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.
## Revision Summary

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision History</th>
<th>Revision Class</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>04/04/2008</td>
<td>0.1</td>
<td></td>
<td>Initial Availability</td>
</tr>
<tr>
<td>06/27/2008</td>
<td>1.0</td>
<td>Major</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>10/06/2008</td>
<td>1.01</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>12/12/2008</td>
<td>1.02</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>07/13/2009</td>
<td>1.03</td>
<td>Major</td>
<td>Changes made for template compliance</td>
</tr>
<tr>
<td>08/28/2009</td>
<td>1.04</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>11/06/2009</td>
<td>1.05</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>02/19/2010</td>
<td>2.0</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>03/31/2010</td>
<td>2.01</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>04/30/2010</td>
<td>2.02</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>06/07/2010</td>
<td>2.03</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>06/29/2010</td>
<td>2.04</td>
<td>Editorial</td>
<td>Changed language and formatting in the technical content</td>
</tr>
<tr>
<td>07/23/2010</td>
<td>2.05</td>
<td>Editorial</td>
<td>Changed language and formatting in the technical content</td>
</tr>
<tr>
<td>09/27/2010</td>
<td>2.05</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>11/15/2010</td>
<td>2.05</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>12/17/2010</td>
<td>2.05</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>03/18/2011</td>
<td>2.05</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>06/10/2011</td>
<td>2.05</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>01/20/2012</td>
<td>2.6</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>04/11/2012</td>
<td>2.6</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>07/16/2012</td>
<td>2.6</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
</tbody>
</table>
# Table of Contents

1 **Introduction** ........................................................................................................... 6  
   1.1 Glossary ................................................................................................................. 6  
   1.2 References ............................................................................................................. 7  
       1.2.1 Normative References ..................................................................................... 7  
       1.2.2 Informative References .................................................................................. 7  
   1.3 Protocol Overview (Synopsis) .................................................................................. 8  
   1.4 Relationship to Other Protocols ............................................................................ 8  
   1.5 Prerequisites/Preconditions .................................................................................... 8  
   1.6 Applicability Statement ......................................................................................... 8  
   1.7 Versioning and Capability Negotiation .................................................................... 9  
   1.8 Vendor-Extensible Fields ....................................................................................... 9  
   1.9 Standards Assignments .......................................................................................... 9  

2 **Messages** .................................................................................................................. 10  
   2.1 Transport ............................................................................................................... 10  
   2.2 Common Data Types ............................................................................................. 10  
       2.2.1 Simple Data Types and Enumerations ............................................................. 10  
       2.2.2 ADConfig ....................................................................................................... 10  
       2.2.3 ProfileServerType ........................................................................................... 10  
       2.2.4 SQL Data Type ............................................................................................... 10  
       2.2.5 UserFormat .................................................................................................... 11  
       2.2.6 ProfilePropertyBlobType ............................................................................... 11  
       2.2.7 SourceConfiguration Schema ........................................................................ 11  
           2.2.7.1 SourceConfigurationElement ................................................................... 12  
           2.2.7.2 SourceConfigurationAttributes ............................................................... 12  
           2.2.7.3 ServerAttributes ...................................................................................... 12  
           2.2.7.4 GroupAttributes ....................................................................................... 13  
           2.2.7.5 Connection InformationProperties ......................................................... 13  
           2.2.7.6 ProfileImportSearchSettings ................................................................. 13  
       2.2.8 MappingListSchema ....................................................................................... 14  
           2.2.8.1 MSPProfileElement ............................................................................... 14  
           2.2.8.2 DataServiceElement ................................................................................. 14  
           2.2.8.3 DataServiceAttributes ............................................................................. 14  
           2.2.8.4 MappingElement ..................................................................................... 15  
           2.2.8.5 DetailedDescriptionofAttributes ............................................................ 15  
       2.2.9 ArrayOfMappingComplexType ....................................................................... 15  
       2.2.10 ArrayOfDataServiceComplexType ................................................................ 16  

3 **Protocol Details** ....................................................................................................... 17  
   3.1 User Profile Import Server Details ........................................................................ 17  
       3.1.1 AbstractDataModel ......................................................................................... 17  
       3.1.2 Timers ............................................................................................................ 17  
       3.1.3 Initialization ................................................................................................... 17  
       3.1.4 Message Processing Events and Sequencing Rules ....................................... 17  
           3.1.4.1 ProfileImport and ProfileImportAltTables ................................................. 19  
           3.1.4.2 profile_GetADConfiguration ................................................................. 19  
           3.1.4.2.1 ADConfigurationResultSet ................................................................. 20  
           3.1.4.3 profile_GetDataService ........................................................................... 21  
           3.1.4.3.1 ProfileDataServiceResultSet ............................................................... 21  
           3.1.4.4 profile_GetDataServicePropMapping ..................................................... 22  

---

*User Profile Import Protocol Specification*

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
3.1.4.4.1 DataServicePropMapping Result Set .................................................. 22
3.1.4.5 profile_GetDataTypeList ................................................................. 23
3.1.4.5.1 DataTypeList Result Set ............................................................ 24
3.1.4.6 profile_GetDomainCookie ............................................................... 25
3.1.4.6.1 DomainCookie Result Set .......................................................... 26
3.1.4.7 profile_LogImportStart ................................................................. 26
3.1.4.7.1 ProfileImportStatus Result Set .................................................... 27
3.1.4.8 profile_LogImportStopForced ......................................................... 27
3.1.4.9 profile_PluginDataImport ............................................................... 28
3.1.4.10 profile_PluginDelete ................................................................. 28
3.1.4.11 profile_PluginOnEndCrawl ........................................................... 29
3.1.4.12 profile_PluginOnStartCrawl .......................................................... 29
3.1.4.13 profile_PluginReset ................................................................. 30
3.1.4.14 profile_GetDeletedUserList ......................................................... 30
3.1.4.14.1 DeletedUsers Result Set .......................................................... 30
3.1.4.15 profile_SetDomainCookie ............................................................ 31
3.1.4.15.1 SetDomainCookie Result Set .................................................... 31
3.1.4.16 profile_UpdateDataService ......................................................... 31
3.1.4.16.1 UpdateDataServiceProcResults Result Set ................................. 32
3.1.4.17 profile_UpdateDataServiceMap ..................................................... 33
3.1.4.18 profile_UpdateADConfiguration ...................................................... 33
3.1.4.19 membership_addRecursiveGroup ................................................... 35
3.1.4.19.1 AddRecursiveGroup Result Set .................................................. 35
3.1.4.20 membership_getAllMemberOf ....................................................... 37
3.1.4.20.1 MemberOf Result Set ............................................................. 37
3.1.4.21 Import Status and Termination ....................................................... 37
3.1.5 Profile Import Termination ................................................................. 37
3.1.6 Other Local Events .................................................................................. 37
3.2 User Profile Import Client Details ......................................................... 38
3.2.1 Abstract Data Model ............................................................................. 38
3.2.2 Timers .................................................................................................... 38
3.2.3 Initialization ........................................................................................... 38
3.2.4 Message Processing Events and Sequencing Rules .............................. 38
3.2.4.1 State Transitions ............................................................................... 39
3.2.4.2 Starting Profile Import Process ......................................................... 39
3.2.4.3 Get Data Connection Information .................................................... 39
3.2.4.4 Filling and Processing Staging Data Area ........................................... 39
3.2.4.5 Finishing Profile Import ................................................................. 40
3.2.4.6 Member Group Import ................................................................. 40

4 Protocol Examples ......................................................................................... 41
4.1 Adding an External Directory ................................................................. 41
4.2 Adding a Property Mapping ................................................................. 41
4.3 Running a Full Import ............................................................................ 41
4.4 Running an Incremental Import ............................................................ 42
4.5 Stopping an Import ................................................................................ 42
4.6 Issuing a Reset Command ................................................................. 42
4.7 Inserting Staging Data ............................................................................. 43

5 Security ...................................................................................................... 44
5.1 Security Considerations for Implementers ............................................... 44
5.2 Index of Security Parameters ................................................................. 44
1 Introduction

The User Profile Import protocol enables the protocol client to import user profiles, member groups and membership from directory services into the protocol server.

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

1.1 Glossary

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

Active Directory
distinguished name (DN)
domain
domain account
domain controller (DC)
forest
fully qualified domain name (FQDN)
GUID
LDAP
Secure Sockets Layer (SSL)
Security Support Provider Interface (SSPI)
user principal name (UPN)

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

Association
binary large object (BLOB)
display name
document identifier
domain cookie
Entity
Hypertext Markup Language (HTML)
import connection
LobSystemInstance
login name
member group
membership
property mapping
result set
return code
stored procedure
Structured Query Language (SQL)
Uniform Resource Locator (URL)
user profile
user profile import
user profile store
UserProfileFilter

The following terms are specific to this document:
crawl type: A setting that specifies whether to evaluate all of the users and member groups in
the directory service that is crawled, or only those users and member groups that were
modified after the last crawl.

profile data type: A data type that is mapped to one or more profile properties and one SQL
Server™ data type.

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

1.2 References

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

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If
you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We
will assist you in finding the relevant information. Please check the archive site,
additional source.


us/library/ms189826(SQL.90).aspx


[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC

[RFC2254] Howes, T., "The String Representation of LDAP Search Filters", RFC 2254, December


[RFC4122] Leach, P., Mealling, M., and Salz, R., "A Universally Unique Identifier (UUID) URN

1.2.2 Informative References


[MSDN-IRowsetFastLoad] Microsoft Corporation, "IRowsetFastLoad (OLE DB)",

[MS-UPSIMP] — v20120630
User Profile Import Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
1.3 Protocol Overview (Synopsis)

This protocol enables user data from a directory service to be imported into a user profile store. The supported directory services are Active Directory, LDAP and Business Data Catalog. Active Directory and LDAP are used to import user profiles. The Business Data Catalog is used to import data for specific user profile properties for existing user profiles. The protocol also enables Active Directory directory services to import member groups.

To accomplish this, the user profile service maintains a list of directory services it is able to import from. Any given user profile property in the user profile store can be mapped to an attribute within a directory service.

The protocol is designed to enable processing of each directory service by a synchronization process. This process finds all new or updated users in the directories and communicates the new user profiles updated properties back to the user profile service. It then finds new or updated member groups for those users.

1.4 Relationship to Other Protocols

The following diagram shows the transport stack for this protocol and the relationship to other protocols:

![Diagram of transport stack]

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

The operations described by this protocol operate between a protocol client and a protocol server. The client is expected to have the location and connection information for the required databases on the protocol server.

This protocol requires that the protocol client has appropriate permissions to call the stored procedures in the required databases on the protocol server.

1.6 Applicability Statement

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

- Five million user profiles
On average 100 member groups per user profile

1.7 Versioning and Capability Negotiation

Security and Authentication Methods: This protocol supports the Security Support Provider Interface (SSPI) and SQL authentication with the protocol server role specified in [MS-TDS].

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.
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

2.2.1   Simple Data Types and Enumerations

2.2.2   ADConfig

ADConfig is a four byte signed integer that tracks the user profile import configuration of the user profile service. The value MUST be listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000001</td>
<td>User profile import is scheduled.</td>
</tr>
<tr>
<td>0x00000002</td>
<td>User profile import source is set to current domain.</td>
</tr>
<tr>
<td>0x00000004</td>
<td>User profile import source is set to entire forest.</td>
</tr>
<tr>
<td>0x00000008</td>
<td>User profile import source is set to administrator defined.</td>
</tr>
</tbody>
</table>

2.2.3   ProfileServerType

ProfileServerType is a string value that tracks the kind of directory service associated with the user profile service. The value MUST be in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>'LDAP'</td>
<td>LDAP directory.</td>
</tr>
<tr>
<td>'AD'</td>
<td>Active Directory[MS-ADTS].</td>
</tr>
<tr>
<td>'AR'</td>
<td>Business Data Catalog.</td>
</tr>
<tr>
<td>'ADR'</td>
<td>An Active Directory from which resource attributes can be imported, but is not the authoritative source for the AccountName, Manager, and SID attributes. For a description of the server attributes, see section 2.2.7.3. When 'ADR' is specified, the protocol client MUST also specify a valid LoginDomain. When crawling, the client MUST import the AccountName, Manager, and SID attributes into the corresponding user profile properties from the directory service specified by LoginDomain. The client MUST import all other mapped attributes from this directory.</td>
</tr>
<tr>
<td>'LOGIN'</td>
<td>Active Directory which is the authoritative source for the AccountName, Manager, and SID attributes.</td>
</tr>
</tbody>
</table>

2.2.4   SQL Data Type

There is a SQL data type associated with each user profile property. The data type of the value MUST be in the following table. For the data type specifications, see [MS-TDS].
### 2.2.5 UserFormat

**UserFormat** is the value that is used to determine the format of the `AccountName` and `Manager` properties.

The `AccountName` value MUST be 0x0001 which means that the `UserName` property is formatted as a domain account name.

The `Manager` value MUST be in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0001</td>
<td>The <code>Manager</code> property is formatted as a domain account name.</td>
</tr>
<tr>
<td>0x0002</td>
<td>The <code>Manager</code> property is formatted as an LDIF ([RFC2849]) distinguished name (DN).</td>
</tr>
<tr>
<td>0x0003</td>
<td>The <code>Manager</code> property is formatted as a user principal name (UPN).</td>
</tr>
<tr>
<td>0x0004</td>
<td>The <code>Manager</code> property is formatted as a display name.</td>
</tr>
<tr>
<td>0x0005</td>
<td>The <code>Manager</code> property is formatted as a GUID.</td>
</tr>
<tr>
<td>0x0006</td>
<td>The <code>Manager</code> property format is unknown.</td>
</tr>
</tbody>
</table>

### 2.2.6 ProfilePropertyBlobType

**ProfilePropertyBlobType** is a 4 byte signed integer enumeration that tracks the format of a user profile property BLOB type. The value MUST be in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000001</td>
<td>Variable-length binary data.</td>
</tr>
<tr>
<td>0x00000000</td>
<td>All other type of data.</td>
</tr>
</tbody>
</table>

### 2.2.7 SourceConfiguration Schema

The complex types, simple types, and elements that are described in this section are used in the `ExtraConfiguration` value of a user profile service.
2.2.7.1 SourceConfiguration Element

The SourceConfiguration element and its child elements are used to store extra configuration values associated with a user profile service.

<?xml version="1.0" encoding="utf-16"?><xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema"><xs:element name="SourceConfiguration"><xs:complexType><xs:sequence><xs:element minOccurs="0" name="Server"><xs:complexType><xs:sequence><xs:element minOccurs="0" name="Group"><xs:complexType><xs:sequence><xs:element minOccurs="0" name="Property"><xs:complexType><xs:attribute name="name" type="xs:string" use="required" /><xs:attribute name="value" type="xs:string" use="required" /></xs:complexType></xs:element></xs:sequence><xs:attribute name="Type" type="xs:string" use="required" /><xs:attribute name="Domain" type="xs:string" use="required" /><xs:attribute name="LoginDomain" type="xs:string" use="required" /><xs:attribute name="ConnectionName" type="xs:string" use="required" /><xs:attribute name="ProviderName" type="xs:string" use="required" /><xs:attribute name="AutoDiscover" type="xs:string" use="required" /><xs:attribute name="ServerIncremental" type="xs:string" use="required" /><xs:attribute name="name" type="xs:string" use="required" /></xs:complexType></xs:element></xs:sequence><xs:attribute name="Version" type="xs:decimal" use="required" /></xs:complexType></xs:element></xs:sequence><xs:attribute name="name" type="xs:string" use="required" /></xs:complexType></xs:element></xs:schema>

2.2.7.2 SourceConfiguration Attributes

Version: Version number supported by the protocol server. MUST be set to 3.0.

2.2.7.3 Server Attributes

Type: Directory service type of the import connection

Domain: Fully qualified domain name (FQDN) for the import connection.

LoginDomain: Domain name for the import connection, if different from the Domain attribute (section 2.2.3).
**ConnectionName:** Name of the connection to a directory service for the import connection.

**ProviderName:** Name of the connection provided by the user profile service administrator. For Active Directory, this value MUST be empty.

**AutoDiscover:** MUST be set to either "Yes" or "No". If the value is "Yes", the client MUST automatically populate the name field with the FQDN of the domain controller (DC). If the value is "No", the client MUST store validated input from the administrator in the name field.

**SeverIncremental:** MUST be set to either "Yes" or "No". If the value is "Yes", the client MUST provide incremental changes to its data from sources even for subsequent full import sessions. If the value is "No", the client MUST provide full set of data from sources for every full import session.

**name:** FQDN of the DC for the domain of the import connection.

### 2.2.7.4 Group Attributes

**name:** Name of the group of configuration attributes. MUST be "Connection information" or "Search Information".

### 2.2.7.5 Connection Information Properties

Connection information properties consist of name/value pairs that store extra configuration for an import connection.

**Port:** Port number used for the import connection.

**UseSecureSocketLayer:** MUST be "Yes" or "No". If the value is "Yes", the client MUST connect to the corresponding user profile service using Secure Sockets Layer (SSL). If the value is "No", an SSL connection is NOT required.

**ServerTimeout:** Numeric value indicating the maximum number of seconds to wait for a response from the server before failing with a timeout error.

The following properties apply only to a Business Data Catalog [MS-BDCSP] connection.

**AppSystem:** Name of the Business Data Catalog LobSystemInstance.

**AppEntity:** Name of the Business Data Catalog Entity to import from.

**AppFilter:** Name of the Business Data Catalog UserProfileFilter.

**UPPropName:** User profile property name corresponding to the property used for a Business Data Catalog connection.

### 2.2.7.6 Profile Import Search Settings

Search information properties consist of name/value pairs that store extra configuration for an import connection.

**UserIDAttribute:** Unique identifier of a user profile in the user profile service. The default value is "uid" for LDAP and "distinguishedname" for Active Directory.

**Search Base:** Distinguished name of the directory node from which to import.

**User Filter:** Filter criteria against Active Directory or LDAP. This string MUST conform to [RFC2254] that defines the string representation of LDAP Search Filters.
**Scope**: Enumeration value corresponding to the search depth. MUST be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubTree</td>
<td>Search the level of Search Base and levels beneath.</td>
</tr>
<tr>
<td>One-Level</td>
<td>Search the level of <strong>Search Base</strong>.</td>
</tr>
</tbody>
</table>

**Page Timeout**: Number of seconds to wait for a response before failing with an Active Directory timeout error.

**Page Size**: Number of results to return per page. The value is applied to Active Directory results.

### 2.2.8 MappingList Schema

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

#### 2.2.8.1 MSProfile Element

The **MSProfile** element is used to request properties that are updated, added, or removed from a user profile service.

```xml
<s:element name="MSProfile">
    <s:complexType>
        <s:element minOccurs="0" maxOccurs="Unbounded" name="DataService" type="tns:ArrayOfDataService"/>
    </s:complexType>
</s:element>
```

#### 2.2.8.2 DataService Element

The **DataService** element is used to request properties that are updated, added, or removed from a user profile service.

```xml
<s:element name="DataService">
    <s:complexType>
        <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="Mapping" type="tns:ArrayOfMapping"/>
            <s:attribute minOccurs="1" maxOccurs="1" name="ProfileName" type="s:string"/>
            <s:attribute minOccurs="1" maxOccurs="1" name="ServiceName" type="s:string"/>
        </s:sequence>
    </s:complexType>
</s:element>
```

#### 2.2.8.3DataService Attributes

- **ProfileName**: MUST be 'UserProfile'.
- **ServiceName**: MUST be 'redmond-AD'.

---

[MS-UPSIMP] — v20120630
User Profile Import Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
2.2.8.4 Mapping Element

The Mapping element is used to request properties that are updated, added, or removed from a user profile service.

```xml
<s:element name="Mapping">
  <s:complexType>
    <s:sequence>
      <s:attribute minOccurs="1" maxOccurs="1" name="PropertyName" type="s:string"/>
      <s:attribute minOccurs="0" maxOccurs="1" name="DataServicePropName" type="s:string"/>
      <s:attribute minOccurs="1" maxOccurs="1" name="AssociationName" type="s:string"/>
      <s:attribute minOccurs="0" maxOccurs="1" name="DataSource" type="s:string"/>
      <s:attribute minOccurs="1" maxOccurs="1" name="bRecordIdentifier" type="s:boolean"/>
    </s:sequence>
  </s:complexType>
</s:element>
```

2.2.8.5 Detailed Description of Attributes

**PropertyName**: Unique name of an existing property. The client MUST specify the name of a valid property that exists in the data store.

**DataServicePropName**: Name of the property used by its originating user profile service. The client MUST provide this attribute for @UpdateMappingList. The client MUST NOT provide this attribute for @RemoveMappingList.

**DataSource**: Name of the directory service connection. This value MUST be NULL when the ProfileServerType value is 'LDAP' or 'AD'. The DataSource value MUST be a valid name of the directory service connection when the ProfileServerType value is 'AR', 'ADR', or 'LOGIN'. The client MUST specify this property for @UpdateMappingList. The client MUST NOT specify this property within @RemoveMappingList.

**AssociationName**: Name of the Association in the profile data service. This value MUST be an empty string if no Association exists for this object. A valid Association is created by proc_ar_CreateAssociation and can be retrieved by proc_ar_GetAssociationByName [MS-BD CSP]. The client MUST specify this property for @UpdateMappingList. The client MUST NOT specify this property within @RemoveMappingList.

**bRecordIdentifier**: MUST be 0.

2.2.8.6 ArrayOfMapping Complex Type

The ArrayOfMapping complex type represents a set of zero or more ServerLink elements.

```xml
<s:complexType name="ArrayOfMapping ">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="Mapping" nillable="true" type="tns:Mapping" />
  </s:sequence>
</s:complexType>
```
2.2.8.7 ArrayOfDataService Complex Type

The ArrayOfDataService complex type represents a set of zero or more DataService elements.

```xml
<xs:complexType name="ArrayOfDataService">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="DataService" nillable="true" type="tns:DataService"/>
  </xs:sequence>
</xs:complexType>
```
3 Protocol Details

3.1 User Profile Import Server Details

3.1.1 Abstract Data Model

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

The protocol server maintains the following data.

- A list of user profile properties and their mappings to directory service attributes.
- 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 staging data area where the protocol client inserts data from directory services.
- State information about whether a profile import session is in progress and if so, what stage the import session is in.

3.1.2 Timers

The protocol server is not required to maintain any timer event.

3.1.3 Initialization

Before using this protocol, a connection that uses the underlying protocol layers specified in section 1.4, Relationship to Other Protocols, MUST be established as specified in [MS-TDS].

The protocol server MUST setup and initialize data structures to store staging data that is received as part of the profile import process.

3.1.4 Message Processing Events and Sequencing Rules

The following tables are available as part of the protocol using standard T-SQL tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProfileImportProfileImportAlt</td>
<td>Stores temporary data for user profiles and member groups during the profile import process.</td>
</tr>
</tbody>
</table>

The following stored procedures are available as part of this protocol using standard T-SQL stored procedure calls.
<table>
<thead>
<tr>
<th>Stored procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>profile_GetADConfiguration</td>
<td>Returns configuration information for the Active Directory associated with user profile import.</td>
</tr>
<tr>
<td>profile_GetDataService</td>
<td>Returns SourceConfiguration associated with a user profile service. See section 2.2.6.1.1.</td>
</tr>
<tr>
<td>profile_GetDataServicePropMapping</td>
<td>Returns the list of property mappings.</td>
</tr>
<tr>
<td>profile_GetDataTypeList</td>
<td>Returns the valid data types for user profile properties.</td>
</tr>
<tr>
<td>Profile_GetDeletedUserList</td>
<td>Returns the list of user profiles marked as deleted.</td>
</tr>
<tr>
<td>profile_GetDomainCookie</td>
<td>Returns the domain cookie data and settings for a user profile service.</td>
</tr>
<tr>
<td>profile_LogImportStart</td>
<td>Logs the beginning of a user profile import.</td>
</tr>
<tr>
<td>profile_LogImportStopForced</td>
<td>Logs a stop attempt for a running user profile import.</td>
</tr>
<tr>
<td>profile_PlugInDataImport</td>
<td>Processes the staging data.</td>
</tr>
<tr>
<td>profile_PlugInDelete</td>
<td>When a user is deleted in a directory service, marks the corresponding user profile for removal by the user profile service.</td>
</tr>
<tr>
<td>profile_PluginOnEndCrawl</td>
<td>Processes an end of user profile import session.</td>
</tr>
<tr>
<td>profile_PluginOnStartCrawl</td>
<td>Registers the beginning of a user profile import session.</td>
</tr>
<tr>
<td>profile_PluginReset</td>
<td>Sets protocol server state to enable deleting previously imported user profiles.</td>
</tr>
<tr>
<td>profile_SetDomainCookie</td>
<td>Updates the domain cookie data and settings for a user profile service.</td>
</tr>
<tr>
<td>profile_UpdateDataService</td>
<td>Updates settings for user profile import.</td>
</tr>
<tr>
<td>profile_UpdateDataServiceMap</td>
<td>Applies and persists the property mapping for user profile service</td>
</tr>
<tr>
<td>profile_UpdateADConfiguration</td>
<td>Updates Active Directory user profile import configuration.</td>
</tr>
<tr>
<td>membership_addRecursiveGroup</td>
<td>Updates a member group or adds a new member group if it does not exist.</td>
</tr>
<tr>
<td>membership_getAllMemberOf</td>
<td>Returns the complete list of users across all member groups.</td>
</tr>
</tbody>
</table>

Unless noted otherwise in the descriptions, all stored procedures:

- **MUST** return an integer value of 0 if the request completes successfully. Conversely, if the request is unable to be finished successfully on the server, the server **MUST** do one of the following:
  - Return a nonzero integer value consistent with T-SQL error codes.
  - Return bits over the wire that cause SQL ADO methods to throw an exception.
MUST NOT return any result sets.

3.1.4.1 ProfileImport and ProfileImportAlt Tables

The ProfileImport and ProfileImportAlt tables store temporary data for user profiles and member groups. They are also referred to as Staging Data Area. The protocol client MAY use the T-SQL INSERT BULK command to fill either table. The protocol server MUST be able to accept this command.

The ProfileImport (ProfileImportAlt) table is defined using T-SQL syntax as follows:

```
TABLE ProfileImport(
   DocID                   int NULL,
   CatalogIDsmall          int NULL,
   PropId                  int NOT NULL,
   Signature               int NOT NULL,
   Flags                   smallint NOT NULL,
   URI                     nvarchar(256) NULL,
   Text                    sql_variant NULL
);
```

**DocID**: Document identifier of the user profile being imported.

**CatalogID**: MUST be ignored.

**PropID**: Identifier for the user profile property.

**Signature**: Indicates whether the property value has changed since the last import. When the property value is the same as the previous import, the client MUST provide the same signature value as the last import and the server MUST NOT update the user profile property. When the property value is different than the previous import, the client MUST provide a different signature value and the server MUST update the user profile property.

**Flags**: MUST be ignored.

**URI**: A URN ([RFC4122]) for the property.

**Text**: Value for the property of the profile being imported.

3.1.4.2 profile_GetADConfiguration

The profile_GetADConfiguration stored procedure is invoked to retrieve user profile Active Directory connection configuration. profile_GetADConfiguration is defined using T-SQL syntax, as follows:

```
PROCEDURE profile_GetADConfiguration(
   @DataServiceName nvarchar(50)
);
```

**@DataServiceName**: MUST be 'redmond-AD'.

**Return Code Values**: profile_GetADConfiguration MUST return 0.

**Result Sets**: profile_GetADConfiguration MUST return one ADConfiguration result set.
3.1.4.2.1 ADConfigurationResult Set

The profile_GetADConfiguration stored procedure returns properties of the requested Active Directory connection configuration. The ADConfiguration result set MUST return one row containing a set of properties of the active directory connection configuration. The ADConfiguration result set is defined using T-SQL syntax, as follows:

```sql
Error                       int,
IsADConfigured              bit,
IsScheduled                 bit,
ADImportChoice              int,
bImportEnabled              bit,
IsRequiredMapped bit,
PropertyCount               int,
MappedPropertyCount         int;
```

**Error**: The error code of this execution.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The execution succeeded.</td>
</tr>
<tr>
<td>1</td>
<td>The execution failed.</td>
</tr>
</tbody>
</table>

**IsADConfigured**: Flag indicating whether the Active Directory connection is configured.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The server configuration is set.</td>
</tr>
<tr>
<td>0</td>
<td>The server configuration is not set.</td>
</tr>
</tbody>
</table>

**IsScheduled**: Flag indicating whether the import has been scheduled.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The import is scheduled.</td>
</tr>
<tr>
<td>0</td>
<td>The import is not scheduled.</td>
</tr>
</tbody>
</table>

**ADImportChoice**: Current import choice. MUST be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>The import source is current domain of the user profile service.</td>
</tr>
<tr>
<td>4</td>
<td>The import source is current forest of the user profile service.</td>
</tr>
<tr>
<td>8</td>
<td>The import source is administrator defined.</td>
</tr>
</tbody>
</table>

**bImportEnabled**: Flag indicating whether the import is enabled.
### Value | Description
--- | ---
1 | The import is enabled.
0 | The import is not enabled.

**bIsRequireMapped:** Flag indicating whether the required properties have been mapped. The required properties are *FirstName*, *LastName*, and *AccountName*.

| Value | Description |
--- | ---
1 | The required properties are all mapped. |
0 | The required properties are not all mapped. |

**PropertyCount:** Number of properties.

**MappedPropertyCount:** Number of properties that are mapped to the connection source.

### 3.1.4.3 profile_GetDataService

The `profile_GetDataService` stored procedure is invoked to retrieve user profile service properties and optionally the property mappings. `profile_GetDataService` is defined using T-SQL syntax, as follows:

```sql
PROCEDURE profile_GetDataService (  
    @DataServiceName            nvarchar(50),  
    @ProfileName                nvarchar(250) N'UserProfile',  
    @RetrieveMapping            bit 0,  
    @bDebug                     bit 0  
) ;
```

- **@DataServiceName:** MUST be 'redmond-AD'.
- **@ProfileName:** MUST be 'UserProfile' or not specified.
- **@RetrieveMapping:** MUST be 0.
- **@bDebug:** If set to nonzero, the server enables return of the message indicating the number of rows affected as part of the T-SQL results. If set to 0, stop the message.

**Return Code Values:** `profile_GetDataService` MUST return 0.

**Result Sets:** `profile_GetDataService` MUST return a ProfileDataService result set.

### 3.1.4.3.1 ProfileDataService Result Set

The `profile_GetDataService` stored procedure returns properties of the requested user profile service. The ProfileDataService result set MUST return one row containing the name and settings of the user profile service. The ProfileDataService result set is defined using T-SQL syntax, as follows:

```sql
DataServiceID                int,
DataServiceName              nvarchar(100),
DataServiceURL               nvarchar(4096),
ProfileID                    int,
```
DataServiceID: Unique identifier of the user profile service. MUST be 1.

DataServiceName: MUST be 'redmond-AD'.

DataServiceURL: MUST be 'redmond-AD-URL'.

ProfileID: MUST be 1.

ExtraConfiguration: SourceConfiguration information. MUST be created by calling profile_UpdateDataService.

Description: Description of the user profile service. This value can be NULL.

PersonDBFormat: UserFormat of the user and manager. The information is stored in the last byte whose last 4 bits specify the UserFormat of the user and first 4 bits specify the UserFormat of the manager.

ConfigInfo: ADConfig value for this user profile service instance.

ADImportChoice: Value indicating the import choice. MUST be one of the values listed for ADImportChoice in section 3.1.4.2.1.

3.1.4.4 profile_GetDataServicePropMapping

The profile_GetDataServicePropMapping stored procedure is invoked to return a list of property mappings for the given user profile service. profile_GetDataServicePropMapping is defined using T-SQL syntax, as follows:

```sql
PROCEDURE profile_GetDataServicePropMapping(
    @DataServiceName nvarchar(50),
    @Debug bit DEFAULT 0
);
```

@DataServiceName: MUST be 'redmond-AD'.

@Debug: If set to nonzero, the protocol server enables return of the message indicating the number of rows affected as part of the T-SQL results. If set to 0, stop the message.

Return Code Values: profile_GetDataServicePropMapping MUST return 0.

Result Sets: profile_GetDataServicePropMapping MUST return one DataServicePropMapping result set.

3.1.4.4.1 DataServicePropMapping Result Set

The profile_GetDataServicePropMapping stored procedure returns the property mapping list for the user profile service. The DataServicePropMapping result set MUST return one or more rows; each row corresponds to a single property. The DataServicePropMapping result set is defined using T-SQL syntax, as follows:
PropertyURI: URI for the property.

DataType: SQL data type for the property.

Length: Maximum length of the value of the property. This value MUST NOT be NULL for the SQL data types 'nvarchar' and 'varbinary'.

BlobType: MUST be one of the values specified in ProfilePropertyBlobType. See section 2.2.6.

DataServicePropName: Display name for this property.

AssociationName: Name of the Association in the user profile service. This value MUST be NULL if no Association exists for this object. A valid Association is created by calling proc_ar_CreateAssociation and can be retrieved by proc_ar_GetAssociationByName [MS-BDCSP].

PropertyName: Display name of the property.

IsMultiValue: If set to 1, this property can store multiple values.

IsURL: If set to 1, the value for this property is a URL.

IsPerson: If set to 1, this property can be used to identify a user profile.

IsEmail: If set to 1, the value for this property MUST be an e-mail address.

DataTypeName: User-friendly name of the profile data type of the property. This value MUST be a valid Name value returned by calling profile_GetDataTypeList.

DataSource: Name of the directory service connection. This value MUST be NULL when the ProfileServerType value is 'LDAP' or 'AD'. The DataSource value MUST be a valid name of the directory service connection when the ProfileServerType value is 'AR', 'ADR', or 'LOGIN'.

### 3.1.4.5 profile_GetDataTypeList

The profile_GetDataTypeList stored procedure is invoked to return the properties for user profile data types. profile_GetDataTypeList is defined using T-SQL syntax, as follows:

```
PROCEDURE profile_GetDataTypeList(
    @Collation nvarchar(60)
);
```
@Collation: MUST be a valid SQL collation name[MS-TDS]. The server MUST use this collation name to sort the output by column FriendlyTypeName.

Return Code Values: profile_GetDataTypeList MUST return 0.

Result Sets: profile_GetDataTypeList MUST return one DataTypeList result set.

3.1.4.5.1   DataTypeList Result Set

The DataTypeList stored procedure returns the list of profile data types defined in the system. The DataTypeList result set MUST contain one row per Profile Data Type. 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: Unique identifier of the current profile data type.

DataTypeName: SQL data type of the current profile data type.

Name: Unique name of the current profile data type.

FriendlyTypeName: User-facing description of the current profile data type.

MaxCharCount: Maximum input length for the value of a property associated with this profile data type. If the value is not NULL, the client MUST limit input length to this value for properties associated with this profile data type.

IsFulltextIndexable: MUST be ignored.

AllowMultiValue: If set to 1, indicates that a property associated with this profile data type supports multiple values.

BlobType: A ProfilePropertyBlobType value indicating how the BLOB value is stored.

IsEmail: If set to 1, indicates the value for the property associated with this profile data type is an e-mail address.

IsURL: If set to 1, indicates the value for a property associated with this profile data type MUST contain a URL.

IsPerson: If set to 1, indicates the value for the property is associated with a user profile object.

IsHTML: If set to 1, indicates the value for the property associated with this profile data type is a fully formatted HTML text data.

The result set MUST contain the following rows where DataTypeID is any valid unique identifier:
3.1.4.6   profile_GetDomainCookie

The profile_GetDomainCookie stored procedure is invoked to retrieve cached Active Directory domain cookie.

profile_GetDomainCookie is defined using T-SQL syntax, as follows:

PROCEDURE profile_GetDomainCookie(
    @DataServiceName            nvarchar(50),
    @DomainName                 nvarchar(50)NULL
)
@DataServiceName: MUST be 'redmond-AD'.

@DomainName: Active Directory domain name to retrieve. This parameter can be NULL.

Return Code Values: profile_GetDomainCookie MUST return 0.

Result Sets: profile_GetDomainCookie MUST return one DomainCookie result set.

### 3.1.4.6.1 DomainCookie Result Set

DomainCookie result set contains zero or more rows with each row holds the domain name and the cached domain cookie. The result set is defined using T-SQL syntax, as follows:

```sql
DataServiceID int,
DomainName nvarchar(50),
Cookie image,
Length int;
```

**DataServiceID:** Unique identifier of the user profile service. MUST be 1.

**DomainName:** Name of the required domain.

**Cookie:** Data of the required domain cookie.

**Length:** Size of the data of the required domain cookie in bytes.

### 3.1.4.7 profile_LogImportStart

The `profile_LogImportStart` stored procedure is invoked to record the start of a user profile import or to return the status and start time for the latest user profile import.

`profile_LogImportStart` is defined using T-SQL syntax, as follows:

```sql
PROCEDURE profile_LogImportStart(
    @Type tinyint NULL
);
```

**@Type:** Import type. MUST be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>Query the current status of the import process. The server MUST return a result set showing the current status of the user profile import.</td>
</tr>
<tr>
<td>0</td>
<td>Record the start time of an import process, the start time of the import process in UTC format, and set the import process to user profiles.</td>
</tr>
<tr>
<td>1</td>
<td>Record the start time of an import process and set the import process to member groups.</td>
</tr>
<tr>
<td>2</td>
<td>Record the start time of an import process and set the import process to member groups or user profiles previously marked as deleted.</td>
</tr>
</tbody>
</table>
Return Code Values: `profile_LogImportStart` MUST return 0.

Result Sets: `profile_LogImportStart` MUST return one `ProfileImportStatus` result set if the @Type parameter is NULL. `profile_LogImportStart` MUST NOT return a result set if the @Type parameter is NOT NULL.

### 3.1.4.7.1 ProfileImportStatus Result Set

ProfileImportStatus result set returns the status and start time of the last user profile import. The ProfileImportStatus result set MUST return one row. The ProfileImportStatus result set is defined using T-SQL syntax, as follows:

```
Status          int,
bInProcess      bit,
StartTime       datetime;
```

**Status**: Returns status of user profile import. This corresponds to @Type parameter value of this stored procedure. The server MUST return one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>User profile import: user profile or member group is being imported into the system</td>
</tr>
<tr>
<td>1</td>
<td>Member group import: memberships are added for existing users.</td>
</tr>
<tr>
<td>2</td>
<td>Previously marked deleted member group or user profile is being processed.</td>
</tr>
<tr>
<td>100</td>
<td>User profile import was cancelled because of a request from a user.</td>
</tr>
<tr>
<td>101</td>
<td>Member group import was cancelled because of a request from a user.</td>
</tr>
<tr>
<td>102</td>
<td>Member group or user profile delete processing was cancelled because of a request from a user</td>
</tr>
</tbody>
</table>

**bInProcess**: The protocol server MUST return 1 if a user profile import is in process. The server MUST return 0 if a user profile import is not in process.

**StartTime**: Start time of the last user profile import session.

### 3.1.4.8 profile_LogImportStopForced

The `profile_LogImportStopForced` stored procedure is invoked to update the protocol server state when the client stops the profile import. `profile_LogImportStopForced` is defined using T-SQL syntax as follows:

```
PROCEDURE profile_LogImportStopForced();
```

Return Code Values: `profile_LogImportStopForced` MUST return 0.

Result Sets: `profile_LogImportStopForced` MAY return one arbitrary result set with zero or more rows. The client MUST ignore this result set.
### 3.1.4.9 profile_PluginDataImport

The **profile_PluginDataImport** stored procedure is invoked to process the staging data, to place the processed data into the user profile store. The server MUST clean up staging data when it’s done. **profile_PluginDataImport** is defined using T-SQL syntax, as follows:

```sql
PROCEDURE profile_PluginDataImport(
  @nCatalogId int,
  @nIsAlt int
);
```

**@nCatalogId**: MUST be ignored.

**@nIsAlt**: Specify which staging table to be used. MUST be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not 1</td>
<td>Process the staging data in the ProfileImport table.</td>
</tr>
<tr>
<td>1</td>
<td>Process the staging data in the ProfileImportAlt table.</td>
</tr>
</tbody>
</table>

**Return Code Values**: **profile_PluginDataImport** returns an integer which MUST be listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution</td>
</tr>
<tr>
<td>Not 0</td>
<td>Unsuccessful execution</td>
</tr>
</tbody>
</table>

**Result Sets**: **profile_PluginDataImport** MUST NOT return a result set.

### 3.1.4.10 profile_PluginDelete

The **profile_PluginDelete** stored procedure is invoked to mark a profile as deleted. **profile_PluginDelete** is defined using T-SQL syntax, as follows:

```sql
PROCEDURE profile_PluginDelete(
  @nCatalogId int,
  @DocID int
);
```

**@nCatalogId**: MUST be ignored.

**@DocID**: Document identifier of the profile to be marked as deleted.

**Return Code Values**: **profile_PluginDelete** MUST return 0.

**Result Sets**: **profile_PluginDelete** MUST NOT return any result set.
3.1.4.11  profile_PluginOnEndCrawl

If the profile_PluginOnEndCrawl stored procedure is invoked at the end of the user profile import, the protocol server MUST resolve the user profile data for users who have a login name in multiple forests. If the delete state has been set by profile_PluginReset, the protocol server MUST permanently remove user profiles that were created by import processes before the current profile import session. The protocol server MUST then clear the delete state.

If the profile_PluginOnEndCrawl stored procedure is invoked at the end of member group import, the server MUST update the recursive membership data.

profile_PluginOnEndCrawl is defined using T-SQL syntax, as follows:

```sql
PROCEDURE profile_PluginOnEndCrawl(
    @nCatalogId                  smallint,
    @nCrawlNumber                int,
    @nCrawlType                  int,
    @nStopped                    int,
    @nSuccessfulTransactions     int,
    @nErrorTransactions          int,
    @nExcludedTransactions       int,
    @nUnvisitedItems             int
);
```

@nCatalogId: MUST be ignored.

@nCrawlNumber: MUST be ignored.

@nCrawlType: Crawl type. MUST be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Full import.</td>
</tr>
<tr>
<td>non-1</td>
<td>Not a full import.</td>
</tr>
</tbody>
</table>

@nStopped: MUST be ignored.

@nSuccessfulTransactions: MUST be ignored.

@nErrorTransactions: MUST be ignored.

@nExcludedTransactions: MUST be ignored.

@nUnvisitedItems: MUST be ignored.

Return Code Values: profile_PluginOnEndCrawl MUST return 0.

Result Sets: profile_PluginOnEndCrawl MUST NOT return any result set.

3.1.4.12  profile_PluginOnStartCrawl

The profile_PluginOnStartCrawl stored procedure is invoked to register a new user profile import session. It updates the status to reflect the user profile import process is in session.

profile_PluginOnStartCrawl is defined using T-SQL syntax, as follows:
PROCEDURE profile_PluginOnStartCrawl(
    @CatID                        int,
    @bFullCrawl small             int
);

@CatID: MUST be ignored.

@bFullCrawl: user profile import type. MUST be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Incremental profile import.</td>
</tr>
<tr>
<td>1</td>
<td>Full profile import.</td>
</tr>
</tbody>
</table>

Return Code Values: profile_PluginOnStartCrawl returns an integer value that MUST be ignored.

Result Sets: profile_PluginOnStartCrawl MUST NOT return any result set.

3.1.4.13 profile_PluginReset

The profile_PluginReset stored procedure is invoked to set protocol server state to handle deleting previously imported user profiles. This state is used by the profile_PluginOnEndCrawl stored procedure.

PROCEDURE profile_PluginReset(
    @nCatalogId            int
);

@nCatalogId: MUST be ignored.

Return Code Values: profile_PluginReset MUST return 0.

Result Sets: profile_PluginReset MUST NOT return any result set.

3.1.4.14 profile_GetDeletedUserList

The profile_GetDeletedUserList stored procedure is invoked to get the list of user profiles marked as deleted. profile_GetDeletedUserList is defined using T-SQL syntax, as follows:

PROCEDURE profile_GetDeletedUserList();

Return Code Values: profile_GetDeletedUserList MUST return 0.

Result Sets: profile_GetDeletedUserList MUST return one DeletedUsers result set.

3.1.4.14.1 DeletedUsers Result Set

The DeletedUsers result set returns the list of user profile marked as deleted. The DeletedUsers result set is defined using T-SQL syntax, as follows:
NTName: Domain account of the user profile marked as deleted.

3.1.4.15  profile_SetDomainCookie

The profile_SetDomainCookie stored procedure is invoked to cache the domain cookie. profile_SetDomainCookie is defined using T-SQL syntax, as follows:

```sql
PROCEDURE profile_SetDomainCookie(
    @DataServiceName            nvarchar(50),
    @DomainName                 nvarchar(250),
    @Cookie                     image,
    @Length                     int
);
```

@DataServiceName: MUST be 'redmond-AD'.

@DomainName: Name of the domain from which to cache the cookie.

@Cookie: The domain cookie data.

@Length: The length of the domain cookie data in bytes.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>1</td>
<td>Unsuccessful execution</td>
</tr>
</tbody>
</table>

Result Sets: profile_SetDomainCookie MUST return one SetDomainCookie result set.

3.1.4.15.1  SetDomainCookie Result Set

The SetDomainCookie result set MUST contain only one row. The SetDomainCookie result set is defined using T-SQL syntax, as follows:

```sql
ERROR                    int NOT NULL;
```

ERROR: If set to 1, indicates that no row is updated; otherwise, set to 0.

3.1.4.16  profile_UpdateDataService

The profile_UpdateDataService stored procedure is invoked to update user profile service import settings. profile_UpdateDataService is defined using T-SQL syntax, as follows:

```sql
PROCEDURE profile_UpdateDataService(
```
@DataServiceName: MUST be 'redmond-AD'.

@DataServiceURL: MUST be 'redmond-AD-URL'.

@ExtraConfiguration: SourceConfiguration for the user profile service.

@Description: Description of the user profile service. This value can be NULL.

@PersonDBFormat: UserFormat of the user and manager. This information is stored in the last byte whose last 4 bits specify the UserFormat of the user and first 4 bits specify the UserFormat of the manager.

@ImportChoice: ADConfig value indicating the import type.

@OrgDataServiceName: MUST be 'redmond-AD'.

@ProfileName: MUST be 'UserProfile' or not specified.

@bDebug: MUST be ignored.

Return Code Values: profile_UpdateDataService MUST return 0.

Result Sets: profile_UpdateDataService MUST return one UpdateDataServiceProcResults result set.

### 3.1.4.16.1 UpdateDataServiceProcResults Result Set

UpdateDataServiceProcResults contains a single row with a user profile service identifier and an error code. The UpdateDataServiceProcResults result set is defined using T-SQL syntax, as follows:

@DataServiceID                    int
@Error                            int;

@DataServiceID: MUST be 1.

@Error: The error code. MUST be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The user profile service was updated successfully, and no error occurred.</td>
</tr>
<tr>
<td>1</td>
<td>The user profile service for the specified @DataServiceName does not exist or does not contain a valid ADConfigInfo value.</td>
</tr>
</tbody>
</table>
### 3.1.4.17 profile_UpdateDataServiceMap

The **profile_UpdateDataServiceMap** stored procedure is invoked to apply and persist property mapping changes for a user profile service. Where an update is requested, the server updates existing property mappings and inserts new property mappings that do not exist. Where a delete is requested, the server removes property mapping and disables import for those properties.

**profile_UpdateDataServiceMap** is defined using T-SQL syntax, as follows:

```sql
PROCEDURE profile_UpdateDataServiceMap (  
    @RemoveMappingList ntext,  
    @UpdateMappingList ntext,  
    @bDebug bit
);
```

- **@RemoveMappingList**: MappingList (section 2.2.8) to remove for a valid user profile service. The server MUST delete the specified property mappings for the profile data service and MUST disable import of these properties. The client MUST specify the Name attribute.

- **@UpdateMappingList**: MappingList to add or update for a valid user profile service. The client MUST pass references to one or more valid profiles, user profile service, and properties. The server MUST update the property mapping if it exists, and MUST insert a new property mapping if it does not exist.

- **@bDebug**: MUST be ignored.

The **profile_UpdateDataServiceMap** stored procedure first removes the mappings listed in the @RemoveMappingList parameter, and then updates the mappings listed in @UpdateMappingList parameter.

**Return Code Values**: **profile_UpdateDataServiceMap** MUST return 0.

**Result Sets**: **profile_UpdateDataServiceMap** MUST return one UpdateDataService result set.

#### 3.1.4.17.1 UpdateDataService Result Set

The UpdateDataService result set returns the error info and statistics for what was updated. The UpdateDataService result set MUST return one row. The UpdateDataService result set is defined using T-SQL syntax, as follows:

```sql
...  
ERROR       int,  
REMOVECOUNT        int,  
XMLRemoveDataServiceErr int,  
XMLRemoveMappingErr int,
...  
```

---

**Copyright © 2012 Microsoft Corporation.**

**Release:** July 16, 2012
UpdateCount int,
XMLUpdateDataServiceErr int,
XMLUpdateMappingErr int;

**ERROR:** The error code. MUST be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The properties were updated successfully, and no error occurred.</td>
</tr>
<tr>
<td>1</td>
<td>A non-NULL @RemoveMappingList value was specified.</td>
</tr>
<tr>
<td>2</td>
<td>The DataService node of the @RemoveMappingList parameter was invalid input.</td>
</tr>
<tr>
<td>3</td>
<td>The stored procedure fetched a DataService row from the @RemoveMappingList XML input.</td>
</tr>
<tr>
<td>4</td>
<td>The stored procedure successfully fetched the corresponding user profile identifier based on the ProfileName specified in the @RemoveMapping XML input.</td>
</tr>
<tr>
<td>5</td>
<td>The Mapping node of the @RemoveMappingList parameter was parsed successfully.</td>
</tr>
<tr>
<td>6</td>
<td>The stored procedure fetched a Mapping row from the @RemoveMappingList XML input.</td>
</tr>
<tr>
<td>10</td>
<td>The server finished processing the @RemoveMappingList XML input.</td>
</tr>
<tr>
<td>11</td>
<td>The stored procedure fetched a DataService row from the @UpdateMappingList XML input.</td>
</tr>
<tr>
<td>12</td>
<td>The stored procedure successfully fetched the corresponding user profile identifier based on the ProfileName specified in the @UpdateMapping XML input.</td>
</tr>
<tr>
<td>13</td>
<td>The stored procedure fetched the identifier for the user profile service from the @UpdateMappingListDataServiceName input.</td>
</tr>
<tr>
<td>14</td>
<td>The Mapping node of the @UpdateMappingList parameter was parsed successfully.</td>
</tr>
<tr>
<td>15</td>
<td>The stored procedure fetched a Mapping row from the @UpdateMappingList XML input.</td>
</tr>
<tr>
<td>16</td>
<td>The server located an existing row for an input property and attempted to update it.</td>
</tr>
<tr>
<td>17</td>
<td>The server did not locate an existing row for an input property and attempted to insert it.</td>
</tr>
</tbody>
</table>

**REMOVECOUNT:** Number of properties entries that were deleted by this stored procedure.

**XMLRemoveDataServiceErr:** Number of DataService nodes in the @RemoveMappingList XML input that contained an invalid user profile service name or profile name.

**XMLRemoveMappingErr:** Number of mapping nodes in the @RemoveMappingList XML input that were not successfully deleted. The protocol server MUST increment this value for properties that were not deleted because of an error. The protocol server SHOULD increment this value for properties that were not deleted because they did not exist. The protocol server MUST disable the deleted property such that it will no longer be imported for this user profile.

**UpdateCount:** Number of properties that were added or updated by this stored procedure.

**XMLUpdateDataServiceErr:** Number of DataService nodes in the @UpdateMappingList XML input that contained an invalid user profile service name or profile name.
**XMLUpdateMappingErr:** Number of **Mapping** nodes in the @UpdateMappingList XML input that were not successfully updated. The server MUST increment this value for properties that do not exist in the data store. The server SHOULD update this value for properties that were not updated because of data store failure.

### 3.1.4.18 profile_UpdateADConfiguration

The **profile_UpdateADConfiguration** stored procedure is invoked to update the Active Directory connection configuration setting. **profile_UpdateADConfiguration** is defined using T-SQL syntax, as follows:

```sql
PROCEDURE profile_UpdateADConfiguration(
    @DataServiceName nvarchar(50),
    @IsScheduled bit NULL,
    @ImportChoice int NULL,
    @IsImportEnabled bit NULL
);
```

@**DataServiceName**: MUST be 'redmond-AD'.

@**IsScheduled**: Flag indicating whether the import is scheduled. MUST be NULL.

@**ImportChoice**: **ADConfig** indicating the import choice.

@**IsImportEnabled**: Flag indicating whether the import is enabled. MUST be NULL.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>1</td>
<td>The Data Service is not found.</td>
</tr>
<tr>
<td>3</td>
<td>The @<strong>ImportChoice</strong> parameter contains an invalid value.</td>
</tr>
</tbody>
</table>

**Result Sets:** **profile_UpdateADConfiguration** MUST NOT return a result set.

### 3.1.4.19 membership_addRecursiveGroup

The **membership_addRecursiveGroup** stored procedure is invoked to update a member group or add a new member group if it does not exist. It also updates or adds the parent child relationship if it does not exist. **membership_addRecursiveGroup** is defined using T-SQL syntax as follows:

```sql
PROCEDURE dbo.membership_addRecursiveGroup(
    @Source uniqueidentifier,
    @DisplayName nvarchar(250),
    @MailNickName nvarchar(250),
    @Description nvarchar(1500),
    @Url nvarchar(2048),
    @SourceReference nvarchar(2048),
    @Type tinyint,
    @ChildGroupId bigint = NULL
);
```
@Source: The identifier of the member group source. MUST be 'A88B9DCB-5B82-41E4-8A19-17672F307B95'.

@DisplayName: Display name of the member group.

@MailNickname: Alternate mail name of the member group.

@Description: Description of the member group.

@Url: URL of the member group.

@SourceReference: Distinguished name (DN) of this member group.

@Type: Type of the member group. MUST be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Regular member group.</td>
</tr>
<tr>
<td>1</td>
<td>Security group with no mail.</td>
</tr>
<tr>
<td>2</td>
<td>Hidden.</td>
</tr>
<tr>
<td>4</td>
<td>Distribution list with no mail.</td>
</tr>
</tbody>
</table>

@ChildGroupId: Identifier of the child member group.

Return Code Values: membership_addRecursiveGroup MUST return 0.

Result Sets: membership_addRecursiveGroup MUST return one AddRecursiveGroup result set.

3.1.4.19.1 AddRecursiveGroup Result Set

The AddRecursiveGroup result set contains the error information and the identifier of the member group that was added or updated. The AddRecursiveGroup result set is defined using T-SQL syntax as follows:

```sql
Error          int,
GroupId        bigint;
```

Error: Error encountered during the adding or updating member group process. MUST be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>The member group has been processed before. Processing signals the external protocol handler not to unwind that member group in the recursive table. Unwinding a recursive table is the process of resolving the parent-child relationships throughout the table.</td>
</tr>
<tr>
<td>0</td>
<td>The add or update process finished successfully.</td>
</tr>
<tr>
<td>1</td>
<td>Update member group definition failed.</td>
</tr>
<tr>
<td>Value</td>
<td>Meaning</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>2</td>
<td>Failed to re-update member group after insertion failure.</td>
</tr>
<tr>
<td>3</td>
<td>Failed to insert member group definition.</td>
</tr>
<tr>
<td>4</td>
<td>Failed to insert MembershipRecursive entry.</td>
</tr>
</tbody>
</table>

**GroupId:** Identifier of the member group that has been added or updated.

### 3.1.4.20 membership_getAllMemberOf

The `membership_getAllMemberOf` stored procedure is invoked to retrieve a list of distinguished names (DN) of the member groups for a directory service. `membership_getAllMemberOf` is defined using T-SQL syntax, as follows:

```sql
PROCEDURE membership_getAllMemberOf(
    @DataSource nvarchar(155)
);
```

**@DataSource:** Name of the directory service connection.

**Return Code Values:** `membership_getAllMemberOf` MUST return 0.

**Result Sets:** `membership_getAllMemberOf` MUST return one MemberOf result set.

#### 3.1.4.20.1 MemberOf Result Set

The MemberOf result set contains a list of distinguished names for member groups. The MemberOf result set MUST return zero or more rows with one row for each member group. The MemberOf result set MUST return zero rows if the last import was incremental. The MemberOf result set is defined using T-SQL syntax, as follows:

```sql
MemberOf nvarchar(2048);
```

**MemberOf:** Distinguished name of the member groups in LDIF ([RFC2849]) format.

### 3.1.4.21 Import Status and Termination

The client MUST query server status. The protocol server MUST NOT send any messages when its state changes.

### 3.1.5 Profile Import Termination

This protocol does not support a pause and resume profile import operation.

### 3.1.6 Other Local Events

None.
3.2 User Profile Import Client Details

3.2.1 Abstract Data Model

The following describes the data and state maintained by the client. The provided data is to explain how the protocol behaves. This does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

- A list of directory services.
- Information about each directory service, which includes the following:
  - ProfileServerType, as described in section 2.2.3.
  - Group name.
  - Other properties that can be used to access the directory service, as described in section 2.2.7.
  - Document: Collection of information for a single profile.
  - Property of the document: Contains specific piece of data for the document. The data comes from directory service.
  - Property mapping: Contains property mapping information between the directory service and the protocol server.

3.2.2 Timers

The protocol client MAY provide a timer job to periodically start a full or incremental user profile import session.

The protocol client MUST have a timer job that periodically checks whether a user profile import session is finished. When the user import session is finished, the timer notifies the client to start a member group import session. For an example, see section 4.3.

3.2.3 Initialization

At least one connection to a directory service MUST be saved to the protocol server before the client can query data from the corresponding directory service.

3.2.4 Message Processing Events and Sequencing Rules

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start profile import</td>
<td>The protocol client checks whether another user profile import session is in progress. If none is in progress, the client updates the protocol server status to perform a user profile import.</td>
</tr>
<tr>
<td>Load configuration</td>
<td>The client loads the configuration that it needs to query data from data sources.</td>
</tr>
<tr>
<td>Update temporary table(s)</td>
<td>The client retrieves the data from data sources and saves it in temporary table(s).</td>
</tr>
<tr>
<td>Processing temporary table(s)</td>
<td>The client triggers the protocol server to process the temporary table and push the data to user profile store.</td>
</tr>
<tr>
<td>End profile import</td>
<td>The client triggers a protocol server call to end the user profile import process and</td>
</tr>
</tbody>
</table>
### State Transitions

There are some assumptions about state that are illustrated in the state diagram in section 3.1. These are referenced informally in section 3.1.4 but are specified formally here.

The state of the user profile import can be "Start", "Running", or "Stop".

![State Diagram](image)

**Figure 2: User profile import states**

#### 3.2.4.2 Starting Profile Import Process

The protocol client MUST query the server status before starting a new user profile import process. If there is no active user profile import session, the protocol client MUST update the protocol server status by calling `profile_LogImportStart` to start a new user profile import session.

#### 3.2.4.3 Get Data Connection Information

The protocol client MUST get data about each connection to a directory service from the server by calling the `profile_GetDataService` stored procedure. The protocol client MUST maintain the data throughout each session and use the information to access the corresponding directory service.

The client MUST be able to pull data from different types of directory services as specified in section 2.2.3.

#### 3.2.4.4 Filling and Processing Staging Data Area

The protocol client MUST insert user profile or member group data into the `ProfileImport` or `ProfileImportAlt` table. The client MUST use one row for each user profile or member group property. The client MUST then start the server processing of the staging data by calling `profile_PluginDataImport`; the client MUST specify the `@nIsAlt` parameter to specify which table the protocol server MUST process. The client MAY insert data into the other table to enable concurrency. The client MUST NOT insert the data into the table if it is not empty.
When importing from an Active Directory resource forest, the client MUST NOT fill the
AccountName and Sid properties. The client MUST copy the values to the corresponding
ResourceAccountName and ResourceSid properties in the staging data area.

3.2.4.5 Finishing Profile Import

The client MUST call profile_PluginOnEndCrawl to finalize the user profile import process.

3.2.4.6 Member Group Import

The client MUST detect an end to a full profile import session to start a member group import
session. The client MUST import member group that is recorded earlier during the user profile
import session.
4 Protocol Examples

4.1 Adding an External Directory

A protocol client adds external directory services to the user profile service by calling `profile_UpdateDataService` with a new directory service server specified in `@ExtraConfiguration`, for example:

```xml
<?xml version="1.0" encoding="utf-16"?>
<SourceConfigurationVersion="3.0">
  <ServerType="AD" Domain="corp.microsoft.com" LoginDomain="">
    <ConnectionName="corp.microsoft.com" ProviderName="">
      <AutoDiscover="Yes" SeverIncremental="Yes" name="corp-dc-05.corp.microsoft.com">
        <Groupname="Connection information">
          <Propertyname="port" value="389" />
          <Propertyname="Use Secure Sockets layer" value="No" />
          <Propertyname="Server Timeout" value="120" />
        </Group>
        <Groupname="Search Information">
          <Propertyname="UserIDAttribute" value="distinguishedname" />
          <Propertyname="Search Base" value="DC=corp,DC=microsoft,DC=com" />
          <Propertyname="User Filter" value="(&(objectCategory=Person)(objectClass=User))" />
          <Propertyname="Scope" value="SubTree" />
          <Propertyname="Page Timeout" value="120" />
          <Propertyname="Page Size" value="10" />
        </Group>
      </Server>
    </SourceConfiguration>
```

4.2 Adding a Property Mapping

To add a property mapping, the protocol client calls `profile_UpdateDataServiceMap` with `@UpdateMappingList` indicating the mapping to add. For example:

```sql
exec dbo.profile_UpdateDataServiceMap @RemoveMappingList=NULL,@UpdateMappingList=N'<?xml version="1.0" encoding="utf-16"?><MSPROFILE><DATASERVICE ProfileName="UserProfile" ServiceName="testdomain"><MAPPING PropertyName="Nickname" DataServicePropName="userPrincipalName" DataSource="datasource" AssociationName="" bRecordIdentifier="0" /></DATASERVICE></MSPROFILE>'
```

4.3 Running a Full Import

The protocol client starts by verifying that a full import is not underway. It does this by calling `profile_LogImportStart` with `@Type` set to null. The protocol server returns a result set indicating whether a user profile import is in progress. If an import is in process, the client cannot start a new import.

To start an import, the client calls `profile_GetDataService` and `profile_GetADConfiguration` with the `@DataServiceName='redmond-AD'` constant. Next, the client calls `profile_PluginOnStartCrawl` with an arbitrary `@CatID` and `@bFullCrawl` set to 1, indicating a
session type of full profile import. This is followed by a call to `profile_LogImportStart` with @Type set to 0, indicating the client will be importing users.

The client gets a record of the property mapping by calling `profile_GetDataServicePropMapping`. This tells the client what attribute changes there are and the corresponding profile properties to update.

After finding the property mapping, the protocol client imports profiles. When it finds user accounts that have been deleted in external directories, it calls `profile_PlugInDelete` with @DocId set to the profile that no longer exists in the external directory. For new or updated users, the client inserts rows in the staging area tables. The client inserts rows in batches sized to match the page size specified in the Page Size property of the profile import search settings.

After a batch of data is put in the buffer table, the client calls `profile_PlugInDataImport`. There can be more than one staging area which the protocol client and server can use alternately so that one does not have to wait for the other to finish each pass. The @nIsAlt parameter indicates the staging area that has been filled to process. The client repeats this process until it has updated all users. The member group data is also saved during this process in a separate table.

When the update is complete, the client calls `profile_SetDomainCookie` to cache an updated domain cookie. Finally, the client calls `profile_PluginOnEndCrawl` with @nCrawlType set to indicate a full import.

The client starts a member group import session via a timer event after the user profile import is finished.

### 4.4 Running an Incremental Import

An incremental import follows a process similar to a full import. Calling `profile_PluginOnStartCrawl` indicates an incremental instead of a full import in @bFullCrawl. The `profile_PluginOnEndCrawl` stored procedure uses the same parameter to indicate an incremental import. The client then calls `profile_GetDomainCookie` for any Active Directory services and uses the returned domain cookie to query Active Directory. For other services, the client updates user profiles with changes after the StartTime is returned from calling `profile_LogImportStart`.

The client begins a member group import session via a timer event after the user import is finished.

### 4.5 Stopping an Import

The protocol client can stop an import by calling `profile_LogImportStopForced` which sets the server status to Force Stop. Subsequent calls to `profile_LogImportStart` return the status @bInProcess as 0 after `profile_PluginOnEndCrawl` is called.

### 4.6 Issuing a Reset Command

An imported profile is marked as deleted in the following two cases.

The protocol client calls `profile_PluginDelete`.

During a full profile import session, the protocol server marks all previously imported user profiles as deleted because they are imported again.

The user profiles marked as deleted are stale. They remain in the database until the protocol client calls `profile_PluginReset`. The `profile_PluginReset` stored procedure does not immediately remove the stale user profiles. The protocol server permanently removes the stale user profiles during the next full profile import by calling `profile_PluginOnEndCrawl`.
4.7 Inserting Staging Data

The protocol client is responsible for querying data from directory services and inserting data. Here is an example where protocol client reads data from Active Directory and inserts data into the ProfileImport and ProfileImportAlt tables.

When the protocol client is ready to insert data into the staging area, the client instantiates an IDirectorySearch object (see [MSDN-IDirectorySearch]). After calling the SetSearchPreference method to specify a search preference, the client calls the ExecuteSearch method. The client then calls the GetNextRow method to enumerate through rows of the result and the GetColumn method to return the values for the attributes that need to be imported. The client inserts the data in the staging area, and then repeats the same process for each row.

To insert data into the staging area (the ProfileImport and ProfileImportAlt tables), the client instantiates an IRowSetFastLoad object (see [MSDN-IRowsetFastLoad]). The client calls the InsertRow method to alternately insert data directly into the ProfileImport and ProfileImportAlt tables. Each row carries data that complies with the ProfileImport (ProfileImportAlt) table definition.

The protocol client inserts data into the ProfileImport and ProfileImportAlt tables in batches. The client begins with the ProfileImport table, insert rows for the first 50 user profiles, and then calls profile_PluginDataImport with @nIsAlt set to 0. While the ProfileImport table is being processed, the client inserts another 50 user profiles into the ProfileImportAlt table, and then calls profile_PluginDataImport with @nIsAlt set to 1. The client repeats this process until all rows from the search result are processed.
5  Security

5.1 Security Considerations for Implementers

None.

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

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.
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

### A
- Abstract data model
  - client 38
  - server 17
- ADConfig type 10
- Applicability 8
- ArrayOfDataService complex type 16
- ArrayOfMapping complex type 15
- Attributes
  - detailed description 15
  - group attributes 13
  - server attributes 12
  - SourceConfiguration attributes 12

### C
- Capability negotiation 9
- Change tracking 46
- Client
  - abstract data model 38
  - filling and processing a staging data area 39
  - finishing a profile import process 40
  - getting data connection information 39
  - importing a member group 40
  - initialization 38
  - message processing 38
  - sequencing rules 38
  - starting a profile import process 39
  - timers 38
  - transitioning between states 39
- Client - message processing
  - filling and processing a staging data area 39
  - finishing a profile import process 40
  - getting data connection information 39
  - importing a member group 40
  - starting a profile import process 39
  - transitioning between states 39
- Client - sequencing rules
  - filling and processing a staging data area 39
  - finishing a profile import process 40
  - getting data connection information 39
  - importing a member group 40
  - starting a profile import process 39
  - transitioning between states 39
- Complex types
  - ArrayOfDataService complex type 16
  - ArrayOfMapping complex type 15
- Connection information properties 13

### D
- Data model - abstract
  - client 38
  - server 17
  - DataService element 14

### E
- Elements
  - DataService element 14
  - Mapping element 15
  - MSProfile element 14
  - SourceConfiguration element 12
- Events
  - local - server 37
- Examples
  - external directory 41
  - full import 41
  - import stop 42
  - incremental import 42
  - property mapping 41
  - reset command 42
  - staging data 43
- External directory example 41

### F
- Fields - vendor-extensible 9
- Full import example 41

### G
- Glossary 6
- Group attributes 13

### I
- Implementer - security considerations 44
- Import Status and Termination method 37
- Import stop example 42
- Incremental import example 42
- Index of security parameters 44
- Informative references 7
- Initialization
  - client 38
  - server 17
- Introduction 6

### L
- Local events
  - server 37

### M
- Mapping element 15
- MappingList schema
  - ArrayOfDataService complex type 16
  - ArrayOfMapping complex type 15
  - attributes 15
- DataService element 14
- Mapping element 15
- MSProfile element 14
  - overview 14
- membership_addRecursiveGroup method 35
- membership_getAllMemberOf method 37
- Message processing
Message processing - client

filling and processing a staging data area 39
finishing a profile import process 40
getting data connection information 39
importing a member group 40
starting a profile import process 39
transitioning between states 39

Messages

ADConfig type 10
ArrayOfDataService complex type 16
ArrayOfMapping complex type 15
attributes 15
collection information properties 13
DataService element 14
group attributes 13
Mapping element 15
MSProfile element 14
profile import search settings 13
ProfilePropertyBlobType type 11
ProfileServerType type 10
server attributes 12
SourceConfiguration attributes 12
SourceConfiguration element 12
SQL Data Type type 10
transport 10
UserFormat type 11

Messages - MappingList schema

ArrayOfDataService complex type 16
ArrayOfMapping complex type 15
Attributes 15
DataService element 14
Mapping element 15
MSProfile element 14
overview 14

Messages - SourceConfiguration schema
collection information properties 13
group attributes 13
overview 11
profile import search settings 13
server attributes 12
SourceConfiguration attributes 12
SourceConfiguration element 12

Methods

Import Status and Termination 37
membership_addRecursiveGroup 35
membership_getAllMemberOf 37
profile_GetADConfiguration 19
profile_GetDataService 21
profile_GetDataServicePropMapping 22
profile_GetDataTypeList 23
profile_GetDeletedUserList 30
profile_GetDomainCookie 25
profile_GetImportStart 26
profile_GetImportStopForced 27
profile_PluginDataImport 28
profile_PluginDelete 28
profile_PluginOnEndCrawl 29
profile_PluginOnStartCrawl 29
profile_PluginReset 30
profile_SetDomainCookie 31
profile_UpdateADConfiguration 35
profile_UpdateDataService 31
profile_UpdateDataServiceMap 33
ProfileImport and ProfileImportAlt Tables 19
MSProfile element 14

N

Normative references 7

O

Overview (synopsis) 8

P

Parameters - security index 44
Preconditions 8
Prerequisites 8
Product behavior 45
Profile import search settings 13
Profile import termination

server 37
profile_GetADConfiguration method 19
profile_GetDataService method 21
profile_GetDataServicePropMapping method 22
profile_GetDataTypeList method 23
profile_GetDeletedUserList method 30
profile_GetDomainCookie method 25
profile_GetImportStart method 26
profile_GetImportStopForced method 27
profile_PluginDataImport method 28
profile_PluginDelete method 28
profile_PluginOnEndCrawl method 29
profile_PluginOnStartCrawl method 29
profile_PluginReset method 30
profile_SetDomainCookie method 31
profile_UpdateADConfiguration method 35
profile_UpdateDataService method 31
profile_UpdateDataServiceMap method 33
ProfileImport and ProfileImportAlt Tables method 19
ProfileImport table 19
ProfileImportAlt table 19
ProfilePropertyBlobType type 11
ProfileServerType type 10
Properties

collection information properties 13
profile import search settings 13
Property mapping example 41

R

References 7

informative 7
normative 7
Relationship to other protocols 8
Reset command example 42

S
Schemas - MappingList
ArrayOfDataService complex type 16
ArrayOfMapping complex type 15
attributes 15
DataService element 14
Mapping element 15
MSProfile element 14
overview 14

Schemas - SourceConfiguration
connection information properties 13
group attributes 13
overview 11
profile import search settings 13
server attributes 12
SourceConfiguration attributes 12
SourceConfiguration element 12

Security
implementer considerations 44
parameter index 44

Sequencing rules
client 38
server 17

Sequencing rules - client
filling and processing a staging data area 39
finishing a profile import process 40
getting data connection information 39
importing a member group 40
starting a profile import process 39
transitioning between states 39

Server
abstract data model 17
Import Status and Termination method 37
initialization 17
local events 37
membership_addRecursiveGroup method 35
membership_getAllMemberOf method 37
message processing 17
profile import termination 37
profile_GetADConfiguration method 19
profile_GetDataService method 21
profile_GetDataServicePropMapping method 22
profile_GetDataTypeList method 23
profile_GetDeletedUserList method 30
profile_GetDomainCookie method 25
profile_LogImportStart method 26
profile_LogImportStopForced method 27
profile_PluginDataImport method 28
profile_PluginDelete method 28
profile_PluginOnEndCrawl method 29
profile_PluginOnStartCrawl method 29
profile_PluginReset method 30
profile_SetDomainCookie method 31
profile_UpdateADConfiguration method 35
profile_UpdateDataService method 31
profile_UpdateDataServiceMap method 33
ProfileImport and ProfileImportAlt Tables method 19
sequencing rules 17
timers 17
Server attributes 12
SourceConfiguration attributes 12

SourceConfiguration element 12
SourceConfiguration schema
connection information properties 13
group attributes 13
overview 11
profile import search settings 13
server attributes 12
SourceConfiguration attributes 12
SourceConfiguration element 12
SQL Data Type type 10
Staging data example 43
Standards assignments 9

T

Tables
ProfileImport 19
ProfileImportAlt 19

Timers
client 38
server 17

Tracking changes 46

Transport 10

Types
ADConfig 10
ProfilePropertyBlobType 11
ProfileServerType 10
SQL Data Type 10
UserFormat 11

U

UserFormat type 11

V

Vendor-extensible fields 9
Versioning 9