[MS-SPSCLSP3]:
SPSCrawl Stored Procedures Version 3 Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

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

Revision Summary

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

Table of Contents

1 Introduction

This document specifies the SPSCrawl Stored Procedures Version 2 Protocol. This protocol is used to read values of user profile properties for user profiles within the context of a site.

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

1.1 Glossary

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

crawl
front-end Web server
group
login name
organization
partition identifier
profile subtype
quick link
result set
return code
Security Account Manager (SAM)
service application
stored procedure
Transact-Structured Query Language (T-SQL)
user display name

The following terms are specific to this document:

organization identifier: A positive 32-bit integer that uniquely identifies an organization.

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

1.2 References

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

1.2.1 Normative References

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

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

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

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

[MS-UPSPROF2] Microsoft Corporation, "User Profile Stored Procedures Version 2 Protocol Specification".

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

1.2.2 Informative References

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

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

1.3 Protocol Overview

This protocol allows clients to read values of user profile properties for user profiles within the context of a site.

The following diagram shows data flow between protocol client and protocol server.

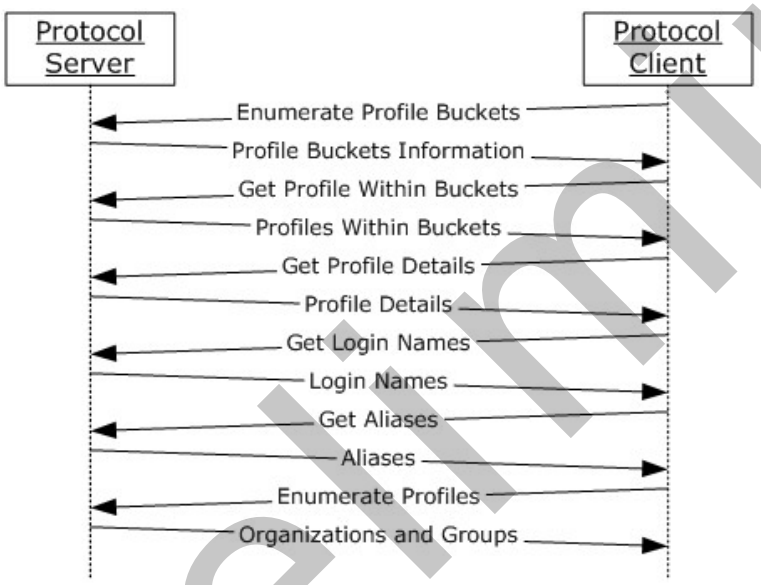

Figure 1: SPS Crawl Stored Procedure Protocol data flow between client and server

The protocol client requests the protocol server to provide a list of all buckets. After the protocol server provides information about all the buckets, the protocol client requests the server to enumerate the user profiles in each bucket. Once this information is provided by the protocol server, protocol client requests the protocol server to provide details of each user profile.

The **GetLoginNames** operation requests the protocol server to provide **login names** of all users in the specified bucket.

The **GetAliases** operation provides the aliases of all users in the specified bucket on the protocol server.

The **EnumProfiles** operation requests the protocol server to provide a list of **organizations** and **groups (2)**.

1.4 Relationship to Other Protocols

The following diagram shows the transport stack that the protocol uses:

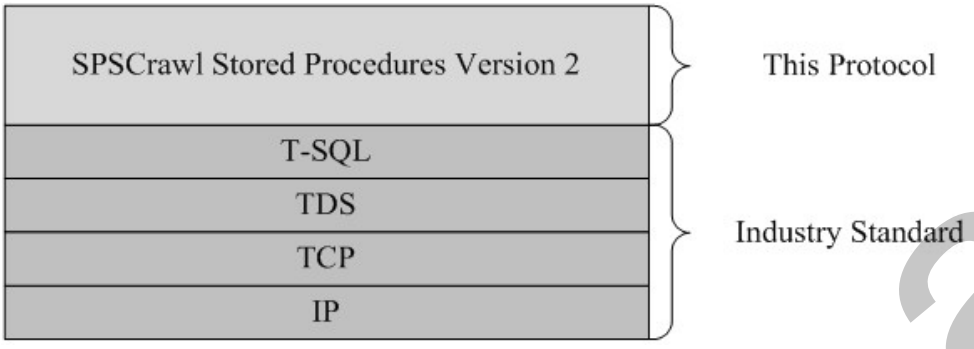

Figure 2: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

This protocol requires that a **service application** is created and is configured correctly on the protocol server.

1.6 Applicability Statement

This protocol is well suited for a client to read up to one million user profile records.

1.7 Versioning and Capability Negotiation

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

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

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

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

2.2 Common Data Types

2.2.1 Simple Data Types and Enumerations

No common simple data types or enumerations are defined in this protocol.

2.2.2 Bit Fields and Flag Structures

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

2.2.3 Binary Structures

No common binary structures are defined in this protocol.

2.2.4 Result Sets

2.2.4.1 ProfileBuckets

The **ProfileBuckets** result set MUST return one or more rows containing three columns if a user profile bucket was found. The result set MUST be empty if no user profile bucket was found. The **T-SQL** syntax for the result set is as follows.

```
BucketID int,
BucketDeleteCount int,
BucketLastModTime datetime,
PartitionID uniqueidentifier,
```

BucketID: The identifier of the user profile bucket.

BucketDeleteCount: The number of deleted records in the corresponding user profile bucket.

BucketLastModTime: The value of the most recent update on records in the corresponding user profile bucket.

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

2.2.4.2 ProfileInBucket

The **ProfileInBucket** result set returns multiple rows, each containing two columns. The result set will be empty if no user profiles were found in the user profile bucket specified by the provided BucketID parameter. The T-SQL syntax for the result set is as follows.

```
RecordID bigint,
LastUpdate datetime,
```

RecordID: The identifier of the user profile.

LastUpdate: The value of the last update on the user profile.

2.2.4.3 ProfileSubtypeResultSet

The **UserSubtypeResultSet** MUST return only one row containing three columns The T-SQL syntax for the result set is as follows.

```
ProfileSubtypeID int,
LevelToTop int,
LastUpdate datetime,
```

ProfileSubtypeID: A **profile subtype** identifier .

LevelToTop: An integer specifying the number of levels to the top of the manager chain.

LastUpdate: The value of the last update on the corresponding user profile.

2.2.4.4 UserProfileInformationResultSet

The **UserProfileInformationResultSet** returns multiple rows, each containing three columns. The result set MUST be returned second, and MUST be empty if no records were found matching the provided RecordID parameter. The T-SQL syntax for the result set is as follows.

```
PropertyID bigint,
Privacy int,
PropertyVal sql_variant,
```

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

Privacy: As defined in the Privacy Policy Type of [MS-UPSPROF2] section 2.2.1.6.

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

2.2.4.5 NTNameResultSet

The **NTNameResultSet** MUST return one or more rows containing a single column if the RecordID identifies the primary record for a user profile with multiple login names. The result set MUST be empty if either the user profile identified by **RecordID** has only one login name, or the **RecordID** specifies an optional secondary login name for a user profile. The T-SQL syntax for the result set is as follows.

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

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

2.2.4.6 GetAliasList

The **GetAliasList** result set contains multiple rows each containing three columns. The T-SQL syntax for the result set is as follows.

```
RecordID bigint,
NAME nvarchar(512),
FLAG int,
```

RecordID: The identifier of the user profile.

NAME: A value for a user profile property marked as an alias.

FLAG: A value that specifies if the NAME column value is a user display name.

Possible parameter values:

Value	Description
0	The NAME column value is not the user display name.
1	The NAME column value is the user display name.

2.2.4.7 EnumProfiles

The **EnumProfiles** record set returns multiple rows, each containing one column if profiles were found. The result set MUST be empty if no profiles were found.

```
RecordID bigint,
```

RecordID: The identifier of the profile.

2.2.4.8 QuickLinkResultSet

The **QuickLinkResultSet** MUST return one or more rows containing two columns if records were found matching the provided **RecordID**. The result set MUST be empty if no records were found matching the provided **RecordID**. The T-SQL syntax for the result set is as follows.

```
QuickLink nvarchar(250),
MemberCount bigint,
```

QuickLink: One or more **quick link** values for the user profile identified by **RecordID**.

MemberCount: The count of members in this membership group.

2.2.4.9 ColleaguesResultSet

The **ColleaguesResultSet** returns multiple rows, each containing a two columns describing the colleagues of a specified user. The result set MUST be returned fifth, and MUST be empty if no records were found matching the provided **RecordID**. The T-SQL syntax for the result set is as follows.

```
UserID uniqueidentifier,
ItemSecurity int,
```

UserID: Identifier of a colleague or a specified user.

ItemSecurity: MUST be a Privacy Type value as defined in the [MS-UPSPROF2] section 2.2.1.2.

2.2.5 Tables and Views

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

2.2.6 XML Structures

No common XML structures are defined in this protocol.

2.2.6.1 Namespaces

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

2.2.6.2 Simple Types

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

2.2.6.3 Complex Types

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

2.2.6.4 Elements

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

2.2.6.5 Attributes

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

2.2.6.6 Groups

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

2.2.6.7 Attribute Groups

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

3 Protocol Details

3.1 Common Details

None.

3.2 Server Details

3.2.1 Abstract Data Model

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

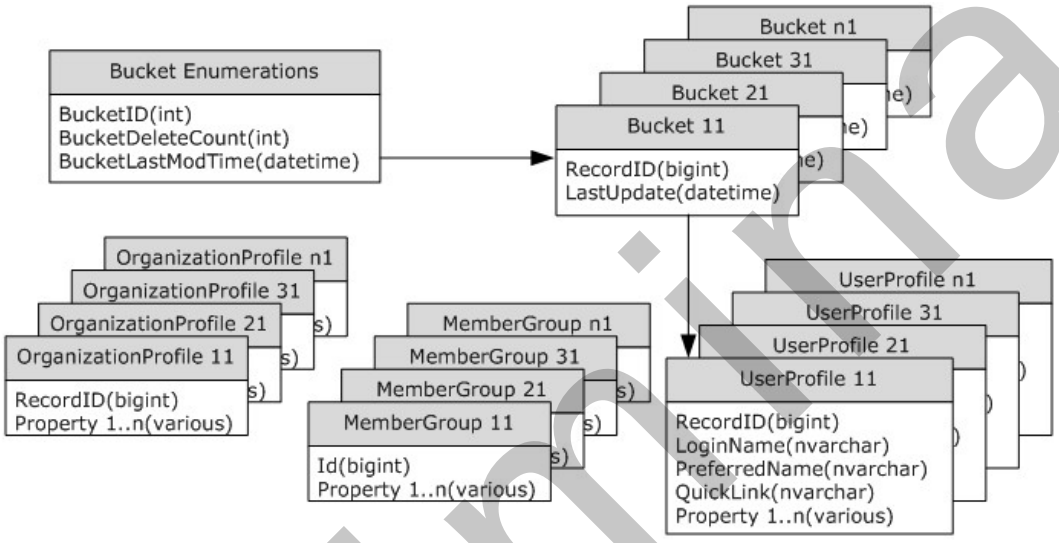

Figure 3: Abstract data model

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

Bucket Enumerations Table: A collection of entries corresponding to the table of information about buckets in the dataset relating to user profiles. A unique **BucketID** MUST identify each entry.

- **BucketID:** A unique identifier assigned to each user profile bucket.
- **BucketDeleteCount:** The number of user profiles deleted from the bucket identified by **BucketID**.
- **BucketLastModTime:** The date and time of the latest update to any user profile enumerated in the bucket identified by **BucketID**.

Bucket11 … Bucket*n1*: A collection of entries corresponding to the tables of user profile buckets in the dataset. A unique **RecordID** MUST identify each entry.

- **RecordID:** An identifier assigned to each user profile.

- **LastUpdate:** The date and time of the last update to the user profile identified by **RecordID**.

UserProfile11…UserProfile*n1*: A collection of identifiers and user profile properties for each user profile in the dataset. A unique **RecordID** MUST identify each entry.

- **RecordID:** An identifier assigned to each user profile.
- **LoginName:** The login name for the user profile identified by **RecordID**.
- **PreferredName:** The **user display name** for the user profile identified by **RecordID**.
- **QuickLink:** One or more quick link values for the user profile identified by **RecordID**.
- **Property1…Property*n***: Additional entries that MAY be defined and populated for a specific dataset implementation. The entries MAY represent values for additional identifiers and user profile properties. The procedures that support **crawl** actions pass these values on to the protocol client as described in the following sections without modifying the values.

OrganizationProfile11… OrganizationProfile*n1*: a collection of identifiers and organization profile information. A unique **RecordID** MUST identify each entry.

- **RecordID:** An identifier assigned to each organization profile.
- **Property1…Property*n*:** Additional entries that MAY be defined and populated for a specific dataset implementation.

MemberGroup11… MemberGroup*n1*: a collection of identifiers and group (2) information. A unique **Id** MUST identify each entry.

- **Id:** An identifier assigned to each group (2).
- **Property1…Property*n*:** Additional entries that MAY be defined and populated for a specific dataset implementation.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 profile_EnumProfileBuckets

The **profile_EnumProfileBuckets** stored procedure is called to get user profile bucket information.

The T-SQL syntax for the stored procedure is as follows.

```
PROCEDURE profile_EnumProfileBuckets (
@partitionID uniqueidentifier
```

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

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

@correlationId: The value MUST be set to NULL and MUST be ignored by the server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a ProfileBuckets.

3.2.5.2 profile_EnumProfileInBucket

The **profile_EnumProfileInBucket** stored procedure is called to get identifiers for user profiles contained in the specified user profile bucket.

The T-SQL syntax for the stored procedure is as follows.

```
PROCEDURE profile_EnumProfileInBucket (
@partitionID uniqueidentifier
,@BucketID int
,@correlationId uniqueidentifier = null
);
```

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

@BucketID: The identifier of the user profile bucket.

@correlationId: The value MUST be set to NULL and MUST be ignored by the server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a ProfileInBucket

3.2.5.3 profile_EnumProfileRecords

The **profile_EnumProfileRecords** stored procedure is called to get information for a specified user profile. The stored procedure MUST return three result sets in the order they are listed following.

```
PROCEDURE profile_EnumProfileRecords (
@partitionID uniqueidentifier
,@RecordID bigint
,@correlationId uniqueidentifier = null
);
```

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

@RecordID: The value of a user profile identifier.

@correlationId: The value MUST be set to NULL and MUST be ignored by the server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a QuickLinkResultSet

This stored procedure MUST return a ProfileSubtypeResultSet.

This stored procedure MUST return a ColleaguesResultSet

This stored procedure MUST return a UserProfileInformationResultSet.

This stored procedure MUST return a NTNameResultSet

3.2.5.4 profile_GetAliasList

The **profile_GetAliasList** stored procedure returns a list of user profile aliases. If **@StartTime** is **NULL**, the result set MUST contain a set of rows for all user profile aliases. If **@StartTime** contains a **datetime** value, the result set MUST contain a set of rows containing aliases for each user profile updated after **@StartTime**, and it MUST be empty if no user profiles updated after @StartTime were found. If @ **StartTime** is NULL, the result set MUST return a set of rows containing aliases for all user profiles.

The T-SQL syntax for the stored procedure is as follows.

```
PROCEDURE profile_GetAliasList (
@StartTime datetime = null
,@LastUpdate datetime OUTPUT
,@correlationId uniqueidentifier = null
);
```

@StartTime: A value for filtering by a specific starting datetime.

@LastUpdate: The most recent update date and time among all user profiles.

@correlationId: The value MUST be set to NULL and MUST be ignored by the server.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a GetAliasList.

3.2.5.5 profile_EnumProfiles

The **profile_EnumProfiles** stored procedure is called to get a list of **Organization** or **Group** profile identifiers between a specified range of identifiers.

```
PROCEDURE profile_EnumProfiles (
@partitionID uniqueidentifier
,@ProfileTypeID smallint
,@BeginID bigint
,@EndID bigint
,@MINID bigint OUTPUT
,@MAXID bigint OUTPUT
,@correlationId uniqueidentifier = null
```

```
);
```

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

@ProfileTypeID: The value of a profile type identifier. This MUST be set to one of the values in the following table.

Value	Description
2	Organization Type
3	Group Type

@BeginID: A value specifying the beginning identifier in the range of identifiers returned by the result set. All identifiers in the result set MUST be greater than or equal to this value.

@EndID: A value specifying the ending identifier in the range of identifiers returned by the result set. All identifiers in the result set MUST be less than or equal to this value.

@MINID: Output value for the minimum identifier among all profiles.

@MAXID: Output value for maximum identifier among all profiles.

@correlationId: The value MUST be set to NULL and MUST be ignored by the server.

Return Values: An integer which MUST be 0.

Result Sets:

For the following combination of parameters,

@ProfileTypeID: 2

This stored procedure MUST return a EnumProfiles.

For the following combination of parameters,

@ProfileTypeID: 3

This stored procedure MUST return a EnumProfiles.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Client Details

None.

3.3.1 Abstract Data Model

None

3.3.2 Timers

None.

3.3.3 Initialization

None.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

None.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

4 Protocol Examples

A protocol client uses the five stored procedures described in this document to crawl a dataset that contains user profiles to create one or more indices of that data. The protocol client may crawl subsets of the dataset based on user profile buckets or crawl the entire dataset.

4.1 Crawl Example Using User Profile Buckets

To crawl based on user profile buckets, the protocol client first uses **profile_EnumProfileBuckets** with a valid **partition identifier** to determine the range of user profile bucket identifiers, called **BucketIDs** in this example. The result set from **profile_EnumProfileBuckets** also contains the most recent update date and time for all of the user profiles in each bucket, and the protocol client may use information cached from previous crawls to ignore buckets that contain only user profiles unchanged since the last crawl.

Example of parameters when calling **profile_EnumProfileBuckets:**

Parameter	Value
@partitionID	0C37852B-34D0-418E-91C6-2AC25AF4BE5B
@correlationId	NULL

This call would return the result set described in the following table.

BucketId	BucketDeleteCount	BucketLastModTime	PartitionID
1	0	2010-01-05 23:17:21.050	0C37852B-34D0-418E-91C6-2AC25AF4BE5B

The protocol client then uses one of the **BucketIDs** and its **PartitionID** as the input parameters for a call to **profile_EnumProfileInBucket**, which returns a result set containing an identifier for each user profile in the user profile bucket, called the **RecordID** in this example. The procedure also returns the date and time of the most recent update for each user profile. The protocol client may use information cached from previous crawls to ignore user profiles unchanged since the last crawl.

Example of parameters when calling **profile_EnumProfileInBucket.**

Parameter	Value
@partitionID	0C37852B-34D0-418E-91C6-2AC25AF4BE5B
@BucketId	The BucketId for user 1
@correlationId	NULL

This call would return the result set described in the following table.

RecordId	LastUpdate
1	2009-12-21 18:14:52.387
2	2010-01-05 14:30:01.203
3	2009-12-21 21:23:40.587

The **RecordID** identifies each user profile in the dataset. The protocol client can use a **RecordID** as an input to **profile_EnumProfileRecords** to get several sets of user profile property values for the user profile for indexing, or to retrieve user profile property values for a user profile previously indexed.

The protocol client creates its indices by making multiple calls to **profile_EnumProfileRecords** for all **RecordIDs** it identifies as appropriate for indexing.

4.2 Crawl Example Using the Full Dataset

The protocol client may crawl the full dataset without first making calls to the stored procedures that support user profile buckets. The protocol client may choose to do that if it has existing indices on the dataset and needs to identify any user profiles that require re-indexing. The protocol client may also crawl the dataset to get alias values for one or more user profiles without the overhead required for a call to **profile_EnumProfileRecords**.

Example of parameters when calling **profile_EnumProfileRecords**.

Parameter	Value
@partitionID	0C37852B-34D0-418E-91C6-2AC25AF4BE5B
@RecordId	The RecordId for user 1
@correlationId	NULL

The call would return the result set consisting of 5 tables. For clarity only the following tables are shown.

ProfileSubtypeID	LevelToTop	LastUpdate
1	0	2009-12-21 18:14:52.387

PropertyID	Privacy	PropertyVal
1	1	CEDFB4DD-B95B-4C56-951D-855420F4D6AC
2	NULL	0x0105000000000000515000000271A6C07352F372AAD20FA5B68450C00
3	1	northamerica\cynthist

4.2.1 Crawling to Request User Profile Alias Values

The protocol client can use **profile_GetAliasList** to get a result set of the user profile alias values for a subset of user profiles, or for all user profiles in the data set. The return from **profile_GetAliasList** flags the user display name in each set of aliases for each user profile.

The protocol client supplies a **StartTime** as an input to **profile_GetAliasList** and the procedure returns aliases for only those user profiles updated after **StartTime**. The protocol client can selectively update its indices by using a **StartTime** based on the update times for recently cached indices. Using a NULL **StartTime** requests alias values for all user profiles.

The protocol client also provides a **LastUpdate** output parameter to **profile_GetAliasList** and the procedure uses **LastUpdate** to return the date and time of the most recently updated user profile in the dataset. The protocol client can use the returned **LastUpdate** value to verify that it has all expected updates indexed.

Example of parameters when calling **profile_GetAliasList.**

Parameter	Value
@StartTime	NULL
@CorrelationId	NULL

This call would return the result set described in the following table.

RecordID	NAME	FLAG
2	redmond\reedpa	1
2	redmond\reedpa	0
3	northamerica\speschka	1

This call will return the value shown in the following table.

LastUpdateTime
2008-02-12 23:48:30.927

4.2.2 Crawling to Request Organization or Group Identifiers

The protocol client can use **profile_EnumProfiles** to get a result set of **organization identifiers** or group identifiers.

The protocol client supplies **BeginID** and **EndID** as the lower and upper bound of identifiers to retrieve. The protocol client also provides **ProfileTypeID** to retrieve either Organization or Group identifiers, as well as **MinID** and **MaxID** output parameters. Upon returning, the procedure will set **MinID** to the smallest **RecordID** that exists, and **MaxID** to the largest **RecordID** that exists.

The protocol client can use multiple calls to **profile_EnumProfiles** to crawl all profile identifiers. On the first call, the protocol client can supply **BeginID** with 0 and **EndID** with an upper bound. The protocol client adjusts the bounds of **BeginID** and **EndID** between each call, using the **MaxID** output to determine when the ending identifier is retrieved and no more calls are needed. The protocol client can also use the **MinID** output after the first call to skip any initial gap in identifiers.

Example of parameters when calling **profile_EnumProfiles.**

Parameter	Value
@partitionId	0C37852B-34D0-418E-91C6-2AC25AF4BE5B
@ProfileTypeId	2
@BeginID	1
@EndId	7

Parameter	Value
@MINID	0
@MAXID	12
@CorrelationId	NULL

This call would return the result set shown in the following tables.

RecordID
1

@MINID	@MAXID
1	1

Return Value
0

5 Security

5.1 Security Considerations for Implementers

This protocol supports the SSPI and SQL Security Authentication Methods with the Protocol Server role. These authentication methods are described in [MS-TDS].

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

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

- Microsoft® SharePoint® Server 2013 Preview
- Microsoft® SQL Server® 2008 R2 SP1
- Microsoft® SQL Server® 2012

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