[MS-UPSSYNC]:
User Profile Synchronization Stored Procedures 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's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). 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

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

Table of Contents

1 Introduction

In an example deployment, individual Web sites will each contain information about the people who contribute content and visit those Web sites (for example, first name, last name, and workgroup) but that data may not always be up to date. The definitive source for data about people is often centralized. Likewise, the individual Web sites will acquire site-specific knowledge relating to which people own which data on that Web site. That data could benefit from centralized storage.

This document specifies a protocol for bi-directional synchronization to:

- Distribute centralized data about people such that it is available at each Web site
- Acquire distributed data about what content people own across multiple Web sites and centralize it.

1.1 Glossary

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

GUID
Security Support Provider Interface (SSPI)

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

back-end database server
change log
change token
content database
member
member group
membership
result set
return code
security group
security principal
site
site collection
stored procedure
Structured Query Language (SQL)
SystemID
user information list
user profile
user profile store

The following terms are specific to this document:

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

1.2 References

1.2.1 Normative References

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

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

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

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

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

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

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

1.2.2 Informative References

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

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

[MS-WSSTS] Microsoft Corporation, "Windows SharePoint Services Technical Specification"

1.3 Protocol Overview (Synopsis)

This protocol specifies the communication between a protocol client and a **user profile store** (protocol server) for the purpose of bi-directional synchronization. While ultimately the bi-directional synchronization of **user profile** and **security principal** data occurs between the user profile store and **content databases**, this protocol does not cover communication between the protocol client and a content database. Rather, in order for the synchronization to be successful, the protocol client is responsible for using [MS-WSSFO] and [MS-WSSTS] to read and write data to and from the content database during the synchronization.

This protocol is designed with the goal to be transactional from the perspective of the protocol server for the synchronization of a given **site collection**. In other words, the protocol is designed in such a way that a situation where the user profile store contains the results of a partially successful synchronization for a given site collection can be avoided.

1.4 Relationship to Other Protocols

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

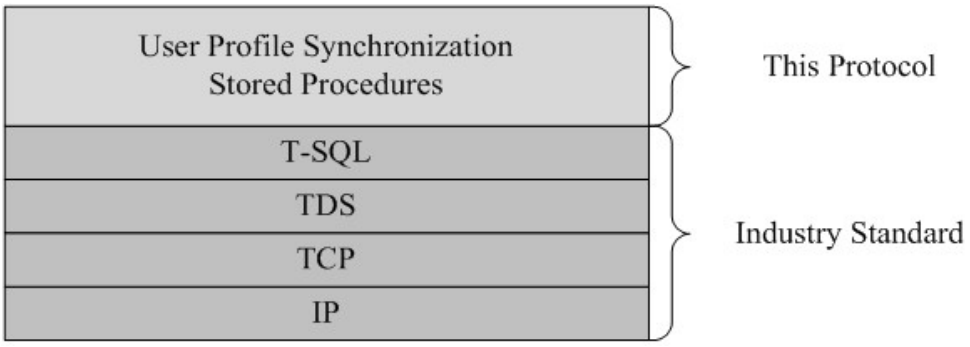

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

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

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

1.6 Applicability Statement

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

- 5 million user profiles
- On average 100 **member group**s per user profile
- 50,000 site collections, each containing an average of 100 security principals which will need to be synchronized with the user profile data.

An additional one million site collections with fewer than 10 security principals which will need synchronization with the user profile data.

This protocol is intended for use by protocol clients and protocol servers that are both connected by high-bandwidth, low latency network connections.

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] is the transport protocol used to call the stored procedures, query SQL views or SQL tables, return result codes, and return **result sets**.

2.2 Common Data Types

None.

2.2.1 Simple Data Types and Enumerations

None.

3 Protocol Details

This protocol allows protocol servers to perform implementation-specific localization of text in various messages. Except where specified, the localization of this text is an implementation-specific behavior of the protocol server and not significant for interoperability.

3.1 Profile Synchronization Server Details

The following diagram shows the possible state transitions for this protocol.

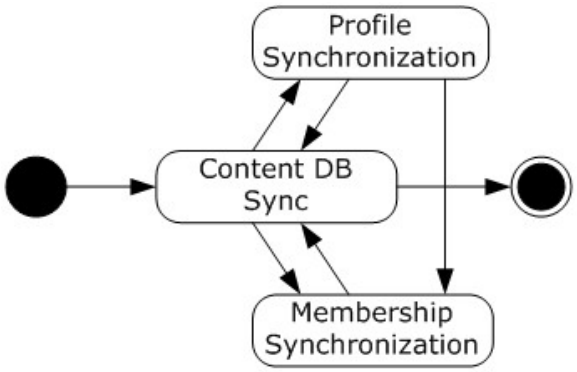

Figure 2: Profile synchronization state transition diagram

Each line represents a possible state transition. As the content database is synchronized with the user profile store, each site collection is synchronized as needed. State transitions occur as a consequence of which stored procedures are called while in a particular state. Only a limited set of stored procedures are allowed to be called while the protocol is in each state.

The following column headers in Table 1 are used to represent the synchronization states shown in the previous diagram, titled Profile synchronization state transition diagram:

- Initial – an initial state
- Content DB – Content DB Sync
- Profile – Profile Synchronization
- Membership – Membership Synchronization

The "New State" column indicates the state that the protocol is in after calling the stored procedure listed in the "Stored Procedure" column. Note that while there is no column for the "Final" state, it is referenced in the table and refers to the final state in the previous diagram (Profile synchronization state transition diagram).

Synchronization stored procedures not containing an "OK" in the box corresponding to a particular stored procedure and the state of the protocol MUST NOT be called by the protocol client while the protocol is in that state.

Stored Procedure	Initial	ContentDB	Profile	Membership	New State
profilesynch_CleanUpDeletedSites	OK	OK			

Stored Procedure	Initial	ContentDB	Profile	Membership	New State
profilesynch_DeleteInfoForDB	OK				
profilesynch_FailedSiteChangeLogConsumption			OK	OK	ContentDB
profilesynch_GetOldDBs	OK				
profilesynch_GetSitesToSynch		OK			
profilesynch_GetUnregisteredSites		OK			
profilesynch_MS_AddUsersToGroup				OK	
profilesynch_MS_AddUserToGroup				OK	
profilesynch_MS_DeleteGroup				OK	
profilesynch_MS_DeleteUserFromGroup				OK	
profilesynch_MS_DeleteWeb				OK	
profilesynch_MS_GetGroupsForSite		OK	OK		Membership
profilesynch_MS_UpdateWeb			OK	OK	Membership
profilesynch_PrepareToMove	OK				
profilesynch_RegisterSitesToSynch		OK			
profilesynch_ScheduleFullSiteSynch	OK				
profilesynch_StartContentDBSynch	OK				ContentDB
profilesynch_StartFullSiteSynch		OK			Profile
profilesynch_SuccessfulContentDBSynch		OK			Final
profilesynch_SuccessfulSiteChangeLogConsumption			OK	OK	ContentDB
profilesynch_SuccessfulSiteProfilePush			OK	OK	Membership
profilesynch_sweep_GetDBToken	OK				
profilesynch_sweep_UpdateDBToken	OK				
profilesynch_UnregisterAllSites		OK			
profilesynch_US_AddProfilesToSynch			OK		
profilesynch_US_IncrementalSynch		OK	OK		Profile

Table 1: Relationship between stored procedures calls and state transitions

The protocol client MUST send the initialization message specified in section 3.1.3 prior to calling a stored procedure that would result in a transition to profile synchronization or membership synchronization.

In some instances it will be convenient for the protocol client to call synchronization stored procedures from the initial state and then terminate the connection without entering another state. However, protocol clients MUST NOT terminate the connection while in a state other than initial or final.

Once the content database synchronization state has been entered in an instance of the protocol, the same site collection MUST be used when calling all legal stored procedures that can be called from the profile or membership synchronization states. The protocol client MUST NOT leave the content database synchronization state and call one legal stored procedure with one site collection and another legal stored procedure with another site collection. To reference a different site collection after the first has been referenced, the state MUST first be changed to content database synchronization.

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that a 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.

3.1.1.1 User Profile Data

The user profile store contains a list of user profiles. The following user profile data is used in this protocol:

- **LastChanged:** A date associated with each user profile that indicates when it was last updated (for example, a job title changed).
- **SiteMembershipList:** The list of **site** memberships associated with each user profile.

3.1.1.2 User Data

The protocol server maintains information about all the security principals that are encountered during synchronization and the mapping of those security principals to user profiles. This data includes:

- **SiteID:** The site collection the security principal belongs to.
- **WSSID:** The identifier of the security principal.
- **SID**: The **SystemID** for the security principal identified by **WSSID**.

3.1.1.3 Membership Data

This data contains relationships between security principals and **security groups** and between security groups and sites.

3.1.1.3.1 UserGroup Data

The UserGroup data tracks which security principals are in which security groups. This data includes:

- **SiteID**: The site collection for a particular security group.
- **GroupID**: The identifier of the security group.
- **WSSID**: The security principal whose security group is **GroupID**.

3.1.1.3.2 GroupSite Data

The GroupSite data tracks the member groups for sites. This data includes:

- **SiteID**: The site collection for this security group.
- **GroupID**: The identifier of the security group.
- **WebID**: The site whose members group is **GroupID**.

This data on relationships is sufficient to calculate the full **SiteMembershipList** for all user profiles in the user profile data.

3.1.1.4 Staging Data

The Staging Data stores pending changes for other parts of the abstract data model for a single site collection. Those pending changes do not take effect for the data relating to that site collection until a flush operation is performed. The order in which these pending operations are performed matters in that when a flush occurs the last update wins.

The following operations on the staging data are defined:

- **User Data** operations
- **DeleteUser** (*U*): Upon flush, **User Data** records will be deleted as needed to represent that security principal U is no longer a security principal for which synchronization is required.
- **Membership Data** operations
- **AddSiteGroupRel** (*S, G*): Upon flush, a **GroupSite** records will be added as needed to represent that security group G is the members group of site S.
- **DeleteSite**(*S*): Upon flush, all records for site S will be deleted from the **UserGroup Data** and **GroupSite Data**.
- **UpdateSiteGroupRel** (*S, GNew, GOld*): Upon flush, **GroupSite** records will be added and deleted as necessary to represent that security group GNew has become the members group of site S and that GOld (the former members group of site S) is no longer the members group of site S.
- **AddUserGroupRel**(*U,G*): Upon flush, **UserGroup** records will be added as necessary to represent that security principal U is in security group G.
- **DeleteUserGroupRel**(*U,G*): Upon flush, **UserGroup** records will be deleted as necessary to represent that security principal U is not in security group G.

3.1.1.5 Synchronization Data

The synchronization data is data about what synchronizations were performed, for example, if they were successful, how long they took, and so on.

- **Site Collection Synchronization Data:** Records containing data about site collections that have synchronized or are registered to synchronize. This data includes:
- **ContentDBID**: The identifier of the content database that this site collection belongs to.
- **Registered:** Specifies whether this site collection is registered for synchronization
- **Moving**: Specifies whether this site collection is scheduled to be moved to a different content database
- **MovingDeleted:** Specifies whether this site collection was found to no longer exist on the content database during a time interval where **Moving** is set to true.
- **LastSynch**: Specifies when this site was last successfully synchronized
- **LastChangeSynchSuccess**: Specifies whether the last synchronization attempt with the site collection was successful. Specifically this refers to the success or failure of operations occurring while in the **membership** synchronization state.
- **ChangeToken**: A **change token** to be used during subsequent synchronizations.
- **SchemaVersion:** A value indicating how up to date the schema of the **user information list** is.
- **Content Database Synchronization Data:** Records containing data about content databases that are synchronized with the user profile store. This data includes:
  - **ChangeTokenFull** - A change token to be used during subsequent synchronizations of the content database.
  - **ChangeTokenQuick** - A change token to be used during subsequent "quick" synchronizations of the content database.
  - **StartSynch** - The start time of the last synchronization of the content database.
  - **EndSynch** - The end time of the last synchronization of the content database.

3.1.1.6 Unambiguous Site Collection References

Because a site collection identifier is expected to be unique but can move from one content database to a different content database, the protocol server will maintain data indicating the content database in which each site collection currently resides. Note that a mechanism (section 3.1.4.14) exists in the protocol that allows the protocol client to notify the protocol server that the site collection has moved or is about to move to a new content database.

Each site collection is expected to reside in exactly one content database at a time. An implication of this is that if two site collections have the same identifier but are contained in different content databases, only the data for the first site collection encountered by the protocol will be stored.

3.1.2 Timers

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

3.1.3 Initialization

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

Prior to executing a stored procedure that will change the protocol state to membership synchronization or profile synchronization, the protocol client MUST send the following.

```
CREATE TABLE #ProfSynchGroupWebAdds (
    [WebID] [uniqueidentifier] NOT NULL,
    [GroupID] [int] NOT NULL,
    UNIQUE CLUSTERED
    (
        [WebID]
    )
)

CREATE  INDEX [IX_GroupWebAdds_GroupID] ON [dbo].[#ProfSynchGroupWebAdds] ([GroupID]) ON
[PRIMARY]

CREATE TABLE #ProfSynchWebDeletes (
    [WebID] [uniqueidentifier] NOT NULL,
    UNIQUE CLUSTERED
    (
        [WebID]
    )
)

CREATE TABLE #ProfSynchGroupWebMoves (
    [WebID] [uniqueidentifier] NOT NULL,
    [SourceGroupID] [int] NOT NULL,
    [TargetGroupID] [int] NOT NULL,
    UNIQUE CLUSTERED
    (
        [WebID]
    )
)

CREATE TABLE #ProfSynchUserGroupAdds (
    [GroupID] [int] NOT NULL,
    [WssId] [int] NOT NULL
)

CREATE  CLUSTERED INDEX CX_UserGroupAdds_Group ON [dbo].[#ProfSynchUserGroupAdds](GroupID,
WssId)

CREATE TABLE #ProfSynchUserGroupDeletes (
    [GroupID] [int] NOT NULL,
    [WssId] [int] NOT NULL
)

CREATE  CLUSTERED  INDEX [CX_UserGroupDeletes_Group] ON
[dbo].[#ProfSynchUserGroupDeletes]([GroupID], [WssId])

CREATE TABLE #ProfSynchSourceGroupMembership (
    [GroupID] [int] NOT NULL,
    [WssId] [int] NOT NULL
)
```

```
CREATE  CLUSTERED  INDEX [CX_SourceGroupMembership_Group] ON
[dbo].[#ProfSynchSourceGroupMembership]([GroupID], [WssId])

CREATE TABLE #ProfSynchTargetGroupMembership (
    [GroupID] [int] NOT NULL,
    [WssId] [int] NOT NULL
)

CREATE  CLUSTERED  INDEX [CX_TargetGroupMembership] ON
[dbo].[#ProfSynchTargetGroupMembership]([GroupID], [WssId])
```

Behaviors:

The server MUST initialize Staging Data to have no pending operations.

3.1.4 Message Processing Events and Sequencing Rules

The following stored procedures are available as part of the protocol using standard T-SQL (Transact-Structured Query Language) stored procedure calls.

Stored Procedure	Description
profilesynch_CleanUpDeletedSites	Deletes synchronization data in the user profile store relating to one or more site collections that no longer exist in the content database currently being synchronized.
profilesynch_DeleteInfoForDB	Deletes synchronization data in the user profile store relating to a content database.
profilesynch_FailedSiteChangeLogConsumption	Changes user profile store data about synchronization for the specified site collection to indicate that it failed its last attempt.
profilesynch_GetOldDBs	Returns a list of content databases that have not been synchronized in a specified number of days.
profilesynch_GetSitesToSynch	Returns a list of site collections for which security principal data is out of date and therefore synchronization is required.
profilesynch_GetUnregisteredSites	Returns a list of site collections which are not registered for synchronization.
profilesynch_MS_AddUsersToGroup	Changes the user profile store to store information that one or more security principals are in a security group on the site collection that is currently being synchronized.
profilesynch_MS_AddUserToGroup	Changes the user profile store to store information that a security principal is a member of a security group on the site collection that is currently being synchronized.
profilesynch_MS_DeleteGroup	Deletes data in the user profile store regarding membership of a security group on the site collection that is currently being synchronized.

Stored Procedure	Description
profilesynch_MS_DeleteUserFromGroup	Deletes data in the user profile store regarding a security principal being a member of a security group on the site collection that is currently being synchronized.
profilesynch_MS_DeleteWeb	Deletes data in the user profile store regarding a site in the site collection that is currently being synchronized.
profilesynch_MS_GetGroupsForSite	Returns the list of security group**s** in the site collection that is currently being synchronized that the user profile store has data for already.
profilesynch_MS_UpdateWeb	Changes the data that the user profile store has regarding a site in the site collection that is currently being synchronized.
profilesynch_PrepareToMove	Changes data in the user profile store so that a site is expected to become associated with a different content database than it has been previously.
profilesynch_RegisterSitesToSynch	Changes data in the user profile store to know about this site and expect to synchronize it going forward.
profilesynch_ScheduleFullSiteSynch	Changes data in the user profile store to record that the next time the specified site collection synchronizes, it should do a full replication of data rather than trying to determine what changed since the last synchronization.
profilesynch_StartContentDBSynch	The protocol client is about to begin the synchronization of the specified content database with this user profile store.
profilesynch_StartFullSiteSynch	The protocol client is about to send all of the security principal data for the specified site collection in the content database.
profilesynch_SuccessfulContentDBSynch	The protocol client has finished sending data for the specified content database with no unrecoverable errors.
profilesynch_SuccessfulSiteChangeLogConsumption	The protocol client has finished sending data about the security principal after consuming the change log for the specified site collection and encountered no unrecoverable errors**.**
profilesynch_SuccessfulSiteProfilePush	The protocol client has successfully received all user profile data and replicated it to the security principal for the specified site collection.
profilesynch_sweep_GetDBToken	Returns the change token that represents the last point of synchronization with a particular content database**.**
profilesynch_sweep_UpdateDBToken	Changes the change token that represents the last point of synchronization with a particular content database**.**

Stored Procedure	Description
profilesynch_UnregisterAllSites	Marks all site collections in the content database that is currently being synchronized as no longer registered for synchronization.
profilesynch_US_AddProfilesToSynch	Records that the specified security principals exist on the site collection currently being synchronized and the matching user profile data.
profilesynch_US_IncrementalSynch	Returns profile data about security principals whose corresponding profiles have changed since the last synchronization with that site collection took place.

Return Code Values and Result Sets:

Unless noted otherwise in the following stored procedures descriptions, all synchronization stored procedures MUST return an integer **return code** which is listed in the following table.

Value	Description
0	The request was finished successfully.
Nonzero	The request was unable to be finished successfully.

Alternatively, the server can indicate an error by returning an error as described in [MS-TDS], section 2.2.2.6.

Unless otherwise noted in the stored procedure descriptions in the subsections that follow, all stored procedures MUST NOT return any result sets. In the case where result sets are specified for a stored procedure, the protocol client MUST ignore those result sets if a nonzero return code is sent. Additionally, the protocol server MUST avoid returning result sets in the cases where a nonzero return code is being sent.

3.1.4.1 profilesynch_CleanUpDeletedSites

The **profilesynch_CleanUpDeletedSites** stored procedure is called to request deletion of all synchronization data in the user profile store for up-to ten site collections. This could occur because the protocol client has determined that those site collections no longer exist in the content database specified by the input parameter.

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

```
PROCEDURE profilesynch_CleanUpDeletedSites(
    @ContentDBID       uniqueidentifier,
    @SiteID0           uniqueidentifier,
    @SiteID1           uniqueidentifier = '00000000-0000-0000-0000-000000000000',
    @SiteID2           uniqueidentifier = '00000000-0000-0000-0000-000000000000',
    @SiteID3           uniqueidentifier = '00000000-0000-0000-0000-000000000000',
    @SiteID4           uniqueidentifier = '00000000-0000-0000-0000-000000000000',
    @SiteID5           uniqueidentifier = '00000000-0000-0000-0000-000000000000',
    @SiteID6           uniqueidentifier = '00000000-0000-0000-0000-000000000000',
    @SiteID7           uniqueidentifier = '00000000-0000-0000-0000-000000000000',
    @SiteID8           uniqueidentifier = '00000000-0000-0000-0000-000000000000',
    @SiteID9           uniqueidentifier = '00000000-0000-0000-0000-000000000000'
);
```

@ContentDBID: The identifier of a content database**.**

@SiteID#: The identifier of a site collection in the content database identified by the *@ContentDBID***.** There are 10 parameters numbered from 0 to 9, each of which optionally identifies a site collection.

The following actions MUST occur for each *@SiteID#* when the protocol server processes this request:

- In the case that the **Moving** field in the **Site Collection Synchronization Data** record corresponding to *@SiteID#* is set to true, the **MovingDeleted** field MUST also be set to true for that record.
- If the **Moving** field is not set to true, all data in the user profile store relating to the synchronization of the site collection identified by *@SiteID#* in the content database identified by *@ContentDBID* MUST be deleted. This includes **Site Collection Synchronization Data, User Data, Membership Data,** and **SiteMembershipList** records.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.2 profilesynch_DeleteInfoForDB

The **profilesynch_DeleteInfoForDB** stored procedure is called to request deletion of all synchronization data in the user profile store relating to a content database.

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

```
PROCEDURE profilesynch_DeleteInfoForDB(
    @ContentDBID              uniqueidentifier
);
```

@ContentDBID: Identifies the content database where the synchronization data should be deleted.

Based on **Site Collection Synchronization Data**, the set of site collections in the content database (identified by *@ContentDBID***)** should be identified. Next, all data in the user profile store relating to the synchronization of those site collections MUST be processed in a manner equivalent to calling the **profilesynch_CleanUpDeletedSites** stored procedure (*@ContentDBID, SiteID*) once for each site collection **SiteID**.

Additionally, **Content Database Synchronization Data** relating to the specified content database MUST be deleted with the exception of **ChangeTokenQuick** field which MAY be deleted.

If the value of *@ContentDBID* is null or does not match any records in the **Site Collection Synchronization Data** or **Content Database Synchronization Data**, then no data is to be modified.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.3 profilesynch_FailedSiteChangeLogConsumption

The **profilesynch_FailedSiteChangeLogConsumption** stored procedure communicates that the protocol client experienced a failure while attempting to synchronize the specified site collection**.**

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

```
PROCEDURE profilesynch_FailedSiteChangeLogConsumption(
    @ContentDBID              uniqueidentifier,
    @SiteID                   uniqueidentifier
);
```

@ContentDBID: The identifier of a content database.

@SiteID: The identifier of a site collection in the content database identified by *@ContentDBID* that experienced a synchronization failure.

The protocol server MUST update the **LastChangeSynchSuccess** field in the **Site Collection Synchronization Data** to false for the failing site collection. Additionally, the server MUST delete all **Staging Data** in preparation for the synchronization of subsequent site collections. The **Staging Data** MUST NOT be flushed because the calling of this stored procedure indicates that the synchronization failed.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.4 profilesynch_GetOldDBs

The **profilesynch_GetOldDBs** stored procedure requests a list of content databases that have not been synchronized in a specified number of days.

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

```
PROCEDURE profilesynch_GetOldDBs(
    @Days                     int
);
```

@Days: The threshold number of days.

The protocol server MUST consult the **StartSynch** and **EndSynch** fields from the **Content Database Synchronization Data**. The most recent date of these two is **LastSynch**.

The protocol server MUST generate the current time (on the protocol server) and subtract the number of days specified by *@Days*. The protocol server MUST return a row for each content database identified that has a **LastSynch** value that is prior to the value obtained through subtraction.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: MUST return the following result set:

3.1.4.4.1 Old Database Result Set

The T-SQL syntax for the result set is as follows.

```
ID              uniqueidentifier,
LastSynch       DateTime;
```

ID: The identifier of the content database.

LastSynch: The most recent protocol server generated UTC from when the content database ID either finished or started synchronizing.

3.1.4.5 profilesynch_GetSitesToSynch

The **profilesynch_GetSitesToSynch** stored procedure is called to request a list of site collections in the specified content database that are due for synchronization.

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

```
PROCEDURE profilesynch_GetSitesToSynch(
    @ContentDBID        Uniqueidentifier
);
```

@ContentDBID: The identifier of a content database.

The result set MUST contain a row for each record in the **Site Collection Synchronization Data** where the **Moving** field is set to false and the **ContentDBID** field matches the *@ContentDBID* parameter.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: MUST return the following result set:

3.1.4.5.1 Sites to Synchronize Result Set

The T-SQL syntax for the result set is as follows.

```
ContentDBID                 uniqueidentifier,
SiteID                      uniqueidentifier,
LastSynch                   DateTime,
ChangeToken                 ntext,
SchemaVersion               int,
LastChangeSynchSuccess      bit,
Moving                      bit,
MovingDeleted               bit,
Registered                  bit,
HasProfileChanges           bit;
```

ContentDBID: The identifier for the content database. This value MUST match *@ContentDBID*.

SiteID: The identifier for the site collection. This value MUST identify a site collection in the content database **ContentDBID** that needs to be synchronized.

LastSynch: The date on which the site collection **SiteID** was last successfully synchronized. The **LastSynch** field from the **Site Collection Synchronization Data** for **SiteID** MUST be returned.

ChangeToken: The **ChangeToken** field from the **Site Collection Synchronization Data** for **SiteID** MUST be returned.

SchemaVersion: The **SchemaVersion** field from the **Site Collection Synchronization Data** for **SiteID** MUST be returned.

LastChangeSynchSuccess: The **LastChangeSynchSuccess** field from the **Site Collection Synchronization Data** for **SiteID** MUST be consulted and a 1 returned if the value is true, otherwise 0.

Moving: The **Moving** field from the **Site Collection Synchronization Data** for **SiteID** MUST be consulted and a 1 returned if the value is true, otherwise 0.

MovingDeleted: The **MovingDeleted** field from the **Site Collection Synchronization Data** for **SiteID** MUST be consulted and a 1 returned if the value is true, otherwise 0.

Registered: The **Registered** field from the **Site Collection Synchronization Data** for **SiteID** MUST be consulted and a 1 returned if the value is true, otherwise 0.

HasProfileChanges: Identifies whether there are profile changes that have not been synchronized yet for **SiteID**. This value MUST be 1 if the protocol server implementation of **profilesynch_US_IncrementalSynch** (*SiteID, 0, 0*) would return 1 or more rows (where *SiteID* is the **SiteID** of the current row). Otherwise, it MUST be 0.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.6 profilesynch_GetUnregisteredSites

The **profilesynch_GetUnregisteredSites** stored procedure requests a list of site collections in a specified content database that are not registered for synchronization.

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

```
PROCEDURE profilesynch_GetUnregisteredSites(
    @ContentDBID       uniqueidentifier
);
```

@ContentDBID: The identifier of a content database**.**

A row MUST be returned for each entry in the **Site Collection Synchronization Data** where **ContentDBID** matches *@ContentDBID* and **Registered** is false.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: MUST return the following result set:

3.1.4.6.1 UnregisteredSites Result Set

The T-SQL syntax for the result set is as follows.

```
SiteID                uniqueidentifier;
```

SiteID: The identifier of a site collection in the content database identified by *@ContentDBID*.

3.1.4.7 profilesynch_MS_AddUsersToGroup

The **profilesynch_MS_AddUsersToGroup** stored procedure is called to request that the protocol server update its data to reflect that the security principals specified are in the security group specified. Up to 10 security principals can be passed to this stored procedure.

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

```
PROCEDURE profilesynch_MS_AddUsersToGroup(
    @SiteID            uniqueidentifier,
    @GroupID           int,
    @WssID0            varbinary(512),
    @WssID1            varbinary(512) = NULL,
    @WssID2            varbinary(512) = NULL,
    @WssID3            varbinary(512) = NULL,
    @WssID4            varbinary(512) = NULL,
    @WssID5            varbinary(512) = NULL,
    @WssID6            varbinary(512) = NULL,
    @WssID7            varbinary(512) = NULL,
    @WssID8            varbinary(512) = NULL,
    @WssID9            varbinary(512) = NULL
);
```

@SiteID: The identifier of a site collection in the content database currently being synchronized.

@GroupID: The identifier of a security group.

@WssID#: The integer identifier of a security principal. Note that despite the type being varbinary(512) the protocol client MUST send an integer value. There are 10 columns numbered from 0 to 9, each of which can identify a security principal.

AddUserGroupRel (*@WssID#, @GroupID*) MUST be performed on the **Staging Data** for each *@WssID#* specified in the parameters.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.8 profilesynch_MS_AddUserToGroup

The **profilesynch_MS_AddUsersToGroup** stored procedure is called to request that the protocol server update its data to reflect that a security principal is a member of a security group.

AddUserGroupRel (*@WssID#, @GroupID*) MUST be performed on the **Staging Data.**

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

```
PROCEDURE profilesynch_MS_AddUsersToGroup(
    @SiteID            uniqueidentifier,
    @GroupID           int,
    @WssID             int
);
```

@SiteID: The identifier of a site collection.

@GroupID: The identifier of a security group in the site collection identified by *@SiteID*.

@WssID: The identifier of a security principal in the security group identified by *@GroupID***.**

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.3.4.

3.1.4.9 profilesynch_MS_DeleteGroup

The **profilesynch_MS_DeleteGroup** stored procedure is called to request that the protocol server update its data to delete all site memberships that depend on the specified security group.

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

```
PROCEDURE profilesynch_MS_DeleteGroup(
    @SiteID            uniqueidentifier,
    @GroupID           int
);
```

@SiteID: The identifier of a site collection.

@GroupID: The identifier of a security group in the site collection identified by *@SiteID*.

For each existing relationship found to exist between security principals and the security group identified by *@GroupID* in the **Membership Data** and **Staging Data**, DeleteUserGroupRel (***WSSID***, *@GroupID*) MUST be performed on the **Staging Data**.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.10 profilesynch_MS_DeleteUserFromGroup

The **profilesynch_MS_DeleteUserFromGroup** stored procedure is called to request that the protocol server delete all site memberships that depend on the specified security principal's membership in the specified security group**.**

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

```
PROCEDURE profilesynch_MS_DeleteUserFromGroup(
    @WssID                  int,
    @SiteID                 uniqueidentifier,
    @GroupID                int
);
```

@WssID: The identifier of a security principal in the site collection identified by *@SiteID***.**

@SiteID: The identifier of a site collection.

@GroupID: The identifier of a security group in the site collection identified by *@SiteID*.

DeleteUserGroupRel (@WSSID, @GroupID) MUST be performed on the **Staging Data**.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.11 profilesynch_MS_DeleteWeb

The **profilesynch_MS_DeleteWeb** stored procedure is called to request that the profile server delete all site memberships that depend on the specified site.

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

```
PROCEDURE profilesynch_MS_DeleteWeb(
    @WebID                  uniqueidentifier
);
```

@WebID: The identifier of a site.

DeleteSite (@WebID) MUST be performed on the **Staging Data.**

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.12 profilesynch_MS_GetGroupsForSite

The **profilesynch_MS_GetGroupsForSite** stored procedure requests that the protocol server return a list of all security groups in the site collection for which it has records.

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

```
PROCEDURE profilesynch_MS_GetGroupsForSite(
    @SiteID                 uniqueidentifier
);
```

@SiteID: The identifier of a site collection.

The **GroupSite Membership Data** MUST be consulted and a row returned for each **GroupID** that is a member of the site collection (identified by *@SiteID*).

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: MUST return the following result set:

3.1.4.12.1 Membership Synchronization Groups Result Set

The T-SQL syntax for the result set is as follows.

```
GroupID               int;
```

GroupID: The identifier of a security group that is a member of the site collection identified by *@SiteID* according to the data stored in **GroupSite Membership Data**.

3.1.4.13 profilesynch_MS_UpdateWeb

The **profilesynch_MS_UpdateWeb** stored procedure is called to request that the protocol server update its data regarding a site in the site collection**.**

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

```
PROCEDURE profilesynch_MS_UpdateWeb(
    @SiteID           uniqueidentifier,
    @WebID            uniqueidentifier,
    @GroupID          int,
    @WebName          nvarchar(250),
    @WebURL           nvarchar(2048),
```

```
    @UnknownGroup      bit = null OUTPUT
);
```

@SiteID: The identifier of a site collection.

@WebID: The identifier of a site in the site collection identified by *@SiteID*.

@GroupID: The identifier of the members group of the site identified by *@WebID*. A null value indicates that the site identified by *@WebID* is not associated with a members group.

@WebName: The name of the site identified by *@WebID*.

@WebURL: The URL of the site identified by *@WebID*.

@UnknownGroup: An output parameter bit indicating whether the user profile store has records identifying the security principals that are in the security group identified by the *@GroupID*..

The **SiteMembershipList** MUST be updated to reflect the new data regarding the site identified by *@WebID*. This includes:

- The name of the site (identified by *@WebName*)
- The URL of the site (identified by *@WebURL*)

If *@GroupID* is null, an equivalent operation to profilesynch_MS_DeleteWeb (*@WebID*) MUST be performed.

Otherwise, if the **GroupSite Membership Data** has a record of a previous relationship between the site identified by *@WebID* and a different security group then UpdateSiteGroupRel(*@WebID, @GroupID, **GroupID***) MUST be performed on the **Staging Data** (where **GroupID** is obtained from the **GroupSite Membership Data** for the site identified by *@WebID* in the site collection identified by *@SiteID***)**.

Otherwise, AddSiteGroupRel(*@WebID, @GroupID*) MUST be performed on the **Staging Data.**

If the security principals in the security group (identified by *@GroupID*) are known either from **AddUserGroupRel** operations that have occurred previously on the **Staging Data** or from the existence of records in the **GroupSite Membership Data** for security group (identified by *@GroupID*)in site collection @SiteID, then the @UnknownGroup parameter MUST be set to 1. Otherwise, it MUST be set to 0.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.14 profilesynch_PrepareToMove

The **profilesynch_PrepareToMove** stored procedure is called to request that the protocol server discontinue all synchronization activity for one or more site collections in a content database. Synchronization should not be performed on those site collections until synchronization is requested with a different content database.

This is useful in backup/restore scenarios where a particular site collection can be moved from one content database to another while retaining the same **GUID** identifier.

There is different handling for two different cases. The first case is where this stored procedure is called to notify the protocol server that a site collection is about to be moved to a different content database. The second is to notify the protocol server that a site collection has already moved to a different content database. The protocol client communicates which of these two cases is needed by specifying a value for the *@OldContentDBID* parameter in the case where the move has already occurred.

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

```
PROCEDURE profilesynch_PrepareToMove(
    @ContentDBID              uniqueidentifier,
    @OldContentDBID           uniqueidentifier = null,
    @SiteID                   uniqueidentifier = null,
    @Undo                     bit = null
);
```

@ContentDBID: The identifier of a content database containing the site collections to be processed.

@OldContentDBID: The identifier of the content database that previously contained the site collections to be processed. If this value is null, the site collections have not yet been moved to a different content database but rather the protocol client is signaling that they will be moved at some time in the future.

@SiteID: The identifier of the site collection to be processed. If this parameter is null, all site collections in the content database for which there exists a **Site Collection Synchronization Data** record MUST be processed.

@Undo: A bit specifying whether the stored procedure should prepare the site for movement or undo existing movement preparation. The **Moving** field in the **Site Collection Synchronization Data** MUST be set to true or false for the specified site collections. If the value of *@Undo* is 1, the stored procedure MUST set the **Moving** field to false for all affected site collections. Otherwise the **Moving** field MUST be set to true for all affected site collections.

The protocol server MUST perform the following actions while processing this stored procedure:

- If the *@OldContentDBID* parameter is set, the **ChangeTokenFull** field in the **Content Database Synchronization Data** for *@ContentDBID* MUST be set to null.
- The **Moving** field in the **Site Collection Synchronization Data** for the set of site collections specified (see the previous *@SiteID* parameter description for details) MUST be updated to true or false based on the value of *@Undo* (see the previous *@Undo* details description for).
- If the *@Undo* parameter is set to 1, the **MovingDeleted** field in the **Site Collection Synchronization Data** must be inspected for all site collections specified (see the previous @SiteID parameter description for details). Any site collections where the value of **MovingDeleted** is found to be true MUST be processed now in a manner consistent with the behaviors specified in the **profilesynch_CleanUpDeletedSites** stored procedure.
- Subsequent to this stored procedure being called, a site collection whose **Moving** field is set to true MUST NOT be returned by profilesynch_GetSitesToSynch and in general its data MUST NOT be updated in reaction to any synchronization stored procedures called until the **profilesynch_RegisterSitesToSynch** stored procedure is called for that site collection with a different content database.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.15 profilesynch_RegisterSitesToSynch

The **profilesynch_RegisterSitesToSynch** stored procedure is called to request that the user profile store register up to 10 site collections for synchronization.

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

```
PROCEDURE profilesynch_RegisterSitesToSynch(
    @ContentDBID              uniqueidentifier,
    @FailedSiteID             uniqueidentifier OUT,
    @SiteID0                  uniqueidentifier,
    @SiteID1                  uniqueidentifier = null,
    @SiteID2                  uniqueidentifier = null,
    @SiteID3                  uniqueidentifier = null,
    @SiteID4                  uniqueidentifier = null,
    @SiteID5                  uniqueidentifier = null,
    @SiteID6                  uniqueidentifier = null,
    @SiteID7                  uniqueidentifier = null,
    @SiteID8                  uniqueidentifier = null,
    @SiteID9                  uniqueidentifier = null
);
```

@ContentDBID: The identifier of a content database.

@FailedSiteID: An output parameter identifying the site collection that encountered a failure, in case of any error during execution.

@SiteID#: The identifier of a site collection in the content database identified by *@ContentDBID*. There are 10 such parameters numbered from 0 to 9, each of which can contain an identifier for a site collection.

The following actions MUST occur for each *@SiteID#* when the protocol server processes this request:

- If the **Site Collection Synchronization Data** already contains a record corresponding to *@SiteID#*
- If the **Moving** field of the **Site Collection Synchronization Data** is set to true for the record corresponding to *@SiteID#* and the value of **ContentDBID** is different than the value of *@ContentDBID* then the value of **ContentDBID** MUST be set to the value *of @ContentDBID* and the value of **Moving** MUST be set to false for that record.
- If the **Moving** field of the **Site Collection Synchronization Data** is set to true for the record corresponding to *@SiteID#* and the value of **ContentDBID** is the same as the value of @ContentDBID then the value of **Moving** MUST remain unchanged for that record.
- The value of **Registered** MUST be set to true.
- Otherwise, if the **Site Collection Synchronization Data** does not contain a record corresponding to *@SiteID#,* the **Site Collection Synchronization Data** MUST be updated to include a new record for the site collection *@SiteID#*. The initial value of each field in the record MUST be.

Field	Value
ContentDBID	*@ContentDBID*
Registered	true
Moving	false
MovingDeleted	false
LastSynch	null
LastChangeSynchSuccess	false
ChangeToken	null
SchemaVersion	0

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.16 profilesynch_ScheduleFullSiteSynch

The **profilesynch_ScheduleFullSiteSynch** stored procedure is called to request that the next synchronization of this site collection be a full synchronization rather than examining what has changed since the last synchronization.

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

```
PROCEDURE profilesynch_ScheduleFullSiteSynch(
    @SiteID             uniqueidentifier
);
```

@SiteID: The identifier of the site collection**.**

The **Site Collection Synchronization Data** MUST be updated for the site collection (as identified by *@SiteID*) as follows.

Field	Value
LastSynch	null
LastChangeSynchSuccess	false
ChangeToken	null

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.17 profilesynch_StartContentDBSynch

The **profilesynch_StartContentDBSynch** stored procedure is called to request that a change token be returned representing the current synchronization point of the specified content database and signals to the protocol server that synchronization activities are about to begin.

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

```
PROCEDURE profilesynch_StartContentDBSynch(
    @ContentDBID              uniqueidentifier
);
```

@ContentDBID: The identifier of a content database.

The server MUST return a single row with the column **CurrentChangeToken** set to the **ChangeTokenFull** field from the **Content Database Synchronization Data** records corresponding to *@ContentDBID*. If the value of this field is null, an empty result set MUST be returned.

Also, the **StartSynch** field from the **Content Database Synchronization Data** MUST be set to the current protocol server time.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: MUST return the following result set:

3.1.4.17.1 StartSynch ChangeToken Result Set

The T-SQL syntax for the result set is as follows.

```
CurrentChangeToken            ntext;
```

CurrentChangeToken: The change token.

3.1.4.18 profilesynch_StartFullSiteSynch

The **profilesynch_StartFullSiteSynch** stored procedure is called to signal the protocol server that the protocol client is about to send and request all data needed for the synchronization of a site collection.

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

```
PROCEDURE profilesynch_StartFullSiteSynch(
    @SiteID            uniqueidentifier,
    @DBTime            DateTime = null OUTPUT
);
```

@SiteID: The identifier of the site collection.

@DBTime: An output parameter containing a protocol server generated UTC timestamp.

The protocol server MUST perform **DeleteUser**, **DeleteSite**, and **DeleteUserGroupRel** operations on the **Staging Data** such that all data stored in the **User Data** and **Membership Data** areas regarding this site collection is slated for deletion.

Additionally the protocol server MUST set **@DBTime** to the current time on the server.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.19 profilesynch_SuccessfulContentDBSynch

The **profilesynch_SuccessfulContentDBSynch** stored procedure is called to request that the protocol server record that the specified content database has finished synchronization successfully.

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

```
PROCEDURE profilesynch_SuccessfulContentDBSynch(
    @ContentDBID            uniqueidentifier,
    @TargetChangeToken      ntext
);
```

@ContentDBID: The identifier of the content database.

@TargetChangeToken: A change token representing the point in the **change log** to which synchronization has occurred.

The **Content Database Synchronization Data** MUST be updated as follows.

Field	Value
ChangeTokenFull	*@TargetChangeToken*
EndSynch	Current protocol server time

Additionally, the **ChangeToken** field of the **Site Collection Synchronization Data** MUST be updated for all site collections in the content database (identified by *@ContentDBID*) to match *@TargetChangeToken*.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.20 profilesynch_SuccessfulSiteChangeLogConsumption

The **profilesynch_SuccessfulSiteChangeLogConsumption** stored procedure is called to request that the protocol server record that a particular site collection is now up to date with a new change token and that site collection data should now become live.

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

```
PROCEDURE profilesynch_SuccessfulSiteChangeLogConsumption(
    @ContentDBID            uniqueidentifier,
    @SiteID                 uniqueidentifier,
    @TargetChangeToken      ntext
);
```

@ContentDBID: The identifier of a content database.

@SiteID: The identifier of a site collection in the content database identified by *@ContentDBID*.

@TargetChangeToken: A change token representing the synchronization point.

The following actions MUST occur:

- The **Staging Data** MUST be flushed to **User Data** and **Membership Data** for the site collection identified by *@SiteID*.
- The consequent changes to the **SiteMembershipList** MUST be made through examination of **Membership Data** and **Staging Data** (using the **Staging Data** is optional because once the **Staging Data** has been flushed, it is a subset of the **Membership Data**):
- Where a security principal maps to a security group through **UserGroup** data and that same security group maps to a site through the **GroupSite** data, the **SiteMembershipList** for the user profile corresponding to that security principal MUST contain an entry for that site membership.
- If during the processing of this stored procedure, such a chain (security principal -> members group -> site) exists in the data where it did not exist previously, it MUST be added to the **SiteMembershipList** for the appropriate user profile.
- Conversely, if during the processing of this stored procedure, such a chain (security principal -> members group -> site) is broken where it did exist previously in the data, it MUST be deleted from the **SiteMembershipList** for the appropriate user profile.
- Also, if during the processing of this stored procedure, such a chain (security principal -> members group -> site) is connected through a new intermediate step (for example, the members group for a site is changed but the same security principal is a member of the new members group as well) then no changes should occur to the **SiteMembershipList** as a result of that change. Further, additional meta-data about the site Membership maintained by the user profile store (see [MS-UPSPROF]) MUST not be deleted or changed.
- The **Site Collection Synchronization Data** for the site collection identified by *@SiteID* MUST be updated as follows.

Field	Value
ChangeToken	*@TargetChangeToken*
LastChangeSynchSuccess	1

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.21 profilesynch_SuccessfulSiteProfilePush

The **profilesynch_SuccessfulSiteProfilePush** stored procedure is called to request that the protocol server record that the protocol client has successfully finished synchronizing the user profile data to the user information list for the specified site collection.

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

```
PROCEDURE profilesynch_SuccessfulSiteProfilePush(
    @SiteID                 uniqueidentifier,
    @StartSynchTime         DateTime,
    @SchemaVersion          int
);
```

@SiteID: The identifier of a site collection.

@StartSynchTime: The UTC identifying when the site collection synchronization began.

@SchemaVersion: An integer value representing the version number of the schema for replicated profile data.

The **Site Collection Synchronization Data** MUST be updated as follows for the site collection identified by *@SiteID*:

Field	Value
LastSynch	*@StartSynchTime*
SchemaVersion	*@SchemaVersion*

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.22 profilesynch_sweep_GetDBToken

The **profilesynch_sweep_GetDBToken** stored procedure is called to request a change token that represents the last point of "quick" synchronization (this is a lightweight synchronization which only pushes user profile data for new security principals in the content database) with the given content database**.**

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

```
PROCEDURE profilesynch_sweep_GetDBToken(
    @ContentDBID              uniqueidentifier
);
```

@ContentDBID: The identifier of a content database.

The server MUST return a single row where the column **ChangeToken** is set to the **ChangeTokenQuick field** from the **Content Database Synchronization Data** record corresponding to *@ContentDBID*. If the value of **ChangeTokenQuick** is null, an empty result set MUST be returned.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: MUST return the following result set:

3.1.4.22.1 SweepSynch GetChangeToken Result Set:

The T-SQL syntax for the result set is as follows.

```
ChangeToken            ntext;
```

ChangeToken: A change token.

3.1.4.23 profilesynch_sweep_UpdateDBToken

The **profilesynch_sweep_UpdateDBToken** stored procedure is called to request that the server store a change token representing the last point of "quick" synchronization for the specified content database**.**

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

```
PROCEDURE profilesynch_sweep_UpdateDBToken(
    @ContentDBID              uniqueidentifier,
    @ChangeToken              ntext
);
```

@ContentDBID: The identifier of a content database.

@ChangeToken: A change token denoting the new value of the last "quick" synchronization point with the content database**.**

The **ChangeTokenQuick** field from the **Content Database Synchronization Data** MUST be set to *@ChangeToken* in the record that corresponds to *@ContentDBID*.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.24 profilesynch_UnregisterAllSites

The **profilesynch_UnregisterAllSites** stored procedure is called to request that all site collections in the content database that is currently being synchronized no longer be registered for synchronization.

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

```
PROCEDURE profilesynch_UnregisterAllSites(
    @ContentDBID              uniqueidentifier
);
```

@ContentDBID: The identifier of the content database**.**

The **Registered** field of the **Site Collection Synchronization Data** MUST be set to false for every site collection that is a member of the content database identified by *@ContentDBID*.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: See "Return Code Values and Result Sets" in section 3.1.4.

3.1.4.25 profilesynch_US_AddProfilesToSynch

The **profilesynch_US_AddProfilesToSynch** stored procedure is called to request that user profile data be returned for up to 10 security principals in the site collection currently being synchronized.

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

```
PROCEDURE profilesynch_US_AddProfilesToSynch(
    @SiteID             uniqueidentifier,
```

```
    @SID0               varbinary(512),
    @UID0               int = null,
    @SID1               varbinary(512)= null,
    @UID1               int = null,
    @SID2               varbinary(512) = null,
    @UID2               int = null,
    @SID3               varbinary(512) = null,
    @UID3               int = null,
    @SID4               varbinary(512) = null,
    @UID4               int = null,
    @SID5               varbinary(512) = null,
    @UID5               int = null,
    @SID6               varbinary(512) = null,
    @UID6               int = null,
    @SID7               varbinary(512) = null,
    @UID7               int = null,
    @SID8               varbinary(512) = null,
    @UID8               int = null,
    @SID9               varbinary(512) = null,
    @UID9               int = null
);
```

@SiteID: The identifier of a site collection.

@SID#: A SystemID identifying the user profile data that is requested. There are 10 such parameters numbered from 0 to 9, each of which can contain a SystemID.

@UID#: A security principal in the site collection identified by @*SiteID* corresponding to the *@SID#* with the same suffix**.** There are 10 such parameters numbered from 0 to 9, each of which can contain a security principal identifier**.**

The user profiles for all requested security principals MUST be returned in the result set if the server has corresponding user profiles. The **WSSID** column MUST be set to the value of *@UID#* for the security principal whose user profile is being returned.

Additionally, for each user profile returned, **User Data** records MUST be added with the following data if it they do not already exist:

- **SiteID** - @SiteID
- **WSSID** - @UID#
- **SID** - @SID#

If there are any pending delete operations for these **User Data** records in the **Staging Data**, those pending operations should themselves be deleted.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: MUST return the following result set:

3.1.4.25.1 User Synchronization Result Set

The **profilesynch_US_AddProfilesToSynch** MUST return a result set based on [MS-UPSPROF] (see "UserProperties Result Set"). One additional column MUST be included in the result set between the 8th and 9th columns of the [MS-UPSPROF] (see "UserProperties Result Set").

```
WSSID           int;
```

WSSID: The identifier of the security principal.

3.1.4.26 profilesynch_US_IncrementalSynch

The **profilesynch_US_IncrementalSynch** stored procedure is called to request that the protocol server return user profile data about security principals whose corresponding user profiles have changed since the last synchronization with the site collection.

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

```
PROCEDURE profilesynch_US_IncrementalSynch(
    @SiteID                  uniqueidentifier,
    @MinNonInclusiveWssID    int,
    @AllProfiles             bit = 0,
    @DBTime                  DateTime OUTPUT
);
```

@SiteID: The identifier of a site collection.

@MinNonInclusiveWssID: A value used to page the **WSSID** field of the **User Data**.

@AllProfiles: A value indicating whether all profiles should be returned or only those that have changed.

@DBTime: An output parameter containing a protocol server-generated time.

The value of *@DBTime* MUST be set to the current protocol server time which MUST be obtained prior to inspecting the values of **LastChanged** for any user profiles.

The set of user profiles returned depends on the value of *@AllProfiles*:

- If it is 1, then all user profiles that match the **User Data** records for site collection (identified by *@SiteID*) MUST be returned.
- If it is 0, then a subset of the user profiles described in the previous bullet MUST be returned. The subset includes all user profiles that have a **LastChanged** field value which is more recent than the **LastSynch** field from the **Site Collection Synchronization Data.** In the case that the **LastSynch** field is null, the comparison to the **LastChanged** field need not be performed, the user profile MUST be included in the result set.

However, a single call to this stored procedure MUST NOT return more than 100 user profiles.

The protocol server MUST also only return rows for **User Data** whose **WSSID** field has a value greater than *@MinNonInclusiveWssID*.

The WSSID column MUST contain the value of the **WSSID** field in the **User Data** that matches the user profile rows being returned.

Return Code Values: See "Return Code Values and Result Sets" in section 3.1.4.

Result Sets: MUST return the following result set:

3.1.4.26.1 User Synchronization Result Set

profilesynch_US_IncrementalSynch MUST return the User Synchronization Result Set (section 3.1.4.25.1).

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

3.2 Synchronization Locking Server Details

The synchronization locking server specifies the interface used for synchronizing user profile and security principal data between the user profile store and content databases. The intent of the synchronization locking interface is that it be used simultaneously and in parallel on a separate SQL connection while synchronization is underway. Specifically, the locking interface is intended to be used to ensure that only a single synchronization is under way at a time between a given user profile store and content database.

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.

To accomplish the locking described here, the protocol server will need to store a list of content databases that are currently in the process of synchronizing.

3.2.2 Timers

None.

3.2.3 Initialization

To initialize the protocol, the following text will be sent.

```
set LOCK_TIMEOUT int0
if not exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[ContentDBLockGUID0]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
BEGIN
    CREATE TABLE [dbo].[ContentDBLockGUID0] (
        [Lock] [bit],
    ) ON [PRIMARY]
    insert into [ContentDBLock" + contentDBID.ToString() + @"] (Lock) values (0)
END

BEGIN TRANSACTION
update [ContentDBLock" + contentDBID.ToString() + @"] set Lock=1
```

The preceding text string contains two words in **bold**. Those should be treated as "parameters" in that they will be replaced with textual strings as follows:

- **Int0:** An integer value indicating the number of seconds to wait for an existing synchronization to terminate. A negative value means to wait indefinitely. A value of zero indicates that a single check should be performed and no waiting should occur.
- **GUID0:** A GUID identifying which content database to ensure is free of synchronization contention. The value of the GUID MUST be represented as a string in the following format:
- xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx where each "x" is replaced with a lower-case hexadecimal digit. Following this pattern, the digits should be arranged in groups of 8, 4, 4, 4, and 12 digits and separated by hyphens.

Behaviors:

The server MUST return an error as described in [MS-TDS], section 2.2.2.6 if

- **GUID0** identifies a content database that is currently in the process of synchronizing and the synchronization does not terminate in the time indicated by the value of **Int0**.
- A synchronization is not initially in progress or it terminates before the time indicated by **Int0** has elapsed the server MUST return a success.

The server MUST also record that a synchronization is in process for the content database identified by **GUID0** (as specified, this MUST block subsequent entrants from achieving successful initialization while the synchronization that has just been initialized is under way).

The current synchronization is considered to be under way until one of two things happens:

- The synchronization termination message is sent (see section 3.2.4).
- The connection used for synchronization is terminated.

3.2.4 Message Processing Events and Sequencing Rules

When the synchronization has finished, the synchronization termination message is sent. This is the only message in this protocol.

3.2.4.1 Synchronization Termination

When the synchronization has finished, the following text string will be transmitted.

```
ROLLBACK TRANSACTION
```

Consistent with section 3.2.4, this message indicates that the synchronization is no longer considered to be under way and other entrants should not be blocked from performing the initialization for the content database specified in the initialization step.

3.2.5 Timer Events

None.

3.2.6 Other Local Events

None.

3.3 Profile Synchronization Client Details

This section provides information about the profile synchronization protocol client.

3.3.1 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 variables will be used by the client:

- **ReplicableSchemaVersion:** A value used to determine whether the schema of the user information list is up to date.
- **ReplicableSchema:** A list of fields and types.
- **SynchStartTime:** A UTC generated by the protocol server.

3.3.2 Timers

None.

3.3.3 Initialization

The protocol client MUST use the protocol described in [MS-UPSPROF] to set the **ReplicableSchemaVersion** variable to the value of the *@ReplicableSchemaVersion* output parameter described in [MS-UPSPROF] (see "profile_GetProfilePropertyInfo"). The value of **ReplicableSchema** MUST be set to a subset of fields and types obtained from the result set described in [MS-UPSPROF] (see "profile_GetProfilePropertyInfo"). The subset of fields and types MUST be those where the **Replicable** column (described in [MS-UPSPROF] in "profile_GetProfilePropertyInfo") has a value of 1.

3.3.4 Message Processing Events and Sequencing Rules

In general, the protocol client MUST accomplish synchronization as described in section 1.3 through usage of the synchronization protocol and locking protocol described in sections section 3.1 and section 3.2 respectively. There are many possible protocol client implementations and stored procedure calling orders to achieve that goal.

The protocol client handles each stored procedure by using the same basic process of calling the stored procedure and waiting for the return code and any result sets that will be returned.

This section describes the additional protocol client behavior when calling some of the stored procedures listed in section 3.1.4 on the user profile store.

3.3.4.1 State Transitions

Section 3.1 describes the state transitions that the protocol client MUST follow.

3.3.4.2 Locking and Synchronization

Prior to calling stored procedures other than **profilesynch_sweep_GetDBToken** and **profilesynch_sweep_UpdateDBToken**, the protocol client MUST lock the content database

synchronization records on the user profile store for exclusive access by using the locking procedure described in section 3.2. The lock MUST be maintained on a separate connection for the duration of synchronization.

3.3.4.3 Stored Procedure Specific Client Requirements

3.3.4.3.1 profilesynch_MS_UpdateWeb

If *@UnknownGroup* is set by the protocol server to a value other than null, the protocol client MUST send the protocol server all **members** of the security group *@GroupID*. The members of the security group MUST be sent through repeatedly calling **profilesynch_MS_AddUsersToGroup**.

3.3.4.3.2 profilesynch_GetSitesToSynch

The **SchemaVersion** column of the result set MUST be compared with the **ReplicableSchemaVersion** variable. If they are found to be different for one of the **SiteIDs**, [MS-WSSFO] MUST be used to update the schema of the user information list for the site collection identified by **SiteID** to match the **ReplicableSchema**.

3.3.4.3.3 profilesynch_StartFullSiteSynch

The **SynchStartTime** variable MUST be set to the value of the *@DBTime* output parameter.

3.3.4.3.4 profilesynch_SuccessfulSiteProfilePush

The *@SchemaVersion* parameter MUST be set to the value of the **ReplicableSchemaVersion** variable.

The *@StartSynchTime* parameter MUST be set to the value of the **StartSynch** variable.

3.3.4.3.5 profilesynch_US_IncrementalSynch

The **SynchStartTime** variable MUST be set to the value of the *@DBTime* output parameter.

4 Protocol Examples

This section provides examples of the synchronization, backup, and restoration processes.

4.1 Synchronization

4.1.1 Example Data

In these synchronization examples, the data is originally arranged this way:

- The user profile store contains five user profiles
  - P1 – Syed Abbas
  - P2 – Lori Kane
  - P3 – Tai Yee
  - P4 – Ellen Adams
  - P5 – Sara Davis
- Content database CDB1 contains 1 site collection
  - SC1 contains
    - Three security principals
      - SP1 – Lori Kane
      - SP2 – Sara Davis
      - SP3 – Steve Masters
    - Two security groups
      - SG1 – SP1 and SP2 are members of this security group
      - SG2 – SP2 and SP3 are members of this security group
    - Two sites
      - S1 – the members group for this site is SG1
      - S2 – the members group for this site is SG2

The following identifiers are used in the examples in this section:

- User Profiles
  - P1 – 0x0105000000000000515000000A065CF7E784B9B5FE77C877003D32000
  - P2 – 0x0105000000000000515000000A065CF7E784B9B5FE77C87704D7A2100
  - P3 – 0x0105000000000000515000000A065CF7E784B9B5FE77C877080D00500
  - P4 – 0x0105000000000000515000000A065CF7E784B9B5FE77C877000311100

- P5 - 0x0105000000000000515000000A065CF7E784B9B5FE77C877088772100
- P6 - 0x0105000000000000515000000A065CF7E784B9B5FE77C877081D00500

- Content Databases
  - CDB1 - CD56ACC0-3E03-4264-B187-786A7B98D49D
- Site Collections
  - SC1 - 595D079D-DB43-4403-8A1D-6DF10295FA75
- Sites
  - S1 - EADD383A-7A5C-4F88-A71F-900D2031F81B
  - S2 - 0F2BE3A3-D9D0-4D8F-BBA5-36BF5EC9BAE8
- Security Principals
  - SP1 – 10
  - SP2 – 8
  - SP3 – 9
  - SP4 – 11
- Security Groups
  - SG1 – 5
  - SG2 – 7
  - SG3 – Not used

In the following diagrams, titled Full synchronization communication between the protocol client and protocol server and Incremental synchronization communication between the protocol client and protocol server, the string "profilesynch_" has been stripped from the beginning of stored procedure names.

4.1.2 Full Synchronization

This example illustrates how this protocol can be used to accomplish the full synchronization of all data in CDB1 from the example data. The following diagram shows the communication between the protocol client and protocol server.

This example assumes that the protocol client has done the following prior to the example:

- Executed the initialization described in section 3.3.3 such that the value of **ReplicableSchemaVersion** and the elements of **ReplicableSchema** are known.
- Obtained a change token for CDB1 (CT1).

The following diagram illustrates the overall flow for the example.

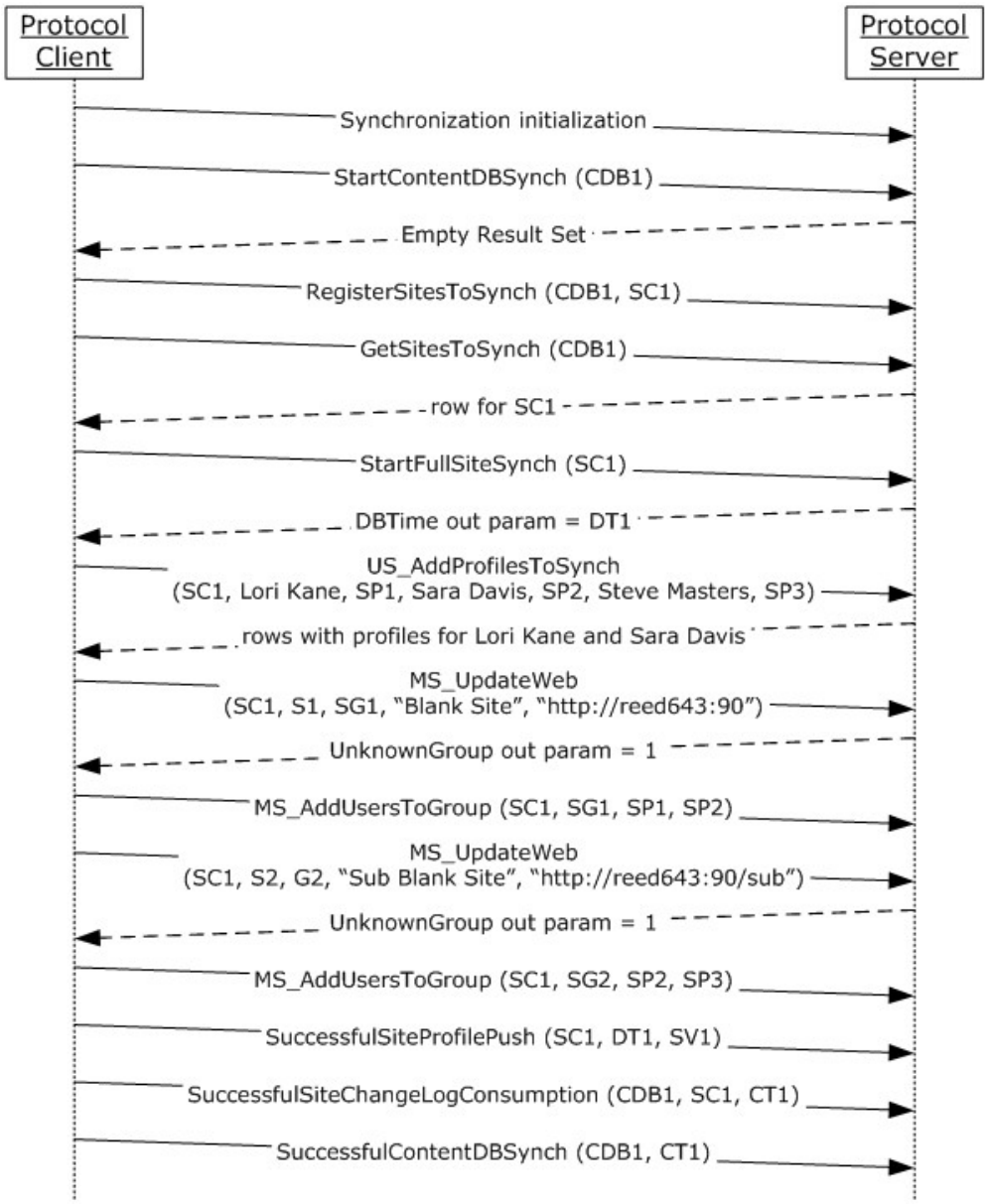

Figure 3: Full synchronization communication between the protocol client and protocol server

In detail:

1. The protocol client starts off the synchronization process by using the locking protocol for the Content database CDB1 on another session with the SSP (not depicted in the previous diagram) to prevent any other synchronization work from being executed while this synchronization is in progress.

```
set LOCK_TIMEOUT 0
if not exists (select * from dbo.sysobjects where id =
```

```
    object_id(N'[dbo].[ContentDBLockCD56ACC0-3E03-4264-B187-786A7B98D49D]')
    and OBJECTPROPERTY(id, N'IsUserTable') = 1)
BEGIN
    CREATE TABLE [dbo].[ContentDBLockCD56ACC0-3E03-4264-B187-786A7B98D49D] (
        [Lock] [bit],
    ) ON [PRIMARY]
    insert into [ContentDBLockCD56ACC0-3E03-4264-B187-786A7B98D49D] (Lock)
    values (0)
END

BEGIN TRANSACTION
update [ContentDBLockCD56ACC0-3E03-4264-B187-786A7B98D49D] set Lock=1
```

2. Next the protocol client executes the initialization described in section 3.1.3.

3. The protocol client next requests synchronization of the content database CDB1 by calling StartContentDBSynch. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure.

```
exec dbo.profilesynch_StartContentDBSynch
    @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D'
```

4. Because the protocol server has never synchronized this content database before, the **ChangeTokenFull** field from the **Content Database Synchronization Data** will be set to null and an empty StartSynchChangeToken Result Set will be returned. This call will also change the protocol state to content database synchronization as detailed in section 3.1.

5. At this point, the protocol is in the content database synchronization state. Before synchronizing individual site collections, the protocol client first tells the user profile store what site collections are in CDB1 by calling RegisterSitesToSynch for SC1. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure.

```
exec dbo.profilesynch_RegisterSitesToSynch
    @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
    @SiteID0 = '595D079D-DB43-4403-8A1D-6DF10295FA75',
    @FailedSiteID = NULL
```

6. The protocol server updates the **Site Collection Synchronization Data** to add a record for SC1. The protocol server then returns a 0 return code for successful execution. It also sets the value of the output parameter *@FailedSiteID* to NULL.

7. Next, the protocol client calls GetSitesToSynch to determine which site collections require synchronization. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure.

```
exec dbo.profilesynch_GetSitesToSynch
    @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D'
```

8. In response, the protocol server returns a SitesToSynchronize Result Set. The following result set is returned to the client.

ContentDBID	SiteID	LastSynch	ChangeToken	SchemaVersion
{CD56ACC0-3E03-4264-B187-786A7B98D49D}	{595D079D-DB43-4403-8A1D-6DF10295FA75}	0	null	0

LastChangeSynchSuccess	Moving	MovingDeleted	Registered	HasProfileChanges
0	0	0	1	0

1. The protocol client compares the value of **SchemaVersion** (zero) to the value of **ReplicableSchemaVersion**. Because this is the first time synchronizing SC1, the two will be different and the protocol client will update the schema of the user information list to match the schema of **ReplicableSchema** as described in section 3.3.4.3.2.

2. Now the protocol client is ready to begin synchronizing individual site collections (in this example case there is only one). It first takes care of synchronizing user profile data down to the content database by calling StartFullSiteSynch. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure.

```
exec dbo.profilesynch_StartFullSiteSynch
    @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
    @DBTime = NULL
```

3. With this call, the protocol state changes to profile synchronization. The protocol server sets the value of the output parameter *@DBTime* to DT1 which is returned to the client. Assuming this synchronization is successful, this UTC will be useful in future synchronizations to determine which user profiles have actually changed since the last synchronization.

```
    DBTime = '2008-03-11 18:01:18.466'
```

4. The protocol client sets **StartSynchTime** to the value of *@DBTime* as described in section 3.3.4.3.3. The protocol client then calls **profilesynch_us_AddProfilesToSynch** which tells the protocol server that Lori Kane is SP1, Sara Davis is SP2, and Steve Masters is SP3. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure.

```
exec dbo.profilesynch_US_AddProfilesToSynch
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @SID0 = 0x0105000000000000515000000a065cf7e784b9b5fe77c87702a040100,
   @UID0 = 1,
   @SID1 = 0x0101000000000000513000000,
   @UID1 = 6,
   @SID2 = 0x0105000000000000515000000a065cf7e784b9b5fe77c877088772100,
   @UID2 = 8,
   @SID3 = 0x0105000000000000515000000a065cf7e784b9b5fe77c877081d00500,
   @UID3 = 9,
   @SID4 = 0x0105000000000000515000000a065cf7e784b9b5fe77c87704d7a2100,
   @UID4 = 10
```

5. The protocol server returns back a User Synchronization Result Set containing user profile data for Lori Kane and Sara Davis. Steve Masters does not presently have a user profile, so no data is returned for him.

RecordID	ProfileSubTypeID	PropertyID	PropertyVal
1	1	2	0x0105000000000000515000000A065CF7E784B9B5FE77C87704D7A2100
1	1	3	CONTOSO\lori
1	1	4	Lori
1	1	7	Lori Kane
1	1	17	lori
2	1	2	0x0105000000000000515000000A065CF7E784B9B5FE77C877088772100
2	1	3	CONTOSO\sara
2	1	4	Sara
2	1	7	Sara Davis
2	1	17	sara

Image	Text	VocValID	OrderRank	Privacy	WSSID
NULL	NULL	NULL	NULL	NULL	10
NULL	NULL	NULL	NULL	NULL	10
NULL	NULL	NULL	NULL	NULL	10
NULL	NULL	NULL	NULL	NULL	10
NULL	NULL	NULL	NULL	NULL	10
NULL	NULL	NULL	NULL	NULL	8
NULL	NULL	NULL	NULL	NULL	8
NULL	NULL	NULL	NULL	NULL	8
NULL	NULL	NULL	NULL	NULL	8
NULL	NULL	NULL	NULL	NULL	8

PropertyName	PropertyURI	VocValValue
SID	urn:schemas-microsoft-com:sharepoint:portal:profile:SID	NULL

PropertyName	PropertyURI	VocValValue
AccountName	urn:schemas-microsoft-com:sharepoint:portal:profile:AccountName	NULL
FirstName	urn:schemas-microsoft-com:sharepoint:portal:profile:FirstName	NULL
PreferredName	urn:schemas-microsoft-com:sharepoint:portal:profile:PreferredName	NULL
UserName	urn:schemas-microsoft-com:sharepoint:portal:profile:UserName	NULL
SID	urn:schemas-microsoft-com:sharepoint:portal:profile:SID	NULL
AccountName	urn:schemas-microsoft-com:sharepoint:portal:profile:AccountName	NULL
FirstName	urn:schemas-microsoft-com:sharepoint:portal:profile:FirstName	NULL
PreferredName	urn:schemas-microsoft-com:sharepoint:portal:profile:PreferredName	NULL
UserName	urn:schemas-microsoft-com:sharepoint:portal:profile:UserName	NULL

1. The protocol client then writes the user profile data it received to the user information list.

2. At this point the protocol client moves on to synchronizing memberships. The protocol client identifies that there are two sites within SC1 (S1 and S2). **profilesynch_MS_UpdateWeb** is called for the first site. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure.

```
exec dbo.profilesynch_MS_UpdateWeb
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @WebID = 'EADD383A-7A5C-4F88-A71F-900D2031F81B',
   @GroupID = 5,
   @WebName = N'Blank Site',
   @WebURL = N'http://reed643:90',
   @UnknownGroup = NULL
```

3. This call switches the state of the protocol to membership synchronization. Because the protocol server doesn't have data about SG1 yet (as described in section 3.1.4.13) the *@UnknownGroup* output parameter is returned with a value of 1.

```
@UnknownGroup = 1
```

4. The protocol client responds by calling **profilesynch_MS_AddUsersToGroup** to tell the protocol server which security principals are in SG1. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure.

```
exec dbo.profilesynch_MS_AddUsersToGroup
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @GroupID = 5,
   @WssID0 = 8,
   @WssID1 = 10
```

5. The protocol server appropriately updates the **Staging Data** and returns a 0 return code for successful execution.

6. The preceding steps (13 – 16) are repeated for the second site S2 in SC1. Consider the following T-SQL syntax, which displays the parameters used to call the **profilesynch_ms_UpdateWeb** stored procedure for the second site.

```
exec dbo.profilesynch_MS_UpdateWeb
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @WebID = '0F2BE3A3-D9D0-4D8F-BBA5-36BF5EC9BAE8',
   @GroupID = 7,
   @WebName = N'Sub Blank Site',
   @WebURL = N'http://reed643:90/sub',
   @UnknownGroup = NULL
```

7. Because the protocol server does not have data about SG2 yet, the *@UnknownGroup* output parameter is returned with a value of 1.

```
@UnknownGroup = 1
```

8. The protocol client responds by calling **profilesynch_ms_AddUsersToGroup** to tell the protocol server which security principals are in SG2. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure.

```
exec dbo.profilesynch_MS_AddUsersToGroup
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @GroupID = 7,
   @WssID0 = 8,
   @WssID1 = 9
```

9. The protocol server appropriately updates the **Staging Data** and returns a 0 return code for successful execution.

10. Assuming no errors are generated during this process, the protocol client then calls SuccessfulSiteProfilePush, passing **ReplicableSchemaVersion** and **StartSynchTime** as described in section 3.3.4.3.4. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure:

```
exec dbo.profilesynch_SuccessfulSiteProfilePush
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @StartSynchTime = 2008-03-11 18:01:18.466,
   @SchemaVersion = 1
```

11. While executing the SuccessfulSiteProfilePush stored procedure, the protocol server updates the **Site Collection Synchronization Data** for SC1. Because this has occurred, in the future, differences can be examined rather than doing a full synchronization. The protocol server returns a 0 return code for successful execution.

12. Now that all data in the site collection has been examined and sent to the user profile store, the protocol client calls SuccessfulSiteChangeLogConsumption for the site collection SC1. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure:

```
exec dbo.profilesynch_SuccessfulSiteChangeLogConsumption
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @TargetChangeToken = N'1;0;cd56acc0-3e03-4264-b187-
```

```
786a7b98d49d;633408552555600000;461'
```

13. The protocol server returns a 0 return code for successful execution. This call also changes the protocol state back to content database synchronization.

14. Because there was only one site collection in CDB1, there is no remaining synchronization work. The protocol client calls SuccessfulContentDBSynch. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure:

```
exec dbo.profilesynch_SuccessfulContentDBSynch
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @TargetChangeToken = N'1;0;cd56acc0-3e03-4264-b187-
     786a7b98d49d;633408552555600000;461'
```

15. The protocol server updates its synchronization data accordingly and returns a 0 return code for successful execution.

16. Finally, the protocol client will unlock CDB1 using the locking protocol on the second session (not depicted) so that other synchronization activity can occur for CDB1.

```
ROLLBACK TRANSACTION
```

4.1.3 Incremental Synchronization

This example describes the incremental synchronization process on CDB1 that occurs after the synchronization described in section 4.1.2.

This example assumes that the data set changed to the following sometime after the full synchronization. *Italic* is used to highlight additions, **bold** highlights updates, and ***bold italic*** highlights deletions.

- The user profile store contains *six user profiles (P1 – P6)*:
  - P1 – Syed Abbas
  - **P2 – Lori Kane**
    - Lori Kane's profile changed because of a new job and a new cost center
  - P3 – Tai Yee
  - P4 – Ellen Adams
  - P5 – Sara Davis
  - *P6 – Steve Masters*
- content database CDB1 contains 1 site collection
  - SC1 contains
    - *Four* security principals
      - SP1 – Lori Kane

- SP2 – Sara Davis
- SP3 – Steve Masters
- *SP4 – Ellen Adams*

- Two sites
  - S1 has members group G1
  - S2 has members group *G1*
- *Three* security groups
  - SG1 – SP2 and *SP4* are members of this security group. ***SP1*** is no longer a member.
  - SG2 – SP2 and SP3 are members of this security group.
  - *SG3 – SP1* is a member of this security group

This example assumes that the protocol client has done the following prior to the example:

- Executed the initialization described in section 3.3.3 such that the value of **ReplicableSchemaVersion** and the elements of **ReplicableSchema** are known.
- Obtained a change token for CDB1 (CT1).

The following diagram illustrates the overall flow for the example.

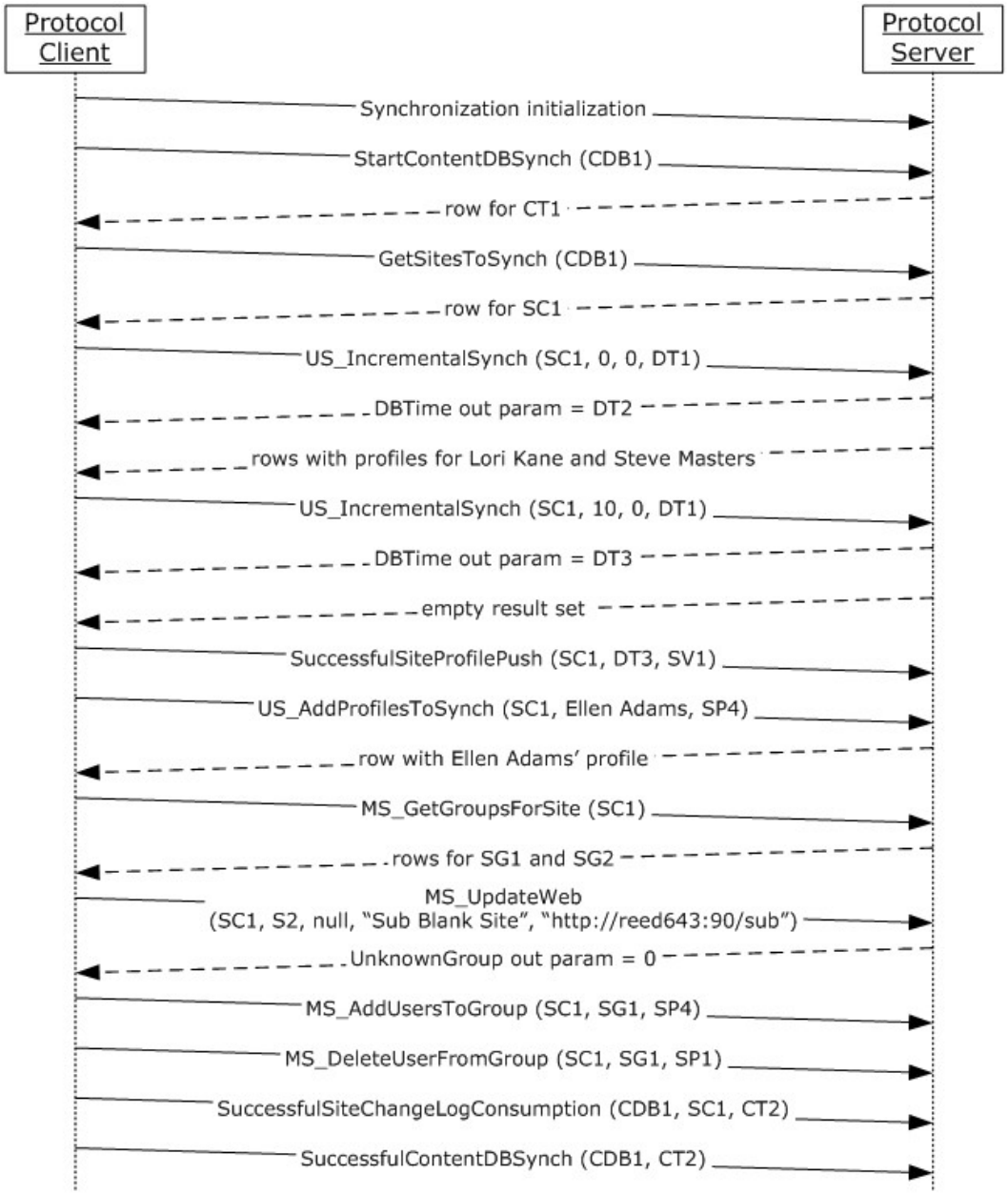

Figure 4: Incremental synchronization communication between the protocol client and protocol server

In detail:

1. The protocol client starts off synchronization by using the locking protocol for the content database CDB1 on another session with the SSP (not depicted in the previous diagram) to prevent any other synchronization work from being executed while this synchronization is in progress.

```
set LOCK_TIMEOUT 0
```

```
if not exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[ContentDBLockCD56ACC0-3E03-4264-B187-786A7B98D49D]')
  and OBJECTPROPERTY(id, N'IsUserTable') = 1)
BEGIN
    CREATE TABLE [dbo].[ContentDBLockCD56ACC0-3E03-4264-B187-786A7B98D49D] (
        [Lock] [bit],
    ) ON [PRIMARY]
    insert into [ContentDBLockCD56ACC0-3E03-4264-B187-786A7B98D49D] (Lock)
values (0)
END

BEGIN TRANSACTION
update [ContentDBLockCD56ACC0-3E03-4264-B187-786A7B98D49D] set Lock=1
```

2. Next the protocol client executes the initialization described in section 3.3.3.

3. The protocol client next calls StartContentDBSynch for CDB1. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure.

```
exec profilesynch_StartContentDBSynch
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D'
```

4. Because the protocol server has synchronized CDB1 before (in the preceding Full Synchronization example), the **ChangeTokenFull** field from the **Content Database Synchronization Data** will not be set to null and the protocol server will return a StartSynchChangeToken Result Set consisting of a single row with the column CurrentChangeToken.

CurrentChangeToken
1;0;cd56acc0-3e03-4264-b187-786a7b98d49d;633408552555600000;461

1. The change token returned (CT1) matches the change token passed to SuccessfulContentDBSynch in the full synchronization example. This call will also change the protocol state to content database synchronization as described in section 3.1.

2. At this point, the protocol is in the content database synchronization state. Before synchronizing individual site collections, the protocol client first:

3. Examines change log events that have taken place between CT1 and CT2 to determine if new site collections have been added to CDB1. If any had been added (they were not in this example) RegisterSitesToSynch would be called for each of those site collections.

4. Calls GetSitesToSynch to determine which site collections require synchronization. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure:

```
exec dbo.profilesynch_GetSitesToSynch
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D'
```

5. In response, the protocol server returns a SitesToSynchronize Result Set. Because this site collection was synchronized earlier, the row will have a value for the ChangeToken column (CT1 from the previous example).

ContentDBID	SiteID	LastSynch	ChangeToken
CD56ACC0-3E03-4264-B187-786A7B98D49D	595D079D-DB43-4403-8A1D-6DF10295FA75	2008-03-11 18:01:18.466	1;0;cd56acc0-3e03-4264-b187-786a7b98d49d;633408552555600000;461

SchemaVersion	LastChange SynchSuccess	Moving	MovingDeleted	Registered	HasProfileChanges
1	1	0	0	1	1

1. Now the protocol client is ready to begin synchronizing individual site collections (in the example case there is only 1). It first takes care of synchronizing user profile data down to the content database.

2. The protocol client first verifies that the value of **SchemaVersion** SV1 matches the current **ReplicableSchemaVersion**. In this example, the **ReplicableSchemaVersion** did not change, so the destination user information list schema already matches the schema in **ReplicableSchema**.

3. Next, the protocol client calls **profilesynch_us_IncrementalSynch** to retrieve user profiles that have changed since the last synchronization.

```
exec dbo.profilesynch_US_IncrementalSynch
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @MinNonInclusiveWssID = 0,
   @DBTime = NULL
```

4. This call causes the protocol state to change to profile synchronization. The protocol server returns a User Synchronization Result Set containing user profiles that have changed – in this example data that includes Lori Kane whose cost center changed and Steve Masters who has a user profile now and did not previously. Additionally, the protocol server sets the output parameter *@DBTime* to DT2.

Record ID	ProfileSubTypeID	Property ID	PropertyVal
1	1	2	0x0105000000000000515000000A065CF7E784B9B5FE77C87704D7A2100
1	1	3	CONTOSO\lori
1	1	4	Lori
1	1	7	Lori Kane
1	1	16	

Record ID	ProfileSubTypeID	Property ID	PropertyVal
1	1	17	lori
1	1	5005	NULL
4	1	2	0xa5024c1d0105000000000000515000000a065cf7e784b9b5fe77c877081d00500
4	1	3	CONTOSO\tai
4	1	5	Yee
4	1	7	Tai Yee
4	1	17	tai

Image	Text	VocValID	OrderRank	Privacy	WSSID
NULL	NULL	NULL	NULL	NULL	10
NULL	NULL	NULL	NULL	NULL	10
NULL	NULL	NULL	NULL	NULL	10
NULL	NULL	NULL	NULL	NULL	10
NULL	NULL	NULL	NULL	NULL	10
NULL	NULL	NULL	NULL	NULL	10
NULL	NULL	1	1	NULL	10
NULL	NULL	NULL	NULL	NULL	9
NULL	NULL	NULL	NULL	NULL	9
NULL	NULL	NULL	NULL	NULL	9
NULL	NULL	NULL	NULL	NULL	9
NULL	NULL	NULL	NULL	NULL	9

PropertyName	PropertyURI	VocValValue
SID	urn:schemas-microsoft-com:sharepoint:portal:profile:SID	NULL
AccountName	urn:schemas-microsoft-com:sharepoint:portal:profile:AccountName	NULL
FirstName	urn:schemas-microsoft-com:sharepoint:portal:profile:FirstName	NULL
PreferredName	urn:schemas-microsoft-com:sharepoint:portal:profile:PreferredName	NULL

PropertyName	PropertyURI	VocValValue
About Me	urn:schemas-microsoft-com:sharepoint:portal:profile:AboutMe	NULL
UserName	urn:schemas-microsoft-com:sharepoint:portal:profile:UserName	NULL
SPS-Responsibility	urn:schemas-microsoft-com:sharepoint:portal:profile:SPS-Responsibility	New job
SID	urn:schemas-microsoft-com:sharepoint:portal:profile:SID	NULL
AccountName	urn:schemas-microsoft-com:sharepoint:portal:profile:AccountName	NULL
LastName	urn:schemas-microsoft-com:sharepoint:portal:profile:LastName	NULL
PreferredName	urn:schemas-microsoft-com:sharepoint:portal:profile:PreferredName	NULL
UserName	urn:schemas-microsoft-com:sharepoint:portal:profile:UserName	NULL

```
@DBTime = '2008-03-11 18:33:30.620'
```

1. The protocol client then writes the user profile data it received to the user information list. Even though the protocol client has processed some user profiles, there could be more. The protocol client calls **profilesynch_us_IncrementalSynch** in a loop until it gets back an empty result set. Because the protocol client received a non empty UserProfile Result Set in the last call, it calls **profilesynch_us_IncrementalSynch** again, passing the highest **WSSID** that was returned in the previous call.

```
exec dbo.profilesynch_US_IncrementalSynch
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @MinNonInclusiveWssID = 10,
   @DBTime = NULL
```

2. This time the protocol server returns an empty UserProfile Result set. It also sets the output parameter *@DBTime* to the new database time DT3.

```
@DBTime = '2008-03-11 18:33:32.310'
```

3. Because the protocol client received an empty result set from **profilesynch_us_IncrementalSynch** it is not called any more. The protocol client will set the **StartSynchTime** variable to the value of @DBTime. The protocol client then calls **profilesynch_SuccessfulSiteProfilePush** passing **ReplicableSchemaVersion** and **StartSynchTime** as described in section 3.3.4.3.4. Calling SuccessfulSiteProfilePush switches the protocol state to membership synchronization. This time the protocol client has the change token CT1 which can be used to look at changes since the last synchronization to make the synchronization faster.

```
exec dbo.profilesynch_SuccessfulSiteProfilePush
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @StartSynchTime = '2008-03-11 18:33:32.310',
   @SchemaVersion = 1
```

4. While executing the **profilesynch_SuccessfulSiteProfilePush** stored procedure, the protocol server updates the **Site Collection Synchronization Data** for SC1. The protocol server returns a 0 return code for successful execution.

5. After consulting the change log for security principal additions that occurred between CT1 and CT2, the protocol client concludes that Ellen Adams is a new security principal in SC1 that requires synchronization and calls **profilesynch_US_AddProfilesToSynch** for Ellen Adams.

```
exec dbo.profilesynch_US_AddProfilesToSynch
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @SID0 = 0x010500000000000515000000a065cf7e784b9b5fe77c877000311100,
   @UID0 = 11
```

6. The protocol server returns back a User Synchronization Result Set containing user profile data for Ellen Adams.

RecordID	ProfileSubTypeID	PropertyID	PropertyVal
5	1	2	0x010500000000000515000000A065CF7E784B9B5FE77C877000311100
5	1	3	CONTOSO\ellen
5	1	4	Ellen
5	1	7	Ellen Adams
5	1	17	ellen

Image	Text	VocValID	OrderRank	Privacy	WSSID
NULL	NULL	NULL	NULL	NULL	11
NULL	NULL	NULL	NULL	NULL	11
NULL	NULL	NULL	NULL	NULL	11
NULL	NULL	NULL	NULL	NULL	11
NULL	NULL	NULL	NULL	NULL	11

PropertyName	PropertyURI	VocValValue
SID	urn:schemas-microsoft-com:sharepoint:portal:profile:SID	NULL
AccountName	urn:schemas-microsoft-com:sharepoint:portal:profile:AccountName	NULL
FirstName	urn:schemas-microsoft-com:sharepoint:portal:profile:FirstName	NULL
PreferredName	urn:schemas-microsoft-com:sharepoint:portal:profile:PreferredName	NULL
UserName	urn:schemas-microsoft-com:sharepoint:portal:profile:UserName	NULL

1. The protocol client then writes the user profile data it received to the user information list.

2. Before looking at security group changes, the protocol client first requests that the protocol server return the full list of security groups that it is currently tracking by calling **profilesynch_MS_GetGroupsForSite**.

```
exec dbo.profilesynch_MS_GetGroupsForSite
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75'
```

3. The protocol server returns Membership Synchronization Groups result sets containing data for SG1 and SG2.

GroupID
5
7

1. The protocol client also discovers an update to S2 in the changes between CT1 and CT2. It calls **profilesynch_ms_UpdateWeb** for S2.

```
exec dbo.profilesynch_MS_UpdateWeb
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @WebID = '0F2BE3A3-D9D0-4D8F-BBA5-36BF5EC9BAE8',
   @GroupID = 5,
   @WebName = N'Sub Blank Site',
   @WebURL = N'http://reed643:90/sub',
   @UnknownGroup = NULL
```

2. This time the *@UnknownGroup* output parameter is 0 because the new members group for S2 is G1 for which the protocol server already has data.

```
@UnknownGroup = 0
```

3. Looking at changes occurring between CT1 and CT2, the protocol client finds that G1 has changed. The protocol client calls **profilesynch_ms_AddUsersToGroup** to update the data for SP4 (Ellen Adams).

```
exec dbo.profilesynch_MS_AddUserToGroup
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @GroupID = 5,
   @WssID = 11
```

4. The protocol server appropriately updates the **Staging Data** and returns a 0 return code for successful execution.

5. Next, the protocol client calls **profilesynch_ms_DeleteUserFromGroup** to update the data for SP1 (Lori Kane).

```
exec dbo.profilesynch_MS_DeleteUserFromGroup
   @WssID = 10,
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @GroupID = 5
```

6. The protocol server appropriately updates the **Staging Data** and returns a 0 return code for successful execution.

7. Now that all data in the site collection has been examined and sent to the user profile store, the protocol client calls SuccessfulSiteChangeLogConsumption for SC1. Calling SuccessfulSiteChangeLogConsumption changes the protocol state back to content database synchronization.

```
exec dbo.profilesynch_SuccessfulSiteChangeLogConsumption
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @TargetChangeToken = N'1;0;cd56acc0-3e03-4264-b187-
      786a7b98d49d;633408571893700000;520'
```

8. The protocol server returns a 0 return code for successful execution. This call also changes the protocol state back to content database synchronization.

9. Because there was only one site collection**,** the content database is also finished synchronizing. The protocol client calls **profilesynch_SuccessfulContentDBSynch**. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure:

```
exec dbo.profilesynch_SuccessfulContentDBSynch
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @TargetChangeToken = N'1;0;cd56acc0-3e03-4264-b187-
      786a7b98d49d;633408571893700000;520'
```

10. The protocol server updates its synchronization data accordingly and returns a 0 return code for successful execution.

11. Finally, the protocol client will unlock CDB1 using the locking protocol on the second session (not depicted) so that other synchronization activity can occur for CDB1.

```
ROLLBACK TRANSACTION
```

4.1.4 New user only synchronization

This example describes synchronization scenario of user profile data for new security principals in a site collection. The protocol client can accomplish this by performing the outlined steps on a periodic basis (for example, every five minutes) for each content database.

In this example, steps 1 through 5 occur in the specified order. The following actions happen.

```
----  Proc profilesynch_sweep_GetDBToken    ---------------->
<---  SweepSynch GetChangeToken Result Set  -----------------
<---  return code                           -----------------
----  Proc profilesynch_sweep_UpdateDBToken ---------------->
<---  return code                           -----------------
```

1. The protocol client begins by calling the stored procedure **profilesynch_sweep_GetDBToken** to get the current change token for the content database. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure.

```
exec profilesynch_sweep_GetDBToken
@ContentDBID='F2179717-1115-4549-9728-EA0EC8ED6069'
```

2. At this point, the protocol server returns a single row matching the **ChangeTokenQuick** field from the **Content Database Synchronization Data** record corresponding to *@ContentDBID.* The example assumes that a previous quick synchronization has occurred before and the following result set is returned to the client.

ChangeToken
1;0;f2179717-1115-4549-9728-ea0ec8ed6069;633416658008370000;7234

1. A return code of 0 is sent back to the protocol client indicating a successful execution of the stored procedure.

2. The protocol client will now use this change token to determine if there are any new security principals that need synchronization. If there are, their user profile data will be obtained using [MS-UPSPROF] and replicated to the user information list. To finish the synchronization, the protocol client then calls the stored procedure **profilesynch_sweep_UpdateDBToken** using the **ChangeToken** from step 2. The T-SQL syntax for the call to this stored procedure is.

```
exec dbo.profilesynch_sweep_UpdateDBToken
   @ContentDBID='F2179717-1115-4549-9728-EA0EC8ED6069',
   @ChangeToken=N'1;0;f2179717-1115-4549-9728-
      ea0ec8ed6069;633416668484370000;7237'
```

3. A return code of 0 is returned to the protocol client, indicating a successful execution of the stored procedure.

4.2 Backup and Restoration

The following identifiers are used in the examples in this section:

- CDB1 - 6D070178-F511-4F22-9229-2A9CF739B525
- CDB2 - 8364ffa6-a96e-4f97-9174-a93d96ea9570
- SC1 - 927CB7B9-5546-4660-AB0A-11964F983C04
- SC2 - 927CB7B9-5546-4660-AB0A-11964F983C04

Suppose that originally a content database (CDB1) had two site collections (SC1 and SC2). Synchronization has been executing successfully between CDB1 and its associated user profile store.

4.2.1 Site Collection Moves Between Content Databases

Now suppose another content database (CDB2) is created and associated with SSP1 and SC1 is backed up from CDB1 and restored into CDB2 in such a way that the GUID for SC1does not change during the restore.

Now there are two site collections (SC1 and SC2) that will attempt to synchronize with SSP1 but they both have the same site collection identifier. At some point SC1 will probably be deleted but depending on when synchronizations happen there may be a temporary state where data for SC1 exists in SSP1 while SC2 is attempting to synchronize.

4.2.1.1 Pre-move notification

This example illustrates the scenario where the protocol server is notified that SC1 will move to CDB2.

1. Prior to making changes to the date for CDB1, the locking interface is used on a separate connection.

```
set LOCK_TIMEOUT 0
if not exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[ContentDBLock6D070178-F511-4F22-9229-2A9CF739B525]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
BEGIN
    CREATE TABLE [dbo].[ContentDBLock6D070178-F511-4F22-9229-2A9CF739B525] (
        [Lock] [bit],
    ) ON [PRIMARY]
    insert into [ContentDBLock6D070178-F511-4F22-9229-2A9CF739B525] (Lock)
values (0)
END
BEGIN TRANSACTION
update [ContentDBLock6D070178-F511-4F22-9229-2A9CF739B525] set Lock=1
```

2. The protocol then prepares to move the content database by calling **profilesynch_PrepareToMove**. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure.

```
exec dbo.profilesynch_PrepareToMove
   @ContentDBID ='6D070178-F511-4F22-9229-2A9CF739B525',
   @SiteID = '927CB7B9-5546-4660-AB0A-11964F983C04'
```

3. A return code of 0 is sent back to the protocol client indicating a successful execution.

4. The connection held open by the locking interface is notified that the synchronization activities are finished for CDB1.

```
ROLLBACK TRANSACTION
```

As documented in **profilesynch_PrepareToMove**, doing this will result in no further synchronization occurring for SC1 until **profilesynch_RegisterSitesToSynch** is called with the CDB2, SC1.

If an error were made (for example the wrong site collection identifier was used), the preceding steps can be repeated with the modification that *@Undo* = 1 be an additional parameter in step 2. This will undo the effects of the preceding sequence.

4.2.1.2 Post-move notification

If the backup/restore of SC1 has already occurred and the pre-move notification steps were not taken, the protocol client will do the following:

1. Prior to making changes to the data for CDB2, the locking interface is used on a separate connection.

```
set LOCK_TIMEOUT 0
if not exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[ContentDBLock8364ffa6-a96e-4f97-9174-a93d96ea9570]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
BEGIN
    CREATE TABLE [dbo].[ContentDBLock8364ffa6-a96e-4f97-9174-a93d96ea9570] (
        [Lock] [bit],
    ) ON [PRIMARY]
    insert into [ContentDBLock8364ffa6-a96e-4f97-9174-a93d96ea9570] (Lock)
values (0)
END
BEGIN TRANSACTION
update [ContentDBLock8364ffa6-a96e-4f97-9174-a93d96ea9570] set Lock=1
```

2. The protocol then prepares to move the content database by calling **profilesynch_PrepareToMove**. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure.

```
exec dbo.profilesynch_PrepareToMove
   @ContentDBID ='8364ffa6-a96e-4f97-9174-a93d96ea9570',
   @OldContentDBID ='6D070178-F511-4F22-9229-2A9CF739B525',
   @SiteID = '927CB7B9-5546-4660-AB0A-11964F983C04'
```

3. A return code of 0 is sent back to the protocol client indicating a successful execution.

4. The connection held open by the locking interface is notified that the synchronization activities are finished for CDB2.

```
ROLLBACK TRANSACTION
```

5 Security

5.1 Security Considerations for Implementers

Interactions with SQL are susceptible to tampering and other forms of security risks. Implementers are advised to sanitize input parameters for stored procedures prior to invoking the stored procedure.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

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

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

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

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

N

O

P

R

S

T

V