiles were deleted.

Then the protocol client can use those values to read the user profile data for each user returned by the preceding call(s).

To obtain user profile details, the protocol client calls the **profile_GetUserProfileData** stored procedure setting the parameter *@UserId* to the GUID identifying the user returned by the **profile_EnumUsers** stored procedure.

4.3 Get Profile Statistics

This protocol can be used to gather quick statistics about the number of users currently stored. Consider the following T-SQL syntax call which a protocol client can make to profile_Admin_GetProfileStatistics.

```
declare @tenantCount int
declare @userProfileCount int
declare @orgProfileCount int

set @correlationId = '806597C7-2A34-4BB3-A807-A8664115E8D1'

exec profile_Admin_GetProfileStatistics
@tenantCount OUTPUT,
@userProfileCount OUTPUT,
@orgProfileCount OUTPUT,
@correlationId='806597C7-2A34-4BB3-A807-A8664115E8D1'
```

```
select @tenantCount as tenantcount, @userProfileCount as userprofilecount, @orgProfileCount
as orgprofilecount
```

The protocol server returns a value of 0, which is ignored by the client. Consider also the following example output data for this call.

```
tenantCount
2

userProfileCount
2

orgProfileCount
3
```

4.4 Managing Links Between Users

4.4.1 Adding User Colleagues

A protocol client can define new colleagues for a specific user. To do this the protocol client would call the **QuickLinksAdd** stored procedure, passing in the following parameters:

@RecordId = The user profile record identifier for which the colleague is added. For this a protocol client can simply use a user profile record identifier retrieved through the method described in example 4.2.

@PageURL = Not specified. NULL by default.

@Title = Not specified. NULL by default.

@ContentClass = not specified (it will be NULL by default.)

@Group = 'General' (string value).

@GroupType = 0 (integer value).

@ItemSecurity = 16 (integer value).

@PolicyId = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C' (GUID value)

@ColleagueRecordId = the user profile record identifier of the desired colleague.

@LinkUserIdIsWorkgroup = 0 (integer value)

@LinkMemberGroupId = not specified (it will be NULL by default)

This call will create a link between the two users, making the user specified by *@ColleagueRecordId* a colleague of the user specified by *@RecordId*.

The owner of this link is the user profile identified by *@RecordId*.

The colleague is added to the "General" group.

The link has a privacy setting of 'Private' because @ItemSecurity was set to 16.

The stored procedure returns the *@LinkID* of the newly created link.

4.4.2 Deleting User Site Memberships

A protocol client can delete a user site membership. Before deleting the membership, a protocol client can list all the site memberships. To do this, the protocol client needs to call the **QuickLinksRetrieveAllItems** stored procedure with the following parameters:

@RecordId = the user profile record identifier for which the list of memberships is to be retrieved.

@PartitionID = A GUID used to filter the current request.

@CorrelationID = The optional request identifier for the current request.

This call will return all the specified user site memberships. Then the protocol client can find among those the 1 that needs to be deleted and call the **QuickLinksDelete** stored procedure and pass the following parameters:

@RecordId = the user profile record identifier for which the membership will be deleted.

@LinkID = the ID of the site membership link (integer value) to be deleted. Here the protocol client will need to pass the value found in the column "ID" in the third result set of the stored procedure call described previously.

@PolicyId = 8BB1220F-DE8B-4771-AC3A-0551242CF2BD (GUID value)

@RemoveCount = 1 (integer value)

The stored procedure will try to delete in this case a site membership, because the specified *@PolicyId* identifies a site membership. If the stored procedure returns 0 the site membership has been successfully deleted.

4.4.3 Retrieving All Colleagues of Colleagues

A protocol client can retrieve the list of colleague properties of colleague properties, for example to search for a contact in another department. To do this the protocol client will have to call **QuickLinksRetrieveColleaguesOfColleagues** stored procedure and pass the following parameter:

@RecordId: The user profile record identifier of the user for which the list of colleague properties colleague properties is obtained.

The stored procedure will return a list with user profile information about each user that is a colleague of any of the colleague properties of the specified user.

The protocol client can then call the **profile_GetUserProfileData** stored procedure, setting the parameter *@UserId* to any of the GUIDs identifying the users returned by the **QuickLinksRetrieveColleaguesOfColleagues** stored procedure, to obtain the user's Department or other user profile details.

4.5 Managing Membership

4.5.1 Enumerating Member Groups

In this example the store contains these membership groups:

ID	Source	Display Name	Email Address
21	distribution list	Group A	groupA@contoso.com
22	distribution list	Group B	
23	distribution list	Group C	groupC@contoso.com
25	site	Group D	
28	site	Group E	
29	distribution list	Group F	groupF@contoso.com

In the group enumeration example the protocol client begins by obtaining the maximum and minimum bounds of the enumeration. The protocol client does this by invoking membership_enumerateGroups(NULL, NULL, *@LowerBound*, *@UpperBound*). This results in the protocol client receiving no result set but with the value of *@LowerBound* being set to 21 and the value of *@UpperBound* being set to 29.

From there, the protocol client enumerate selects an enumeration set size. This example uses a set size of 4. The protocol client then calculates bound using the value of *@LowerBound*, the selected enumeration set size and the value of *@UpperBound*. The protocol client then establishes an enumeration set. The protocol client does this by invoking membership_enumerateGroups(21, 25, NULL, NULL)<1>. This results in the protocol client obtaining the following result set:

Id
21
23
25

Note that Group B is not included as it is does not have an e-mail address. The protocol client can then use these Member Group Identifiers to perform most membership group operations.

Once the protocol client has exhausted the enumeration it then calculates new bound using the previous bounds, the enumeration set size and value of *@UpperBound*. The protocol client then refreshes the enumeration set. The protocol client does this by invoking membership_enumerateGroups(26, 29, NULL, NULL). This results in the protocol client obtaining following result set:

Id
28
29

Note that at this point the enumeration set upper bound has equaled the *@UpperBound* value. All that remains if for the protocol client to exhaust this enumeration set and the enumeration of all membership groups is complete.

4.5.2 Creating a Member Group

A protocol client creates the following membership group<2>:

DisplayName	MailNickName	Source	SourceReference	Url	Type
Group Q	groupQ	A88B9DCB-5B82-41E4-8A19-17672F307B95	CN=GroupQ,OU=Distribution Lists,DC=contoso,DC=com	mailto:groupQ@contoso.com	0

To create the membership group, the protocol client calls **membership_updateGroup** using NULL as the value of the *@Id* input parameter and with all other input parameters set appropriately.

The protocol client makes the following call.

```
membership_updateGroup(NULL, 'A88B9DCB-5B82-41E4-8A19-17672F307B95',
    'Group Q', 'groupQ', 'This is a group with name Q',
                'mailto:groupQ@contoso.com',
    'CN=GroupQ,OU=Distribution Lists,DC=contoso,DC=com',
                NULL, 0, 0, NULL, @CreatedMemberGroupId, @PossibleErrorCode)
```

This results in the protocol client receiving no result set but with the values of *@CreatedMemberGroupId* and *@PossibleErrorCode* being set. The protocol client then examines the value of the return code and the value of *@PossibleErrorCode*. If both are 0 then the value of *@CreatedMemberGroupId* can be used by the protocol client for further membership group operations. If they are not both 0, then the protocol client can take appropriate action.

When creating a membership group, there is no requirement for unique user display names. However, not having a unique user display name can make invoking **membership_getGroup** more difficult.

4.5.3 Updating a Membership Group

In this example the store contains this membership group<3>:

Id	DisplayName	MailNickName	Source	SourceReference	Url	Type
439	Group U	(null)	A88B9DCB-5B82-41E4-8A19-17672F307B95	cn=groupU,ou=useraccounts,dc=contoso,dc=com	mailto:	1

To update a membership group, the protocol client calls **membership_updateGroup** using the Member Group Identifier as the value of the *@Id* input parameter. All other input parameters are set to either their updated values or their current values. In this example, an e-mail address is assigned to the group.

To make the desired change, the protocol client calls membership_updateGroup(439, 'A88B9DCB-5B82-41E4-8A19-17672F307B95', 'Group U', 'groupU', 'This is a group with name U', 'mailto:groupU@contoso.com', 'CN=GroupU,OU=Distribution Lists,DC=contoso,DC=com', NULL, 0, 0, NULL, NULL, *@PossibleErrorCode*). This results in the protocol client receiving no result set but with the values of *@PossibleErrorCode* being set.

The protocol client then examines the value of the return code and the value of *@PossibleErrorCode*. If both are 0 then the group has been updated to have an e-mail address. If they are not both 0 then the protocol client can take appropriate action.

When assigning an e-mail address to a group, it is crucial to remember to also update the group's type from 1 to 0. If this is not done, inconsistent behavior can result.

When updating a membership group be aware that there is no requirement for uniqueness of user display names but not having a unique user display name can make invoking **membership_getGroup** more difficult.

4.5.4 Retrieving Membership Data

In this example, the data is as follows:

- The user profile store contains these 5 user profiles representing 5 different accounts:

RecordID	PreferredName
4312453	Bob Robertson
3452432	Ed Williams
1434312	Fred Fleinhart
5432525	Fred Fleinhart
6635545	Fred Fleinhart
2341241	Steve Steveson

- The store contains this member group:

Id	DisplayName
72	Group M

To retrieve a partial list of members of a membership group the protocol client calls membership_getGroupMembershipsPaged. For this example the protocol client will set the *@Count* parameter to 3 to illustrate the use of the *@ItemBeforeFirst* and *@RecordIdBeforeFirst* parameters. The protocol client, initially, calls membership_getGroupMembershipsPaged(72, 4312453, 3, 7, 0, NULL, NULL, NULL). This results in the protocol client obtaining following result set<4>:

PreferredName	RecordID	DisplayName	MemberCount
Bob Robertson	4312453	Group M	6
Ed Williams	3452432	Group M	6
Fred Fleinhart	1434312	Group M	6

The protocol client then calls membership_getGroupMembershipsPaged(72, 4312453, 3, 7, 0, 'Fred Fleinhart', 1434312, NULL) to obtain the remaining memberships. This results in the protocol client obtaining following result set<5>:

PreferredName	RecordID	DisplayName	MemberCount
Fred Fleinhart	5432525	Group M	6
Fred Fleinhart	6635545	Group M	6
Steve Steveson	2341241	Group M	6

To retrieve a full list of members of a membership group the protocol client calls membership_getGroupMemberships. The protocol client calls membership_getGroupMemberships(72). This results in the protocol client obtaining following result set<6>:

RecordId	DisplayName	MemberCount	PreferredName
4312453	Group M	6	Bob Robertson
3452432	Group M	6	Ed Williams
1434312	Group M	6	Fred Fleinhart
5432525	Group M	6	Fred Fleinhart
6635545	Group M	6	Fred Fleinhart
2341241	Group M	6	Steve Steveson

4.6 Managing User Profile Data

4.6.1 Retrieving a User GUID

A protocol client can obtain a GUID identifying the user for a user's primary account.

To achieve this, the protocol client calls **profile_GetUserGUID** stored procedure passing at least 1 of the following parameters:

@PartitionID: A GUID used to filter the current request.

@CorrelationID: The optional request identifier for the current request.

@NTName

@SID

The **profile_GetUserGUID** stored procedure will take as input either the user's login name or SID. If login name is specified then its value will be used and the SID will be ignored, otherwise the SID will be used.

The call to **profile_GetUserGUID** returns output parameter GUID (*@GUID*) that will specify the GUID value of the user's userID. This value can be used later to call other stored procedures**.** Note: If a valid user is not found, the output parameter *@GUID* will be NULL.

4.6.2 Retrieving User Profile Data

A protocol client can obtain the user profile data for the user. Profile data could be a user's home address, telephone number, and so forth.

To achieve this the protocol client calls **profile_G**etUserProfileData** stored procedure passing at least 1 of the following parameters:

@UserId, *@NTName*, *@SID*, and *@RecordId*.

@PartitionID: A GUID used to filter the current request.

@CorrelationID: The optional request identifier for the current request.

The **profile_GetUserProfileData** stored procedure will take as input either a GUID, login name, SID or matching record identifier for the requested user. If *@UserId* is NULL, then the user will be identified by *@SID*. If *@SID* is also NULL, then the user will be identified by *@NTName*. If *@NTName* is also NULL, then the user will be identified by *@RecordId*.

The call to **profile_GetUserProfileData** returns a **result set** describing the available properties for the specified **user**. If no properties are found for a specified **user**, then the **result set** will contain 0 rows. Otherwise it will contain as many rows as there are properties for the specified **user**.

4.6.3 Updating User Profile Data

A protocol client can add, update or delete the user profile data for a user. Profile data could be a user's home address, telephone number, and so forth.

To achieve this, the protocol client calls **profile_UpdateUserProfileData** stored procedure passing the *@UpdatePropertyList* parameter. *@UpdatePropertyList* parameter contains only the XML text describing the list of properties that need to be updated. This can be used for 1 or more users.

The *@UpdatePropertyList* XML parameter can be constructed in such a way to either add, update or delete properties. If the PROPERTY xml element attribute "RemoveFlag" is 1, then the property record is removed; in a multiple values field, all values are removed. If the USER xml element attribute "UserID" is empty, then a new user is created with the specified property information.

The following is an example of XML code which adds a user.

```
@UpdatePropertyList=N'<?xml version="1.0" encoding="utf-16"?>
<MSPROFILE>
  <PROFILE ProfileName="UserProfile">
    <USER NewUser="1" NTAccount="<Some Domain>\User1"
          UserID="12345678-1234-400e-b72e-88887c9a6f31">
      <PROPERTY PropertyName="AccountName" PropertyValue="="<Some Domain>\User1" />
      <PROPERTY PropertyName="LastName" PropertyValue="Smith" />
      <PROPERTY PropertyName="FirstName" PropertyValue="John" />
      <PROPERTY PropertyName="PreferredName" PropertyValue="John Smith" />
      <PROPERTY PropertyName="WorkPhone"
                PropertyValue="+1 (425) 1(425)5550100 x50100 " />
      <PROPERTY PropertyName="Office" PropertyValue="88/1234" />
      <PROPERTY PropertyName="Department" PropertyValue="Office" />
      <PROPERTY PropertyName="Title" PropertyValue="Vice President" />
      <PROPERTY PropertyName="Manager" PropertyValue="<Some Domain>\User1" />
      <PROPERTY PropertyName="UserName" PropertyValue="user1" />
      <PROPERTY PropertyName="PublicSiteRedirect"
                PropertyValue="http://my/sites/user1/" />
      <PROPERTY PropertyName="SPS-SipAddress" PropertyValue="user1@contoso.com" />
      <PROPERTY PropertyName="SPS-ProxyAddresses"
                PropertyValue="X500:/o=MSNBC/ou=Servers/cn=Recipients/cn=user1" />
      <PROPERTY PropertyName="SPS-ProxyAddresses"
                PropertyValue="X500:/O=Microsoft/OU=APPS-WGA/cn=Recipients/cn=JohnSmith" />
      <PROPERTY PropertyName="SPS-ProxyAddresses"
```

```
                PropertyValue="gwise:Exchange.APPS-WGA.user1" />
        <PROPERTY PropertyName="SPS-ProxyAddresses"
                PropertyValue="smtp:user1@contoso.com" />
        <PROPERTY PropertyName="SPS-ProxyAddresses"
                PropertyValue="SMTP:user1@exchange.contoso.com" />
        <PROPERTY PropertyName="WorkEmail" PropertyValue="user1@contoso.com" />
        <PROPERTY PropertyName="Userprofile_GUID"
                PropertyValue="12345678-1234-abcd-1234-123456781234" />
      </USER>
    </PROFILE>
  </MSPROFILE>'
```

The following is an example of XML code which updates a user profile:

```
@UpdatePropertyList=N'<?xml version="1.0" encoding="utf-16"?>
<MSPROFILE>
  <PROFILE ProfileName="UserProfile">
    <USER NewUser="0" NTAccount="="<Some Domain>\User1"
          UserID="12345678-1234-400e-b72e-88887c9a6f31 ">
      <PROPERTY PropertyName="Manager" PropertyValue="="<Some Domain>\User2" />
      <PROPERTY PropertyName="AboutMe" PropertyValue="&lt;div&gt;&lt;/div&gt;" />
      <PROPERTY PropertyName="SPS-Dotted-line" PropertyValue="" />
      <PROPERTY PropertyName="SPS-HireDate" PropertyValue="" />
      <PROPERTY PropertyName="SPS-LastColleagueAdded" PropertyValue="" />
      <PROPERTY PropertyName="SPS-OWAUrl" PropertyValue="http://www.someurl.com" />
      <PROPERTY PropertyName="Assistant" PropertyValue="" />
      <PROPERTY PropertyName="xx-html" PropertyValue="&lt;div&gt;&lt;/div&gt;" />
      <PROPERTY PropertyName="xx-html2-2000"
                PropertyValue="&lt;div&gt;&lt;/div&gt;" />
      <PROPERTY PropertyName="xx-html3-555"
                PropertyValue="&lt;div&gt;&lt;/div&gt;" />
      <PROPERTY PropertyName="xx-Person" PropertyValue="" />
      <PROPERTY PropertyName="xx-person2" PropertyValue="" />
      <PROPERTY PropertyName="xx-date" PropertyValue="" />
      <PROPERTY PropertyName="xx-datetime" PropertyValue="" />
    </USER>
  </PROFILE>
</MSPROFILE>
```

4.7 Managing Commonalities

The following examples assume the following hierarchical management structure of users and their respective identifiers:

- Syed Abbas (B8C750FC-E3E3-11DC-AFA1-EFA756D89593)
  - Brenda Diaz (B8C750FC-E3E3-11DC-AFA1-EFA756D89594)
    - Lori Kane(B8C750FC-E3E3-11DC-AFA1-EFA756D89595)
  - Steve Masters(B8C750FC-E3E3-11DC-AFA1-EFA756D89599)
    - Tai Yee(B8C750FC-E3E3-11DC-AFA1-EFA756D89597)
    - Roy Antebi (B8C750FC-E3E3-11DC-AFA1-EFA756D89598)

In this example, Brenda and Steve report to Syed, Tai and Roy report to Steve, and Lori reports to Brenda.

4.7.1 Retrieving a Common Manager

In this example, the protocol client retrieves the common managers of Tai Yee and Roy Antebi. The protocol client would make the following call:

exec profile_GetCommonManager 'B8C750FC-E3E3-11DC-AFA1-EFA756D89597', 'B8C750FC-E3E3-11DC-AFA1-EFA756D89598'

The following result set would be returned:

RecordId	UserID	NTName	Email	SipAddress	PreferredName	ProfileSubTypeId	PictureURL	Title	FirstCommon
1	B8C750FC-E3E3-11DC-AFA1-EFA756D89599	domain\steve.masters	Steve.masters@domain.com	NULL	Steve Masters		http://my/sites/stevemasters/picture.jpg	NULL	True
2	B8C750FC-E3E3-11DC-AFA1-EFA756D89593	domain\syed.abbas	Syed.abbas@domain.com	NULL	Syed Abbas		http://my/sites/syedabbas/picture.jpg	NULL	False

4.7.2 Retrieving Reporting Data

A protocol client can retrieve information about employees from a user, and the colleagues and manager of the user. To find employees from Steve Masters, the protocol client would make the following query.

```
exec profile_GetUserReportToData NULL,
    'B8C750FC-E3E3-11DC-AFA1-EFA756D89599', NULL, NULL, False
```

The following three result sets would be returned:

Direct Employees

RecordId	UserID	NTName	PreferredName	Email	SipAddress	PersonTitle
1	B8C750FC-E3E3-11DC-AFA1-EFA756D89597	domain\tai.yee	Tai Yee	Tai.yee@domain.com	NULL	NULL

RecordId	UserID	NTName	PreferredName	Email	SipAddress	PersonTitle
2	B8C750FC-E3E3-11DC-AFA1-EFA756D89598	domain\roy.antebi	Roy Antebi	Roy.antebi@domain.com	NULL	NULL

Manager (only applies when the user has a manager)

RecordId	UserID	NTName	PreferredName	Email	SipAddress	PersonTitle
1	B8C750FC-E3E3-11DC-AFA1-EFA756D89593	domain\Syed.abbas	Syed Abbas	Syed.abbas@domain.com	NULL	NULL

Peers

RecordId	UserID	NTName	PreferredName	Email	SipAddress	PersonTitle
1	B8C750FC-E3E3-11DC-AFA1-EFA756D89594	domain\Brenda.diaz	Brenda Diaz	Brenda.diaz@domain.com	NULL	NULL

4.8 Controlling Policy

The protocol client uses can expose policy to an administrator that defines the policy for users of the user profile service. The client would first call **privacy_getAllPolicy** and use the result set to display each policy shown inside its *@GroupName* labeled by *@DisplayName*. The client makes calls to **privacy_deletePolicy** or **privacy_UpdatePolicy** as changes are made. As calls to **privacy_deletePolicy** are made, the server is responsible for no longer returning this data in calls to **privacy_getAllPolicy** or **privacy_getFeaturePolicy**. As calls to privacy_**UpdatePolicy** are made, the server is responsible for returning this same data in subsequent calls to **privacy_getAllPolicy** or **privacy_getFeaturePolicy**.

4.9 Enforcing Policy

The protocol client is responsible for enforcing the privacy and policy stored by the server. As the protocol client services requests, it checks whether to enable or disable features by calling **privacy_GetFeaturePolicy**. As the client services requests that involve editing of Profile properties, it calls **privacy_getAllPolicy**. If the policy type is set to disabled, it does not show the property. If the policy type is set to optional, it shows the property and allows it to be left empty. If the policy type is set to mandatory, it shows the property and requires that it be set. The client uses the result of **IsItemSecurityOverridable** to determine whether to let users override the privacy of their property. Lastly, as the client displays these features and properties to other users they use the **DefaultItemSecurity** result to determine whether another user can see the property.

4.10 Managing Organizations

The protocol client is able to use this protocol to create and manage organizations, a tool to organize users into logical groups. The following examples will create and manage an organization whose parent is the root organization.

4.10.1 Creating an Organization

First, the protocol client gets information about the root organization. Consider the following T-SQL call to **profile_GetRootOrganization** that can be used to get this information.

```
exec dbo.profile_GetRootOrganization @partitionID='0C37852B-34D0-418E-91C6-
2AC25AF4BE5B',@correlationId='2796BE61-28F9-4BC0-8AA0-9C9582298FE7'
```

The protocol server will return the following result set:

Organization Id	Profile SubTypeID	Organization DisplayName	Organization Guid	Parent Type	Parent RecordID	Children Count
1	2	Root Organization	07193C68-A8FD-4C90-BDB8-550582A574FC	NULL	-1	2

This information can then be used by the protocol client to find more information about the Root Organization. Consider the following T-SQL call to **profile_GetOrganizationData**.

```
exec dbo.profile_GetOrganizationData @partitionID='0C37852B-34D0-418E-91C6-
2AC25AF4BE5B',@correlationId='2796BE61-28F9-4BC0-8AA0-9C9582298FE7',@OrgID=NULL,@RecordID=1
```

The protocol server returns the following two result sets.

RecordID	ProfileSubTypeID	ChildrenCount
1	2	2

Record ID	Property ID	Property Val	Secondary Val	Text	Order Rank	Privacy	PartitionId	Property Name
1	1	07193C68-A8FD-4C90-BDB8-550582A574FC	NULL	NULL	NULL	NULL	0C37852B-34D0-418E-91C6-2AC25AF4BE5B	UserProfile_GUID
1	7	Root Organization	NULL	NULL	NULL	NULL	0C37852B-34D0-418E-91C6-2AC25AF4BE5B	PreferredName

Now consider the following T-SQL call to **profile_OrganizationMembersCount**.

```
declare @p4 bigint
set @p4=0
exec dbo.profile_OrganizationMembersCount @partitionID='0C37852B-34D0-418E-91C6-
2AC25AF4BE5B',@correlationId='2796BE61-28F9-4BC0-8AA0-
9C9582298FE7',@OrganizationID=1,@MembersCount=@p4 output
select @p4
```

In the case that there are no members of the Root Organization, the value of @p4 is 0. Now consider the following T-SQl call to **profile_GetOrganizationMemberships**.

```
exec dbo.profile_GetOrganizationMemberships @partitionID='0C37852B-34D0-418E-91C6-
2AC25AF4BE5B',@correlationId='2796BE61-28F9-4BC0-8AA0-9C9582298FE7',@RecordID=1
```

Given that there were no members, this call returns an empty result set. Otherwise it could be used to add members of the parent organization, which in this case is also the root organization, to the organization being created. Now consider the following call to **profile_CreateOrganization**.

```
declare @p4 bigint
set @p4=6
exec dbo.profile_CreateOrganization @partitionID='0C37852B-34D0-418E-91C6-
2AC25AF4BE5B',@correlationId='2796BE61-28F9-4BC0-8AA0-
9C9582298FE7',@ProfileName=N'OrganizationProfile',@RecordID=@p4 output
select @p4
```

The value of @p4 in this case will depend on how many other organizations there are already. For the purpose of this example, assume it to be 4. Now consider the following T-SQL call to **profile_GetOrganizationData** to see the data of the newly created org.

```
exec dbo.profile_GetOrganizationData @partitionID='0C37852B-34D0-418E-91C6-
2AC25AF4BE5B',@correlationId='2796BE61-28F9-4BC0-8AA0-9C9582298FE7',@OrgID=NULL,@RecordID=6
```

The protocol server would return the following two result sets.

RecordID	ProfileSubTypeID	ChildrenCount
1	2	0

Record ID	Property ID	PropertyVal	Secondary Val	Text	Order Rank	Privacy	PartitionId	Property Name
6	1	05A0A338-572C-4580-AE66-5C6632EC713B	NULL	NULL	NULL	NULL	0C37852B-34D0-418E-91C6-2AC25AF4BE5B	UserProfile_GUID
6	7	Organization	NULL	NUL	NULL	NULL	0C37852B-	PreferredNam

Record ID	Property ID	PropertyVal	Secondary Val	Text	Order Rank	Privacy	PartitionId	Property Name
		Name		L			34D0-418E-91C6-2AC25AF4BE5B	e

The protocol client would now make the following T-SQL call to **profile_GetOrganizationMemberships** to verify that there currently are no members of the organization.

```
exec dbo.profile_GetOrganizationMemberships @partitionID='0C37852B-34D0-418E-91C6-
2AC25AF4BE5B',@correlationId='2796BE61-28F9-4BC0-8AA0-9C9582298FE7',@RecordID=6
```

Given that there are no members yet, this call would return empty. Now the protocol client can begin to add users to the organization's memberships. For this example, assume two users are being added to the organization. Consider the following T-SQL call to **profile_AddUserToOrganization** to add a user who is only a member.

```
exec dbo.profile_AddUserToOrganization @partitionID='0C37852B-34D0-418E-91C6-
2AC25AF4BE5B',@correlationId='2796BE61-28F9-4BC0-8AA0-
9C9582298FE7',@RecordID=6,@OrganizationID='05A0A338-572C-4580-AE66-
5C6632EC713B',@UserRecordID=4,@MembershipType=1
```

The protocol server does not return any results for this. Now consider the following T-SQL call to **profile_AddUserToOrganization** to add a user who is a leader of the organization.

```
exec dbo.profile_AddUserToOrganization @partitionID='0C37852B-34D0-418E-91C6-
2AC25AF4BE5B',@correlationId='2796BE61-28F9-4BC0-8AA0-
9C9582298FE7',@RecordID=6,@OrganizationID='05A0A338-572C-4580-AE66-
5C6632EC713B',@UserRecordID=3,@MembershipType=2
```

The protocol server does not return any results for this call. Now the protocol client can update the rest of the properties of the organization that were desired upon creation. Consider the following T-SQL with example data for the new organization.

```
exec dbo.profile_UpdateOrganizationProfileData @partitionID='0C37852B-34D0-418E-91C6-
2AC25AF4BE5B',@correlationId='2796BE61-28F9-4BC0-8AA0-
9C9582298FE7',@UpdatePropertyList=N'<?xml version="1.0" encoding="utf-16"?>
<MSPROFILE>
  <PROFILE ProfileName="OrganizationProfile">
    <ORGANIZATION RecordID="6">
      <PROPERTY PropertyName="SPS-ParentType" Privacy="1" PropertyValue="2" />
      <PROPERTY PropertyName="SPS-Parent" Privacy="1" PropertyValue="1" />
      <PROPERTY PropertyName="PreferredName" Privacy="1" PropertyValue="Organization Name" />
      <PROPERTY PropertyName="SPS-AboutUs" Privacy="1" PropertyValue="About the
organization." />
      <PROPERTY PropertyName="SPS-FormerNames" Privacy="1" PropertyValue="c8e4a41b-94ba-496a-
adef-8e9a56b40c84" PropertySecondaryValue="Organization Former Name" />
      <PROPERTY PropertyName="SPS-Team-Site" Privacy="1"
PropertyValue="http://server/orgsite" />
    </ORGANIZATION>
```

```
  </PROFILE>
</MSPROFILE>'
```

The protocol server does not return any results for this call. Now, the new organization is ready.

4.10.2 Retrieving Organization Data

The previous example covered uses of the **profile_GetOrganizationData** and **profile_GetOrganizationMemberships** portions of the protocol, which can be used to view information about organizations.

4.10.3 Managing Organization Data

Example 4.9.1 covered uses of the **profile_AddUserToOrganization** and **profile_UpdateOrganizationProfileData** portions of the protocol, which can be used to update information about organizations.

4.10.4 Deleting Organizations

To delete the organization created in example 4.9.1, the protocol client would issue the following series of calls to the protocol server. First, the protocol client would find data related to the organization. Consider the following T-SQL call to **profile_GetOrganizationData**.

```
exec dbo.profile_GetOrganizationData @partitionID='0C37852B-34D0-418E-91C6-
2AC25AF4BE5B',@correlationId='8D91C316-5734-416E-8ACC-B978D6EBF758',@OrgID='05A0A338-572C-
4580-AE66-5C6632EC713B'
```

The protocol server returns the following two result sets.

RecordID	ProfileSubTypeID	ChildrenCount
6	2	0

Record ID	Property ID	PropertyVal	Secondary Val	Text	Order Rank	Privacy	PartitionId	Property Name
6	1	05A0A338-572C-4580-AE66-5C6632EC713B	NULL	NULL	NULL	NULL	0C37852B-34D0-418E-91C6-2AC25AF4BE5B	UserProfile_GUID
6	7	Organization Name	NULL	NULL	NULL	NULL	0C37852B-34D0-418E-91C6-2AC25AF4BE5B	PreferredName
6	5025	1	NULL	NULL	NULL	NULL	0C37852B-34D0-418E-91C6-	SPS-Parent

Record ID	Property ID	PropertyVal	Secondary Val	Text	Order Rank	Privacy	PartitionId	Property Name
							2AC25AF4BE5B	
6	5030	http://server/orgsite	NULL	NULL	NULL	NULL	0C37852B-34D0-418E-91C6-2AC25AF4BE5B	SPS-Team-Site
6	5031	Organization About us.	NULL	NULL	NULL	NULL	0C37852B-34D0-418E-91C6-2AC25AF4BE5B	SPS-AboutUs
6	5033	2	NULL	NULL	NULL	NULL	0C37852B-34D0-418E-91C6-2AC25AF4BE5B	SPS-ParentType
6	5037	C8E4A41B-94BA-496A-ADEF-8E9A56B40C84	Organization Former name	NULL	1	NULL	0C37852B-34D0-418E-91C6-2AC25AF4BE5B	SPS-FormerNames

With the information about the organization, the protocol client can verify that the correct organization is about to be deleted, and then delete it. Consider the following T-SQL request to delete the organization.

```
exec dbo.profile_RemoveOrganization @partitionID='0C37852B-34D0-418E-91C6-
2AC25AF4BE5B',@correlationId='8D91C316-5734-416E-8ACC-B978D6EBF758',@RecordID=6
```

Now the protocol server deletes the organization, and returns a value of 0 to indicate success.

5 Security

5.1 Security Considerations for Implementers

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

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

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

- Microsoft® SharePoint® Server 2010
- 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.

<1> Section 4.5.1: The protocol client can also refresh its bounds by invoking

```
membership_enumerateGroups(21, 25, @LowerBound, @UpperBound).
```

<2> Section 4.5.2: The following columns have been omitted for simplicity of presentation: Id, Description, cs_SourceReference, MemberCount, LastUpdate, WebID, and UserCreated.

<3> Section 4.5.3: The following columns have been omitted for simplicity of presentation: Description, cs_SourceReference, MemberCount, LastUpdate, WebID, UserCreated.

<4> Section 4.5.4: The following columns have been omitted for simplicity of presentation: Title, Department, NTName, Email, SipAddress, UserId, AboutMe, PictureURL, IsAboutMeVisible, IsPictureUrlVisible, Id, RecordId, MemberGroupId, GroupType, GroupTitle, SID, PolicyId, ItemSecurity, Id, MailNickName, Description, Source, SourceReference, cs_SourceReference, Url, LastUpdate, WebID, Type, and UserCreated.

<5> Section 4.5.4: The following columns have been omitted for simplicity of presentation: Title, Department, NTName, Email, SipAddress, UserId, AboutMe, PictureURL, IsAboutMeVisible, IsPictureUrlVisible, Id, RecordId, MemberGroupId, GroupType, GroupTitle, SID, PolicyId, ItemSecurity, Id, MailNickName, Description, Source, SourceReference, cs_SourceReference, Url, LastUpdate, WebID, Type, UserCreated.

<6> Section 4.5.4: The following columns have been omitted for simplicity of presentation:. Id, MemberGroupId, GroupType, GroupTitle, SID, PolicyId, ItemSecurity, Id, MailNickName, Description, Source, SourceReference, cs_SourceReference, Url, LastUpdate, WebID, Type, UserCreated, RecordId, NTName, Email, SipAddress.

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

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

X