[MS-UPSPROF]:
User Profile Stored Procedures Protocol Specification

Revision Summary

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

Table of Contents

1 Introduction

This document provides specific details of the User Profile Service Stored Procedure 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.

1.1 Glossary

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

Active Directory
GUID
language code identifier (LCID)
security identifier (SID)
Unicode
user principal name (UPN)

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

alternate account
audience
back-end database server
collation sequence
colleague
Colleague Tracker Web Part
content type
distribution list
e-mail address
language pack
login name
mailto URI
master account
membership
membership group
multivalue property
personal site
quick link
Replicable
result set
return code
section
Security Account Manager (SAM)
server-relative URL
Session Initiation Protocol (SIP) address
Shared Services Provider (SSP)
stored procedure
Uniform Resource Identifier (URI)
user display name
user name
user profile privacy policy
user profile record identifier
user profile store
XML schema

The following terms are specific to this document:

vocabulary identifier: A string that specifies a database operation for a property in a user profile record. Valid vocabulary identifiers are Add, Rename, and Delete.

vocabulary value: The value portion of a name/value pair. The value can be associated with one or more documents in a workspace, and can include property values, document profile terms, categories, and other keywords that are used to apply categorized tags to documents.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

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

[MS-SQL] Microsoft Corporation, "SQL Server 2000 Architecture and XML/Internet Support", Volume 1 of Microsoft SQL Server 2000 Reference Library, Microsoft Press, 2001, ISBN 0-7356-1280-3, http://msdn.microsoft.com/en-us/library/dd631854(v=SQL.10).aspx

[MS-TDS] Microsoft Corporation, "Tabular Data Stream Protocol Specification".

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

[MS-UPSSYNC] Microsoft Corporation, "User Profile Synchronization Stored Procedures Protocol Specification"

[MS-WSSFO] Microsoft Corporation, "Windows SharePoint Services (WSS): File Operations Database Communications 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

[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".

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 various attribute objects which represent information about users. This protocol provides way for the client to retrieve this information, write new information or update existing information for each user.

All the information about a particular user is referred to as user profile. Each user profile contains a set of user profile properties. The protocol allows the protocol client to create a common set of user profile properties across all user profiles, and set metadata on each user profile property. 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.

This protocol also provides a way for protocol clients to control the policy and privacy issues associated with a user profile service. 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, policy also describes whether user profile properties are disabled, enabled and allowed to be null or enabled or mandatory. Privacy describes the privacy 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 both privacy and 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

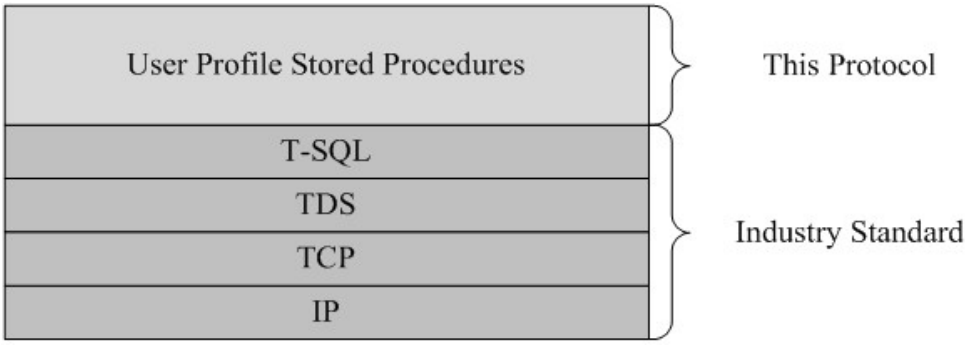

Figure 1: This protocol in relation to other protocols

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

The user profile service 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 defined in [MS-WSSFO] 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

None.

2.2.2 Simple Data Types

2.2.2.1 Group Type

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
2	General
5	Users who share the same **Manager** property
7	Distribution list default grouping.
8	Site default grouping.

2.2.2.2 Is Item Security Overridable Type

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.

2.2.2.3 Is MLS Enabled Type

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.

2.2.2.4 Is User Created Type

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

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

2.2.2.5 Name Format Type

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.
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 cross-domain linked with 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.

2.2.2.6 Policy Link Type

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 distribution list sourced membership group.
8BB1220F-DE8B-4771-AC3A-0551242CF2BD	Specifies a site sourced membership group.
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.

2.2.2.7 Privacy Type

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 workgroup colleagues.
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.

2.2.2.8 Privacy Policy Type

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 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.

2.2.2.9 Property Choice Type

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.

2.2.2.10 Property Data Type

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.
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.

2.2.2.11 Separator Type

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

2.2.2.12 Short Group Type

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 default grouping.

2.2.2.13 Short Link Type

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 distribution list sourced membership group.
8BB1220F-DE8B-4771-AC3A-0551242CF2BD	Specifies a site sourced membership group.

2.2.2.14 Visibility Type

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.

2.2.3 Bit Fields and Flag Structures

None.

2.2.4 Binary Structures

None.

2.2.5 Result Sets

None.

2.2.6 Tables and Views

None.

2.2.7 XML Structures

None.

3 Protocol Details

3.1 User Profile Service Protocol Specification 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.

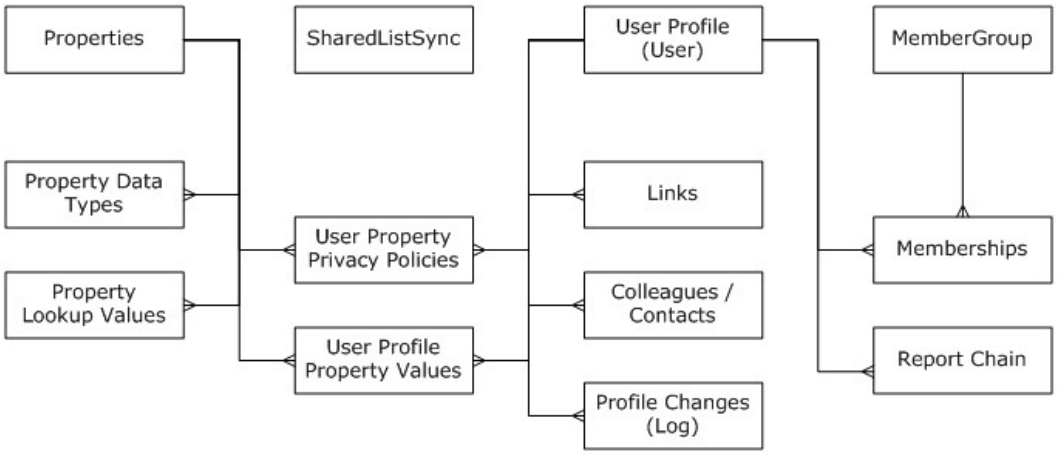

Figure 2: Types in the data model of the User Profile Stores Procedures Protocol

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.
- **User Profile**: a collection of entries containing information about a user profile.
- **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.
- **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 Message Processing Events and Sequencing Rules

The following table summarizes the types that are defined in this specification.

Name	Description
membership_deleteGroup	The **membership_deleteGroup** stored procedure is called to remove a member group and its members from the database.
membership_enumerateGroups	The **membership_enumerateGroups** stored procedure is called to retrieve a subset of the set of distribution list sourced membership groups that have e-mail address and site sourced membership groups.
membership_getColleagueSuggestions	The **membership_getColleagueSuggestions** stored procedure is called to retrieve a set of probable Colleague property for an entity specified by a user profile .
membership_getGroup	The **membership_getGroup** stored procedure is called to retrieve information about a particular membership group.
membership_getGroupCount	The **membership_getGroupCount** stored procedure is called to retrieve the count of the distribution list sourced membership groups that have an e-mail address and site sourced membership groups
membership_getGroupMemberships	The **membership_getGroupMemberships** stored procedure is called to retrieve the all membership information about a particular membership group.
membership_getGroupMembershipsPaged	The **membership_getGroupMembershipsPaged** stored procedure is called to retrieve a subset of the membership information about a particular membership group.

Name	Description
membership_getRelatedGroups	The **membership_getRelatedGroups** stored procedure is called to retrieve information about membership groups that are related to a particular membership group.
membership_updateGroup	The **membership_updateGroup** stored procedure is called to either create a new membership group or alter an existing membership group.
privacy_deletePolicy	The **privacy_deletePolicy** stored procedure is called to delete a privacy policy which is associated with a specific user profile property.
privacy_getAllPolicy	The **privacy_getAllPolicy** stored procedure is called to return the properties of all the privacy policies.
privacy_getFeaturePolicy	The **privacy_getFeaturePolicy** stored procedure is called to return the definition of all privacy policies which are not associated with a specific user profile property.
privacy_updatePolicy	The **privacy_updatePolicy** stored procedure is called to update the properties of a privacy policy or to create a new privacy policy.
profile_EnumUsers	The **profile_EnumUsers** stored procedure is called to retrieve the user profiles from the user profile store whose IDs are within the specified interval.
profile_FindPropertyChoiceList	The **profile_FindPropertyChoiceList** stored procedure is called to retrieve all the choice list values that match the specified search pattern for a property.
profile_GetCommonManager	The **profile_GetCommonManager** stored procedure is called to retrieve the common **Manager** property between 2 users.
profile_GetDataTypeList	The **profile_GetDataTypeList** stored procedure is called to return the properties for profile data types
profile_GetMultiloginAccounts	The **profile_GetMultiloginAccounts** stored procedure is called to find each **login name** registered to the specified value of *@Recorded*.
profile_GetNextUserProfileData	The **profile_GetNextUserProfileData** stored procedure is called to retrieve the next user profile information: the record identifier and the GUID identifying the user.
profile_GetPersonalSiteInfo	The **profile_GetPersonalSiteInfo** stored procedure is called to retrieve the personal site configuration properties from the server.
profile_GetProfileCount	The **profile_GetProfileCount** stored procedure is called to retrieve the number of user profiles contained in the user profile store.
profile_GetProfileCountWithProperty	The **profile_GetProfileCountWithProperty** stored procedure is called to retrieve the number of user

Name	Description
	profiles contained in the user profile store that have a value defined for a specified property.
profile_GetProfilePropertyInfo	The **profile_GetProfilePropertyInfo** stored procedure is called to retrieve property information.
profile_GetProfilePropertyLoc	The **profile_GetProfilePropertyLoc** stored procedure is called to retrieve the localized **user display name** and description for each property.
profile_GetPropertyChoiceList	The **profile_GetPropertyChoiceList** stored procedure is called to retrieve all the choice list values for the specified property.
profile_GetSharedListSync	The **profile_GetSharedListSync** stored procedure is called to retrieve a shared list object based on the specified *@ListId*.
profile_GetUserFormat	The **profile_GetUserFormat** stored procedure is called to retrieve the user display name format that is used by the Protocol server.
profile_GetUserGUID	The **profile_GetUserGUID** stored procedure is called to retrieve the GUID of a user's primary account.
profile_GetUserProfileData	The **profile_GetUserProfileData** stored procedure is called by a user who request the user profile for the user specified by any of *@UserID*, *@NTName*, *@SID* or *@RecordId*.
profile_GetUserProfilesByEmail	The **profile_GetUserProfilesByEmail** stored procedure is called to get user profile information based on the e-mail addresses specified.
profile_GetUserReportToData	The **profile_GetUserReportToData** stored procedure is called to retrieve a user's **Manager** property, the **Colleague** property of a specified user, and the reports associated with a specific user.
profile_GetViewerRights	The **profile_GetViewerRights** stored procedure is called to determine what rights the user specified by *@ViewerNTName* has to see items owned by the user specified by *@RecordId*.
profile_MigrateUserProfile	The **profile_MigrateUserProfile** stored procedure is called to update the **security identifier (SID)** and user display name of an existing user profile.
profile_OnSqlRestore	The **profile_OnSqlRestore** stored procedure is called to apply database changes after a restore operation
profile_RemoveUser	The **profile_RemoveUser** stored procedure is called to delete a user profile from the Protocol server.
profile_ResetDeletedUser	The **profile_ResetDeletedUser** stored procedure is called to unmark the Profile Pending Deletion Flag on a user profile and set the account to active.
profile_SearchUser	The **profile_SearchUser** stored procedure is called to

Name	Description
	return the users that match the search criteria.
profile_UpdateOrgColleagues	The **profile_UpdateOrgColleagues** stored procedure is called to update colleagues organizational structure links for newly added users or users with updated **Manager** properties in the user profile store.
profile_UpdatePersonalSiteInfo	The **profile_UpdatePersonalSiteInfo** stored procedure is called to update the personal site configuration properties.
profile_UpdatePersonalSpace	The **profile_UpdatePersonalSpace** stored procedure is called to update a user's personal site URL.
profile_UpdateProfileDisplay	The **profile_UpdateProfileDisplay** stored procedure is called to update the display order of any property and **section** associated with the specified user profile.
profile_UpdateProperty	The **profile_UpdateProfileProperty** stored procedure is called to add, update and remove properties.
profile_UpdatePropertyLoc	The **profile_UpdatePropertyLoc** stored procedure is called to update the user display names and descriptions of a property in multiple languages.
profile_updateSharedListSync	The **profile_updateSharedListSync** stored procedure is called to update a shared list object.
profile_UpdateUserProfileBlobData	The **profile_UpdateUserProfileBlobData** stored procedure is called to update the user profile properties which cannot be updated through **profile_UpdateUserProfileData** (Section 3.1.4.46).
profile_UpdateUserProfileData	The **profile_UpdateUserProfileData** stored procedure is called to update the user profile data.
QuickLinksAdd	The **QuickLinksAdd** stored procedure is called to add a new link for a user in the user profile store.
QuickLinksDelete	The **QuickLinksDelete** stored procedure is called to delete link for a user in the user profile store with a specified link type (*@PolicyId*) and identifier (*@LinkID*).
QuickLinksDeleteUser	The **QuickLinksDeleteUser** stored procedure is called to delete all link data for the specified user in the user profile store with the specified link type (specified by *@PolicyId*).
QuickLinksEdit	The **QuickLinksEdit** stored procedure is called to modify or add 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.
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.
QuickLinksRetrieveColleaguesOfColleagues	The **QuickLinksRetrieveColleaguesOfColleagues** stored procedure is called to retrieve the colleagues of

Name	Description
	the specified user's colleagues.
QuickLinksRetrieveGroupList	The **QuickLinksRetrieveGroupList** stored procedure is called to retrieve the groups of a specified type for a user profile.

3.1.4.1 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 transaction to restore data to the original state.

membership_deleteGroup is defined using T-SQL syntax as follows:

```
PROCEDURE membership_deleteGroup(
    @Id                     bigint,
    @SourceId               uniqueidentifier
);
```

@Id: The **identifier of** the Member **Group** to delete. The protocol client MUST specify a valid ID obtained by procedure **membership_getGroup** (Section 3.1.4.4), specified in [MS-UPSIMP].

@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 constant GUID with the value of 'A88B9DCB-5B82-41E4-8A19-17672F307B95'.

Return Code Values: membership_deleteGroup MUST return 0.

Result Sets: membership_deleteGroup MUST NOT return any result set.

3.1.4.2 membership_enumerateGroups

The **membership_enumerateGroups** stored procedure is called to retrieve 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,
    @EndId                  bigint,
    @MINID                  bigint OUTPUT,
    @MAXID                  bigint OUTPUT
);
```

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

@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. 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. The output value MUST be set to a number that is greater than or equal to the maximum membership group record identifier.

Return Code Values: membership_enumerateGroups MUST return 0.

Result Sets: membership_enumerateGroups MUST always return the Valid Identifier result set.

3.1.4.2.1 Valid Identifier Result Set

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. Should the value of the preceding calculation be less than or equal to 0 then 0 rows MUST be returned. Should 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.

3.1.4.3 membership_getColleagueSuggestions

The **membership_getColleagueSuggestions** stored procedure is called to retrieve 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
);
```

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

Return Code Values: membership_getColleagueSuggestions MUST return 0.

Result Sets: membership_getColleagueSuggestions MUST return the Suggested Colleagues result set.

3.1.4.3.1 Suggested Colleagues Result Set

Suggested Colleagues result set returns user profile data for Colleague property. The Suggested Colleagues result set MUST NOT contain more than 75 rows. The Suggested Colleagues 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 Suggested Colleagues 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),
    PersonTitle                 nvarchar(150),
    Weight                      float;
```

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.

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.

3.1.4.4 membership_getGroup

The **membership_getGroup** stored procedure is called to retrieve information about a particular membership group.

membership_getGroup is defined using T-SQL syntax as follows:

```
PROCEDURE membership_getGroup(
    @Source             uniqueidentifier = NULL,
    @DisplayName        nvarchar(250) = NULL,
    @SourceReference    nvarchar(2048) = NULL,
    @Id                 bigint = NULL
);
```

@Source: MUST be a Short Link (Section 2.2.2.13) Type identifier specifying the source of the membership group. Value MUST be ignored when parameter *@Id* is not NULL.

@DisplayName: A descriptive name for the membership group. The **Unicode** String Trim operation MUST have been performed on the value. Value MUST be ignored when parameter *@Id* is not NULL. Value MUST be ignored when parameter *@Source* is NULL.

@SourceReference: Text used to distinguish the membership group within a particular *@Source*. The Unicode String Trim operation MUST have been performed on the value. The value MUST be ignored when parameter *@Id* is not NULL. Value MUST be ignored when both parameters *@Source* and *@DisplayName* are not NULL. Value MUST be ignored when parameter *@Source* is NULL.

@Id: A membership group record identifier specifying the membership group for which to retrieve information.

Return Code Values: membership_getGroup MUST return 0.

Result Sets: membership_getGroup MUST NOT return any result set when all input parameters are NULL. **membership_getGroup** MUST NOT return any result set when the *@Source* input parameter is NOT NULL but all other input parameters are NULL. In all other cases the Matching Member Group result set MUST be returned.

3.1.4.4.1 Matching Member Group Result Set

Matching Member Group result set returns the membership groups that match the input parameters.

The Matching Member Group result set MUST return 1 row if *@Id* corresponds to an existing membership group.

The Matching Member Group result set MUST return 1 row if *@Source* and *@SourceReference* corresponds to an existing membership group.

The Matching Member Group result set MUST return 1 or more rows if *@Source* and *@DisplayName* correspond to existing membership groups, but the second and subsequent rows MUST be ignored.

If the *@Id* parameter is not NULL and refers to a non existing membership group then the Matching Member Group result set MUST return 0 rows.

The Matching Member Group 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),
MemberCount             bigint,
LastUpdate              datetime,
WebID                   uniqueidentifier,
Type                    tinyint,
UserCreated             bit;
```

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 2.2.2.13) Type identifier specifying the source of the membership group.

SourceReference: A section of text used to distinguish the various membership groups within a particular Source.

cs_SourceReference: This field MUST be ignored.

Url: A **URI** for the membership group.

MemberCount: The count of members in this membership group.

LastUpdate: A UTC value specifying the last time **membership_updateGroup** (Section 3.1.4.9) was successfully called.

WebID: This field MUST be ignored.

Type: This field MUST be ignored.

UserCreated: This field MUST be ignored.

3.1.4.5 membership_getGroupCount

The **membership_getGroupCount** stored procedure is called to retrieve 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();
```

Return Code Values: membership_getGroupCount MUST return 0.

Result Sets: membership_getGroupCount MUST return the Count result set.

3.1.4.5.1 Count Result Set

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 a 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.

3.1.4.6 membership_getGroupMemberships

The **membership_getGroupMemberships** stored procedure is called to retrieve the all membership information about a particular membership group.

membership_getGroupMemberships is defined using T-SQL syntax as follows:

```
PROCEDURE membership_getGroupMemberships(
    @Id                 bigint
);
```

@Id: A membership group record identifier specifying the membership group to retrieve membership data for.

Return Code Values: membership_getGroupMemberships MUST return 0.

Result Sets: membership_getGroupMemberships MUST always return a Membership result set.

3.1.4.6.1 Membership Result Set

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,
RecordId                bigint,
MemberGroupId           bigint,
GroupType               tinyint,
GroupTitle              nvarchar(400),
SID                     varbinary(512),
PolicyId                uniqueidentifier,
ItemSecurity            int,
Id                      bigint,
DisplayName             nvarchar(250),
MailNickName            nvarchar(250),
Description             nvarchar(1500),
Source                  uniqueidentifier,
SourceReference         nvarchar(2048),
cs_SourceReference      int,
Url                     nvarchar(2048),
MemberCount             bigint,
LastUpdate              datetime,
WebID                   uniqueidentifier,
Type                    tinyint,
UserCreated             bit,
RecordId                nvarchar(400),
PreferredName           nvarchar(256),
Email                   nvarchar(256),
SipAddress              nvarchar(250);
```

These fields contain information about the membership relation:

Id: A GUID identifying 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 (Section 2.2.2.12) Type value specifying the membership relation grouping.

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

SID: This field MUST be ignored.

PolicyId: This field MUST be ignored.

ItemSecurity: MUST be a Privacy (Section 2.2.2.7) 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 2.2.2.13) Type identifier specifying the source of the membership group.

SourceReference: A section of text used to distinguish the various membership groups within a particular Source.

cs_SourceReference: This field MUST be ignored.

Url: A URI for the membership group.

MemberCount: The count of members in this membership group.

LastUpdate: A UTC value specifying the last time **membership_updateGroup** (Section 3.1.4.9) was successfully called.

WebID: This field MUST be ignored.

Type: This field MUST be ignored.

UserCreated: This field MUST be ignored.

These fields contain information about the member's user profile:

RecordId: A user profile record identifier.

NTName: A 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 specified by the user profile.

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

3.1.4.7 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(
```

```
    @Id                     bigint,
    @ViewerRecordId         bigint,
    @Count                  int,
    @SortPropertyId         bigint = 7,
    @SortDirection          bit = 0,
    @ItemBeforeFirst        nvarchar(1000),
    @RecordIdBeforeFirst    bigint,
    @Collation              nvarchar(60)
);
```

@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.

Return Code Values: membership_getGroupMembershipsPaged MUST return 0.

Result Sets: membership_getGroupMembershipsPaged MUST NOT return any result set when the *@Count* input parameter is NULL or when the *@SortPropertyId* input parameter is NULL. When the *@SortPropertyId* input parameter is set to 13 then the Title Paged Membership result set MUST be returned. When the *@SortPropertyId* input parameter is set to 14 then the Department Paged Membership result set MUST be returned. For all other values of the *@SortPropertyId* input parameter the Preferred Name Paged Membership result set MUST be returned.

3.1.4.7.1 Preferred Name Paged Membership Result Set

Preferred Name Paged Membership result set returns data about memberships in a particular membership group. The Preferred Name 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 Preferred Name Paged Membership result set will be ordered on the PreferredName field in the order specified by the *@SortDirection* input parameter and then in ascending order on the RecordId field.

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

```
PreferredName                   nvarchar(256),
Title                           sql_variant,
Department                      sql_variant,
RecordID                        bigint,
NTName                          nvarchar(400),
Email                           nvarchar(256),
SipAddress                      nvarchar(250),
UserId                          uniqueidentifier,
AboutMe                         sql_variant,
PictureURL                      sql_variant,
IsAboutMeVisible                int,
IsPictureUrlVisible             int,
Id                              bigint,
RecordId                        bigint,

MemberGroupId                   bigint,
GroupType                       tinyint,
GroupTitle                      nvarchar(400),
SID                             varbinary(512),
PolicyId                        uniqueidentifier,
ItemSecurity                    int,
Id                              bigint,
DisplayName                     nvarchar(250),
MailNickName                    nvarchar(250),
Description                     nvarchar(1500),
Source                          uniqueidentifier,
SourceReference                 nvarchar(2048),
cs_SourceReference              int,
Url                             nvarchar(2048),
MemberCount                     bigint,
LastUpdate                      datetime,
WebID                           uniqueidentifier,
Type                            tinyint,
UserCreated                     bit;
```

These fields contain information about the user profile of the specified member.

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

Title: The 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.

RecordID: A user profile record identifier.

NTName: A 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 2.2.2.14) 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 2.2.2.14) 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 (Section 2.2.2.12) Type value 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 2.2.2.7) 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: An 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 2.2.2.13) Type identifier specifying the source of the membership group.

SourceReference: An section of text used to distinguish the various membership groups within a particular Source.

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 3.1.4.9) was successfully called.

WebID: This field MUST be ignored.

Type: This field MUST be ignored.

UserCreated: This field MUST be ignored.

3.1.4.7.2 Title Paged Membership Result Set

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,

Department                      sql_variant,
PreferredName                   nvarchar(256),
RecordID                        bigint,
NTName                          nvarchar(400),
Email                           nvarchar(256),
SipAddress                      nvarchar(250),
UserId                          uniqueidentifier,
AboutMe                         sql_variant,
PictureURL                      sql_variant,
IsAboutMeVisible                int,
IsPictureUrlVisible             int,
Id                              bigint,
RecordId                        bigint,
MemberGroupId                   bigint,
GroupType                       tinyint,
GroupTitle                      nvarchar(400),
SID                             varbinary(512),
PolicyId                        uniqueidentifier,
```

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

These fields contain information about the member's user profile:

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 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 2.2.2.14) 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 2.2.2.14) 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 2.2.2.12) 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 2.2.2.7) 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 2.2.2.13) Type identifier specifying the source of the membership group.

SourceReference: An section of text used to distinguish the various membership groups within a particular Source.

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 3.1.4.9) was successfully called.

WebID: This field MUST be ignored.

Type: This field MUST be ignored.

UserCreated: This field MUST be ignored.

3.1.4.7.3 Department Paged Membership Result Set

Department Paged Membership result set returns data about memberships in a particular membership group. The Department 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 Department Paged Membership result set will be ordered on the Department field in the order specified by the *@SortDirection* input parameter and then in ascending order on the RecordId field.

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

```
Department                      sql_variant,
Title                           sql_variant,
PreferredName                   nvarchar(256),
```

```
RecordID                        bigint,
NTName                          nvarchar(400),
Email                           nvarchar(256),
SipAddress                      nvarchar(250),
UserId                          uniqueidentifier,
AboutMe                         sql_variant,
PictureURL                      sql_variant,
IsAboutMeVisible                int,
IsPictureUrlVisible             int,
Id                              bigint,
RecordId                        bigint,
MemberGroupId                   bigint,
GroupType                       tinyint,
GroupTitle                      nvarchar(400),
SID                             varbinary(512),
PolicyId                        uniqueidentifier,
ItemSecurity                    int,
Id                              bigint,
DisplayName                     nvarchar(250),
MailNickName                    nvarchar(250),
Description                     nvarchar(1500),
Source                          uniqueidentifier,
SourceReference                 nvarchar(2048),
cs_SourceReference              int,
Url                             nvarchar(2048),
MemberCount                     bigint,
LastUpdate                      datetime,
WebID                           uniqueidentifier,
Type                            tinyint,
UserCreated                     bit;
```

These fields contain information about the member's user profile:

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

Title: The title 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 SAM user name for 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 2.2.2.14) 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 2.2.2.14) 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 (Section 2.2.2.12) Type value specifying the membership relation grouping.

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

SID: This field MUST be ignored.

PolicyId: This field MUST be ignored.

ItemSecurity: MUST be a Privacy (Section 2.2.2.7) 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 2.2.2.13) Type identifier specifying the source of the membership group.

SourceReference: A section of text used to distinguish the various membership groups within a particular source.

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 3.1.4.9) was successfully called.

WebID: This field MUST be ignored.

Type: This field MUST be ignored.

UserCreated: This field MUST be ignored.

3.1.4.8 membership_getRelatedGroups

The **membership_getRelatedGroups** stored procedure is called to retrieve 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
);
```

@Id: A membership group record identifier specifying the membership group to retrieve related membership groups for.

Return Code Values: membership_getRelatedGroups MUST return 0.

Result Sets: membership_getRelatedGroups MUST return the Related Member Group result set.

3.1.4.8.1 Related Member Group Result Set

Related Member Group 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 Related Member Group result set MUST contain 1 row per descendant of the input membership group if any descendants exist. The Related Member Group result set MUST contain 0 rows when the *@Id* input parameter is NULL or when *@Id* does not match a valid membership group.

The Related Member Group result set is defined using T-SQL syntax as follows:

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

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 2.2.2.13) Type identifier specifying the source of the membership group.

SourceReference: Text used to distinguish the various membership groups within the source identified by Source.

cs_SourceReference: This field MUST be ignored.

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 3.1.4.9) was successfully called.

WebID: This field MUST be ignored.

Type: This field MUST be ignored.

UserCreated: This field MUST be ignored.

3.1.4.9 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(
    @Id                bigint = NULL,
    @Source            uniqueidentifier,
    @DisplayName       nvarchar(250),
    @MailNickName      nvarchar(250),
    @Description       nvarchar(1500),
    @Url               nvarchar(2048),
    @SourceReference   nvarchar(2048),
    @WebID             uniqueidentifier = NULL,
    @UserCreated       bit = 0,
    @Type              tinyint = 0,
    @LastUpdate        datetime OUTPUT,
    @NewId             bigint OUTPUT,
    @Error             int = NULL OUTPUT
);
```

@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 2.2.2.13) 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. The Unicode String Trim operation MUST have been performed on the input value.

@MailNickName: The membership group's ms-Exch-Mail-Nickname Active Directory attribute. This value MUST NOT be NULL. The Unicode String Trim operation MUST have been performed on the input value.

@Description: A description of the membership group. This value MUST NOT be NULL. The Unicode String Trim operation MUST have been performed on the input value.

@Url: A URI for the membership group. Value MUST NOT be NULL. 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 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 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. When the *@Source* input parameter specifies a site sourced membership group the value MUST be the site identifier.

@WebID: A site identifier associated with this membership group. When the *@Source* input parameter specifies a distribution list sourced membership group the value MUST be NULL. When the *@Source* input parameter specifies a site sourced membership group the value MUST be the site identifier of the specified site.

@UserCreated: MUST be an Is User Created (Section 2.2.2.4) 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)".

@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.

Return Code Values: membership_updateGroup returns an integer return code which MUST be listed in the following table:

Value	Description
0	Possibly successful execution. Check *@Error* output parameter for further data.
-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: membership_updateGroup MUST NOT return any result set.

3.1.4.10 privacy_deletePolicy

The **privacy_deletePolicy** stored procedure is called to delete a privacy policy which is associated with a specific user profile property.

privacy_delete policy is defined using T-SQL syntax as follows:

```
PROCEDURE privacy_deletePolicy(
    @Id               uniqueidentifier
);
```

@Id: The GUID identifying the privacy policy to be deleted.

Return Code Values: privacy_deletePolicy MUST return 0.

Result Sets: privacy_deletePolicy MUST NOT return any result set.

3.1.4.11 privacy_getAllPolicy

The **privacy_getAllPolicy** stored procedure is called to return the properties of all the privacy policies.

privacy_getAllPolicy is defined using T-SQL syntax as follows:

```
PROCEDURE privacy_getAllPolicy();
```

Return Code Values: privacy_getAllPolicy MUST return 0.

Result Sets: privacy_getAllPolicy MUST return exactly 1 AllPrivacyPolicy result set.

3.1.4.11.1 AllPrivacyPolicy Result Set

AllPrivacyPolicy result set returns the properties associated with each privacy policy in the system. The AllPrivacyPolicy result set MUST always be returned, and MUST contain 0 or more rows where each row defines the profile property of an existing privacy policy. The result set also includes 1 row for each privacy policy that does not have an associated profile property. The result set excludes 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,
IsPolicyOverridable         bit,
FilterPrivacyItems          bit,
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 2.2.2.8) Type.

DefaultItemSecurity: The default protection level assigned to the privacy policy. The value MUST be a Privacy (Section 2.2.2.7) type.

IsItemSecurityOverridable: MUST be an Is Item Security Overridable (Section 2.2.2.2) 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.

IsPolicyOverridable: This value MUST be ignored.

FilterPrivacyItems: This value MUST be ignored.

DisplayOrder: An ascending number by which the rows in the result set MUST be ordered. 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.

3.1.4.12 privacy_getFeaturePolicy

The **privacy_getFeaturePolicy** stored procedure is called to return 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();
```

Return Code Values: privacy_getFeaturePolicy MUST return 0.

Result Sets: privacy_getFeaturePolicy MUST return exactly 1 FeaturePrivacyPolicy result set.

3.1.4.12.1 FeaturePrivacyPolicy Result Set

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 always be returned, and 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,
```

```
IsPolicyOverridable          bit,
FilterPrivacyItems           bit;
```

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 2.2.2.8) Type.

DefaultItemSecurity: The default protection level assigned to the privacy policy. The value MUST be a Privacy (Section 2.2.2.7) Type.

IsItemSecurityOverridable: MUST be an "Is Item Security Overridable" (Section 2.2.2.2) Type value.

PropertyId: MUST be NULL.

IsPolicyOverridable: This value MUST be ignored.

FilterPrivacyItems: This value MUST be set to 1.

3.1.4.13 privacy_updatePolicy

The **privacy_updatePolicy** stored procedure is called to update the properties of a privacy policy or to create a new privacy policy.

privacy_updatePolicy is defined using T-SQL syntax as follows:

```
PROCEDURE privacy_updatePolicy(
    @Id                          uniqueidentifier = NULL,
    @PropertyId                  bigint = NULL,
    @DisplayName                 nvarchar(256),
    @GroupName                   nvarchar(256),
    @Policy                      int,
    @DefaultItemSecurity         int,
    @IsItemSecurityOverridable   bit,
    @FilterPrivacyItems          bit = 1,
    @NewId                       uniqueidentifier OUTPUT
);
```

@Id: The identifier assigned to the privacy policy to be updated. If this value is NOT NULL, then parameter *@PropertyId* 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 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 2.2.2.8) type.

@DefaultItemSecurity: The default protection level assigned to the privacy policy. The value MUST be a Privacy (Section 2.2.2.7) Type.

@IsItemSecurityOverridable: MUST be an Is Item Security Overridable (Section 2.2.2.2) Type value.

@FilterPrivacyItems: This value MUST be set to 1.

@NewId: Any input value MUST be ignored. When a privacy policy is created, this output parameter MUST be set to the assigned identifier of the added privacy policy. When a privacy policy is updated *@NewId* MUST be NULL.

Return Code Values: privacy_updatePolicy MUST return 0.

The privacy policy to be updated MUST be specified either through its assigned identifier or through the assigned identifier of the user profile property with which it is associated. If no such privacy policy exists, a new privacy policy will be added.

Result Sets: privacy_updatePolicy MUST NOT return any result set.

3.1.4.14 profile_EnumUsers

The **profile_EnumUsers** stored procedure is called to retrieve the user profiles from the user profile store whose IDs are within the specified interval.

The **profile_EnumUsers** stored procedure is defined using T-SQL syntax as follows:

```
PROCEDURE profile_EnumUsers (
    @BeginID              bigint,
    @EndID                bigint,
    @MINID                bigint OUTPUT,
    @MAXID                bigint OUTPUT
);
```

@BeginID: The 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 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.

Return Code Values: profile_EnumUsers MUST return 0.

Result Sets: profile_EnumUsers MUST return one result set.

3.1.4.14.1 profile_EnumUsers Result Set

profile_EnumUsers MUST return a result set of 0 or more rows. For a record to be included in the result set, it MUST NOT have a NULL login name or user display 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.

3.1.4.15 profile_FindPropertyChoiceList

The **profile_FindPropertyChoiceList** stored procedure is called to retrieve all the choice list values that match the specified search pattern for a property.

profile_FindPropertyChoiceList is defined using T-SQL syntax as follows:

```
PROCEDURE profile_FindPropertyChoiceList (
    @PropertyName        nvarchar(50),
    @search              nvarchar(250)
);
```

@PropertyName: The name of a property. This parameter MUST be specified and MUST NOT be NULL.

@search: A search pattern. This parameter MUST be specified and MUST NOT be NULL. The server MUST search the choice list values for any values that contain the search pattern.

Return Code Values: profile_FindPropertyChoiceList MUST return 0.

Result Sets: profile_FindPropertyChoiceList MUST return 1 result set.

3.1.4.15.1 Choice List Values Result Set

The choice list values result set returns the existing choice list values that match the search pattern for the specified *@PropertyName* sorted in ascending order. The result set MUST contain 0 or more rows.

The choice list values result set is defined using T-SQL syntax as follows:

```
VocValValue             sql_variant;
```

VocValValue: A choice list **value** that matches the search pattern for the property.

3.1.4.16 profile_GetCommonManager

The **profile_GetCommonManager** stored procedure is called to retrieve the common **Manager** property between 2 users. The stored procedure is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetCommonManager (
    @MyUserId               uniqueidentifier,
    @YourUserId             uniqueidentifier
);
```

@MyUserId: The GUID for the first user. This MUST NOT be NULL.

@YourUserId: The GUID for the second user. This MUST NOT be NULL.

Return Code Values: profile_GetCommonManager MUST return 1 of the following values:

Value	Description
0	Successful
1	Encountered **Manager** property loop
2	Exceeded the maximum **Manager** property chain length of 40.
3	Encountered **Manager** property loop and exceeded the maximum **Manager** property chain length of 40.

profile_GetCommonManager MUST return 1 result set.

3.1.4.16.1 profile_GetCommonManager Result Set

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),
Title                       nvarchar(150),
FirstCommon                 bit;
```

RecordID: Contains the record identifier of the **Manager** property. The value MUST be returned and MUST NOT be NULL.

UserID: Contains the GUID identifying the **Manager** property. The 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.

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.

3.1.4.17 profile_GetDataTypeList

The **profile_GetDataTypeList** stored procedure is called to return the properties for Profile Data Types.

profile_GetDataTypeList is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetDataTypeList(
    @Collation              nvarchar(60)
);
```

@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.

Return Code Values: profile_GetDataTypeList MUST NOT return a value.

Result Sets: profile_GetDataTypeList MUST return one result set.

3.1.4.17.1 DataTypeList Result Set

DataTypeList returns the list of profile data types defined in the system. The **DataTypeList** result set will return one result set.

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;
```

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

DataTypeName: Contains the underlying SQL data type for this Profile Data Type. Valid values are defined in the enumeration **TYPE:SQLServerDataType**.as specified in [MS-UPSIMP], section 3.1.4.4.1 and section 3.1.4.5.1.

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

FriendlyTypeName: Contains the user-facing description 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 that the data type is eligible for Full-Text Indexing. The server MUST ignore this value.

AllowMultiValue: Contains a bit that if 1 indicates that a profile property associated with this Profile Data Type supports multiple values.

BlobType: Contains a **TYPE:ProfilePropertyBlobType** (as specified in [MS-UPSIMP] section 2.2.5) enumeration value indicating how the Blob value 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.

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.

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 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.

3.1.4.18 profile_GetMultiLoginAccounts

The **profile_GetMultiLoginAccounts** stored procedure is called to find 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(
    @RecordId               BigInt
);
```

@RecordId: record of the user to return the list of login names for. This MUST NOT be NULL.

Return Code Values: profile_GetMultiLoginAccounts MUST return 0.

profile_GetMultiLoginAcccounts MUST return a result set.

3.1.4.18.1 MultiLoginAccounts Result Set

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 SAM user name for the entity specified by the user profile.

3.1.4.19 profile_GetNextUserProfileData

The **profile_GetNextUserProfileData** stored procedure is called to retrieve the next user profile information: the record identifier and the GUID identifying the user.

The profile_**GetNetUserProfileData** stored procedure is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetNextUserProfileData (
    @CurrentRecordID          bigint,
    @NextCurrentRecordID      bigint OUTPUT,
    @UserID                   uniqueidentifier OUTPUT
);
```

@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

@UserID: The Protocol server MUST set this parameter to the GUID of the user whose profile is identified by the *@NextCurrentRecordID*. If the *@NextCurrentRecordID* is -1, the parameter MUST be NULL.

Return Code Values: profile_GetNextUser ProfileData MUST return 0.

profile_GetNextUserProfileData MUST NOT return any result set.

3.1.4.20 profile_GetPersonalSiteInfo

The **profile_GetPersonalSiteInfo** stored procedure is called to retrieve the protocol server's personal site configuration properties.

profile_GetPersonalSiteInfo is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetPersonalSiteInfo();
```

Return Code Values: profile_GetPersonalSiteInfo MUST return 0.

profile_GetPersonalSiteInfo MUST return 1 result set.

3.1.4.20.1 ProfilePersonalSite Result Set

profile_GetPersonalSiteInfo 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) NOT NULL,
NameFormat              smallint NOT NULL
SiteReader              ntext NULL,
EnableMLS               bit NOT NULL;
```

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

NameFormat: MUST be a Name Format (Section 2.2.2.5) 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.

EnableMLS: MUST be an Is MLS Enabled (Section 2.2.2.3) Type value.

3.1.4.21 profile_GetProfileCount

The **profile_GetProfileCount** stored procedure is called to retrieve the number of user profiles contained in the user profile store.

profile_GetProfileCount is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetProfileCount();
```

Return Code Values: profile_GetProfileCount MUST return 0.

Result Sets: profile_GetProfileCount MUST return one result set.

3.1.4.21.1 profile_GetProfileCount Result Set

The **profile_GetProfileCount result set** MUST contain 1 row. The result set is defined using T-SQL syntax as follows:

```
CountTrack              int;
```

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

3.1.4.22 profile_GetProfileCountWithProperty

The **profile_GetProfileCountWithProperty** stored procedure is called to retrieve 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),
    @NoOfProfiles                   int OUTPUT,
    @Error                          int OUTPUT
);
```

@PropertyName: The name of a property . This parameter MUST be specified and MUST NOT be NULL.

@NoOfProfiles: The number of user profiles that contain the specified *@PropertyName*.

@Error: If *@PropertyName* is a valid property name the Protocol server MUST set this parameter to 0. If *@PropertyName* is invalid, the Protocol server MUST set this parameter to -1.

Return Code Values: profile_GetProfileCountWithProperty MUST return 0.

profile_GetProfileCountWithProperty MUST NOT return any result set.

3.1.4.23 profile_GetProfilePropertyInfo

The **profile_GetProfilePropertyInfo** stored procedure is called to retrieve property information. It can also be used to retrieve information about sections.

profile_GetProfilePropertyInfo is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetProfilePropertyInfo (
    @PropertyURI                nvarchar(250) NULL,
    @PropertyName               nvarchar(50) NULL,
    @ProfileName                nvarchar(250) N'UserProfile',
    @Collation                  nvarchar(60) NULL,
    @Lcid                       int NULL,
    @ReplicableSchemaVersion    int NULL OUTPUT,
    @bDebug                     bit 0
);
```

@PropertyURI: The URI of the property.

@PropertyName: The name of the property.

@ProfileName: This parameter MUST NOT be specified.

@Collation: The **collation sequence** used to determine the sort order for the **result set**. The Protocol server MUST use this value to sort the output by the DisplayName column in the **result set**.

@Lcid: The **language code identifier (LCID)** that indicates the language of the property associated with the user display name.

@ReplicableSchemaVersion: An monotonically-increasing value that indicates if changes are made to the Replicable Schema as defined in [MS-UPSSYNC], Section 3.1.1.

@bDebug: This parameter MUST be ignored.

Return Code Values: profile_GetProfilePropertyInfo MUST return 0.

profile_GetProfilePropertyInfo MUST return one of the following three user profile property result sets. These three result sets are referred to as Profile Property Result Set A (Section 3.1.4.23.1), Profile Property Result Set B (Section 3.1.4.23.2), and Profile Property Result Set C (Section 3.1.4.23.3), respectively.

If *@PropertyURI* is NULL and *@PropertyName* is NULL and either *@Collation* is NULL or *@Lcid* is NULL, then the columns specified in Profile Property Result Set A (Section 3.1.4.23.1) MUST be returned, ordered by DisplayOrder.

If *@PropertyURI* is NULL and *@PropertyName* is NULL and both *@Collation* and *@Lcid* are not NULL, then the columns specified in Profile Property Result Set B (Section 3.1.4.23.2) MUST be returned. Profile Property Result Set B (Section 3.1.4.23.2) is ordered by DisplayName according to the value of *@Collation*.

If either *@PropertyURI* is not NULL or *@PropertyName* is not NULL, then the columns specified in Profile Property Result Set C (Section 3.1.4.23.3) MUST be returned.

3.1.4.23.1 Profile Property Result Set A

Return Code Values: Profile Property Result Set A MUST include all rows in which the IsAuxiliary value is 0.

Result Sets: The T-SQL syntax for the Profile Property Result Set A is defined in Profile Property Result Set Columns (Section 3.1.4.23.4).

3.1.4.23.2 Profile Property Result Set B

Return Code Values: Profile Property Result Set B MUST include all rows in which the IsAuxiliary value is 0.

Result Sets: The T-SQL syntax for the Profile Property Result Set B is defined in Profile Property Result Set Columns (Section 3.1.4.23.4).

3.1.4.23.3 Profile Property Result Set C

Return Code Values: Profile Property Result Set C MUST include all rows in which either the PropertyURI value matches the *@PropertyURI* input parameter or the PropertyName value matches the *@PropertyName* input parameter.

Result Sets: The T-SQL syntax for the Profile Property Result Set C is defined in Profile Property Result Set Columns (Section 3.1.4.23.4).

3.1.4.23.4 Profile Property Result Set Columns

The three Profile Property result sets each return property information. The data columns contained by each result set are summarized in the following table.

This table includes the column name, the data type in T-SQL syntax, and an indication as to whether this column appears in Result Set A, Result Set B, or Result Set C.

Column Name	T-SQLData Type	ResultSet A	ResultSet B	ResultSet C
PropertyID	bigint	•	•	•
PropertyName	nvarchar(250)	•	•	•
PropertyURI	nvarchar(250)	•	•	•
DataTypeID	int	•	•	•
DataType	nvarchar(50)	•	•	•
VocValTypeID	int	•	•	•
Length	int	•	•	•
BlobType	tinyint	•	•	•
IsSection	bit	•	•	•
IsMultiValue	bit	•	•	•
IsSystem	bit	•	•	•
IsEditable	bit	•	•	•
IsAdminEditOnly	bit	•	•	•
IsImport	bit	•	•	•
IsAlias	bit	•	•	•
IsAuxiliary	bit	•	•	•

Column Name	T-SQLData Type	ResultSet A	ResultSet B	ResultSet C
IsColleagueEventLog	bit	•	•	•
IsUpgrade	bit	•	•	•
IsUpgradePrivate	bit	•	•	•
IsSearchable	bit	•	•	•
replicable	bit	•	•	•
ChoiceType	int	•	•	•
MaximumShown	int	•	•	•
DisplayName	nvarchar(512)		•	
Policy	int	•	•	•
DefaultItemSecurity	int	•	•	•
IsItemSecurityOverridable	bit	•	•	•
IsPolicyOverridable	bit	•	•	•
IsExpand	bit	•	•	•
IsVisible	bit	•	•	•
IsVisibleOnViewer	bit	•	•	•
DisplayOrder	int	•	•	•
Separator	tinyint	•	•	•
Name	nvarchar(500)	•	•	•
FriendlyTypeName	nvarchar(500)	•	•	•
IsEmail	bit	•	•	•
IsURL	bit	•	•	•
IsPerson	bit	•	•	•
IsHTML	bit	•	•	

PropertyID: Contains the identifier of the property.

PropertyName: Contains the name of the property.

PropertyURI: Contains the URI for this property.

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

DataType: Contains the data type of the property which MUST be listed in the following table:

T-SQL Type Name
int
bigint
datetime
float
nvarchar
varbinary
uniqueidentifier
bit

VocValTypeID: Contains the choice list identifier of the property.

Length: Contains the maximum length of the property's value.

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

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

IsMultiValue: Contains a value that indicates whether this property is a **multivalue property**. This value MUST be 1 to indicate a multi-value property.

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

IsEditable: Contains a value that indicates whether this property is editable. This value MUST be 1 to indicate that this property is 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 a property editable only by the administration.

IsImport: Contains a value that indicates whether this property is an imported property. This value MUST be 1 to indicate an imported property**.**

IsAlias: Contains a value that indicates whether this 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 e-mail alias of the user for user search purposes.

IsAuxiliary: Contains a value that indicates whether this property is an auxiliary property. This value MUST be 1 to indicate an auxiliary property**.**

IsColleagueEventLog: Contains a value that indicates whether this property is displayed on the **Colleague Tracker Web Part**. This value MUST be 1 to indicate that this property is displayed on the Colleague Tracker Web Part.

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.

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.

IsSearchable: Contains a value that indicates whether this property is indexed by the search server. This value MUST be 1 to indicate that this property is indexed by Search Server.

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

ChoiceType: MUST be a Property Choice (Section 2.2.2.9) 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 when rendering on a profile site.

DisplayName: Contains the localized user display name of the property based on the specified *@Lcid*.

Policy: MUST be a Privacy Policy (Section 2.2.2.8) Type value. The value of this attribute MUST be 0 for updated or built-in sections.

DefaultItemSecurity: Contains a value indicating the default Privacy (Section 2.2.2.7) Type of the property**.** The value of this attribute MUST be 0 for updated sections.

IsItemSecurityOverridable: MUST be an "Is Item Security Overridable" (Section 2.2.2.2) Type value.

IsPolicyOverridable: Contains a value that indicates that a policy override can be changed by administration. This value MUST be 1 to indicate that a policy override can be changed by administration.

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

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

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

DisplayOrder: Contains a value specifying the display order for the property.

Separator: Contains the multi-value user interface separator which MUST be a value listed in the following table:

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

Name: Contains the name of the property's T-SQL datatype.

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

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

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

IsPerson: Contains a value that indicates the value for this property, if non-NULL, MUST be a valid Domain Account. This value MUST be 1 to indicate that the value for this property MUST be a valid Domain Account.

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

3.1.4.24 profile_GetProfilePropertyLoc

The **profile_GetProfilePropertyLoc** stored procedure is called to retrieve all localized user display names and descriptions for each property.

profile_GetProfilePropertyLoc is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetProfilePropertyLoc();
```

Return Code Values: profile_GetProfilePropertyLoc MUST return 0.

Results Sets: profile_GetProfilePropertyLoc MUST return 1 result set.

3.1.4.24.1 Get Localized Profile Property Result Set

The Get Localized Profile Property result set returns localized property information. The result set MUST contain 1 or more rows.

The Get Localized Profile Property result set is defined using T-SQL syntax as follows:

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

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:

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 LCID indicating what language the specified text is in.

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

3.1.4.25 profile_GetPropertyChoiceList

The **profile_GetPropertyChoiceList** stored procedure is called to retrieve all the choice list values for the specified property.

profile_GetPropertyChoiceList is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetPropertyChoiceList (
    @PropertyName               nvarchar(50)
);
```

@PropertyName: The name of a property. This parameter MUST be specified. This parameter MUST NOT be NULL.

Return Code Values: profile_GetPropertyChoiceList MUST return 0.

Result Sets: profile_GetPropertyChoiceList MUST return 1 result set.

3.1.4.25.1 Choice List Values Result Set

The choice list values result set returns the existing choice list values for the *@PropertyName*. The result set MUST contain 0 or more rows.

The choice list values result set is defined using T-SQL syntax as follows:

```
VocValValue               sql_variant;
```

VocValValue: A choice list **value** associated with a property.

3.1.4.26 profile_GetSharedListSync

The **profile_GetSharedListSync** stored procedure is called to retrieve a shared list object based on the specified *@ListId*.

profile_GetSharedListSync is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetSharedListSync (
    @ListId                 uniqueidentifier
);
```

@ListId: The GUID value that uniquely identifies the shared list object being retrieved.

Return Code Values: This MUST return 0.

Result Sets: profile_GetSharedListSync MUST return 1 shared list result set.

3.1.4.26.1 Shared List Result Set

The Shared List result set returns 0 or more rows of shared list items.

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

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

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 group. The triplet separator MUST be ";;".

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

3.1.4.27 profile_GetUserFormat

The **profile_GetUserFormat** stored procedure is called to retrieve the **user name** format that is used by the Protocol server.

profile_GetUserFormat is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetUserFormat();
```

Return Code Values: profile_GetUserFormat MUST return 0.

Result Sets: profile_GetUserFormat MUST return one result set.

3.1.4.27.1 ProfileGetUserFormat Result Set

profile_GetUserFormat 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 NOT NULL;
```

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

Value	Description
1	Indicates that the system uses login names to identify users.
2	Indicates that the system uses distinguished names to identify users.

Value	Description
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.

3.1.4.28 profile_GetUserGUID

The **profile_GetUserGUID** stored procedure is called to retrieve 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,
    @SID                   varbinary(512) = NULL,
    @GUID                  uniqueidentifier OUTPUT,
    @RequireValues         bit = 0,
    @Debug                 bit = 0
);
```

@NTName: A SAM user name for the entity specified by the user profile.

@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.

Return Code Values: profile_GetUserGUID MUST return 0.

Result Sets: profile_GetUserGUID MUST NOT return any result set. Either *@NTName* or *@SID* MUST be specified. If both *@NTName* and *@SID* are specified, *@NTName* parameter MUST be used to identify the user and *@SID* MUST be ignored by the Protocol server.

3.1.4.29 profile_GetUserProfileData

The **profile_GetUserProfileData** stored procedure is called by a user who request 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,
    @NTName                         nvarchar(400) = NULL,
    @SID                            varbinary(512)= NULL,
    @RecordId                       bigint = NULL,
    @ViewerRights                   int OUTPUT,
    @ViewerNTName                   nvarchar(400) = NULL,
    @AllowAlternateAccountName      bit = 0,
    @bQuickLoad                     bit = 0,
    @bDebug                         bit = 0
);
```

@UserID: Identifies the user requested.

@NTName: A SAM user name for the entity specified by the user profile.

@SID: Contains the SID for the user requested.

@RecordId: Contains the matching GUID identifying the user requested.

@ViewerRights: Contains the rights of the **user** requesting the information. This MUST be a Privacy (Section 2.2.2.7) type value. If this parameter is set to (0x40000000) to indicate that a privacy policy is unset , then *@ViewerRights* MUST be set according to the requesting user's privacy policy based on *@ViewerNTName*.

@ViewerNTName: Contains the requesting NT Name. This MUST be NULL unless the *@ViewerRights* is set to PRIVACY_NOTSET.

@AllowAlternateAccountName: Contains 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.

Return Code Values: profile_GetUserProfileData MUST return 0.

One of *@UserID*, *@SID*, *@NTName*, or *@RecordID* MUST be specified.

If *@UserID* is NULL then the user will be identified by *@SID*. If *@SID* is NULL then user will be identified by *@NTName*. If *@NTName* is NULL then user will be identified by *@RecordId*.

Result Sets: profile_GetUserProfileData MUST return 1 result set of UserProperties:

3.1.4.29.1 UserProperties Result Set

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.

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

```
RecordID                    bigint,
PropertyID                  bigint,
PropertyVal                 sql_variant,
Image                       image,
```

```
Text                        ntext,
VocValID                    int,
OrderRank                   int,
Privacy                     int,
PropertyName                nvarchar(250),
PropertyURI                 nvarchar(250)
VocValValue                 sql_variant;
```

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.

PropertyID: The identifier of the property**.**

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

Image: The image of the property.

Text: The text description of the property**.**

VocValID: The **vocabulary identifier** of the property.

OrderRank: This value MUST be NULL.

Privacy: The privacy policy value.

PropertyName: The user-friendly name of the property**.**

PropertyURI: The URI reference of the property**.**

VocValValue: The **vocabulary value** of the property.

3.1.4.30 profile_GetUserProfilesByEmail

The **profile_GetUserProfilesByEmail** stored procedure is called to get user profile information based on the e-mail addresses specified.

profile_GetUserProfilesByEmail is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetUserProfilesByEmail(
    @RecordId               bigint,
    @Email0                 nvarchar(400) = NULL,
    @Email1                 nvarchar(400) = NULL,
    @Email2                 nvarchar(400) = NULL,
    @Email3                 nvarchar(400) = NULL,
    @Email4                 nvarchar(400) = NULL,
    @Email5                 nvarchar(400) = NULL,
    @Email6                 nvarchar(400) = NULL,
    @Email7                 nvarchar(400) = NULL,
    @Email8                 nvarchar(400) = NULL,
    @Email9                 nvarchar(400) = NULL,
    @Email10                nvarchar(400) = NULL,
    @Email11                nvarchar(400) = NULL,
    @Email12                nvarchar(400) = NULL,
    @Email13                nvarchar(400) = NULL,
    @Email14                nvarchar(400) = NULL,
    @Email15                nvarchar(400) = NULL,
    @Email16                nvarchar(400) = NULL,
```

```
    @Email17                nvarchar(400) = NULL,
    @Email18                nvarchar(400) = NULL,
    @Email19                nvarchar(400) = NULL,
    @Email20                nvarchar(400) = NULL,
    @Email21                nvarchar(400) = NULL,
    @Email22                nvarchar(400) = NULL,
    @Email23                nvarchar(400) = NULL,
    @Email24                nvarchar(400) = NULL,
    @Email25                nvarchar(400) = NULL,
    @Email26                nvarchar(400) = NULL,
    @Email27                nvarchar(400) = NULL,
    @Email28                nvarchar(400) = NULL,
    @Email29                nvarchar(400) = NULL,
    @Email30                nvarchar(400) = NULL,
    @Email31                nvarchar(400) = NULL,
    @Email32                nvarchar(400) = NULL,
    @Email33                nvarchar(400) = NULL,
    @Email34                nvarchar(400) = NULL,
    @Email35                nvarchar(400) = NULL,
    @Email36                nvarchar(400) = NULL,
    @Email37                nvarchar(400) = NULL,
    @Email38                nvarchar(400) = NULL,
    @Email39                nvarchar(400) = NULL
);
```

@RecordId: The record identifier of a user. The stored procedure MUST NOT return the user profile of the user corresponding to the specified e-mail address, if this user is a colleague of a user corresponding to a specified e-mail address.

@Email0-Email39: The e-mail address of the specified user for which to return the user profile. The stored procedure MUST return the profile information of all the different e-mail addresses that are not NULL.

Return Code Values: profile_GetUserProfilesByEmail MUST return 0.

Result Sets: profile_GetUserProfilesByEmail MUST return 1 result set.

3.1.4.30.1 UserProfile Result Set

profile_GetUserProfilesByEmail MUST return 0 or more rows corresponding to records that match a specified user's e-mail address.

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

```
    RecordID                bigint IDENTITY (1, 1) NOT NULL,
    UserID                  uniqueidentifier NOT NULL,
    NTName                  nvarchar(400) NOT NULL,
    PreferredName           nvarchar(256) NULL,
    Email                   nvarchar(256) NULL,
    SipAddress              nvarchar(250) NULL,
    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 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 SIP address of the user.

PersonTitle: Contains the user title.

3.1.4.31 profile_GetUserReportToData

The **profile_GetUserReportToData** stored procedure is called to retrieve the **Manager** property associated with a specific user, the colleague properties associated with a specific user, and the reports associated with a specific user. This procedure is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetUserReportToData(
    @Collation              nvarchar(60),
    @UserID                 uniqueidentifier,
    @NTName                 nvarchar(400) = NULL,
    @SID                    varbinary(512) = NULL,
    @bDebug                 bit = 0
);
```

@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 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.

Return Code Values: profile_GetUserReportToData MUST return 0.

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.

Result Sets: profile_GetUserReportToData MUST return three result sets when the user has a **Manager** property.

profile_GetUserReportToData MUST return two result sets when the user does not have a **Manager** property. All result sets are of the same structure, MUST be sorted by PreferredName on *@Collation* and ordered as follows:

1. Reports
2. **Manager** property
3. **Peer**

3.1.4.31.1 profile_GetUserReportToDataResult Set

The **profile_GetUserReportToData** result set returns user profile data pertaining to the specified user's reports, manager, and peers.

profile_GetUserReportToData is defined using T-SQL syntax as follows:

```
RecordID                    bigint,
UserID                      nvarchar(400),
PreferredName               nvarchar(256),
Email                       nvarchar(256),
SipAddress                  nvarchar(250),
PersonTitle                 nvarchar(150);
```

RecordID: Contains the record identifier for the user.

UserID: Contains the GUID identifying the user.

NTName: A SAM user name for the entity specified by the user profile.

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

Email: The e-mail address of the user.

SipAddress: The SIP address of the user.

PersonTitle: The title of the user.

3.1.4.32 profile_GetViewerRights

The **profile_GetViewerRights** stored procedure is called to determine what rights the user specified by *@ViewerNTName* has to see items owned by the user specified by *@RecordId*.

profile_GetViewerRights is defined using T-SQL syntax as follows:

```
PROCEDURE profile_GetViewerRights(
    @RecordId               BigInt,
    @ViewerNTName           nvarchar(400),
    @ManagerNTName          nvarchar(400) = null
);
```

@RecordId: The *@RecordId* of the user who owns the items. This MUST NOT be NULL.

@ViewerNTName: The login name of user requesting rights. This MUST NOT be NULL.

@ManagerNTName: The **Manager** property of user specified by *@RecordId* parameter.

Return Value: profile_ GetViewerRights returns a bitmask of the permission values listed in the following table:

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*.

Value	Description
4	Workgroup. This bit MUST be set if the *@ViewerNTName* corresponds to a user in the *@RecordId* 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 return 1 result set.

3.1.4.33 profile_MigrateUserProfile

The **profile_MigrateUserProfile** stored procedure is called to update 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(
    @OldNTName              nvarchar(400),
    @OldSID                 varbinary(512),
    @NewNTName              nvarchar(400),
    @NewSID                 varbinary(512),
    @PersonalSiteReaders    ntext
);
```

@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*, it MUST be deleted to avoid multiple profiles for a single user. This value MUST be specified and MUST NOT be NULL.

@NewSID: The new SID that 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** (Section 3.1.4.20) stored procedure and then all occurrences of the *@OldNTName* MUST be replaced with *@NewNTName* prior to the call of the **profile_MigrateUserProfile** stored procedure.

Return Code Values: profile_MigrateUserProfile returns an integer return code which MUST be listed 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: profile_MigrateUserProfile MUST NOT return any result set.

3.1.4.34 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(
  @NewSSP         bit,
  @ClearListData  bit
);
```

@NewSSP: This parameter specifies if a new **Shared Services Provider (SSP)** was created as part of the restore operation. It MUST be 1 of these values:

Value	Description
1	A new SSP has been created as part of the restore operation.
0	The restore operation was performed on an existing SSP.

@ClearListData: Specifies whether the stored procedure deletes all shared list objects. *@ClearListData* MUST be one of the following values:

Value	Description
1	All shared list objects MUST be deleted.
0	Shared list objects MUST NOT deleted.

Return Code Values: profile_OnSqlRestore MUST return a return code of 0.

Result Sets: profile_OnSqlRestore MUST NOT return any result set.

3.1.4.35 profile_RemoveUser

The **profile_RemoveUser** stored procedure is called to delete a user profile from the Protocol server. The procedure is defined using T-SQL syntax as follows:

```
PROCEDURE profile_RemoveUser(
    @UserID            uniqueidentifier,
    @NTName            nvarchar(400) = null,
    @SID               varbinary(512) = null
);
```

@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.

Return Code Values: profile_RemoveUser returns an integer return code that MUST be listed in the following table:

Value	Description
0	A user was deleted.
1	A user could not be found to delete.
Other	Operation not successful.

Result Sets: profile_RemoveUser MUST NOT return any result set.

3.1.4.36 profile_ResetDeletedUser

The **profile_ResetDeletedUser** stored procedure is called to unmark the Profile Pending Deletion Flag on a user profile and set the account to active.

profile_ResetDeletedUser is defined using T-SQL syntax as follows:

```
PROCEDURE profile_ResetDeletedUser(
   @UserID                 uniqueidentifier
);
```

@UserID: Contains the GUID of the user to activate. This value MUST be specified.

Return Code Values: profile_ResetDeletedUser MUST return 0.

Result Sets: profile_ResetDeletedUser MUST NOT return any result set.

3.1.4.37 profile_SearchUser

The **profile_SearchUser** stored procedure is called to return the users that match the search criteria.

profile_SearchUser is defined using T-SQL syntax as follows:

```
PROCEDURE DBO.profile_SearchUser(
   @SearchString           nvarchar(250),
   @RowCountStart          int,
   @RowCountEnd            int,
   @AccountName            bit,
   @PreferredName          bit = NULL,
   @Email                  bit = NULL,
   @WildSearch             bit = 1,
   @DoGeneralSearch        bit = 0,
   @Collation              nvarchar(60),
   @bActiveOnly            bit = 1,
   @TotalRowCount          int OUTPUT
);
```

@SearchString: Specifies the search input text for matching user records. This Parameter MUST be specified and MUST NOT be NULL.

@RowCountStart: Specifies which row the protocol server starts the search from. This value MUST be specified if *@DoGeneralSearch* is not set to 1. If *@DoGeneralSearch* is set to 1, then this parameter MUST be ignored. If this value is set to 0, then the protocol server MUST return 0 records.

@RowCountEnd: Specifies which row the protocol server stops returning records after. This value MUST be specified if *@DoGeneralSearch* is not set to 1. If *@DoGeneralSearch* is set to 1, then this parameter MUST be ignored. If specified, this value MUST be greater than or equal to *@RowCountStart*.

@AccountName: Specifies whether the search uses the login name. If this parameter is set to 1, it specifies that search MUST use the login name accounts to match the results.

@PreferredName: Specifies whether the search uses the user display name. If this parameter is set to 1, it specifies that search MUST use the user display name to match the results.

@Email: Specifies whether the search uses the e-mail address. If this parameter is set to 1, it specifies that search MUST use the e-mail address to match the results.

@WildSearch: This parameter specifies how the protocol server does the search. This value MUST be 1 of the following values:

Value	Description
0	The protocol server MUST search exactly by *@SearchString*.
1	The protocol server MUST search for values that start with *@SearchString*.

@DoGeneralSearch: Specifies whether the protocol server searches on all possible values in login name, user display name, and e-mail address.

@Collation: This specifies that the results MUST be ordered by the collation sequence specified in this parameter and not according to the collation sequence of the column as defined in the table or view. If *@DoGeneralSearch* and *@PreferredName* are 1, the protocol server MUST ignore this parameter. If *@DoGeneralSearch* and *@PreferredName* are not 1, this parameter MUST be specified and MUST NOT be NULL.

@bActiveOnly: This value specifies if the protocol server MUST search through records that are no longer active. This value MUST be 1 of the following values:

Value	Description
0	The protocol server MUST search only through the records that are no longer active.
1	The protocol server MUST search only through the records that are active.

@TotalRowCount: This value specifies the total number of records found in the user profile table. This value is not related to the search criteria. If *@bActiveOnly* is set to 1, the protocol server MUST set this to the number all active user records in the list. If *@bActiveOnly* is set to 0, the protocol server MUST set this to the number of all inactive records.

Return Code Values: profile_SearchUser returns an integer return code which MUST be listed in the following table:

Value	Description
0	Successful execution.
1	No search column is specified.

Result Sets: profile_SearchUser MUST NOT return any result set when the stored procedure's return code is 1. When the return code of **profile_SearchUser** is 0, the stored procedure MUST return 1 result set UserInfo which MUST have 0 or more rows:

If *@SearchString* is empty and *@DoGeneralSearch* is not set to 1, then the protocol server MUST return all rows. Note the protocol server still takes into account *@bActiveOnly*.

What values are searched on depends on *@DoGeneralSearch*, *@AccountName*, *@PreferredName*, and *@Email*. All values MUST be searched on if *@DoGeneralSearch* and *@PreferredName* are set to 1. In that case, the values in *@Email* and *@AccountName* MUST be ignored.

If *@DoGeneralSearch* is not 1, then the protocol server looks to *@AccountName*, *@PreferredName*, and *@Email* to see what to search on.

If *@Email*Address is set to 1, the protocol server MUST only search for matching e-mail addresses and ignores the values in *@PreferredName* and *@AccountName*.

If *@Email*Address is not set to 1 but *@PreferredName* is set 1 then, then the protocol server MUST only search for matching user display names.

If *@Email*Address and *@PreferredName* aren't set to 1 but *@AccountName* is set to 1, then the protocol server MUST only search for matching login names.

3.1.4.37.1 UserInfo Result Set

profile_SearchUser returns a row for each record that matches the search string.

Return Code Values: The UserInfo result set MUST return 0 or more rows that contain the user Info for each matching record.

Result Sets: The result set MUST be sorted by *@RecordID* if both *@PreferredName* and *@DoGeneralSearch* are set to 1. Otherwise the result set MUST be sorted by login name.

The UserInfo result set uses T-SQL syntax, as follows:

```
ID                      int,
RecordID                bigint,
UserID                  uniqueidentifier,
AccountName             nvarchar(400),
PreferredName           nvarchar(256),
Email                   nvarchar(256),
Department              nvarchar(250),
Title                   nvarchar(150);
```

ID: Contains the sequential Id for rows in the result set starting at 1.

RecordID: Contains the record identifier for the user.

UserID: Contains the GUID for the user.

AccountName: Contains the full name of the user.

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

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

Department: Contains the Department name of the user.

Title: Contains the user's title.

3.1.4.38 profile_UpdateOrgColleagues

The **profile_UpdateOrgColleagues** stored procedure is called to update 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();
```

Return Code Values: profile_UpdateOrgColleagues returns an integer return code which MUST be listed 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: **profile_UpdateOrgColleagues** MUST NOT return any result set.

3.1.4.39 profile_UpdatePersonalSiteInfo

The **profile_UpdatePersonalSiteInfo** stored procedure is called to update the personal site configuration properties.

profile_UpdatePersonalSiteInfo is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdatePersonalSiteInfo(
    @Inclusion            nvarchar(300),
    @NameFormat           smallint,
    @SiteReader           ntext,
    @EnableMLS            bit
);
```

@Inclusion: The site provider under which user personal sites are created. The value MUST be a valid **server-relative URL**.

@NameFormat: MUST be a Name Format (Section 2.2.2.5) 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 2.2.2.3) Type value.

Return Code Values: profile_UpdatePersonalSiteInfo MUST return 0.

Result Sets: profile_UpdatePersonalSiteInfo MUST NOT return any result set.

3.1.4.40 profile_UpdatePersonalSpace

The **profile_UpdatePersonalSpace** stored procedure is called to update a user's personal site URL.

profile_UpdatePersonalSpace is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdatePersonalSpace(
    @UserGuid               uniqueidentifier,
    @PersonalSpaceURL       nvarchar(2048)
);
```

@UserGuid: This MUST be the GUID for the user whose personal site is being updated.

@PersonalSpaceURL: The server-relative URL of the personal site. This parameter MUST be specified and MUST be a server-relative URL.

Return Code Values: profile_UpdatePersonalSpace MUST 0.

Result Sets: profile_UpdatePersonalSpace MUST NOT return any result set.

3.1.4.41 profile_UpdateProfileDisplay

The **profile_UpdateProfileDisplay** stored procedure is called to update the display order of **properties** and **sections**

profile_UpdateProfileDisplay is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdateProfileDisplay(
    @UpdateList             ntext,
    @Debug                  bit = 0
);
```

@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 3.1.4.41.2) schema.

@Debug: This parameter MUST be ignored.

Return Code Values: profile_UpdateProfileDisplay MUST return 0.

Result Sets: profile_UpdateProfileDisplay MUST return 1 result set.

3.1.4.41.1 UpdateProfileDisplay Result Set

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; 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.

3.1.4.41.2 UpdateList Schema

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

Usage Example:

```
<MSPROFILE>
   <PROFILE ProfileName="UserProfile">
      <ITEM PropertyName="HomeAddress" DisplayOrder="1" />
      <ITEM PropertyName="ZipCode" DisplayOrder="2" />
   </PROFILE>
</MSPROFILE>
```

3.1.4.41.2.1 MSPROFILE Element

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

PROFILE: Must be a PROFILE (Section 3.1.4.41.2.2) Type element. This element MUST be specified.

3.1.4.41.2.2 PROFILE Element

```
<s:element name="PROFILE">
   <s:complexType>
     <s:sequence>
       <s:element name="ITEM" type="s0:ITEM" 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".

ITEM: Must be an **ITEM** (Section 3.1.4.41.2.3) Type element.

3.1.4.41.2.3 ITEM Element

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

PropertyName: The name of the property or section.

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**.**

3.1.4.42 profile_UpdateProperty

The **profile_UpdateProperty** stored procedure is called to add, update and remove **properties**. It can also be used to update sections.

profile_UpdateProperty is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdateProperty(
    @RemovePropertyList        ntext,
    @UpdatePropertyList        ntext,
    @Debug                     bit = 0
);
```

@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 3.1.4.42.3)**.**

@UpdatePropertyList: XML code which lists the properties to add or update. This parameter MUST be specified. It MUST adhere to the UpdateProperty Schema (Section 3.1.4.42.3).

@Debug: This parameter MUST be ignored.

Return Code Values: profile_UpdateProperty MUST return 0.

Result Sets: profile_UpdateProperty MUST return one or two result sets. The Update Property result set MUST always be returned. If *@RemovePropertyList* contains a property used in Audience then the Audience result set MUST be returned; otherwise it MUST NOT be returned.

3.1.4.42.1 Update Property Result Set

The Update Property 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 .

Return Code Values: The Update Property result set MUST contain only 1 row.

Result Sets: The Update Property result set is defined using T-SQL syntax for the stored procedure as follows:

```
ERROR                              int,
RemovedPropertyCount               int,
XMLRemoveProfileERR                int,
```

```
XMLRemovePropertyERR              int,
UpdatePropertyCount               int,
XMLUpdateProfileErr               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.
2	The value of the **ProfileName** attribute in the *@RemovePropertyList* XML could not be found.
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.
21	The value of the **ProfileName** attribute in the *@UpdatePropertyList* XML could not be found.
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.
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 ignored.

XMLRemoveProfileERR: If *@RemovePropertyList* is not NULL, this value MUST be set to 1 if the user profile (whose **ProfileName** value is set to "UserProfile") could not be opened; otherwise it MUST be set to 0.

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

UpdatePropertyCount: This value MUST be ignored.

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

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

3.1.4.42.2 Audience Result Set

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.

3.1.4.42.3 UpdateProperty Schema

The complex types, simple types, and elements that are specified in this section are used in the **profile_UpdateProperty** (Section 3.1.4.42) stored procedure.

3.1.4.42.3.1 MSPROFILE Element

The **MSPROFILE** element contains a **PROFILE** (Section 3.1.4.42.3.2) element.

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

3.1.4.42.3.2 PROFILE Element

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

```
<s:element name="PROFILE">
   <s:complexType>
     <s:element minOccurs="0" maxOccurs="1" name="PROPERTY" type="s0:PROPERTY" />
     <s:attribute name="ProfileName" type="s:string" use="required" />
   </s:complexType>
</s:element>
```

ProfileName: This value MUST be "UserProfile".

Property: This value MUST be in 1 of the 3 following forms depending on whether the protocol client is using an UpdateProperty Schema (Section 3.1.4.42.3) value to remove a property, add a property, or update a property.

3.1.4.42.3.2.1 PROPERTY Element for Remove Operations

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="PropertyName" type="s:string" use="required" />
    </s:complexType>
</s:element>
```

PropertyName: Contains the name of a property.

3.1.4.42.3.2.2 PROPERTY Element for Add Operations

The PROPERTY element contains XML code for adding a property.

```
<s:element name="PROPERTY">
  <s:complexType>
    <s:element name="VOCABULARY" type="s0:VOCABULARY" minOccurs="0"
               maxOccurs="1" />
    <s:attribute name="PropertyName" type="s:string" 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="ChoiceType" type="s:int" use="required" />
    <s:attribute name="IsEditable" type="s:boolean" use="required" />
    <s:attribute name="IsAdminEditOnly" type="s:boolean" use="required" />
    <s:attribute name="IsColleagueEventLog" 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.

DataTypeId: MUST contain a Property Data (Section 2.2.2.10) 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 2.2.2.10) Type definition.

DefaultPrivacy: Contains a value indicating the default Privacy (Section 2.2.2.7) Type of the property**.**

UserOverridePrivacy: MUST be an Is Item Security Overridable (Section 2.2.2.2) 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 2.2.2.8) 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.

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.

ChoiceType: MUST be a Property Choice (Section 2.2.2.9) Type value.

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

IsColleagueEventLog: Indicates whether changes to this property are displayed in a Colleague Tracker Web Part. This value MUST be 1 to indicate that changes to this property are displayed in a Colleague Tracker Web Part.

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.

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.

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.

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.

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

IsExpand: This value MUST be 1.

Separator: MUST be a Separator (Section 2.2.2.11) 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.

3.1.4.42.3.2.3 PROPERTY Element for Update Operations

The PROPERTY element contains XML code for updating a property.

```
<s:element name="PROPERTY">
  <s:complexType>
    <s:element name="VOCABULARY" type="s0:VOCABULARY" minOccurs="0"
               maxOccurs="1" />
    <s:attribute name="PropertyName" type="s:string" use="required" />
    <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="ChoiceType" type="s:int" use="required" />
      <s:attribute name="IsEditable" type="s:boolean" use="required" />
      <s:attribute name="IsAdminEditOnly" type="s:boolean" use="required" />
      <s:attribute name="IsColleagueEventLog" 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.

DefaultPrivacy: Contains a value indicating the default Privacy (Section 2.2.2.7) 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 2.2.2.2) 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 2.2.2.8) 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.

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.

ChoiceType: MUST be a Property Choice (Section 2.2.2.9) Type value.

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

IsColleagueEventLog: Indicates whether changes to this property are displayed in a Colleague Tracker Web Part. This value MUST be 1 to indicate that changes to this property are displayed in a Colleague Tracker Web Part.

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.

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.

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.

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.

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

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 2.2.2.11) 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.

3.1.4.42.3.3 VOCABULARY Element

The VOCABULARY element contains 1 or more term elements that represent items in a choice list if this property is a multi-value property.

```
<s:element name="VOCABULARY">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="1" maxOccurs="unbounded" name="TERM" type="s0:TERM" />
    </s:sequence>
  </s:complexType>
</s:element>
```

The VOCABULARY element MUST NOT be specified unless the property is a multi-value property. Such properties will have the IsMultiValue attribute of the PROPERTY element set to 1.

3.1.4.42.3.4 TERM Element

The TERM element is used to specify choice list items to be updated, added or removed.

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

```
      <s:attribute name="Type" type="s:string" use="required" />
      <s:attribute name="Value" type="s:string" use="required" />
      <s:attribute name="NewValue" type="s:string" />
    </s:complexType>
  </s:element>
```

Type: The value MUST be listed in the following table:

Value	Description
Add	The specified term is being added to the list of choice list items.
Delete	The specified term is being deleted from the list of choice list items.
Rename	The specified term is being renamed. The original name is specified in the Value attribute and the new name is specified in the NewValue attribute. The NewValue attribute is required only if the term is being renamed.

Value: The text value of the choice list item.

NewValue: The renamed text value of the choice list item. This MUST be specified if Type is set to the string literal "Rename".

3.1.4.43 profile_UpdatePropertyLoc

The **profile_UpdatePropertyLoc** stored procedure is called to update 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),
    @DisplayNamesXml           ntext,
    @DescriptionsXml           ntext
);
```

@PropertyName: The name of a property. This parameter MUST be specified and MUST NOT be NULL.

@DisplayNamesXml: XML code which lists user display names. This parameter MUST be specified and MUST adhere to the UpdatePropertyLoc Schema (Section 3.1.4.43.1).

@DescriptionsXml: XML code which lists descriptions. This parameter MUST be specified and MUST adhere to the UpdatePropertyLoc Schema (Section 3.1.4.43.1).

Return Code Values: profile_UpdatePropertyLoc stored procedure returns an integer return code which MUST be listed in the following table:

Value	Description
0	Successful execution.
Non-	Failure. The *@PropertyName* did not reference an existing property or the XML parameters

Value	Description
zero	could not be processed.

Result Sets: profile_UpdatePropertyLoc MUST NOT return any result set.

3.1.4.43.1 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>
```

3.1.4.43.1.1 Loc Element

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

Item: An **Item** (Section 3.1.4.43.1.2) element (from the UpdatePropertyLoc schema) that MUST be specified in the *@DisplayNamesXml* input parameter.

3.1.4.43.1.2 Item Element

```
<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<1>.

3.1.4.44 profile_UpdateSharedListSync

The **profile_updateSharedListSync** stored procedure is called to update a shared list object.

profile_updateSharedListSync is defined using T-SQL syntax as follows:

```
PROCEDURE profile_updateSharedListSync(
    @ListId                uniqueidentifier,
```

```
    @ItemsXml              ntext
);
```

@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 that MUST conform to the following **XML schema**. For **more information about XML schemas, see** [XML10], [XMLNS], [XMLINFOSET], [XMLSCHEMA1], and [XMLSCHEMA2].

This value MUST be specified.

```
<xs:element name="List" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Item">
        <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" minOccurs="0"/>
          <xs:attribute name="TargetTo" type="xs:string" minOccurs="0"/>
          <xs:attribute name="Int1" type="xs:int" minOccurs="0"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

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.

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 parameter MUST be ignored.

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.

Return Code Values: profile_updateSharedListSync MUST return 0.

Result Sets: profile_updateSharedListSync MUST NOT return any result set.

3.1.4.45 profile_UpdateUserProfileBlobData

The **profile_UpdateUserProfileBlobData** stored procedure is called to update the user profile properties which can't be updated through **profile_UpdateUserProfileData** (Section 3.1.4.46).

profile_UpdateUserProfileBlobData is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdateUserProfileBlobData(
@UserID                         uniqueidentifier,
@NTName                         nvarchar(400),
@PropertyName                   nvarchar(250),
@PropertyValVarbinary           varbinary(7500) = null,
@PropertyValText               ntext = null,
@PropertyValImage              image = null,
@Debug                          bit = 0
);
```

@UserID: Contains the GUID for the user. If this value is NULL then the user MUST be identified by *@NTName*.

@NTName: The NTName for the user. 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 to be updated.

@PropertyValImage: Contains the image value of the property to be updated.

@Debug: This parameter MUST be ignored.

Return Code Values: profile_UpdateUserProfileBlobData MUST return 0.

Result Sets: profile_UpdateUserProfileBlobData MUST return one result set.

If both *@UserId* and *@NTName* are specified, *@UserId* is used by the protocol server.

If *@UserId* and *@NTName* don't 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*, *@PropertyValText*, or *@PropertyValImage*

3.1.4.45.1 UpdateUserProfileBlobDataResult Result Set

profile_UpdateUserProfileBlobData returns a UpdateUserProfileBlobDataResult that 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 **profile_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.

3.1.4.46 profile_UpdateUserProfileData

The **profile_UpdateUserProfileData** stored procedure is called to update the user profile data.

profile_UpdateUserProfileData is defined using T-SQL syntax as follows:

```
PROCEDURE profile_UpdateUserProfileData(
    @UpdatePropertyList           ntext,
    @Debug                        bit = 0
);
```

@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 3.1.4.46.1)**.**

@Debug: This parameter MUST be ignored.

Return Code Values: profile_UpdateUserProfileData MUST return 0.

Result Sets: profile_UpdateUserProfileData MUST return 1 result set OperationResult. This result set MUST contain 1 row.

3.1.4.46.1 UpdateUserProfileData schema

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

3.1.4.46.1.1 Usage Example

```
<MSPROFILE >
   <PROFILE ProfileName ="UserProfile">
      <USER UserID="12345" NTAccount="redmond\user">
        <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>
```

3.1.4.46.1.2 MsProfile Element

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

PROFILE: A Profile (Section 3.1.4.46.1.3) Type element.

3.1.4.46.1.3 Profile Element

```
<xs:element name="PROFILE" type="">
```

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

USER: An ArrayOfUser (Section 3.1.4.46.1.4) Type element.

3.1.4.46.1.4 ArrayOfUser Element

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

UserID: The GUID of the user

NTAccount: A SAM user name for the entity specified by the user profile.

PROPERTY: An ArrayOfProperty (Section 3.1.4.46.1.5) Type element.

3.1.4.46.1.5 ArrayOf Property Element

```
<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="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 will 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, the protocol server MUST compose the list of values in sequence and MUST NOT be interwoven with other properties.

3.1.4.46.2 OperationResult Result Set

profile_UpdateUserProfileData returns OperationResult that contains Error information in case of failure or the count of properties updated in the case of a success.

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

```
ERROR                       int,
XMLUpdateUserErr            int,
XMLUpdatePropertyErr        int,
UpdatePropertyCount         int,
NEWUSERGUID                 uniqueidentifier;
```

ERROR: Contains the SQL process error not including the XML document error. This MUST be set if there is a SQL error.

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: This MUST always be NULL.

3.1.4.47 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 types membership, Colleague, or **quick link**.

QuickLinksAdd is defined using T-SQL syntax as follows:

```
PROCEDURE QuickLinksAdd(
     @RecordId                bigint,
     @PageURL                 nvarchar(1250) NULL,
     @Title                   nvarchar(500) NULL,
     @ContentClass            nvarchar(200) NULL,
     @Group                   nvarchar(400),
     @GroupType               int,
     @ItemSecurity            int,
     @PolicyId                uniqueidentifier,
     @ColleagueRecordId       bigint NULL,
     @LinkUserIdIsWorkgroup   bit NULL,
     @LinkMemberGroupId       bigint NULL
);
```

@RecordId: The ID of a user**.** This parameter MUST be specified and it MUST NOT be NULL.

@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 2.2.2.1) Type value that specifies the link group type. This parameter MUST be specified.

@ItemSecurity: MUST be a v Type value that defines security for the link.

@PolicyId: MUST be a Policy Link (Section 2.2.2.6) Type value that specifies the link type. This parameter MUST be specified.

@ColleagueRecordId: The ID 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.

Return Code Values: QuickLinksAdd returns an integer return code which MUST be listed in the following table:

Value	Description
A positive number	Success. This is the identifier of the link that was added.
-1	Failure – Duplicate information exists. No data is added.

Result Sets: **QuickLinksAdd** MUST NOT return any result set.

3.1.4.48 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,
    @LinkID              bigint,
    @PolicyId            uniqueidentifier,
    @RemoveCount         int OUTPUT
```

```
);
```

@RecordId: The record identifier of a user**.** This parameter MUST be specified and it MUST NOT be NULL.

@LinkID: The ID of a link. This parameter MUST be specified and it MUST NOT be NULL.

@PolicyId: MUST be a Policy Link (Section 2.2.2.6) Type **value** that specifies the link type. This parameter MUST be specified.

@RemoveCount: This is an output parameter specifying the number of data rows being deleted. If a link is deleted *@RemoveCount* MUST be 1.

Return Code Values: QuickLinksDelete MUST return 0.

Result Sets: **QuickLinksDelete** MUST NOT return any result set.

3.1.4.49 QuickLinksDeleteUser

The **QuickLinksDeleteUser** stored procedure is called to delete 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,
    @PolicyId               uniqueidentifier
);
```

@RecordId: The ID of a user**.** This parameter MUST be specified and it MUST NOT be NULL.

@PolicyId: MUST be a Policy Link (Section 2.2.2.6) Type value that specifies the link type.

Return Code Values: QuickLinksDeleteUser MUST return 0.

Result Sets: QuickLinksDeleteUser MUST NOT return any result set.

3.1.4.50 QuickLinksEdit

The **QuickLinksEdit** stored procedure is called to modify or add 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,
    @LinkID                 bigint,
    @PageURL                nvarchar(1250) NULL,
    @Title                  nvarchar(500) NULL,
    @ContentClass           nvarchar(200) NULL,
    @Group                  nvarchar(400),
    @GroupType              int,
    @ItemSecurity           int,
    @PolicyId               uniqueidentifier,
    @ColleagueRecordId      bigint NULL,
```

```
    @LinkUserIdIsWorkgroup   bit NULL,
    @LinkMemberGroupId       bigint NULL
);
```

@RecordId: The ID of a user. This parameter MUST be specified and it MUST NOT be NULL.

@LinkID: The ID 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 2.2.2.1) Type value that specifies the link group type. This parameter MUST be specified.

@ItemSecurity: MUST be a Privacy (Section 2.2.2.7) Type value that defines security for the item**.**

@PolicyId: MUST be a Policy Link (Section 2.2.2.6) Type value that specifies the link type. This parameter MUST be specified.

@ColleagueRecordId: The ID 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 ID. 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 ID for a user. If this parameter is not NULL, it specifies that the link data is a Membership link.

Return Code Values: QuickLinksEdit MUST return 0.

Result Sets: QuickLinksEdit MUST NOT return any result set.

3.1.4.51 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(
    @RecordId                bigint,
    @ViewerItemSecurity      int,
```

```
    @RequestedItemSecurity     int
);
```

@RecordId: The identifier of a user. This parameter MUST be specified and it MUST NOT be NULL.

@ViewerItemSecurity: A value indicating the rights of the user who is requesting to see this user's personal data. Each bit corresponds to a privacy level. This parameter MUST be specified and its 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 workgroup colleagues.
8	The only two users allowed to access the resource are the owner of the resource and the owner's manager.
16	The only user allowed to access the resource is the owner of the resource.

@RequestedItemSecurity: This value defines the maximum level of privacy for returned data as a Privacy (Section 2.2.2.7) Type. This parameter MUST be specified.

Return Code Values: QuickLinksRetrieveAllItems MUST return 0.

Result Sets: QuickLinksRetrieveAllItems MUST return three result sets defined in the following section.

3.1.4.51.1 UserColleagues Result Set

UserColleagues result set returns property data about colleagues.

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

```
PolicyId                uniqueidentifier,
Id                      bigint,
RecordId                bigint,
GroupType               tinyint,
GroupTitle              nvarchar(400),
ItemSecurity            int,
IsWorkgroup             bit,
ColleagueRecordId       bigint,
UserId                  uniqueidentifier,
NTName                  nvarchar(400),
Email                   nvarchar(256),
SipAddress              nvarchar(250),
PreferredName           nvarchar(256),
PersonTitle             nvarchar(150);
```

PolicyId: Contains the link type, the value MUST be listed in the following table:

Value	Description
EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C	PolicyId_Colleague

Id: Contains the unique ID for the colleague / user pair.

RecordId: Contains the record ID of the user's Colleague.

GroupType: MUST be a Group (Section 2.2.2.1) Type value which specifies the group type of the Colleague.

GroupTitle: Contains the group title of the colleague..

ItemSecurity: MUST be a Privacy (Section 2.2.2.7) Type value that defines security for the item.

IsWorkgroup: Contains a flag specifying whether the record is a workgroup.

ColleagueRecordId: Contains the ID of the user.

UserId: Contains the GUID of the user.

NTName: A SAM user name for the entity specified by the user profile.

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

SipAddress: Contains the SIP address for the user.

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

PersonTitle: This MUST always be NULL.

Return Code Values: MUST return 0.

Result Sets: The result set MUST contain 0 or more rows,

3.1.4.51.2 UserLinks Result Set

UserLinks returns user-defined link data.

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

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

PolicyId: Contains the link type, the value MUST be listed in the following table:

Value	Description
861D8FB6-7012-4CD9-A7A0-A615AED038B3	PolicyId_QuickLink

Value	Description
E21FE63D-0BF6-42C0-85BC-AC8031552558	PolicyId_PersonalizationLink

Id: Contains the unique ID for the link / user pair.

RecordId: Contains the record ID of the user.

GroupType: MUST be a Group (Section 2.2.2.1) Type value that specifies the link group type.

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

ItemSecurity: MUST be a Privacy (Section 2.2.2.7) Type value that defines security for the user.

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

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

ContentClass: Contains the content class for the user's link.

Return Code Values: MUST return 0.

Result Sets: The result set MUST contain 0 or more rows.

3.1.4.51.3 UserMemberships Result Set

UserMemberships result set returns a user's membership data. The result set is defined using T-SQL syntax as follows:

```
PolicyId                uniqueidentifier,
Id                      bigint,
RecordId                bigint,
GroupType               tinyint,
GroupTitle              nvarchar(400),
ItemSecurity            int,
MemberGroupId           bigint,
Id                      bigint,
DisplayName             nvarchar(250),
MailNickName            nvarchar(250),
Description             nvarchar(1500),
Source                  uniqueidentifier,
SourceReference         nvarchar(2048),
cs_SourceReference      int,
Url                     nvarchar(2048),
MemberCount             bigint,
LastUpdate              datetime,
WebID                   uniqueidentifier,
Type                    tinyint,
UserCreated             bit;
```

PolicyId: Contains a Short Link (Section 2.2.2.13) Type identifier specifying the link type.

Id: Contains the unique ID for the Membership / user pair.

RecordId: Contains the ID for the user.

GroupType: MUST be a Group (Section 2.2.2.1) Type value that specifies the Membership Group type.

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

ItemSecurity: MUST be a Privacy (Section 2.2.2.7) Type value that defines security for the membership of the specified user.

MemberGroupId: Contains the membership group ID for the user.

Id: Contains the Membership Group ID 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 2.2.2.13) Type identifier for the Membership Group.

SourceReference: Contains the source reference for the Membership Group.

cs_SourceReference: Contains the source reference checksum for the Membership Group.

Url: Contains the URL for the Membership Group.

MemberCount: Contains the members count for the Membership Group.

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

WebID: Contains the unique web ID for the Membership Group.

Type: Contains the type for the Membership Group.

UserCreated: MUST be an Is User Created (Section 2.2.2.4) Type value.

Return Code Values: MUST return 0.

Result Sets: The result set MUST contain 0 or more rows

3.1.4.52 QuickLinksRetrieveColleaguesOfColleagues

The **QuickLinksRetrieveColleaguesOfColleagues** stored procedure is called to retrieve the specified user's colleagues' colleagues.

QuickLinksRetrieveColleaguesOfColleages is defined using T-SQL syntax as follows:

```
PROCEDURE QuickLinksRetrieveColleaguesOfColleagues (
    @RecordId                       bigint
);
```

@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.

Return Code Values: QuickLinksRetrieveColleaguesOfColleagues MUST return 0.

Result Sets: QuickLinksRetrieveColleaguesOfColleagues MUST return 1 result set.

3.1.4.52.1 QuickLinksRetrieveColleaguesOfColleagues Result Set

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 information 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:

```
RecordID                bigint,
UserID                  uniqueidentifier,
SipAddress              nvarchar(250),
NTName                  nvarchar(400),
Email                   nvarchar(256),
PreferredName           nvarchar(256),
PersonTitle             nvarchar(150);
```

RecordID: The record identifier of the user profile.

UserID: The GUID of the user.

SipAddress: Contains the SIP address of the colleague.

NTName: A SAM user name for the entity specified by the user profile.

Email: The e-mail address of the user.

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

PersonTitle: The user specified title.

Return Code Values: MUST return 0.

Result Sets: MUST return a result set of 0 or more rows, containing 1 row for each user that is a colleague of any of the colleague properties of the user specified through the *@RecordId* parameter.

3.1.4.53 QuickLinksRetrieveGroupList

The QuickLinksRetrieveGroupList stored procedure is called to retrieve the groups of a specified type for a user profile.

QuickLinksRetrieveGroupList is defined using T-SQL syntax as follows:

```
PROCEDURE QuickLinksRetrieveGroupList (
    @RecordID               bigint,
    @PolicyID               uniqueidentifier = NULL
);
```

@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 2.2.2.6) Type **value** that specifies the link type.

Return Code Values: QuickLinksRetrieveGroupList MUST return 0.

Result Sets: QuickLinksRetrieveGroupList MUST return one result set.

3.1.4.53.1 QuickLinksRetrieveGroupList Result Set

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.

Return Code Values: MUST return 0.

Result Sets: MUST return 1 result set of 0 or more rows.

3.1.5 Timer Events

No timer events impact the operation of this protocol.

3.1.6 Other Local Events

No other local events impact the operation of this protocol.

3.2 User Profile Service Protocol Specification Client Details

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Message Processing Events and Sequencing Rules

None.

3.2.5 Timer Events

None.

3.2.6 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 property will have a choice list with "High", "Medium" and "Low" as choice list values. The T-SQL syntax is:

```
exec dbo.profile_UpdateProperty
@RemovePropertyList=NULL,
@UpdatePropertyList=N
'<?xml version="1.0" encoding="utf-16"?>
<MSPROFILE>
<PROFILE ProfileName="UserProfile">
<PROPERTY PropertyName="TestProperty" DataTypeID="6" Length="25" DefaultPrivacy="16"
UserOverridePrivacy="0" Replicable="0" PrivacyPolicy="2" IsSection="0" IsMultiValue="0"
ChoiceType="2"
IsEditable="1" IsAdminEditOnly="1" IsColleagueEventLog="0" IsUpgrade="0" IsUpgradePrivate="0"
IsSearchable="1" IsAlias="0" IsVisible="1" IsVisibleOnViewer="0" IsExpand="1" Separator="0"
MaximumShown="10" bUpdate="0">
<VOCABULARY>
<TERM Type="Add" Value="High" />
<TERM Type="Add" Value="Medium" />
<TERM Type="Add" Value="Low" />
</VOCABULARY>
</PROPERTY>
</PROFILE>
</MSPROFILE>'
```

4.1.2 Reading a Property

To read all information about a property the protocol client uses the **profile_GetProfilePropertyInfo** stored procedure. The T-SQL syntax is:

```
exec dbo.profile_GetProfilePropertyInfo @PropertyName=N'TestProperty'
```

4.1.3 Modifying Choice List Values

This property uses a choice list as possible values. The **profile_GetPropertyChoiceList** stored procedure is used to read all choice list values available. The T-SQL syntax is:

```
exec dbo.profile_GetPropertyChoiceList @PropertyName=N'TestProperty'
```

In the next example, the **profile_UpdateProperty** stored procedure is used to remove the "High" choice list value, rename "Medium" to "Moderate" and add "Extreme" as a new choice list value. The T-SQL syntax is:

```
exec dbo.profile_UpdateProperty
@RemovePropertyList=NULL,
@UpdatePropertyList=N
'<?xml version="1.0" encoding="utf-16"?>
<MSPROFILE>
  <PROFILE ProfileName="UserProfile">
    <PROPERTY PropertyName="John" DefaultPrivacy="16" UserOverridePrivacy="0"
              Replicable="0" PrivacyPolicy="2" IsSection="0" IsMultiValue="0"
              ChoiceType="2" IsEditable="1" IsAdminEditOnly="1"
              IsColleagueEventLog="0" IsUpgrade="0" IsUpgradePrivate="0"
              IsSearchable="1" IsAlias="0" IsVisible="1"
              IsVisibleOnViewer="0" IsExpand="0" Separator="0" MaximumShown="10"
              bUpdate="1">
        <VOCABULARY>
             <TERM Type="Add" Value="Extreme" />
             <TERM Type="Delete" Value="High" />
             <TERM Type="Rename" Value="Medium" NewValue="Moderate" />
        </VOCABULARY>
    </PROPERTY>
  </PROFILE>
</MSPROFILE>'
```

4.1.4 Localizing The 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 above 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 Managing Links between Users

4.3.1 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.

4.3.2 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.

@ViewerItemSecurity = 31(integer value – representing the rights of the user making the request.

@RequestedItemSecurity=16 (integer value – representing the maximum level of privacy for the returned data)

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 previously described stored procedure call.

@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.

4.3.3 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.

4.4 Managing Membership

4.4.1 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)<2>. 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.

4.4.2 Retrieving Member Group Data

In this example the store contains this membership group<3>:

Id	DisplayName	MailNickName	Source	SourceReference	Url
321	Group G	groupG	A88B9DCB-5B82-41E4-8A19-17672F307B95	CN=groupG,OU=Distribution Lists,DC=contoso,DC=com	mailto:groupG@contoso.com

To retrieve membership group information the protocol client calls membership_getGroup. This can be done in 3 different ways.

The first method is for the protocol client to specify the membership group with its Member Group Identifier. The protocol client does this by invoking membership_getGroup(NULL, NULL, NULL, 321).

The second method is for the protocol client to specify the membership group with its source reference and source. The protocol client does this by invoking membership_getGroup('A88B9DCB-5B82-41E4-8A19-17672F307B95', NULL, 'CN=GroupG,OU=Distribution Lists,DC=contoso,DC=com', NULL).

The third method is for the protocol client to specify the membership group with its user display name and source. The protocol client does this by invoking membership_getGroup('A88B9DCB-5B82-41E4-8A19-17672F307B95', 'Group G', NULL, NULL).

All three methods result in the protocol client obtaining following result set<4>:

Id	DisplayName	MailNickName	Source	SourceReference	Url
321	Group G	groupG	A88B9DCB-5B82-41E4-8A19-17672F307B95	CN=groupG,OU=Distribution Lists,DC=contoso,DC=com	mailto:groupG@contoso.com

Because the Member Group Identifier is guaranteed to be unique and uniqueness is enforced on the source reference field within a particular source it is recommended that the protocol client use the first and second methods when invoking this procedure.

4.4.3 Creating a Member Group

A protocol client creates the following membership group<5>:

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.

4.4.4 Updating a Member Group

In this example the store contains this membership group<6>:

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.

It is crucial when assigning an e-mail address to a group to remember to also update the group's type from 1 to 0. If this is not done then 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.

4.4.5 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<7>:

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<8>:

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<9>:

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

4.5 Managing User Profile Data

4.5.1 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:

@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.

4.5.2 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_GetUserProfileData** stored procedure passing at least 1 of the following parameters:

@UserId, *@NTName*, *@SID*, and *@RecordId*.

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**.

4.5.3 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>
```

4.6 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.

4.6.1 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	Title	FirstCommon
1	B8C750FC-E3E3-11DC-AFA1-EFA756D89599	domain\steve.masters	Steve.masters@domain.com	NULL	Steve Masters	NULL	True
2	B8C750FC-E3E3-11DC-AFA1-EFA756D89593	domain\syed.abbas	Syed.abbas@domain.com	NULL	Syed Abbas	NULL	False

4.6.2 Retrieving Reporting Data

A protocol client can retrieve information about reports from a user, and the colleagues and manager of the user. To find reports 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 Reports

RecordId	UserID	NTName	PreferredName	Email	SipAddress	PersonTitle
1	B8C750FC-E3E3-11DC-AFA1-EFA756D89597	domain\tai.yee	Tai Yee	Tai.yee@domain.com	NULL	NULL
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

4.7 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**.

4.8 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 finished. 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.

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® Office SharePoint® Server 2007
- Microsoft® SQL Server® 2005
- Microsoft® SQL Server® 2008
- Microsoft® SQL Server® 2008 R2
- Windows® SharePoint® Services 3.0

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 3.1.4.43.1.2: In the case that the value of Text includes more than 256 characters, the value will be truncated to 256 characters.

<2> Section 4.4.1: The protocol client can also refresh its bounds by invoking

```
membership_enumerateGroups(21, 25, @LowerBound, @UpperBound).
```

<3> Section 4.4.2: The following columns have been omitted for simplicity of presentation: MailNickName, Description, cs_SourceReference, MemberCount, LastUpdate, WebID, Type, UserCreated.

<4> Section 4.4.2: The following columns have been omitted for simplicity of presentation: MailNickName, Description, cs_SourceReference, MemberCount, LastUpdate, WebID, Type, UserCreated.

<5> Section 4.4.3: The following columns have been omitted for simplicity of presentation: Id, Description, cs_SourceReference, MemberCount, LastUpdate, WebID, and UserCreated.

<6> Section 4.4.4: The following columns have been omitted for simplicity of presentation: Description, cs_SourceReference, MemberCount, LastUpdate, WebID, UserCreated.

<7> Section 4.4.5: 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.

<8> Section 4.4.5: 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.

<9> Section 4.4.5: 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

Q

R

S

T

U

V

X