[MS-UPSSYNC2]:
User Profile Synchronization Stored Procedures Version 2 Protocol Specification

---

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
07/13/2009	0.1	Major	Initial Availability
08/28/2009	0.2	Editorial	Revised and edited the technical content
11/06/2009	0.3	Editorial	Revised and edited the technical content
02/19/2010	1.0	Major	Updated and revised the technical content
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Editorial	Revised and edited the technical content
06/29/2010	1.04	Major	Significantly changed the technical content.
07/23/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
04/11/2012	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	1.04	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies the User Profile Synchronization Stored Procedures, 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.

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.

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

1.1 Glossary

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

Coordinated Universal Time (UTC)
Security Support Provider Interface (SSPI)

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

back-end database server
change token
content database
member
member group
membership
profile subtype
request identifier
result set
return code
security group
security principal
Shared Services Provider (SSP)
site
site collection
stored procedure
Structured Query Language (SQL)
SystemID
Transact-Structured Query Language (T-SQL)
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

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

1.2.1 Normative References

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

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

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

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

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

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

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

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

1.2.2 Informative References

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

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

1.3 Protocol Overview (Synopsis)

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 (1)** 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-WSSFO2] 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:

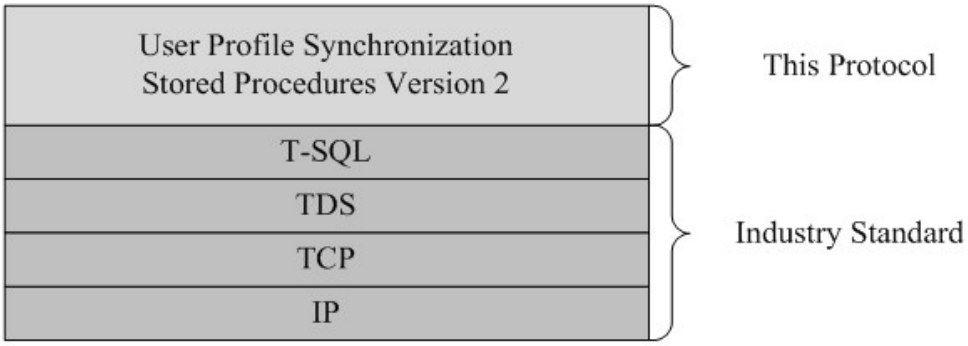

Figure 1: This protocol in relation to other protocols

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 (1) which will need to be synchronized with the user profile data.
- An additional one million site collections with fewer than 10 security principals (1) 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 **Structured Query Language (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

2.2.1 Simple Data Types and Enumerations

None.

2.2.2 Bit Fields and Flag Structures

None.

2.2.3 Binary Structures

None.

2.2.4 Result Sets

2.2.4.1 OldDatabase Result Set

The **OldDatabase** result set returns a list of content databases that have not been synchronized in number of days specified. The **OldDatabase** result set MUST contain 0 or more rows.

```
ID uniqueidentifier NOT NULL,
LastSynch DateTime NOT NULL,
```

ID: The identifier of the content database.

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

2.2.4.2 SitestoSynchronize Result Set

The **SitesToSynchronize** result set returns a list of site collections due for synchronization for the content database specified. The **SitesToSynchronize** result set MUST contain 0 or more rows.

```
ContentDBID uniqueidentifier NOT NULL,
SiteID uniqueidentifier NOT NULL,
LastSynch datetime NOT NULL,
ChangeToken ntext NOT NULL,
SchemaVersion int NOT NULL,
LastChangeSynchSuccess bit NOT NULL,
Moving bit NOT NULL,
MovingDeleted bit NOT NULL,
Registered bit NOT NULL,
PartitionID uniqueidentifier NOT NULL,
HasProfileChanges bit NOT NULL,
```

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

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 ignored.

MovingDeleted: The **MovingDeleted** field from the **Site Collection Synchronization Data** for **SiteID** MUST be ignored.

Registered: Specifies The **Registered** field from the **Site Collection Synchronization Data** for **SiteID** MUST be ignored.

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

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** (*PartitionID, ContentDBID, SiteID, 0, 0,DBTime*) would return 1 or more rows (where *PartitionID* is the **PartitionID** of the current row, *ContentDBID* is the **ContentDBID** of the current row, *SiteID* is the **SiteID** of the current row, and *DBTime* is the output parameter of the stored procedure). Otherwise, it MUST be 0.

2.2.4.3 UnregisteredSites Result Set

The **UnregisteredSites** result set returns a list of all site collections that are not registered for synchronization for the content database specified. The **UnregisteredSites** result set MUST contain 0 or more rows.

```
SiteID uniqueidentifier NOT NULL,
```

SiteID: The identifier of a site collection in the content database.

2.2.4.4 MembershipSynchronizationGroups Result Set

The **MembershipSynchronizationGroups** result set returns a list of all security groups in the site collection specified. The **MembershipSynchronizationGroup** MUST contain 0 or more rows.

```
GroupID int NOT NULL,
```

GroupID: The identifier of a **security group** that is a **member (1)** of the site collection identified by SiteID according to the data stored in GroupSite Membership Data.

2.2.4.5 StartSynchChangeToken Result Set

The **StartSynchChangeToken** result set returns the requested current change token representing the last change that was synchronized in change log of the specified content database.

```
CurrentChangeToken ntext,
```

CurrentChangeToken: The **change token** representing the last change that was synchronized in the change log of the specified content database.

2.2.4.6 SweepSynchGetChangeToken Result Set

The **SweepSynchGetChangeToken** result set returns the requested change token.

```
ChangeToken  ntext,
```

ChangeToken: The change token representing the last change in a specified content database's change log that was synchronized with "quick" synchronization**.** A quick synchronization is lightweight synchronizations which only push user profile data for new security principals (1) in the content database,

2.2.4.7 UserSynchronization Result Set

The **UserSynchronization** result set returns available properties for the users specified. The **UserSynchronization** result set MUST contain 0 or more rows of property values.

```
RecordId bigint NOT NULL,
ProfileSubtypeId int NOT NULL,
PropertyId bigint NOT NULL,
PropertyVal sql_variant NOT NULL,
Text ntext,
OrderRank int,
Privacy int,
WssId int NOT NULL,
PropertyName nvarchar(250) NOT NULL,
PropertyURI nvarchar(250) NOT NULL,
```

RecordId: The record identifier for the user. RecordId MUST NOT be NULL.

ProfileSubtypeId: A **profile subtype** identifier.

PropertyId: The identifier of the property. PropertyId MUST NOT be NULL.

PropertyVal: The value of the property that is specified by PropertyId. PropertyVal MUST NOT be NULL.

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

OrderRank: The order that property values are returned for multi-valued properties. The result set MUST include one row for each property value the user has for a multi-valued property, ordered from lowest to highest **OrderRank**.

Privacy: The privacy policy value.

WssId: The identifier of the security principal (1).

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

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

2.2.5 Tables and Views

None.

2.2.6 XML Structures

No common XML Structures are defined in this protocol.

2.2.6.1 Namespaces

No common XML Structures are defined in this protocol.

2.2.6.2 Simple Types

No common XML Structures are defined in this protocol.

2.2.6.3 Complex Types

No common XML Structures are defined in this protocol.

2.2.6.4 Elements

No common XML Structures are defined in this protocol.

2.2.6.5 Attributes

No common XML Structures are defined in this protocol.

2.2.6.6 Groups

No common XML Structures are defined in this protocol.

2.2.6.7 Attribute Groups

No common XML Structures are defined in this protocol.

3 Protocol Details

3.1 Profile Synchronization Server 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.

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

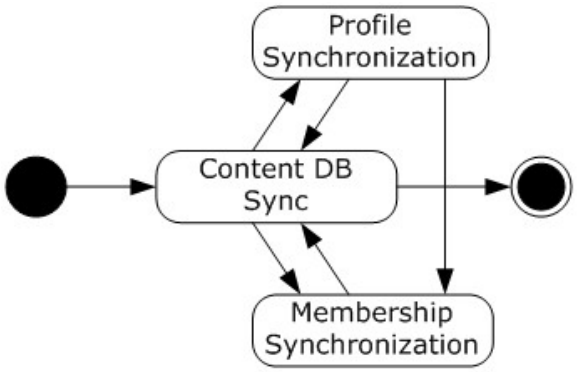

Figure 2: Profile synchronization state transition diagram

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 column headers in the following table are used to represent the synchronization states shown in the preceding figure:

- 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 preceding 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_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

The protocol client MUST send the initialization message specified in section 3.1.3 before any stored procedures are called that 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 (2)** memberships associated with each user profile.

3.1.1.2 User Data

The protocol server maintains information about all the security principals (1) that are encountered during synchronization and the mapping of those security principals (1) to user profiles. This data includes:

- **SiteID:** The site collection the security principal (1) belongs to.
- **WSSID:** The identifier of the security principal (1).
- **SID**: The **SystemID** for the security principal (1) identified by **WSSID**.
- **PartitionID:** A GUID used to filter the current request.

3.1.1.3 Membership Data

This data contains relationships between security principals (1) and security groups and between security groups and sites.

3.1.1.3.1 UserGroup Data

The UserGroup data tracks which security principals (1) 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 (1) whose security group is **GroupID**.

3.1.1.3.2 GroupSite Data

The GroupSite data tracks the member groups for sites (2). This data includes:

- **SiteID**: The site collection for this security group.
- **GroupID**: The identifier of the security group.
- **WebID**: The site (2) whose member 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 is persisted to the store.

The following operations on the staging data are defined:

- **User Data** operations
  - **DeleteUserSiteColRel** (*U, SC*): Upon flush, **User Data** records will be deleted as needed to represent that security principal (1) U is no longer a security principal (1) of site collection SC 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 (2) S.
  - **DeleteSite**(*S*): Upon flush, all records for site (2) 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 (2) S and that GOld (the former members group of site (2) S) is no longer the members group of site (2) S.
  - **AddUserGroupRel**(*U,G*): Upon flush, **UserGroup** records will be added as necessary to represent that security principal (1) U is in security group G.
  - **DeleteUserGroupRel**(*U,G*): Upon flush, **UserGroup** records will be deleted as necessary to represent that security principal (1) 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:

- **SiteID:** The identifier of the site collection.
- **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 collection 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.
- **PartitionID:** A GUID used to filter the current request.

- **Content Database Synchronization Data:** Records containing data about content databases that are synchronized with the user profile store. This data includes:
  - **ContentDBID**: The identifier of the content database.
  - **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 (lightweight synchronizations which only push user profile data for new security principals (1) in the content database) 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.
  - **PartitionID**: A GUID used to filter the current request.

3.1.1.6 Unambiguous Site Collection References

The protocol server will maintain data indicating the content database in which each site collection resides. If a site collection moves to a new content database the protocol client can register the site collection again using the new content database's identifier (section 3.1.5.14). The protocol client can also remove the site collection reference to the moved site collection with the old content database's identifier (section 3.1.5.1).

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 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

The following stored procedures are available as part of the protocol using standard **Transact-Structured Query Language (T-SQL)** 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 (1) 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 (1) 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 (1) is a member (2) of a security group on the site collection that is currently being synchronized.

Stored Procedure	Description
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.
profilesynch_MS_DeleteUserFromGroup	Deletes data in the user profile store regarding a security principal being a member (2) 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 (2) 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 (2) in the site collection that is currently being synchronized.
profilesynch_RegisterSitesToSynch	Changes data in the user profile store to know about these site collections and expect to synchronize them 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	Returns a change token representing the last change in the change log of the specified content database that was synchronized, indicating that the protocol client is about to begin the synchronization of the specified content database with this user profile store.
profilesynch_StartFullSiteSynch	Signals that the protocol client is about to send all of the security principal (1) data for the specified site collection in the content database.
profilesynch_SuccessfulContentDBSynch	Signals that the protocol client has finished sending data for the specified content database with no unrecoverable errors.
profilesynch_SuccessfulSiteChangeLogConsumption	Signals that the protocol client has finished sending data about the security principal (1) after consuming the change log for the specified site collection and encountered no unrecoverable errors**.**
profilesynch_SuccessfulSiteProfilePush	Signals that the protocol client has successfully received all user profile data and replicated it to the security principal (1) for the specified site collection.
profilesynch_sweep_GetDBToken	Returns the change token that represents the last change in the change log of a particular content database that was synchronized.

Stored Procedure	Description
profilesynch_sweep_UpdateDBToken	Changes the change token that represents the last change in the change log of a particular content database' that was synchronized**.**
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 (1) and the matching user profile data exist on the site collection currently being synchronized.
profilesynch_US_IncrementalSynch	Returns profile data about security principals (1) whose corresponding profiles have changed since the last synchronization with the site collection.
profilesynch_RegisterSiteToSynch	Changes data in the user profile store to register this site collection to be synchronized.

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 2.2.2.7.

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.

3.1.5.1 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(
  @partitionID    uniqueidentifier,
  @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',
  @correlationId  uniqueidentifier = NULL
);
```

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

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

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

@correlationId: The optional **request identifier** for the current request. The value MUST be set to NULL and MUST be ignored by the server.

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#* and @partitionId MUST be deleted and all the data in the content database identified by *@ContentDBID* MUST be deleted. This includes **Site Collection Synchronization Data, User Data, Membership Data,** and **SiteMembershipList** records.

Return Values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: MUST NOT return any result sets.

3.1.5.2 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(
  @partitionID         uniqueidentifier,
  @ContentDBID         uniqueidentifier,
  @correlationId       uniqueidentifier
);
```

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

@ContentDBID: The identifier of a content database.

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

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

Based on **Site Collection Synchronization Data**, the set of site collections in the content database is identified by *@ContentDBID* and *@partitionID*. 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 (*@partitionID, @ContentDBID, @SiteID, @correlationId*) 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** for the specified *@partitionID*, then no data is to be modified.

Return Values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: MUST NOT return any result sets.

3.1.5.3 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(
  @partitionID              uniqueidentifier,
  @ContentDBID              uniqueidentifier,
  @SiteID                   uniqueidentifier,
  @correlationId            uniqueidentifier = NULL
);
```

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

@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.

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

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

The protocol server MUST update the **LastChangeSynchSuccess** field in the **Site Collection Synchronization Data** to false for the failing site collection where **PartitionID** matches *@partitionID*. 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 Values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: MUST NOT return any result sets.

3.1.5.4 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(
  @partitionID          niqueidentifier,
  @Days                 int,
  @correlationId        uniqueidentifier = NULL
);
```

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

@Days: The threshold number of days.

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

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

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 where **PartitionID** matches *@partitionID* and the **LastSynch** value is prior to the value obtained through subtraction.

Return Values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: This procedure MUST return an OldDatabase.

3.1.5.5 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(
  @partitionID    uniqueidentifier,
  @ContentDBID    uniqueidentifier,
  @correlationId  uniqueidentifier = NULL
);
```

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

@ContentDBID: The identifier of a content database.

@correlationId: The optional request identifier for the current request. The value MUST be set to NULL and MUST be ignored by the server.

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

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

Return Values:

Value	Description
0	The requested was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: This procedure MUST return a SitesToSynchronize.

3.1.5.6 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(
  @partitionID     uniqueidentifier,
  @ContentDBID     uniqueidentifier,
  @correlationId   uniqueidentifier = null
);
```

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

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

@correlationId: The optional request identifier for the current request. The value MUST be set to NULL and MUST be ignored by the server.

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

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

Return Values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: This procedure MUST return an UnregisteredSites.

3.1.5.7 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 (1) 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(
  @partitionID    uniqueidentifier,
  @ContentDBID    uniqueidentifier,
  @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,
  @correlationId  uniqueidentifier = NULL
);
```

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

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

@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 (1). 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 (1).

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

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

All data related to each *@WssID#* specified in the parameters MUST be processed in a manner equivalent to profilesynch_MS_AddUserToGroup(*@partitionID, @ContentDBID, @SiteID, @GroupID, @WssID, @correlationId*).

Return Values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: MUST NOT return any result sets.

3.1.5.8 profilesynch_MS_AddUserToGroup

The **profilesynch_MS_AddUserToGroup** stored procedure is called to request that the protocol server update its data to reflect that a security principal (1) is a member (2) 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_AddUserToGroup (
@contentDBID uniqueidentifier,
@partitionID uniqueidentifier,
@SiteID uniqueidentifier,
@GroupID int,
@WssID int,
@correlationId uniqueidentifier = null,
);
```

@ContentDBID: The identifier of the content database. This value MUST NOT be null or empty.

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

@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 (1) in the security group identified by *@GroupID***.**

@correlationID: The optional request identifier for the current request. The value MUST be set to NULL and MUST be ignored by the server.

Return Code Values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: MUST NOT return any result sets.

3.1.5.9 profilesynch_MS_DeleteGroup

The **profilesynch_MS_DeleteGroup** stored procedure is called to request that the protocol server update its data to delete all site (2) memberships that depend on the specified security group.

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

```
PROCEDURE profilesynch_MS_DeleteGroup (
@partitionID uniqueidentifier,
@ContentDBID uniqueidentifier,
@SiteID uniqueidentifier,
@GroupID int,
@correlationId uniqueidentifier = null,
);
```

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

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

@SiteID: The identifier of a site collection.

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

@correlationID: The optional request identifier for the current request. The value MUST be set to NULL and MUST be ignored by the server.

For each relationship found to exist between security principals (1) and the security group identified by *@GroupID* in the **Membership Data** and **Staging Data**, **DeleteUserGroupRel** (***WSSID, @GroupID***) MUST be performed on the **Staging Data**. **WSSID** is the integer identifier of the security principal (1) that is a member (2) of the security group identified by *@GroupID*.

Return Code Values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: MUST NOT return any result sets.

3.1.5.10 profilesynch_MS_DeleteUserFromGroup

The **profilesynch_MS_DeleteUserFromGroup** stored procedure is called to request that the protocol server delete all site (2) memberships that depend on the specified security principal (1)'s membership in the specified security group**.**

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

```
PROCEDURE profilesynch_MS_DeleteUserFromGroup (
@partitionID uniqueidentifier,
@WssID int,
@SiteID uniqueidentifier,
@ContentDBID uniqueidentifier,
@GroupID int,
```

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

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

@WssID: The identifier of a security principal (1) in the site collection identified by *@SiteID* and *@PartitionID.*

@SiteID: The identifier of a site collection.

@ContentDBID: The identifier of a content database.

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

@correlationID: The optional request identifier for the current request. The value MUST be set to NULL and MUST be ignored by the server.

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

Return Code Values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: MUST NOT return any result sets.

3.1.5.11 profilesynch_MS_DeleteWeb

The **profilesynch_MS_DeleteWeb** stored procedure is called to request that the protocol server delete all site (2) memberships that depend on the specified site (2).

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

```
PROCEDURE profilesynch_MS_DeleteWeb (
@partitionID uniqueidentifier,
@WebID uniqueidentifier,
@correlationId uniqueidentifier = null,
);
```

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

@WebID: The identifier of a site (2).

@CorrelationID: The optional request identifier for the current request. The value MUST be set to NULL and MUST be ignored by the server.

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

Return Code Values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: MUST NOT return any result sets.

3.1.5.12 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 (
@partitionID uniqueidentifier,
@ContentDBID uniqueidentifier,
@SiteID uniqueidentifier,
@correlationId uniqueidentifier = null,
);
```

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

@ContentDBID: The identifier of a content database.

@SiteID: The identifier of a site collection.

@correlationId: The optional request identifier for the current request. The value MUST be set to NULL and MUST be ignored by the server.

The **GroupSite Membership Data** MUST be consulted and a row returned for each **GroupID** that is a member (2) of the site collection (identified by *@SiteID*).

Return Code Values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: This procedure MUST return a MembershipSynchronizationGroups.

3.1.5.13 profilesynch_MS_UpdateWeb

The **profilesynch_MS_UpdateWeb** stored procedure is called to request that the protocol server update its data regarding a site (2) in the site collection**.**

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

```
PROCEDURE profilesynch_MS_UpdateWeb (
@contentDBID uniqueidentifier,
@partitionID uniqueidentifier,
@SiteID uniqueidentifier,
```

```
@WebID uniqueidentifier,
@GroupID int,
@WebName nvarchar(250),
@WebURL nvarchar(2048),
@UnknownGroup bit = null OUTPUT,
@correlationId uniqueidentifier = null,
);
```

@ContentDBID: The identifier of a content database. This value MUST NOT be null or empty.

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

@SiteID: The identifier of a site collection.

@WebID: The identifier of a site (2) in the site collection identified by *@SiteID* and *@partitionID*.

@GroupID: The identifier of the members group of the site (2) identified by *@WebID* and *@partitionID*. A null value indicates that the site (2) identified by *@WebID* and *@PartitionID* is not associated with a members group.

@WebName: The name of the site (2) identified by *@WebID* and *@partitionID*.

@WebURL: The URL of the site (2) identified by *@WebID* and *@partitionID*.

@UnknownGroup: An output parameter bit indicating whether the user profile store has records identifying the security principals (1) that are in the security group identified by the *@GroupID*.

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

The **SiteMembershipList** MUST be updated to reflect the new data regarding the site (2) identified by *@WebID*. This includes:

- The name of the site (2) (identified by *@WebName*)
- The URL of the site (2) (identified by *@WebURL*)

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

Otherwise, if the **GroupSite Membership Data** has a record of a previous relationship between the site (2) 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 (2) identified by *@WebID* in the site collection identified by *@SiteID*,*@partitionID and @contentDBID***)**.

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

If the security principals (1) 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:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: MUST NOT return any result sets.

3.1.5.14 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 (
@partitionID uniqueidentifier,
@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,
@correlationId uniqueidentifier = null,

);
```

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

@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.

@correlationId: The optional request identifier for the current request. This value MUST be set to NULL and MUST be ignored by the server.

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 both *@SiteID#* and *@partitionID*
  - If the **Moving** field of the **Site Collection Synchronization Data** is set to true for the record corresponding to both *@SiteID#* and @partitionID, 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 both *@SiteID#* and *@partitionID*, 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 both *@SiteID#* and *@partitionID*, 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*
PartitionID	*@partitionID*
Registered	true
Moving	false
MovingDeleted	false
LastSynch	null
LastChangeSynchSuccess	false
ChangeToken	null
SchemaVersion	0

Return values:

Value	Description
0	The request was finished successfully.
-1	For one of the specified *@SiteID#* parameters, a site collection was found corresponding to the specified *@partitionID* and *@SiteID#*, however the **ContentDBID** field of the **Site Collection Synchronization Data** is different than the value of *@ContentDBID*, and the **Moving** field is set to false.
Positive integer number	A T-SQL error code.

Result Sets: MUST NOT return any result sets

3.1.5.15 profilesynch_ScheduleFullSiteSynch

The **profilesynch_ScheduleFullSiteSynch** stored procedure is called to request that the next synchronization of this site collection is 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 (
@partitionID uniqueidentifier,
@ContentDBID uniqueidentifier,
@SiteID uniqueidentifier,
@correlationId uniqueidentifier = null,

);
```

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

@ContentDBID: The identifier of a content database.

@SiteID: The identifier of the site collection.

@correlationId: The optional request identifier for the current request. This value MUST be set to NULL and MUST be ignored by the server.

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

Field	Value
PartitionID	*@partitionID*
LastSynch	null
LastChangeSynchSuccess	false
ChangeToken	null

Return values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: MUST NOT return any result sets.

3.1.5.16 profilesynch_StartContentDBSynch

The **profilesynch_StartContentDBSynch** stored procedure is called to request that a change token be returned representing the last change in the specified content database's change log that was synchronized 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 (
@partitionID uniqueidentifier,
@ContentDBID uniqueidentifier,
@correlationId uniqueidentifier = null,
```

);

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

@ContentDBID: The identifier of a content database.

@correlationId: The optional request identifier for the current request. This value MUST be set to NULL and MUST be ignored by the server.

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 both *@ContentDBID* and *@partitionId*. 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 values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets:

This procedure MUST return a StartSynchChangeToken.

3.1.5.17 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 (
@partitionID uniqueidentifier,
@ContentDBID uniqueidentifier,
@SiteID uniqueidentifier,
@DBTime datetime = null OUTPUT,
@correlationId uniqueidentifier = null,
```

);

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

@ContentDBID: The identifier of a content database.

@SiteID: The identifier of the site collection.

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

@correlationId: The optional request identifier for the current request. This value MUST be set to NULL and MUST be ignored by the server.

The protocol server MUST perform the following operations on the **Staging Data** such that all data stored in the **User Data** and **Membership Data** areas regarding the site collection identified by *@SiteID*, *@ContentDBID* and *@PartitionID* are slated for deletion:

- **DeleteUserSiteColRel**(**WSSID**, *@SiteID*) operations MUST be performed for all existing relationships found between a security principal (1) with integer identifier **WSSID** and the site collection.
- **DeleteUserGroupRel**(**WSSID**, **GroupID**) operations MUST be performed for all existing relationships found between a security principal (1) with integer identifier **WSSID** and a security group with integer identifier **GroupID** that belongs to the site collection.
- **DeleteSite**(**WebID**) operations MUST be performed for all existing relationships found between a site (2) with identifier **WebID** and the site collection.

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

Return values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: MUST NOT return any result sets.

3.1.5.18 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 (
@partitionID uniqueidentifier,
@ContentDBID uniqueidentifier,
@TargetChangeToken ntext,
@correlationId uniqueidentifier = null,

);
```

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

@ContentDBID: The identifier of the content database.

@TargetChangeToken: A change token representing the most recent change in the change log of the content databasethat was synchronized.

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

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

Field	Value
ChangeTokenFull	*@TargetChangeToken*

Field	Value
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 and @partitionID*) to match *@TargetChangeToken*.

Return values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: This procedure MUST NOT return any result sets.

3.1.5.19 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 (
@partitionID uniqueidentifier,
@ContentDBID uniqueidentifier,
@SiteID uniqueidentifier,
@TargetChangeToken ntext,
@correlationId uniqueidentifier = null,

);
```

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

@ContentDBID: The identifier of a content database.

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

@TargetChangeToken: A change token representing the last change in the change log of the content databasethat was synchronized.

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

The following actions MUST occur:

- The **Staging Data** MUST be flushed to **User Data** and **Membership Data** for the site collection identified by *@SiteID,@partitionID* and *@ContentDBID*.
- 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 (1) maps to a security group through **UserGroup** data and that same security group maps to a site (2) through the **GroupSite** data, the **SiteMembershipList** for the user profile corresponding to that security principal (1) MUST contain an entry for that site (2) membership.
- If during the processing of this stored procedure, such a chain (security principal (1) -> members group -> site (2)) exists in the data where it did not exist previously, an entry for the site (2) MUST be added to the **SiteMembershipList** of the user profile corresponding to that security principal (1).
- Conversely, if during the processing of this stored procedure, such a chain (security principal (1) -> members group -> site (2)) 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 (1) -> members group -> site (2)) is connected through a new intermediate step (for example, the members group for a site (2) is changed but the same security principal (1) is a member (2) 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 (2) Membership maintained by the user profile store (see [MS-UPSPROF2]) MUST not be deleted or changed.

- The **Site Collection Synchronization Data** for the site collection identified by *@SiteID,@partitionID* and *@ContentDBID* MUST be updated as follows.

Field	Value
ChangeToken	*@TargetChangeToken*
LastChangeSynchSuccess	1

Return values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: This procedure MUST NOT return any result sets.

3.1.5.20 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 (
@partitionID uniqueidentifier,
@ContentDBID uniqueidentifier,
@SiteID uniqueidentifier,
@StartSynchTime datetime,
@SchemaVersion int,
```

```
  @correlationId uniqueidentifier = null,
```

);

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

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

@correlationId: The optional request identifier for the current request. The value MUST be set to NULL and MUST be ignored by the server.

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

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

Return values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: This procedure MUST NOT return any result sets.

3.1.5.21 profilesynch_sweep_GetDBToken

The **profilesynch_sweep_GetDBToken** stored procedure is called to request a change token that represents the last change in the change log of the content database that was synchronized during a "quick" synchronization. Quicl synchronization is a lightweight synchronization that only pushes user profile data for new security principals (1) in the content database)**.**

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

```
PROCEDURE profilesynch_sweep_GetDBToken (
@partitionID uniqueidentifier,
@ContentDBID uniqueidentifier,
@correlationId uniqueidentifier = null,
```

);

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

@ContentDBID: The identifier of a content database.

@correlationId: The optional request identifier for the current request. The value MUST be set to NULL and MUST be ignored by the server.

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 values:

Value	Description
0	The request was finished successfully.

Result Sets: This procedure MUST return a SweepSynchGetChangeToken:

3.1.5.22 profilesynch_sweep_UpdateDBToken

The **profilesynch_sweep_UpdateDBToken** stored procedure is called to request that the server store a change token representing the most recent change in the change log of the content database that was synchronized during the quick synchronization**.**

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

```
PROCEDURE profilesynch_sweep_UpdateDBToken (
@partitionID uniqueidentifier,
@ContentDBID uniqueidentifier,
@ChangeToken ntext,
@correlationId uniqueidentifier = null,
```

);

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

@ContentDBID: The identifier of a content database.

@ChangeToken: A change token representing the most recent change in the change log of the content database that was synchronized during the quick synchronization**.**

@correlationId: The optional request identifier for the current request. The value MUST be set to NULL and MUST be ignored by the server.

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

Return values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: This procedure MUST NOT return any result sets.

3.1.5.23 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(
  @partitionID      uniqueidentifier,
  @ContentDBID      uniqueidentifier,
  @correlationId    uniqueidentifier = NULL
);
```

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

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

@correlationId: The optional request identifier for the current request. This value MUST be set to NULL and MUST be ignored by the server.

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

Return values:

Value	Description
Nonnegative integer number	A T-SQL error code.

Result Sets: MUST NOT return any result sets.

3.1.5.24 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 (1) in the site collection currently being synchronized.

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

```
PROCEDURE profilesynch_US_AddProfilesToSynch(
  @partitionID    uniqueidentifier,
  @ContentDBID    uniqueidentifier,
  @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,
    @correlationId  uniqueidentifier = null
  );
```

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

@ContentDBID: The identifier of a content database.

@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#: The identifier of a security principal (1) in the site collection identified by @SiteID and @partitionID 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 (1) identifier**.**

@correlationId: The optional request identifier for the current request. This value MUST be set to NULL and MUST be ignored by the server.

The user profiles for all requested security principals (1) 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 (1) 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:

Field	Value
PartitionID	*@partitionID*
ContentDBID	@ContentDBID
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 MUST themselves be deleted.

Return values:

Value	Description
0	The request was finished successfully.
Positive integer number	A T-SQL error code.

Result Sets: This procedure MUST return a UserSynchronization.

3.1.5.25 profilesynch_US_IncrementalSynch

The **profilesynch_US_IncrementalSynch** stored procedure is called to request that the protocol server return user profile data about security principals (1) 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(
  @partitionID           uniqueidentifier,
  @ContentDBID           uniqueidentifier,
  @SiteID                uniqueidentifier,
  @MinNonInclusiveWssID  int,
  @AllProfiles           bit = 0,
  @DBTime                DateTime OUTPUT,
  @correlationId         uniqueidentifier = NULL
);
```

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

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

@SiteID: The identifier of a site collection.

@MinNonInclusiveWssID: Used to page the **WSSID** field of the **User Data**.

@AllProfiles: Indicates whether all profiles MUST be returned or only those that have changed.

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

@correlationId: The optional request identifier for the current request. This value MUST be set to NULL and MUST be ignored by the server.

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 this procedure MUST return all the user profiles that match the **User Data** records for the site collection, identified by *@SiteID* and *@partitionID*.
- If it is 0, then this procedure MUST return a subset of the user profiles that match the **User Data** records for the site collection, identified by *@SiteID* and *@partitionID*. 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.** If the **LastSynch** field is null, all the user profiles that match the **User Data** records for the site collection MUST be returned.

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 values: The stored procedure MUST return 0.

Result Sets: This procedure MUST return a UserSynchronization.

3.1.5.26 profilesynch_RegisterSiteToSynch

The **profilesynch_RegisterSiteToSynch** stored procedure is called to request that the user profile store register a site collection for synchronization.

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

```
PROCEDURE profilesynch_RegisterSiteToSynch (
@partitionID uniqueidentifier,
@ContentDBID uniqueidentifier,
@SiteID uniqueidentifier,
@correlationId uniqueidentifier = null,
);
```

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

@ContentDBID: The identifier of a content database. This value MUST NOT be null or empty.

@SiteID: The identifier of a site collection in the content database. This value MUST NOT be null or empty.

@correlationId: The optional request identifier for the current request. This value MUST be set to NULL and MUST be ignored by the server.

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

- If the **Site Collection Synchronization Data** already contains a record corresponding to *@SiteID*, *@partitionID and @ContentDBID*

  - The value of **Registered** MUST be set to true.

- Otherwise, if the **Site Collection Synchronization Data** does not contain a record corresponding to both *@SiteID, @partitionID and @ContenDBID*, 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*
PartitionID	*@partitionID*
SiteID	▪ *@siteID*
Registered	true
Moving	false
MovingDeleted	false
LastSynch	null

Field	Value
LastChangeSynchSuccess	false
ChangeToken	null
SchemaVersion	0

Return values:

Value	Description
0	The request was finished successfully.
-1	A site collection was found corresponding to the specified *@SiteID* and *@partitionID*, however the **ContentDBID** field of the **Site Collection Synchronization Data** is different than the value of *@ContentDBID*, and the **Moving** field is set to false.
Positive integer number	A T-SQL error code.

Result Sets: MUST NOT return any result sets.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Synchronization Locking Server Details

Section 3.1 specifies the interface used for synchronizing user profile and security principal (1) data between the user profile store and content databases. The intent of the synchronization locking interface is that it is 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 [ContentDBLockGUID0] (Lock) values (0)
END

BEGIN TRANSACTION
update [ContentDBLockGUID0] 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 2.2.2.7 if

- synchronization is in progress for the content database identified by **GUID0** and the synchronization does not terminate before the time indicated by **Int0** has elapsed.

The server MUST return a success if

- synchronization is not initially in progress for the content database identified by **GUID0.**
- synchronization is in progress for the content database identified by **GUID0** but it terminates before the time indicated by **Int0** has elapsed.

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 3.2.5.1).
- The connection used for synchronization is terminated.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 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.

3.2.5.1 Synchronization Termination

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

```
ROLLBACK TRANSACTION
```

Consistent with section 3.2.5, 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.

3.2.6 Timer Events

None.

3.2.7 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:** 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 Higher-Layer Triggered Events

None.

3.3.5 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 3.1.5 on the user profile store.

3.3.5.1 State Transitions

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

3.3.5.2 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.

3.3.5.3 Stored Procedure Specific Client Requirements

3.3.5.3.1 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**.

3.3.5.3.2 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-WSSFO2] MUST be used to update the schema of the user information list for the site collection identified by **SiteID** to match the **ReplicableSchema**.

3.3.5.3.3 profilesynch_StartFullSiteSynch

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

3.3.5.3.4 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.

3.3.5.3.5 profilesynch_US_IncrementalSynch

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

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

3.4 Synchronization Locking Client Details

None.

4 Protocol Examples

The section provides an example of the synchronization processes that are used in this protocol.

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, 0x0105000000000000515000000A065CF7E784B9B5FE77C87704D7A2100
  - SP2 – 8, 0x0105000000000000515000000A065CF7E784B9B5FE77C877088772100
  - SP3 – 9, 0x0105000000000000515000000A065CF7E784B9B5FE77C877081D00500
  - SP4 – 11, 0x0105000000000000515000000a065cf7e784b9b5fe77c877000311100
- Security Groups
  - SG1 – 5
  - SG2 – 7
  - SG3 – Not used
- PartitionID
  - EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C

In Figures 3, and 4 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 figure illustrates 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 the most recent change token for CDB1 (CT2) which has the value '1;0;cd56acc0-3e03-4264-b187-786a7b98d49d;633408552555600000;461'.

The following figure illustrates the overall flow for the example.

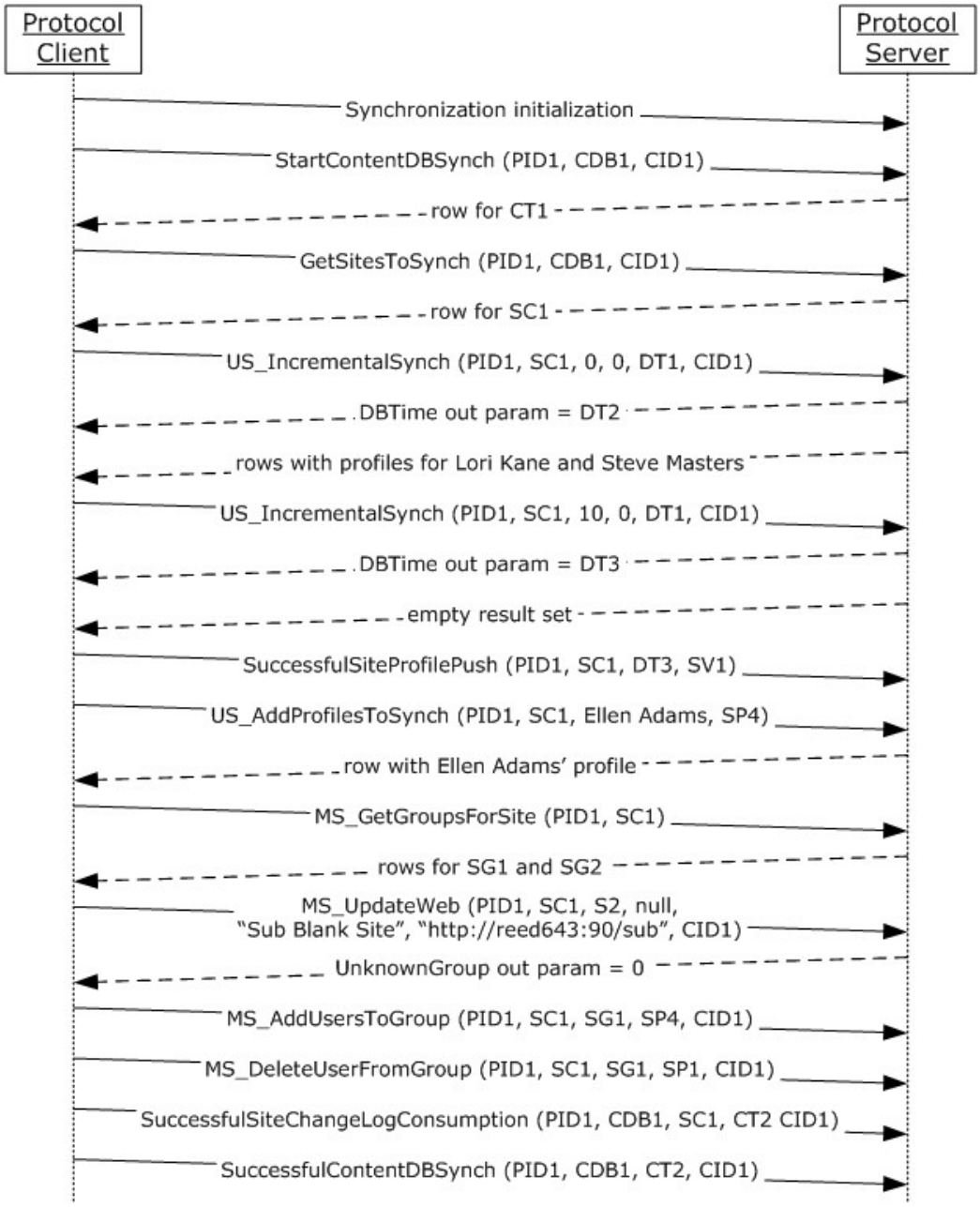

Figure 3: Full synchronization communication between the protocol client and protocol server

In detail:

Step 1: The protocol client starts off the synchronization process by using the locking protocol for the Content database CDB1 on another session with the **Shared Services Provider (SSP)** (not depicted in Figure 3) 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
```

Step 2: 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
    @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
    @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D'
```

Step 3: Because the protocol server has never synchronized this content database before, the **ChangeTokenFull** field from the **Content Database Synchronization Data** (CT1) 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.

Step 4: 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
    @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
    @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
    @SiteID0 = '595D079D-DB43-4403-8A1D-6DF10295FA75',
    @FailedSiteID = NULL
```

Step 5: 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.

Step 6: 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
    @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
    @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D'
```

Step 7: 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

LastChange SynchSuccess	Moving	MovingDeleted	Registered	PartitionID	HasProfileChanges
0	0	0	1	{EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C}	0

Step 8: 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 3.3.5.3.2.

Step 9: 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
    @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
    @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
    @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
    @DBTime = NULL
```

Step 10: 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'
```

Step 11: The protocol client sets **StartSynchTime** to the value of *@DBTime* as described in section 3.3.5.3.3. 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
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @SID0 = 0x010500000000000515000000a065cf7e784b9b5fe77c877088772100,
   @UID0 = 8,
   @SID1 = 0x010500000000000515000000a065cf7e784b9b5fe77c877081d00500,
   @UID1 = 9,
   @SID2 = 0x010500000000000515000000a065cf7e784b9b5fe77c87704d7a2100,
   @UID2 = 10
```

Step 12: 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

Text	OrderRank	Privacy	WSSID
NULL	NULL	NULL	10
NULL	NULL	NULL	10
NULL	NULL	NULL	10
NULL	NULL	NULL	10
NULL	NULL	NULL	10
NULL	NULL	NULL	8
NULL	NULL	NULL	8
NULL	NULL	NULL	8
NULL	NULL	NULL	8
NULL	NULL	NULL	8

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

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

Step 13: The protocol client then writes the user profile data it received to the user information list.

Step 14: 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
   @partitionID = ‘EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C’,
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @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
```

Step 15: 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 3.1.5.13) the *@UnknownGroup* output parameter is returned with a value of 1.

```
@UnknownGrou p = 1
```

Step 16: 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
   @partitionID = ‘EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C’,
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @GroupID = 5,
   @WssID0 = 8,
   @WssID1 = 10,
   @correlationId = NULL
```

Step 17: The protocol server appropriately updates the **Staging Data** and returns a 0 return code for successful execution.

Step 18: The preceding steps (15 – 18) 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
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @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
```

Step 19: Because the protocol server does not have data about SG2 yet, the *@UnknownGroup* output parameter is returned with a value of 1.

```
@UnknownGroup = 1
```

Step 20: 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
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @GroupID = 7,
   @WssID0 = 8,
   @WssID1 = 9,     @correlationId = NULL
```

Step 21: The protocol server appropriately updates the **Staging Data** and returns a 0 return code for successful execution.

Step 22: Assuming no errors are generated during this process, the protocol client then calls SuccessfulSiteProfilePush, passing **ReplicableSchemaVersion** and **StartSynchTime** as described in section 3.3.5.3.4. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure:

```
exec dbo.profilesynch_SuccessfulSiteProfilePush
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @StartSynchTime = 2008-03-11 18:01:18.466,
   @SchemaVersion = 1
```

Step 23: 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.

Step 24: 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 using CDB1's current change token (CT2) for the *TargetChangeToken* parameter. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure:

```
exec dbo.profilesynch_SuccessfulSiteChangeLogConsumption
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @TargetChangeToken = N'1;0;cd56acc0-3e03-4264-b187-
     786a7b98d49d;633408552555600000;461'
```

Step 25: The protocol server returns a 0 return code for successful execution. This call also changes the protocol state back to content database synchronization.

Step 26: Because there was only one site collection in CDB1, there is no remaining synchronization work. The protocol client calls **SuccessfulContentDBSynch** using CT2 for the *TargetChangeToken* parameter. Consider the following T-SQL syntax, which displays the parameters used to call this stored procedure:

```
exec dbo.profilesynch_SuccessfulContentDBSynch
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @TargetChangeToken = N'1;0;cd56acc0-3e03-4264-b187-
     786a7b98d49d;633408552555600000;461'
```

Step 27: The protocol server updates its synchronization data accordingly and returns a 0 return code for successful execution.

Step 28: 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. Additions are noted with the word "(addition)", **bold** highlights updates, while deletions are noted with the word "(deletion)".

- The user profile store contains six user profiles (P1 – P6) (addition):
  - 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 (addition)

- content database CDB1 contains 1 site collection
  - SC1 contains
    - Four security principals
      - SP1 – Lori Kane
      - SP2 – Sara Davis
      - SP3 – Steve Masters
      - SP4 – Ellen Adams (addition)
    - Two sites
      - S1 has members group G1
      - S2 has members group G1 (addition)
    - Two security groups
      - SG1 –SP2, and SP4 (addition) are members of this security group. SP1 (deletion) is no longer a member.
      - SG2 – SP2 and SP3 are members of this security group.
      - SG3 – SP1 (addition) 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 the most recent change token for CDB1 (CT2) which has the value '1;0;cd56acc0-3e03-4264-b187-786a7b98d49d;633408571893700000;520'.

The following figure illustrates the overall flow for the example.

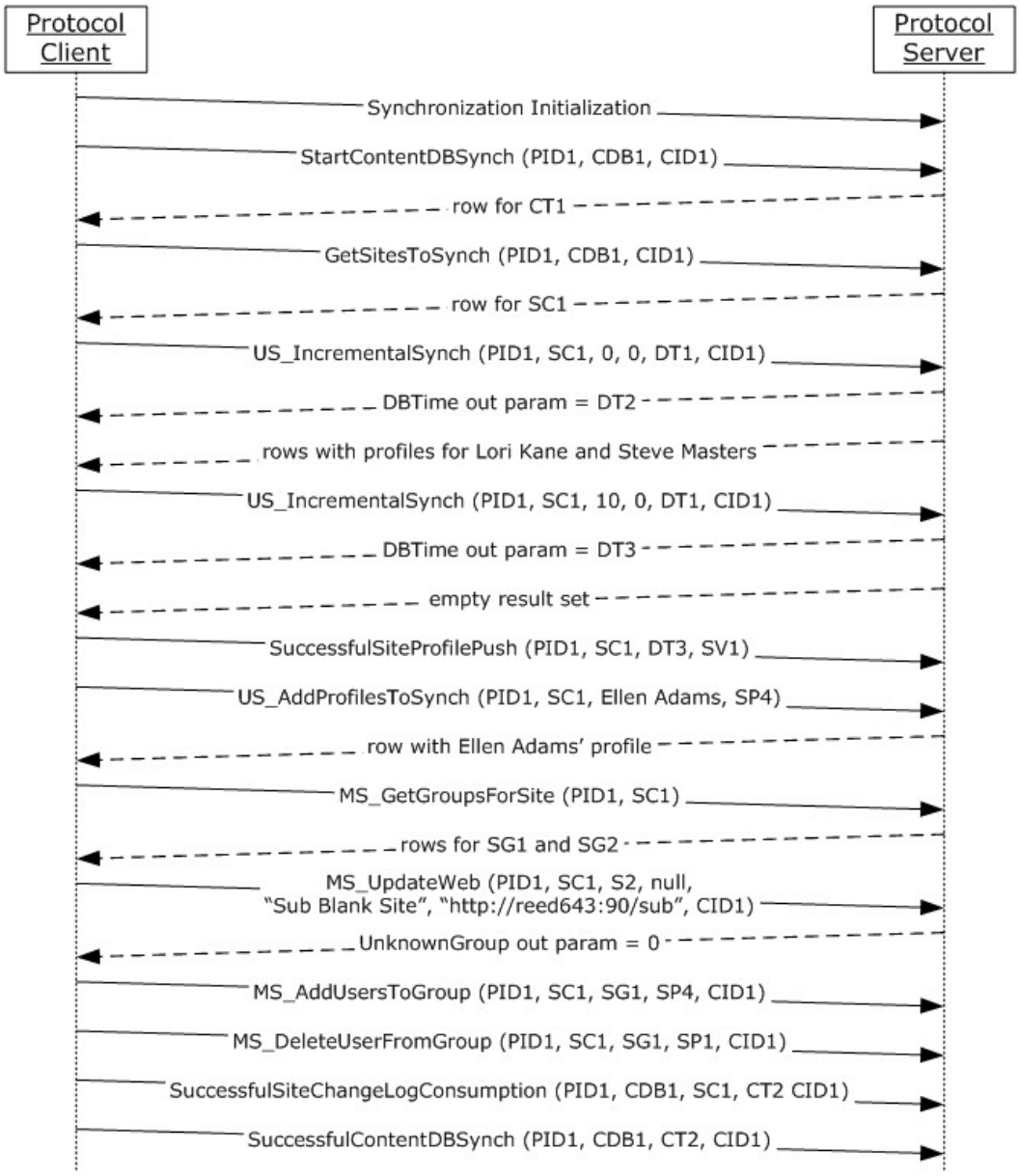

Figure 4: Incremental synchronization communication between the protocol client and protocol server

In detail:

Step 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 Figure 4) 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
```

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

Step 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
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D'
```

Step 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

Step 5: 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.

Step 6: At this point, the protocol is in the content database synchronization state. Before synchronizing individual site collections, the protocol client first:

1. Examines change log events that have taken place between CT1 and CDB1's current change token (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.
2. 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
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D'
```

Step 7: 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	SchemaVersion
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	1

LastChange SynchSuccess	Moving	MovingDeleted	Registered	PartitionID	HasProfileChanges
1	0	0	1	{EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C}	1

Step 8: 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.

1. 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**.

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

```
exec dbo.profilesynch_US_IncrementalSynch
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @MinNonInclusiveWssID = 0,
   @DBTime = NULL,
   @correlationId = NULL
```

Step 9: 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.

RecordID	ProfileSubTypeID	PropertyID	PropertyVal
1	1	2	0x0105000000000000515000000A065CF7E784B9B5FE77C87704D7A2100
1	1	3	CONTOSO\lori
1	1	4	Lori

Record ID	ProfileSubTypeID	Property ID	PropertyVal
1	1	7	Lori Kane
1	1	16	
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

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

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

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

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

Step 10: 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
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @MinNonInclusiveWssID = 10,
   @DBTime = NULL,
   @correlationId = NULL
```

Step 11: 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'
```

Step 12: 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 3.3.5.3.4. 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
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C'
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
```

```
    @StartSynchTime = '2008-03-11 18:33:32.310',
    @SchemaVersion = 1,     @correlationId uniqueidentifier = NULL
```

Step 13: 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.

Step 14: 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
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @SID0 = 0x010500000000000515000000a065cf7e784b9b5fe77c877000311100,
   @UID0 = 11
```

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

Text	OrderRank	Privacy	WSSID
NULL	NULL	NULL	11
NULL	NULL	NULL	11
NULL	NULL	NULL	11
NULL	NULL	NULL	11
NULL	NULL	NULL	11

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

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

Step 15: The protocol client then writes the user profile data it received to the user information list.

Step 16: 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
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75'
```

Step 17: The protocol server returns ***Membership Synchronization Groups*** result sets containing data for SG1 and SG2.

GroupID
5
7

Step 18: 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
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @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
```

Step 19: 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
```

Step 20: Looking at changes occurring between CT1 and CT2, the protocol client finds that G1 has changed. The protocol client calls **profilesynch_MS_AddUserToGroup** to update the data for SP4 (Ellen Adams).

```
exec dbo.profilesynch_MS_AddUserToGroup
```

```
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @GroupID = 5,
   @WssID = 11
```

Step 21: The protocol server appropriately updates the **Staging Data** and returns a 0 return code for successful execution.

Step 22: Next, the protocol client calls **profilesynch_MS_DeleteUserFromGroup** to update the data for SP1 (Lori Kane).

```
exec dbo.profilesynch_MS_DeleteUserFromGroup
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @WssID = 10,
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @GroupID = 5,      @correlationId uniqueidentifier = NULL
```

Step 23: The protocol server appropriately updates the **Staging Data** and returns a 0 return code for successful execution.

Step 24: 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 **SuccessfulSiteChangeLogConsuption** changes the protocol state back to content database synchronization.

```
exec dbo.profilesynch_SuccessfulSiteChangeLogConsumption
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @SiteID = '595D079D-DB43-4403-8A1D-6DF10295FA75',
   @TargetChangeToken = N'1;0;cd56acc0-3e03-4264-b187-
      786a7b98d49d;633408571893700000;520',
   @correlationId uniqueidentifier = NULL
```

Step 25: The protocol server returns a 0 return code for successful execution. This call also changes the protocol state back to content database synchronization.

Step 26: 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
     @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID = 'CD56ACC0-3E03-4264-B187-786A7B98D49D',
   @TargetChangeToken = N'1;0;cd56acc0-3e03-4264-b187-
      786a7b98d49d;633408571893700000;520',          @correlationId uniqueidentifier = NULL,
```

Step 27: The protocol server updates its synchronization data accordingly and returns a 0 return code for successful execution.

Step 28: 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.

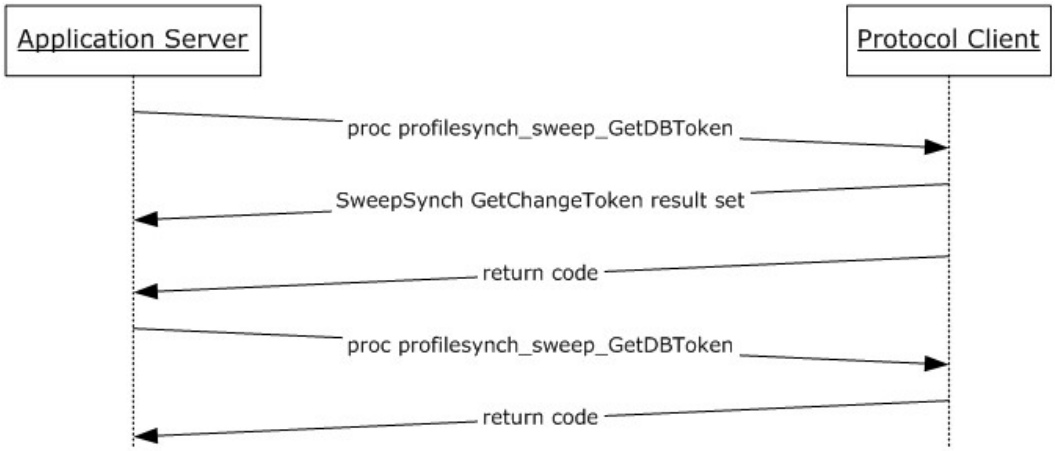

Figure 5: New-user synchronization

Step 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
    @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
    @ContentDBID='F2179717-1115-4549-9728-EA0EC8ED6069',
    @correlationId uniqueidentifier = NULL,
```

Step 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

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

Step 4: 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

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
   @partitionID = 'EE96E8D6-FBC6-4BC1-838F-25C8F0535E4C',
   @ContentDBID='F2179717-1115-4549-9728-EA0EC8ED6069',
   @ChangeToken=N'1;0;f2179717-1115-4549-9728-
      ea0ec8ed6069;633416668484370000;7237',
   @correlationId uniqueidentifier = NULL
```

Step 5: A return code of 0 is returned to the protocol client, indicating a successful execution of the stored procedure.

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® SharePoint® Server 2010
- Microsoft® SQL Server® 2005
- Microsoft® SQL Server® 2008
- Microsoft® SQL Server® 2008 R2

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

U

V

X