[MS-UPIESP2]:
User Profile Import and Export Stored Procedures Version 2 Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Preliminary

Copyright © 2012 Microsoft Corporation.

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release (beta) version of this technology. This Open Specification is final documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release (beta) versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Revision History	Revision Class	Comments
01/20/2012	0.1	New	Released new document.
04/11/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.

Preliminary

Copyright © 2012 Microsoft Corporation.

Table of Contents

Preliminary

Copyright © 2012 Microsoft Corporation.

Preliminary

Copyright © 2012 Microsoft Corporation.

Preliminary

Copyright © 2012 Microsoft Corporation.

1 Introduction

This document specifies the User Profile Import and Export Stored Procedures Protocol. This protocol is used to import and export information about users and member groups.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

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

Active Directory
credential
directory service (DS)
distinguished name (DN)
domain controller (DC)
domain user
GUID
LDAP

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

account
back-end database server
Business Data Connectivity (BDC)
contact
cookie
datetime
display name
distribution list
group
import connection
member group
membership
organizational unit
partition
property mapping
request identifier
result set
return code
stored procedure
table-valued parameter
tenant
user profile
user profile store

The following terms are specific to this document:

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

Preliminary

Copyright © 2012 Microsoft Corporation.

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

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

[Iseminger] Microsoft Corporation, "SQL Server 2000 Architecture and XML/Internet Support", Volume 1 of Microsoft SQL Server 2000 Reference Library, Microsoft Press, 2001, ISBN 0-7356-1280-3, http://www.microsoft.com/mspress/books/5001.aspx

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

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

[MS-UPIEWS] Microsoft Corporation, "User Profile Import and Export Web Service Protocol Specification".

[MS-UPSPROF3] Microsoft Corporation, "User Profile Stored Procedures Version 3 Protocol Specification".

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

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

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

1.2.2 Informative References

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

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

1.3 Overview

This protocol is used to import and export **user profile** and **member group** data to and from the **user profile store**. A typical scenario for using this protocol is a synchronization application that runs at fixed intervals to keep the user profile store and an **LDAP directory service (DS)** in sync.

The protocol supports methods to retrieve all user profiles or only user profiles that have changed since a specific time. The protocol also allows importing **Business Data Connectivity (BDC)** data for specific user profile properties for existing user profiles.

Preliminary

Copyright © 2012 Microsoft Corporation.

1.4 Relationship to Other Protocols

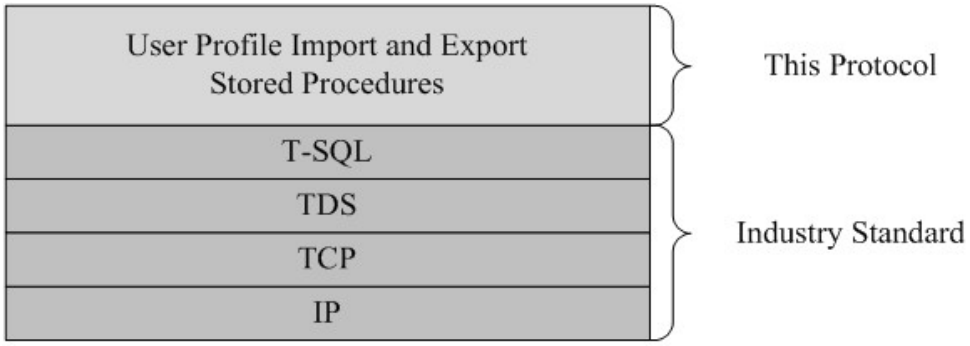

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

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

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

1.6 Applicability Statement

This protocol is designed for flowing user and group data across the user profile store and external directory services (DS). It is applicable when the protocol client is acting as a broker between directory services and the user profile store.

This protocol is designed with the intention of supporting a scale point of approximately:

- 2 million users
- On average 100 member groups per user profile, up to a total of 1 million member groups
- 10 million group **memberships**

This protocol does not specify how the data should be stored in the external directory services, how the protocol client should connect to external directory services, or what synchronization logic should be used by the protocol client when flowing data between the user profile store and the external DS.

1.7 Versioning and Capability Negotiation

Versions of the data structures or stored procedures in the database must 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-WSSFO2] section 1.7.

Preliminary

Copyright © 2012 Microsoft Corporation.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

Preliminary

Copyright © 2012 Microsoft Corporation.

2 Messages

2.1 Transport

[MS-TDS] section 2 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

The following simple types and enumerations are specified in this protocol.

2.2.1.1 Staging Status Type

An integer that specifies the status of a member in the staging data set. The value MUST be one of the values listed in the following data set. If the value is not one of the values listed in the following data set then the member in that row of the staging data set will be ignored and will remain there after post import processing.

Value	Description
-1	Omit this member from import.
0	Member has not yet been processed.
1	Member has been identified as a valid user profile or member group.
2	Member post import processing is finished.

2.2.1.2 Member Type

An integer that specifies the type of member of a member group. The value MUST be one of the values listed in the following table. If the value is not one of the values listed in the following table then the member will not be imported into the user profile store.

Value	Description
0	Member type is unknown.
1	Member is a user.
2	Member is a member group.

2.2.1.3 Short Group Type

A 1-byte unsigned integer that specifies the member group type. These values are a subset of the Group Type value (section 2.2.1.4). The value MUST be one of the values listed in the following table:

Preliminary

Copyright © 2012 Microsoft Corporation.

Value	Description
0	User Specified grouping.
7	**Distribution list** default grouping.
8	Site default grouping.

2.2.1.4 Group Type

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

Value	Description
0	User Specified grouping.
1	Best Bet. User specified group which has an emphasized link in the user interface.
2	General.
5	Users who share the same **Manager** property.
7	Distribution list default grouping.
8	Site default grouping.

Values 3, 4, and 6 are undefined.

2.2.1.5 Is Expanded Type

A bit specifying whether the relation was added as a result of expanding the members of groups within a group. This value MUST be one of the values listed in the following table. If a value is used which is not in the following table then the behavior is undefined.

Value	Description
0	The user is a member of the group.
1	This user is a member of a subgroup of the group.

2.2.1.6 Short Link Type

A **GUID** that specifies the source of a member group. This value MUST be listed in the following table. If a value is used which is not in the following table then the behavior is undefined.

Value	Description
A88B9DCB-5B82-41E4-8A19-17672F307B95	Specifies a member group which is a distribution list.
8BB1220F-DE8B-4771-AC3A-0551242CF2BD	Specifies a site sourced member group.

2.2.2 Bit Fields and Flag Structures

No common bit field or flag structures are defined in this protocol.

Preliminary

Copyright © 2012 Microsoft Corporation.

2.2.3 Binary Structures

No common binary structures are defined in this protocol.

2.2.4 Result Sets

2.2.4.1 ImportExport_GetGroupMembers.ResultSet0

This result set returns the **distinguished names (DNs) (1)** of the users and groups which are members of a group. The ImportExport_GetGroupMembers.ResultSet0 MUST contain 0 rows when the *@GroupId* input parameter does not specify a valid group.

```
DistinguishedName nvarchar(2048),
```

DistinguishedName: A string compatible with the LDAP standard DN (1), see [RFC2251]. This string specifies the DN (1) of a member in a member group.

2.2.4.2 profile_ADImportGetConstantPropertyMapping.ResultSet0

This result set returns the mapping information for properties for the **Tenant** identified by PartitionId that have constant values for an **import connection** identified by DCId.

```
PartitionID uniqueidentifier,
DCId uniqueidentifier,
ConstantValue nvarchar(1000),
PropertyName nvarchar(250),
```

PartitionID: A GUID which identifies the tenant **partition (1)** identifier.

DCId: A GUID which identifies the **Active Directory** import connection.

ConstantValue: The value of the property identified by **PropertyName**.

PropertyName: The name of the user profile property that is being mapped.

2.2.4.3 profile_ADImportGetConstantPropertyMappingForDC.ResultSet0

This result set returns the mapping information for properties for the Tenant identified by PartitionId that have constant values for an Active Directory import connection identified by DCId.

```
PartitionID uniqueidentifier,
DCId uniqueidentifier,
ConstantValue nvarchar(1000),
PropertyName nvarchar(250),
```

PartitionID: A GUID which identifies the tenant partition (1) identifier.

DCId: A GUID which identifies the Active Directory import connection.

ConstantValue: The value of the property identified by **PropertyName**.

PropertyName: The name of the user profile property that is being mapped.

Preliminary

Copyright © 2012 Microsoft Corporation.

2.2.4.4 profile_ADImportGetConstantPropertyMappingForPartition.ResultSet0

This result set returns the mapping information for properties for the Tenant identified by PartitionId that have constant values for an Active Directory import connection identified by DCId.

```
PartitionID uniqueidentifier,
DCId uniqueidentifier,
ConstantValue nvarchar(1000),
PropertyName nvarchar(250),
```

PartitionID: A GUID which identifies the tenant partition (1) identifier.

DCId: A GUID which identifies the Active Directory import connection.

ConstantValue: The value of the property identified by **PropertyName**.

PropertyName: The name of the user profile property that is being mapped.

2.2.4.5 profile_ADImportGetDCMapping.ResultSet0

This result set returns the details of the Active Directory import connection identified by DCId.

```
DCId uniqueidentifier,
ConnectionName nvarchar(1000),
RootDn nvarchar(1000),
DCName nvarchar(1000),
DCUserName nvarchar(1000),
DCPassword varbinary(256),
SyncCookie varbinary(max),
```

DCId: A GUID which identifies the Active Directory import connection.

ConnectionName: The name of this Active Directory Import Connection. It MUST be unique.

RootDn: The Root Active Directory partition (1) that this Active Directory import connection refers to.

DCName: The name of the **domain controller (DC)** which hosts the Active Directory partition identified by **RootDn**.

DCUserName: The **domain user** that has the replicating directory changes rights on the Active Directory Partition identified by **RootDn**.

DCPassword: An opaque sequence of bytes that contains the password associated with the **DCUserName**.

SyncCookie: An opaque sequence of bytes that contains a cookie associated with this Active Directory import connection.

2.2.4.6 profile_ADImportGetFailedItems.ResultSet0

This result set returns the identifiers of items that failed to import from an Active Directory import connection.

```
ItemId uniqueidentifier,
```

Preliminary

Copyright © 2012 Microsoft Corporation.

```
Expiration datetime,
RetryCount tinyint,
```

ItemId: The objectGUID property of the item in Active Directory that could not be imported.

Expiration: The **datetime** after which this item will not be retried for import.

RetryCount: The number of times this item was retried for import.

2.2.4.7 profile_ADImportGetImportOUMapping_DCId.ResultSet0

This result set returns the **Organizational Units** being imported by the Active Directory import connection.

```
DCId uniqueidentifier,
ImportOU nvarchar(1000),
```

DCId: A GUID which identifies the Active Directory Import Connection.

ImportOU: The distinguished name (DN) of the Organizational Units being imported by this Active Directory import connection.

2.2.4.8 profile_ADImportGetProfileExportItems.ResultSet0

This result set returns the identifiers of items that were imported most recently from an Active Directory import connection.

```
ItemId uniqueidentifier,
Success bit,
ErrorMessage nvarchar(max),
```

ItemId: The objectGUID property of the item in Active Directory that was imported.

Success: A bit True or False, indicating whether the import was successful or not respectively.

ErrorMessage: A string containing a description of any error associated with the imported item.

2.2.4.9 profile_ADImportGetPropertyMapping.ResultSet0

This result set returns the mapping information for properties for the Tenant identified by PartitionId whose values are obtained from the Active Directory through the Active Directory import connection identified by DCId.

```
PartitionID uniqueidentifier,
DCId uniqueidentifier,
DCAttribute nvarchar(1000),
PropertyName nvarchar(250),
```

PartitionID: A GUID which identifies the tenant partition (1) identifier.

DCId: A GUID which identifies the Active Directory import connection.

DCAttribute: The name of the property in Active Directory that is being mapped.

Preliminary

Copyright © 2012 Microsoft Corporation.

PropertyName: The name of the user profile property that is being mapped.

2.2.4.10 profile_ADImportGetPropertyMappingForDC.ResultSet0

This result set returns the mapping information for properties for the Tenant identified by PartitionId whose values are obtained from the Active Directory through the Active Directory import connection identified by DCId.

```
PartitionID uniqueidentifier,
DCId uniqueidentifier,
DCAttribute nvarchar(1000),
PropertyName nvarchar(250),
```

PartitionID: A GUID which identifies the tenant partition (1) identifier.

DCId: A GUID which identifies the Active Directory import connection.

DCAttribute: The name of the property in Active Directory that is being mapped.

PropertyName: The name of the user profile property that is being mapped.

2.2.4.11 profile_ADImportGetPropertyMappingForPartition.ResultSet0

This result set returns the mapping information for properties for the Tenant identified by PartitionId whose values are obtained from the Active Directory through the Active Directory import connection identified by DCId.

```
PartitionID uniqueidentifier,
DCId uniqueidentifier,
DCAttribute nvarchar(1000),
PropertyName nvarchar(250),
```

PartitionID: PartitionID: A GUID which identifies the tenant partition (1) identifier.

DCId: A GUID which identifies the Active Directory import connection.

DCAttribute: The name of the property in Active Directory that is being mapped.

PropertyName: The name of the user profile property that is being mapped.

2.2.4.12 profile_EnumDNs.ResultSet0

This result set returns record identifiers as well as their external identifiers for objects synchronized with the profile synchronization service.

```
DN nvarchar(2048),
RecordId bigint,
```

DN: A string used by the external system to identify the object.

RecordId: The internal object identifier. For users this is the same as the RecordID value from the UserProfile_Full data set. For groups this is the Id value from the MemberGroup data set. For contacts this is the ContactID value from DNContactLookup data set.

Preliminary

Copyright © 2012 Microsoft Corporation.

2.2.4.13 profile_EnumerateUsersForBDCImport.ResultSet0

This result set returns a page full of imported users present in the profile store with their DN (1). The result set MUST contain at-maximum @pageSize number of rows.

```
DN nvarchar(2048),
RecordId bigint,
MossJoinValue sql_variant,
DistinguishedName sql_variant,
```

DN: A string used by the external system to identify the object.

RecordId: RecordId of the user profile as present in the user profile store.

MossJoinValue: The value the user profile property passed in as @mossJoinAttribute.

DistinguishedName: A string compatible with the LDAP standard DN (1).

2.2.4.14 profile_GetDNFromAccountName.ResultSet0

This result set returns the DN (1) used by the external system to identify a user.

```
DistinguishedName nvarchar(2048),
```

DistinguishedName: The DN (1) used by the external system.

2.2.4.15 ImportExport_GetNonimportedObjects.ResultSet0

This result set returns the object type and a user-readable identifier of the objects that have not been imported. An object is recognized as imported when it has a corresponding row in the DNLookup data set (section 3.2.1). An imported object has an external identifier stored in the DNLookup data set which is the identifier used by the data source from which it is imported. User profiles added to the profile store by means other than import do not have a corresponding row in the DNLookup data set.

```
ObjectType nvarchar(7),
Name nvarchar(400),
```

ObjectType: Defines the type of the non-imported object. It can contain one of these 3 strings – 'user', 'group' or 'contact'.

Name: For users it contains the user NT Name. For groups and contacts if contains the display name except in cases when it is not set; then it shows the object DN (1).

2.2.5 Tables and Views

No common table or view structures are defined for this protocol.

2.2.6 XML Structures

This section describes the XML schema used in this protocol.

Preliminary

Copyright © 2012 Microsoft Corporation.

2.2.6.1 Members XML

The Members XML structure MUST be used for the **@members** parameter of the **ImportExport_ImportMembers** stored procedure. (section 3.2.5.1).

The Members XML is an XML fragment which MUST contain one and only one <Ms> element and SHOULD contain one or more <M> elements.

The <Ms> element is the root element of the XML fragment. If the XML fragment contains additional <Ms> elements at the root level, then an error will occur when attempting to read the XML fragment. Additional <Ms> elements under the root <Ms> are ignored. If no <M> element is specified then the XML fragment is ignored because it contains no member data to process.

2.2.6.2 Namespaces

This specification does not define any common XML schema namespaces.

2.2.6.3 Simple Types

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

2.2.6.4 Complex Types

This specification does not define any common XML schema complex type definitions.

2.2.6.5 Elements

The following are XML elements specified in this protocol:

Element	Description
Ms	Root element of a list of members of a group.
M	Element which specifies a member of a group. The parent of this element MUST be the <Ms> element. The <M> element MUST have a DN and an OU attribute.

2.2.6.6 Attributes

The following are XML attributes specified in this protocol:

Attribute	Description
DN	The DN (1) of the member.
OU	The organizational unit of the member.

2.2.6.7 Groups

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

2.2.6.8 Attribute Groups

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

Preliminary

Copyright © 2012 Microsoft Corporation.

3 Protocol Details

3.1 Common Details

None.

3.2 Server Details

The back-end database protocol responds to stored procedure calls. It returns result sets and return codes and never initiates communication with other endpoints.

3.2.1 Abstract Data Model

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

The protocol server maintains the following data:

- A list of member groups and the user profiles that belong to them.
- A reconciliation of member groups that contain other member groups such that users belonging to a child member group are identified as belonging in the parent member group.
- A list of partitions (1).
- A list of user profile properties and the values that belong to them.
- The state information about whether a profile import or export session is in progress.

The following diagram illustrates the relationships between the data sets referenced in this protocol:

Preliminary

Copyright © 2012 Microsoft Corporation.

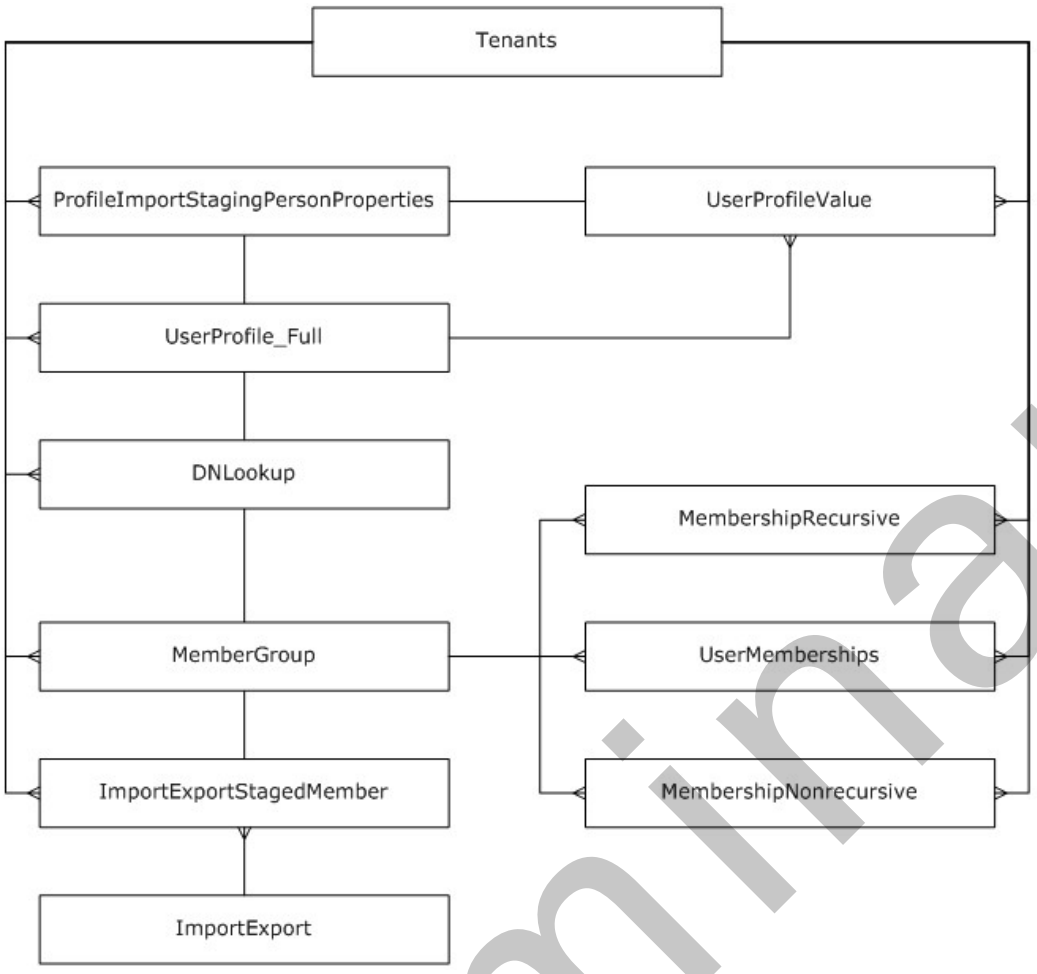

Figure 2: Abstract Data Model

The UserProfile_Full, MemberGroup and DNLookup data sets are expected to contain existing users and member groups. This protocol's import process loads member group members into the ImportExportStagedMember data set and user profile properties into the ProfileImportStagingPersonProperties data set through the ImportExport_ImportMembers (section 3.2.5.1) and profile_UpdateStagingPersonProperty (section 3.2.5.9) stored procedures respectively.

Once the import of data is finished, ImportExport_PostImportMembers (section 3.2.5.4) and ImportExport_PostImportUserProperties (section 3.2.5.5) stored procedures are called to update the member group members and user profile properties in the operational data sets. ImportExport_PostImportMembers (section 3.2.5.4) updates the membership data in the MembershipRecursive, MembershipNonRecursive and UserMemberships data sets. The ImportExport_PostImportUserProperties (section 3.2.5.5) stored procedure updates the user profile attributes in the UserProfileValue and UserProfile_Full data sets.

The ImportExport_GetGroupMembers (section 3.2.5.8) stored procedure retrieves the groups from the operational data sets MembershipNonrecursive and UserMemberships.

Preliminary

Copyright © 2012 Microsoft Corporation.

The Tenants data set contains the partition (1) identifier which partitions the data in all data sets except the ImportExport data set. The ImportExport_GetPartitionId (section 3.2.5.7) retrieves the tenant partition (1) identifier from the Tenants data set.

3.2.2 Timers

None.

3.2.3 Initialization

When performing an import of user profiles and member groups, the ImportExport_ImportStart (section 3.2.5.3) stored procedure MUST be the first stored procedure called.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

The following stored procedures MUST be called in the order shown when importing user and group data. The call sequence for ImportExport_PostImportUserProperties (section 3.2.5.5) and ImportExport_PostImportMembers (section 3.2.5.4) MUST be called after ImportExport_ImportEnd (section 3.2.5.2). If these stored procedures are called before the import batch has finished, then errors may occur due the import data lack of referential integrity.

Stored procedure	Description
ImportExport_ImportStart	See section 3.2.5.3.
ImportExport_ImportMembers	See section 3.2.5.1.
ImportExport_ImportEnd	See section 3.2.5.2.
ImportExport_PostImportUserProperties	See section 3.2.5.5.
ImportExport_PostImportMembers	See section 3.2.5.4.

Only one import batch MUST be in process at a time. The ImportExport_PostImportUserProperties and ImportExport_PostImportMembers stored procedures MUST be after the end of the data import, which is signified by the call to the ImportExport_ImportEnd stored procedure, and MUST be called before additional another import is started. If ImportExport_PostImportUserProperties and ImportExport_PostImportMembers are called while data is being imported the stored procedures MAY fail because of the lack of referential integrity in the import data.

3.2.5.1 ImportExport_ImportMembers

This stored procedure stores the members of a group in staging data sets during the import process. After the import process has finished, the stored procedure **ImportExport_PostImportMembers** (section 3.2.5.4) will be called to complete the transference of imported members to the operational data sets of the user profile store. If this stored procedure fails then none of the group members will be stored in the staging data set.

```
PROCEDURE ImportExport_ImportMembers (
@importExportId bigint
,@members xml
```


Copyright © 2012 Microsoft Corporation.

Preliminary

```
,@parentGroupId bigint
,@partitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@importExportId: The 64-bit integer identifier of the import or export batch being processed. This value MUST be the value of the *@importExportId* output parameter of the last call to the **ImportExport_ImportStart** (section 3.2.5.3) stored procedure.

@members: A Members XML (section 2.2.6.1) string which specifies the members of the member group.

@parentGroupId: The 64-bit identifier that identifies the member group which contains the members which are being added.

@partitionID: A GUID which identifies the tenant partition (1) identifier of the member group which the members are being added.

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

Return Values: An **integer** which MUST be 0.

Result Sets:

This stored procedure MUST NOT return any result sets.

3.2.5.2 ImportExport_ImportEnd

This stored procedure is called to signify the completion of the specified import batch. If this stored procedure fails, then the import batch is not updated to signify completion of the import batch.

```
PROCEDURE ImportExport_ImportEnd (
@importExportId bigint
,@correlationId uniqueidentifier = null
);
```

@importExportId: The 64-bit integer identifier of the import or export batch being processed. This value MUST be the value of the @importExportId output parameter of the last call to the ImportExport_ImportStart (section 3.2.5.3) stored procedure. Supplying a value other than the value corresponding call to ImportExport_ImportStart will result in error.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.3 ImportExport_ImportStart

This stored procedure is called prior to importing a batch of users and groups. The importExportId output by this stored procedure is used in subsequent calls to stored procedures in this protocol. When the import has finished, a call to the stored procedure ImportExport_ImportEnd (section 3.2.5.2) MUST be made using the importExportId output from this stored procedure. Only one import batch MUST be processed at a time, so this stored procedure MUST NOT be called more than once before the call to ImportExport_ImportEnd (section 3.2.5.2). If this stored procedure is called

Preliminary

Copyright © 2012 Microsoft Corporation.

more than once before ImportExport_ImportEnd it will succeed, however the state of the import process will then be undefined. If this stored procedure fails then the importExportId that is output will not contain a valid identifier.

```
PROCEDURE ImportExport_ImportStart (
@importExportId bigint OUTPUT
,@correlationId uniqueidentifier = null
);
```

@importExportId: A 64-bit integer identifier used to represent the import being started.

Value	Description
@@identity	The importExportId output is the value of @@identity after calling this stored procedure.

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

Return Values: An integer which MUST be zero.

Result Sets: MUST NOT return any result sets.

3.2.5.4 ImportExport_PostImportMembers

This stored procedure is called to process the member information in the staging data set and MUST be called after the completion of the import batch to ensure the imported data has referential integrity. This stored procedure MUST only be run after the call to ImportExport_ImportEnd (section 3.2.5.2) has been made and before the next import is started. If this stored procedure fails, then all the post processing work may not have been finished and the stored procedure MUST be called again to complete the post processing of the imported members.

```
PROCEDURE ImportExport_PostImportMembers (
@correlationId uniqueidentifier = null
);
```

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

Return Values: An integer which MUST be 0 on success and nonzero on failure.

Result Sets: MUST NOT return any result sets.

3.2.5.5 ImportExport_PostImportUserProperties

This stored procedure is called to process the user profile properties which were saved to the ProfileImportStagingPersonProperties data set (section 2.2.5.4) during the user profile import. This stored procedure MUST be called after the completion of the import batch to ensure the imported data has referential integrity. This stored procedure MUST only be run after the call to ImportExport_ImportEnd (section 3.2.5.2) has been made and before the next import is started.

```
PROCEDURE ImportExport_PostImportUserProperties (
@correlationId uniqueidentifier = null
);
```

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

Preliminary

Copyright © 2012 Microsoft Corporation.

Return Values: MUST NOT return any values.

Result Sets: MUST NOT return any result sets.

3.2.5.6 ImportExport_IsRunning

This stored procedure is invoked to determine if an import or export is currently running.

```
PROCEDURE ImportExport_IsRunning (
@correlationId uniqueidentifier = null
);
```

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

Return Values: An integer which MUST be zero when no import or export is currently running or MUST be one when an import or export is running.

Result Sets: MUST NOT return any result sets.

3.2.5.7 ImportExport_GetPartitionId

This stored procedure retrieves the partition (1) identifier of the tenant with the corresponding organizational unit when multiple tenants exist or the default partition (1) identifier for a single tenant. If this stored procedure fails then the partition (1) identifier output is undefined and MUST not be used.

```
PROCEDURE ImportExport_GetPartitionId (
@organizationalUnit nvarchar(64)
,@correlationId uniqueidentifier = null
,@partitionId uniqueidentifier OUTPUT
);
```

@organizationalUnit: The name of the organizational unit that corresponds to the partition (1) identifier. When multiple tenants exist, this value MUST correspond to a value in the SynchronizationOU column in the Tenants data set (section 3.2.1).

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

@partitionId: The partition (1) identifier of the tenant with the corresponding organizational unit. This value MUST NOT be null or empty.

Return Values: An integer which MUST be zero.

Result Sets: MUST NOT return any result sets.

3.2.5.8 ImportExport_GetGroupMembers

This stored procedure return a result set with the DN (1) of all the members of the specified member group from the operational user profile store. The DN (1) of both user profiles and member group members are returned in the result set.

```
PROCEDURE ImportExport_GetGroupMembers (
@partitionID uniqueidentifier
,@Id bigint
```

Preliminary

Copyright © 2012 Microsoft Corporation.

```
,@correlationId uniqueidentifier = null
);
```

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

@Id: The identifier of the member group whose members are to be retrieved.

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

Return Values: An integer that MUST be zero.

Result Sets:

This stored procedure MUST return a ImportExport_GetGroupMembers.ResultSet0

3.2.5.9 profile_UpdateStagingPersonProperty

This stored procedure stores the user profile properties in the **ProfileImportStagingPersonProperties** data set (section 2.2.5.3) during the import process. After the import process has finished, the stored procedure **ImportExport_PostImportUserProperties** (section 3.2.5.5) MUST be called to transfer the members to the operational data sets of the user profile store.

```
PROCEDURE profile_UpdateStagingPersonProperty (
@partitionID uniqueidentifier
,@RecordId bigint
,@ProfileType nvarchar(20)
,@PropertyURI nvarchar(250)
,@PropertyDN nvarchar(2048)
,@correlationId uniqueidentifier = null
);
```

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

@RecordId: An **integer** that specifies the user profile where the properties are to be updated.

@ProfileType: A string that specifies the type of profile to be updated. The values for this parameter MUST be either "UserProfile" or "OrganizationalProfile". Use of other values results in errors.

@PropertyURI: A string that contains the name that identifies the user profile property.

@PropertyDN: A string used by the external system to identify the object. Later **ImportExport_PostImportUserProperties** will resolve this identifier to a user account name.

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

Return Values: An integer that MUST be zero.

Result Sets: MUST NOT return any result sets.

3.2.5.10 ImportExport_CleanGroupMembers

This stored procedure removes all member groups that are members of the specified member group from the **MembershipRecursive** and **MembershipNonRecursive** data sets (section 3.2.1).

Copyright © 2012 Microsoft Corporation.

Preliminary

```
PROCEDURE ImportExport_CleanGroupMembers (
@memberGroupId bigint
,@partitionId uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@memberGroupId: The identifier of the member group that contains the members that are to be deleted.

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

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

Return Values: An integer that MUST be zero.

Result Sets: MUST NOT return any result sets.

3.2.5.11 ImportExport_PurgeNonimportedObjects

This stored procedure will purge the user profiles and member groups that are not imported using this protocol (section 1.3) nor using MS-UPIEWS protocol ([MS-UPIEWS] section 1.3).

For user profiles, the stored procedure marks all the user profiles in the UserProfile_Full data set (section 3.2.1) that do not have a corresponding row in the DNLookup data set as deleted.

For member groups, the stored procedure deletes all the member groups in the MemberGroup data set (section 3.2.1) that do not have a corresponding row in the DNLookup data set.

```
PROCEDURE ImportExport_PurgeNonimportedObjects (
@isUsersOnly bit = null
,@correlationId uniqueidentifier = null
);
```

@isUsersOnly: Specifies if the operation is performed for user profiles only, or for both user profiles and member groups. If this value is NULL or zero, the operation MUST be performed for both user profiles and member groups. For all other values the operation MUST be performed on user profiles only and member groups MUST remain unchanged.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.12 ImportExport_GetNonimportedObjects

This stored procedure returns result sets containing the user profiles, member groups and **contacts (3)** that are not imported using this protocol (see section 1.3) nor using the User Profile Import and Export Web Service Protocol Specification (see [MS-UPIEWS] section 1.3).

The stored procedure returns a ImportExport_GetNonimportedObjects.ResultSet0 (section 2.2.4.15) containing all objects in the UserProfile_Full, MemberGroup and DNContactLookup data sets that do not have a corresponding row in the DNLookup data set ([MS-UPSPROF3] section 2.2.5.1).

```
PROCEDURE ImportExport_GetNonimportedObjects (
```

Preliminary

Copyright © 2012 Microsoft Corporation.

```
@isUsersOnly bit = null
,@correlationId uniqueidentifier = null
);
```

@isUsersOnly: Specifies if the operation returns the records for user profiles only, or for all object types. If this value is NULL or zero, the operation MUST return records for all object types. For all other values the operation MUST return the user profiles only.

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

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a ImportExport_GetNonimportedObjects.ResultSet0

3.2.5.13 profile_GetBusinessDataCatalogConnections

This stored procedure returns the list of Business Data Connectivity (BDC) Profile Synchronization connections.

```
PROCEDURE profile_GetBusinessDataCatalogConnections (
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = NULL
);
```

@partitionID: A GUID which identifies the tenant partition (1) identifier of the member group which the members are being added.

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

Return Values: MUST NOT return any values.

Result Sets: This stored procedure MUST return one result set.

3.2.5.13.1 profile_GetBusinessDataCatalogConnections Result Set

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

```
DisplayName                string,
SystemName                 string,
EntityName                 string,
EntityNamespace            string,
FilterName                 string,
ProfilePropertyName        string,
MappedAttribute             string;
```

DisplayName: Name of the Profile Synchronization Connection.

SystemName: Name of the BDC System for Synchronize data from.

EntityName: Name of the BDC Entity for Synchronize data from.

Preliminary

Copyright © 2012 Microsoft Corporation.

EntityNamespace: NameSpace of the BDC Entity for Synchronize data from.

FilterName: Name of the BDC filter to use on Profile synchronization.

ProfilePropertyName: Name of the user profile property used to identify the User profile to add BDC data to.

MappedAttribute: Name of the BDC entity attribute whose value should match the **ProfileProperty** value for a successful join to happen.

3.2.5.14 profile_DeleteBusinessDataCatalogConnection

This stored procedure deletes the Profile Synchronization connection for the BusinessDataCatalog with the given displayName.

```
PROCEDURE profile_DeleteBusinessDataCatalogConnection (
@partitionID uniqueidentifier
,@displayName nvarchar(128)
,@correlationId uniqueidentifier = NULL
);
```

@partitionID: A GUID which identifies the tenant partition (1) identifier of the member group which the members are being added.

@displayName: Name of Profile Synchronization connection.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.15 profile_UpdateBusinessDataCatalogConnection

This stored procedure updates and creates the Profile Synchronization connection for the BusinessDataCatalog.

```
PROCEDURE DBO.profile_UpdateBusinessDataCatalogConnection (
@partitionID uniqueidentifier
,@correlationId uniqueidentifier = NULL
,@displayName nvarchar(128)
,@systemName nvarchar(250)
,@entityName nvarchar(250)
,@entityNamespace nvarchar(250)
,@filterName nvarchar(250)
,@profilePropertyName nvarchar(50,
,@mappedAttribute nvarchar(250)
);
```

@partitionID: A GUID which identifies the tenant partition (1) identifier of the member group which the members are being added.

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

@displayName: Name of the Profile Synchronization Connection.

Preliminary

Copyright © 2012 Microsoft Corporation.

@systemName: Name of the Business Data Connectivity (BDC) System for Synchronize data from.

@entityName: Name of the BDC Entity for Synchronize data from.

@entityNamespace: NameSpace of the BDC Entity for Synchronize data from.

@filterName: Name of the BDC filter to use on Profile synchronization.

@profilePropertyName: Name of the user profile property used to identify the User profile to add BDC data to.

@mappedAttribute: Name of the BDC entity attribute whose value should match the **ProfileProperty** value for a successful join to happen.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.16 ImportExport_DeleteStagedLinks

This stored procedure deletes all records in the staging data sets that refer to a particular object.

```
PROCEDURE ImportExport_DeleteStagedLinks (
@DistinguishedName nvarchar(2048)
,@correlationId uniqueidentifier = null
);
```

@DistinguishedName: A string used by the external system to identify the object.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.17 ImportExport_GetUserByPropertyValue

This stored procedure searches for a user profile by the value of one of its properties and returns its import/export identifier.

```
PROCEDURE ImportExport_GetUserByPropertyValue (
@partitionID uniqueidentifier
,@mossJoinAttribute nvarchar(250)
,@propertyVal sql_variant
,@distinguishedName nvarchar(2048) OUTPUT
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID which identifies the tenant partition (1) identifier where the search must take place.

@mossJoinAttribute: The name of the property being searched for.

@propertyVal: The value the property being searched for.

Preliminary

Copyright © 2012 Microsoft Corporation.

@distinguishedName: A string used by the external system to identify the object. If no object is found this parameter will return NULL.

Value	Description
null	The object was not found

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.18 profile_AddDNLookupEntry

This stored procedure adds a new record in the DNLookup data set. It is called when a new object is brought to the profile store by the profile import/export service.

```
PROCEDURE profile_AddDNLookupEntry (
@partitionID uniqueidentifier
,@RecordId bigint
,@ObjectType nvarchar(20)
,@DistinguishedName nvarchar(2048)
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID which identifies the tenant partition (1) identifier of the added object.

@RecordId: This parameter identifies the added object. For users this is the same as the RecordID value from the UserProfile_Full data set. For groups this is the Id value from the MemberGroup data set. For contacts this is the ContactID value from DNContactLookup data set.

@ObjectType: Defines the type of the added object. It can contain one of these 3 strings – 'user', 'group' or 'contact'.

@DistinguishedName: A string used by the external system to identify the object.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.19 profile_ADImportAddDCConstantPropertyMapping

This stored procedure adds a new property mapping from a property in the Active Directory to a user profile property for a tenant.

```
PROCEDURE profile_ADImportAddDCConstantPropertyMapping (
@PartitionId uniqueidentifier
,@DCId uniqueidentifier
,@ConstantValue nvarchar(1000)
,@ProfileProperty nvarchar(1000)
,@correlationId uniqueidentifier = null
);
```

Preliminary

Copyright © 2012 Microsoft Corporation.

@PartitionId: A GUID that identifies the tenant partition (1) receiving the property mapping.

@DCId: A GUID which identifies the Active Directory Import Connection for which this property mapping is being defined.

@ConstantValue: The constant value of the property identified by **@ProfileProperty**.

@ProfileProperty: The name of the user profile property being mapped.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.20 profile_ADImportAddDCMapping

This stored procedure adds a new Active Directory import connection to a specified root directory partition (1) in a domain controller (DC).

```
PROCEDURE profile_ADImportAddDCMapping (
@DCId uniqueidentifier
,@ConnectionName nvarchar(1000)
,@RootDn nvarchar(1000)
,@DCName nvarchar(1000)
,@DCUserName nvarchar(1000)
,@DCPassword varbinary(256)
,@SyncCookie varbinary(max)
,@correlationId uniqueidentifier = null
);
```

@DCId: A GUID which identifies the Active Directory import connection being created.

@ConnectionName: The name of this connection, which MUST be unique across all connections.

@RootDn: The Distinguished Name of the root directory partition of this connection.

@DCName: The name of the **DC** for this connection.

@DCUserName: The domain user **account** that has replicating directory changes rights on the root directory partition identified by *@RootDn*.

@DCPassword: A sequence of bytes that identifies the password associated with *@DCUserName*.

@SyncCookie: A sequence of bytes that identifies a cookie associated with this connection.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.21 profile_ADImportAddDCPropertyMapping

This stored procedure adds a **property mapping** for a given Active Directory import connection.

```
PROCEDURE profile_ADImportAddDCPropertyMapping (
```

Preliminary

Copyright © 2012 Microsoft Corporation.

```
@PartitionId uniqueidentifier
,@DCId uniqueidentifier
,@DCAttribute nvarchar(1000)
,@ProfileProperty nvarchar(1000)
,@correlationId uniqueidentifier = null
);
```

@PartitionId: A GUID which identifies the tenant partition (1) receiving the property mapping.

@DCId: A GUID which identifies the Active Directory import connection being created.

@DCAttribute: The name of the property in Active Directory that is being mapped.

@ProfileProperty: The name of the user profile property being mapped.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.22 profile_ADImportAddFailedItems

This stored procedure stores the identifiers of items that could not be imported from the Active Directory to the user profile database.

```
PROCEDURE profile_ADImportAddFailedItems (
@DCId uniqueidentifier
,@Items tvpProfileExportItems
,@correlationId uniqueidentifier = null
);
```

@DCId: A GUID which identifies the Active Directory import connection from which items are being imported.

@Items: A **table-valued parameter** that contains one row per failed item, where each row has the following columns:

ItemId: The objectGUID attribute of the Active Directory item that failed to import.

- **Expiration**: The datetime after which no retries will be made to reimport this item.
- **ErrorMessage**: An optional string description of any errors encounted in the import process.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.23 profile_ADImportAddImportOUMapping

This stored procedure adds an Organizational Unit to be imported via an Active Directory import connection.

Preliminary

Copyright © 2012 Microsoft Corporation.

```
PROCEDURE profile_ADImportAddImportOUMapping (
@DCId uniqueidentifier
,@ImportOU nvarchar(1000)
,@correlationId uniqueidentifier = null
);
```

@DCId: A GUID which identifies the Active Directory import connection from which items are being imported.

@ImportOU: The distinguished name (DN) of the Organizational Unit to be imported.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.24 profile_ADImportAddProfileExportItems

This stored procedure stores the most recent list of items imported from Active Directory into the user profile database.

```
PROCEDURE profile_ADImportAddProfileExportItems (
@DCId uniqueidentifier
,@Success bit
,@Items tvpProfileExportItems
,@correlationId uniqueidentifier = null
);
```

@DCId: A GUID which identifies the Active Directory import connection from which items are being imported.

@Success: TRUE or FALSE depending on whether the item was successfully imported into the user profile database or not.

@Items: A table-valued parameter that contains one row per imported item, where each row has the following columns:

ItemId: The objectGUID attribute of the Active Directory Item that was imported.

- **Expiration**: The datetime after which no retries will be made to reimport this item.
- **ErrorMessage**: An optional string description of any errors encounted in the import process.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.25 profile_ADImportGetConstantPropertyMapping

This stored procedure gets the constant property mappings for all Active Directory import connections.

Preliminary

Copyright © 2012 Microsoft Corporation.

```
PROCEDURE profile_ADImportGetConstantPropertyMapping (
@correlationId uniqueidentifier = null
);
```

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

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a profile_ADImportGetConstantPropertyMapping.ResultSet0

3.2.5.26 profile_ADImportGetConstantPropertyMappingForDC

This stored procedure gets the constant property mappings for a specific Active Directory import connection.

```
PROCEDURE profile_ADImportGetConstantPropertyMappingForDC (
@DCId uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@DCId: A GUID which identifies the Active Directory import connection containing the constant property mappings.

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

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a profile_ADImportGetConstantPropertyMappingForDC.ResultSet0

3.2.5.27 profile_ADImportGetConstantPropertyMappingForPartition

This stored procedure gets the constant property mappings for all Active Directory import connections for a specific tenant.

```
PROCEDURE profile_ADImportGetConstantPropertyMappingForPartition (
@PartitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@PartitionID: A GUID which identifies the tenant partition (1) identifier for which constant property mappings need to be returned.

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

Return Values: An integer which MUST be 0.

Result Sets:

Preliminary

Copyright © 2012 Microsoft Corporation.

This stored procedure MUST return a profile_ADImportGetConstantPropertyMappingForPartition.ResultSet0

3.2.5.28 profile_ADImportGetDCMapping

This stored procedure gets all the Active Directory import connections defined for this user profile database.

```
PROCEDURE profile_ADImportGetDCMapping (
@correlationId uniqueidentifier = null
);
```

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

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a profile_ADImportGetDCMapping.ResultSet0

3.2.5.29 profile_ADImportGetFailedItems

This stored procedure gets all the items that could not be imported from a specified Active Directory import connection.

```
PROCEDURE profile_ADImportGetFailedItems (
@DCId uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@DCId: A GUID which identifies the Active Directory import connection from which items could not be imported.

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

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a profile_ADImportGetFailedItems.ResultSet0

3.2.5.30 profile_ADImportGetImportOUMapping_DCId

This stored procedure gets all the Organizational Units from which items are being imported for a specified Active Directory import connection.

```
PROCEDURE profile_ADImportGetImportOUMapping_DCId (
@DCId uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@DCId: A GUID which identifies the Active Directory import connection from which items need to be imported.

Preliminary

Copyright © 2012 Microsoft Corporation.

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

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a profile_ADImportGetImportOUMapping_DCId.ResultSet0

3.2.5.31 profile_ADImportGetProfileExportItems

This stored procedure gets the most recent items that were imported from a specified Active Directory import connection.

```
PROCEDURE profile_ADImportGetProfileExportItems (
@DCId uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@DCId: A GUID which identifies the Active Directory import connection from which items were imported.

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

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a profile_ADImportGetProfileExportItems.ResultSet0

3.2.5.32 profile_ADImportGetProfileStatistics

This stored procedure gets some statistics on the number of items being imported from Active Directory, and the number of items and tenants in the user profile database.

```
PROCEDURE profile_ADImportGetProfileStatistics (
@userInDBCount int OUTPUT
,@groupInDBCount int OUTPUT
,@userImportedCount int OUTPUT
,@groupImportedCount int OUTPUT
,@tenantCount int OUTPUT
,@correlationId uniqueidentifier = null
);
```

@userInDBCount: An output parameter that contains the count of the number of users in the user profile database when the stored procedure returns.

Value	Description
UserProfile_Full.select	Count of the number of users in the user profile database.

@groupInDBCount: An output parameter that contains the count of the number of **groups** in the user profile database when the stored procedure returns.

Preliminary

Copyright © 2012 Microsoft Corporation.

Value	Description
MemberGroup.select	Count of the number of groups in the user profile database.

@userImportedCount: An output parameter that contains the count of the number of users that have been imported into the user profile database when the stored procedure returns. This can be different from *@userInDBCount* because some of the users in the user profile database could have been created without using Active Directory import connections.

Value	Description
0	Default return value.
@objects.select	Count of the number of users that have been imported into the user profile database.

@groupImportedCount: An output parameter that contains the count of the number of groups that have been imported into the user profile database when the stored procedure returns. This can be different from *@groupInDBCount* because some of the groups in the user profile database could have been created without using Active Directory import connections.

Value	Description
0	Default return value.
@objects.select	Count of the number of groups that have been imported into the user profile database.

@tenantCount: An output parameter that contains the count of number of tenants in the user profile database when the stored procedure returns.

Value	Description
Tenants.select	Count of the number of Tenants in the user profile database.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.33 profile_ADImportGetPropertyMapping

This stored procedure gets the property mappings for all Active Directory import connections.

```
PROCEDURE profile_ADImportGetPropertyMapping (
@correlationId uniqueidentifier = null
);
```

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

Return Values: An integer which MUST be 0.

Result Sets:

Preliminary

Copyright © 2012 Microsoft Corporation.

This stored procedure MUST return a profile_ADImportGetPropertyMapping.ResultSet0

3.2.5.34 profile_ADImportGetPropertyMappingForDC

This stored procedure gets the property mappings for a specified Active Directory import connection.

```
PROCEDURE profile_ADImportGetPropertyMappingForDC (
@DCId uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@DCId: GUID which identifies the Active Directory import connection from which items were imported.

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

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a profile_ADImportGetPropertyMappingForDC.ResultSet0

3.2.5.35 profile_ADImportGetPropertyMappingForPartition

This stored procedure gets the property mappings for all Active Directory import connections for a specific tenant.

```
PROCEDURE profile_ADImportGetPropertyMappingForPartition (
@PartitionID uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@PartitionID: A GUID which identifies the tenant partition (1) identifier for which property mappings need to be returned.

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

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a profile_ADImportGetPropertyMappingForPartition.ResultSet0

3.2.5.36 profile_ADImportRemoveAllFailedItems

This stored procedure removes all the items for which import was not successful from a specific Active Directory import connection.

```
PROCEDURE profile_ADImportRemoveAllFailedItems (
@DCId uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

Preliminary

Copyright © 2012 Microsoft Corporation.

@DCId: A GUID which identifies the Active Directory import connection from which items were not imported successfully.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.37 profile_ADImportRemoveConstantPropertyMapping

This stored procedure removes the constant property mapping for a specific property for a specific tenant for a specific Active Directory import connection.

```
PROCEDURE profile_ADImportRemoveConstantPropertyMapping (
@PartitionId uniqueidentifier
,@DCId uniqueidentifier
,@ProfileProperty nvarchar(1000)
,@correlationId uniqueidentifier = null
);
```

@PartitionId: A GUID which identifies the tenant partition (1) identifier for which constant property mappings need to be removed.

@DCId: A GUID which identifies the Active Directory import connection from which constant property mappings need to be removed.

@ProfileProperty: The name of the user profile property for which constant property mappings need to be removed.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.38 profile_ADImportRemoveDCMapping

This stored procedure removes a specific Active Directory import connection.

```
PROCEDURE profile_ADImportRemoveDCMapping (
@PartitionId uniqueidentifier
,@DCId uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@PartitionId: This value is ignored.

@DCId: A GUID which identifies the Active Directory import connection which needs to be removed.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

Preliminary

Copyright © 2012 Microsoft Corporation.

3.2.5.39 profile_ADImportRemoveFailedItems

This stored procedure removes all items that could not be imported successfully for a specific Active Directory import connection.

```
PROCEDURE profile_ADImportRemoveFailedItems (
@DCId uniqueidentifier
,@Items tvpProfileExportItems
,@correlationId uniqueidentifier = null
);
```

@DCId: A GUID which identifies the Active Directory import connection from which failed items need to be removed.

@Items: A table-valued parameter that contains one row per imported item, where each row has the following columns:

ItemId: The objectGUID attribute of the Active Directory item was imported.

- **Expiration**: The datetime after which no retries will be made to reimport this item.
- **ErrorMessage**: An optional string description of any errors encounted in the import process.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.40 profile_ADImportRemoveImportOUMapping

This stored procedure removes a specific Organizational Unit from the list of Organizational Units that need to be imported for a specific Active Directory import connection.

```
PROCEDURE profile_ADImportRemoveImportOUMapping (
@DCId uniqueidentifier
,@ImportOU nvarchar(1000)
,@correlationId uniqueidentifier = null
);
```

@DCId: A GUID which identifies the Active Directory import connection from which Organizational Units need to be removed.

@ImportOU: The distinguished name (DN) of the Organizational Unit to be removed.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.41 profile_ADImportRemoveProfileExportItems

This stored procedure removes the most recently imported items from the Active Directory by a specific Active Directory import connection.

Preliminary

Copyright © 2012 Microsoft Corporation.

```
PROCEDURE profile_ADImportRemoveProfileExportItems (
@DCId uniqueidentifier
,@correlationId uniqueidentifier = null
);
```

@DCId: A GUID which identifies the Active Directory import connection from which Organizational Units need to be removed.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.42 profile_ADImportRemovePropertyMapping

This stored procedure removes the property mapping for a specific property for a specific tenant for a specific Active Directory import connection.

```
PROCEDURE profile_ADImportRemovePropertyMapping (
@PartitionId uniqueidentifier
,@DCId uniqueidentifier
,@ProfileProperty nvarchar(1000)
,@correlationId uniqueidentifier = null
);
```

@PartitionId: A GUID which identifies the tenant partition (1) identifier for which property mappings need to be removed.

@DCId: A GUID which identifies the Active Directory import connection from which property mappings need to be removed.

@ProfileProperty: The name of the user profile property for which property mappings need to be removed.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.43 profile_ADImportUpdateDCMappingCredentials

This stored procedure updates the **credentials** used to access the Active Directory for a specific Active Directory import connection.

```
PROCEDURE profile_ADImportUpdateDCMappingCredentials (
@DCId uniqueidentifier
,@DCUserName nvarchar(1000)
,@DCPassword varbinary(256)
,@correlationId uniqueidentifier = null
);
```

Preliminary

Copyright © 2012 Microsoft Corporation.

@DCId: A GUID which identifies the Active Directory import connection whose credentials need to be updated.

@DCUserName: The domain user account that has replicating directory changes rights on the root directory partition identified by *@RootDn*.

@DCPassword: A sequence of bytes that identifies the password associated with *@DCUserName*.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.44 profile_ADImportUpdateDCMappingSyncCookie

This stored procedure updates the **cookie** associated with a specific Active Directory import connection.

```
PROCEDURE profile_ADImportUpdateDCMappingSyncCookie (
@DCId uniqueidentifier
,@SyncCookie varbinary(max)
,@correlationId uniqueidentifier = null
);
```

@DCId: A GUID which identifies the Active Directory import connection whose cookie needs to be updated.

@SyncCookie: A sequence of bytes that identifies a cookie associated with this connection

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.45 profile_CleanupDNLookupTable

This stored procedure deletes all records from the DNLookup, staging and ProfileImportStagingPersonProperties data sets.

```
PROCEDURE profile_CleanupDNLookupTable (
@correlationId uniqueidentifier = null
);
```

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.46 profile_DeleteBusinessDataCatalogConnection

This stored procedure deletes the Profile Synchronization connection for the BusinessDataCatalog with the given displayName.

Preliminary

Copyright © 2012 Microsoft Corporation.

```
PROCEDURE profile_DeleteBusinessDataCatalogConnection (
@partitionID uniqueidentifier
,@displayName nvarchar(128)
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID which identifies the tenant partition (1) identifier of the member group which the members are being added.

@displayName: Name of Profile Synchronization connection.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.47 profile_DeleteContactEntry

This stored procedure deletes a contact (3) from DNContactLookup data set.

```
PROCEDURE profile_DeleteContactEntry (
@partitionID uniqueidentifier
,@contactID bigint
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID which identifies the tenant partition (1) identifier of the deleted contact.

@contactID: The contact identifier.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.48 profile_DeleteDNLookupEntry

This stored procedure deletes a contact from DNContactLookup data set.

```
PROCEDURE profile_DeleteDNLookupEntry (
@partitionID uniqueidentifier
,@RecordId bigint
,@ObjectType nvarchar(20)
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID which identifies the tenant partition (1) identifier of the deleted object.

@RecordId: This parameter identifies the deleted object. For users this is the same as the RecordID value from the UserProfile_Full data set. For groups this is the Id value from the MemberGroup data set. For contacts this is the ContactID value from DNContactLookup data set.

Preliminary

Copyright © 2012 Microsoft Corporation.

@ObjectType: Defines the type of the deleted object. It can contain one of these 3 strings – 'user', 'group' or 'contact'.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.49 profile_EnumDNs

This stored procedure selects a batch of object identifiers together with their external identifiers. It is used to implement the export part of the profile import/export service.

```
PROCEDURE profile_EnumDNs (
@partitionID uniqueidentifier
,@objectType nvarchar(20)
,@beginID bigint
,@pageSize int
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID which identifies the tenant partition (1) identifier of the queried objects.

@objectType: The type of the queried objects. It can contain one of these 3 strings – 'user', 'group' or 'contact'.

@beginID: The first identifier to return.

@pageSize: The desired number of returned objects.

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

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a profile_EnumDNs.ResultSet0

3.2.5.50 profile_EnumerateUsersForBDCImport

This stored procedure returns all the imported user profiles with the property value for the property that is set as the **Join** attribute for a Business Data Connectivity (BDC) import.

```
PROCEDURE profile_EnumerateUsersForBDCImport (
@partitionID uniqueidentifier
,@mossJoinAttribute nvarchar(250)
,@beginID bigint
,@pageSize int
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID which identifies the partition (1) identifier of the tenant for which the import is done.

Preliminary

Copyright © 2012 Microsoft Corporation.

@mossJoinAttribute: Name of the user profile property whose value is used to identify the row from the BDC entity data which is used to add data to mapped profile properties.

@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.

@pageSize: The value which determines the number of user profiles to retrieve.

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

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a profile_EnumerateUsersForBDCImport.ResultSet0

3.2.5.51 profile_GetDNFromAccountName

Get the string used by the external system to identify the user from the user account name.

```
PROCEDURE profile_GetDNFromAccountName (
@partitionID uniqueidentifier
,@accountName nvarchar(400)
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID which identifies the tenant partition (1) identifier of the queried object.

@accountName: The user account name.

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

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a profile_GetDNFromAccountName.ResultSet0

3.2.5.52 profile_GetDNFromRecordId

This stored procedure gets the string used by the external system to identify the object from the object record identifier.

```
PROCEDURE profile_GetDNFromRecordId (
@partitionID uniqueidentifier
,@RecordId bigint
,@ObjectType nvarchar(20)
,@distinguishedName nvarchar(2048) OUTPUT
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID which identifies the tenant partition (1) identifier of the queried object.

@RecordId: The object identifier. For users this is the same as the RecordID value from the UserProfile_Full data set. For groups this is the Id value from the MemberGroup data set. For contacts this is the ContactID value from DNContactLookup data set.

Preliminary

Copyright © 2012 Microsoft Corporation.

@ObjectType: Defines the type of the queried object. It can contain one of these 3 strings – 'user', 'group' or 'contact'.

@distinguishedName: The string used by the external system to identify the object.

Value	Description
null	The object was not found.
DNLookup.DN	The string used by the external system to identify the object.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.53 profile_GetRecordIdFromDN

This stored procedure gets the object identifier and type from the string used by the external system to identify the object.

```
PROCEDURE profile_GetRecordIdFromDN (
@DistinguishedName nvarchar(2048)
,@partitionID uniqueidentifier OUTPUT
,@recordId bigint OUTPUT
,@objectType nvarchar(20) OUTPUT
,@correlationId uniqueidentifier = null
);
```

@DistinguishedName: A string used by the external system to identify the object.

@partitionID: A GUID which identifies the tenant partition (1) identifier of the returned object.

Value	Description
null	The object was not found.
DNLookup.PartitionID	A GUID which identifies the tenant partition (1) identifier of the queried object.

@recordId: The object identifier. For users this is the same as the RecordID value from the UserProfile_Full data set. For groups this is the Id value from the MemberGroup data set. For contacts this is the ContactID value from DNContactLookup data set.

Value	Description
0	The object was not found.
dn.recordid	The object identifier.

@objectType: The object type. It can contain one of these 3 strings – 'user', 'group' or 'contact'.

Value	Description
null	The object was not found.

Preliminary

Copyright © 2012 Microsoft Corporation.

Value	Description
DNLookup.ObjectType	The object type.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.54 profile_ResetDNLookupTable

This stored procedure marks all objects as not imported. This stored procedure is called at the beginning of a full import.

```
PROCEDURE profile_ResetDNLookupTable (
@organizationalUnit nvarchar(256) = null
,@correlationId uniqueidentifier = null
);
```

@organizationalUnit: The Organizational Unit for which to clear the imported flag. If it is NULL all objects will be marked as not imported.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.55 profile_UpdateDNContactEntry

This stored procedure updates the information about an existing contact (3) or creates a new one.

```
PROCEDURE profile_UpdateDNContactEntry (
@partitionID uniqueidentifier
,@contactID bigint = null
,@DN nvarchar(2048) = null
,@ProfileID nvarchar(400) = null
,@DistinguishedName nvarchar(2048) = null
,@SourceObjectDN nvarchar(2048) = null
,@PreferredName nvarchar(256) = null
,@correlationId uniqueidentifier = null
);
```

@partitionID: A GUID which identifies the tenant partition (1) identifier of the updated contact.

@contactID: A logon name of the contact – used when the contact represents a secondary logon of an existing user.

@DN: A string used by the external system to identify the contact.

@ProfileID: An optional logon name. This parameter is used when the contact represents a second logon credential for an existing user.

@DistinguishedName: The distinguished name (DN) of the contact record.

Preliminary

Copyright © 2012 Microsoft Corporation.

@SourceObjectDN: The distinguished name of the user this contact represents.

@PreferredName: The contact **display name**.

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

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

Value	Description
-1	Invalid arguments.
-2	Trying to add a contact that already exists.
0	The operation completed successfully.
2	The account record was not found.

Result Sets: MUST NOT return any result sets.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Client Details

None.

3.3.1 Abstract Data Model

None.

3.3.2 Timers

None.

3.3.3 Initialization

None.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

None.

3.3.6 Timer Events

None.

Preliminary

Copyright © 2012 Microsoft Corporation.

3.3.7 Other Local Events

None.

Preliminary

Copyright © 2012 Microsoft Corporation.

4 Protocol Examples

In this example the stored procedures are used to import a single group with 5 users from Active Directory. After the import has finished, the imported member group members are added to the user profile store.

```
-- Sample value for default partition identifier
DECLARE @partitionId   uniqueidentifier SET @partitionId = '0c37852b-34d0-418e-91c6-
2ac25af4be5b'
DECLARE @correlationId   uniqueidentifier SET @partitionId = '00000000-0000-0000-0000-
000000000000'

-- Sample member group values
DECLARE @parentGroupId bigint           SET @parentGroupId = 100

-- Sample XML structure passed to the procedure for ImportExport_ImportMembers
DECLARE @members nvarchar(max)
SET @members = '
<Ms>
    <M DN="CN=UserOne,OU=UserAccounts,DC=DOMAINNAME,DC=corp,DC=COMPANYNAME,DC=com", OU=
"UserAccounts" />
    <M DN="CN=UserTwo,OU=UserAccounts,DC=DOMAINNAME,DC=corp,DC=COMPANYNAME,DC=com", OU=
"UserAccounts" />
    <M DN="CN=UserThree,OU=UserAccounts,DC=DOMAINNAME,DC=corp,DC=COMPANYNAME,DC=com", OU=
"UserAccounts" />
    <M DN="CN=UserFour,OU=UserAccounts,DC=DOMAINNAME,DC=corp,DC=COMPANYNAME,DC=com", OU=
"UserAccounts" />
    <M DN="CN=UserFive,OU=UserAccounts,DC=DOMAINNAME,DC=corp,DC=COMPANYNAME,DC=com", OU=
"UserAccounts" />
</Ms>'

-- Start the import process
DECLARE @ImportExportId uniqueidentifier
EXEC @ImportExportId = [dbo].[ImportExport_ImportStart] @correlationId
```

OUTPUT: After successful execution of this stored procedure, it returns 1 (sample data) which is assigned to importExportId.

```
-- Store the members temporarily in staging data sets for later processing
EXEC [dbo].[ImportExport_ImportMembers] @ImportExportId, @members, @parentGroupId,
@partitionId, @correlationId
```

OUTPUT: The stored procedure returns 0. After successful execution of this stored procedure, the sample data is inserted in the ImportExportStagedMember data set.

```
-- End the import process
EXEC [dbo].[ImportExport_ImportEnd] @ImportExportId, @correlationId

-- Perform the post processing of temporarily stored members
DECLARE @PostImportUserPropRet uniqueidentifier
EXEC @PostImportUserPropRet = [dbo].[ImportExport_PostImportUserProperties]
```

OUTPUT: After successful execution of this stored procedure, it returns 0 which is assigned to PostImportUserPropRet.

Preliminary

Copyright © 2012 Microsoft Corporation.

```
DECLARE @PostImportUserPropRet uniqueidentifier
EXEC @PostImportMemPropRet = [dbo].[ImportExport_PostImportMembers] @correlationId
```

OUTPUT: After successful execution of this stored procedure, it returns 0 which is assigned to PostImportMemPropRet.

```
-- Confirm that the members were successfully imported
-- The result set from ImportExport_GetGroupMembers displays the
-- distinguished names of the members in the member group.
EXEC [dbo].[ImportExport_GetGroupMembers] @parentPartitionId, @parentGroupId, @correlationId
```

OUTPUT: The output after successful execution of this stored procedure is as follows:

DistinguishedName
CN=UserOne,OU=UserAccounts,DC=DOMAINNAME,DC=corp,DC=COMPANYNAME,DC=com
CN=UserTwo,OU=UserAccounts,DC=DOMAINNAME,DC=corp,DC=COMPANYNAME,DC=com
CN=UserThree,OU=UserAccounts,DC=DOMAINNAME,DC=corp,DC=COMPANYNAME,DC=com
CN=UserFour,OU=UserAccounts,DC=DOMAINNAME,DC=corp,DC=COMPANYNAME,DC=com
CN=UserFive,OU=UserAccounts,DC=DOMAINNAME,DC=corp,DC=COMPANYNAME,DC=com

Preliminary

Copyright © 2012 Microsoft Corporation.

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

Preliminary

Copyright © 2012 Microsoft Corporation.

6 Appendix A: Product Behavior

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

- Microsoft® SharePoint® Server 2010
- Microsoft® SharePoint® Server 2013 Preview

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.

Preliminary

Copyright © 2012 Microsoft Corporation.

7 Change Tracking

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

Preliminary

Copyright © 2012 Microsoft Corporation.

8 Index

Preliminary

Copyright © 2012 Microsoft Corporation.

Copyright © 2012 Microsoft Corporation.

Preliminary

R

S

Preliminary

Copyright © 2012 Microsoft Corporation.

Preliminary

Copyright © 2012 Microsoft Corporation.