[MS-ECS]:
Enterprise Client Synchronization Protocol

Revision Summary

Date	Revision History	Revision Class	Comments
08/08/2013	1.0	New	Released new document.

Contents

1 Introduction

This document specifies the Enterprise Client Synchronization (ECS) Protocol.

The Enterprise Client Synchronization Protocol is designed for synchronizing files across multiple devices in an enterprise network.

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]:

MD5 hash
URI
URL

The following terms are specific to this document:

Sync Framework: A data synchronization platform that can be used to synchronize data across multiple data stores.

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 documents, which are updated frequently. References to other documents include a publishing year when one is available.

A reference marked "(Archived)" means that the reference document was either retired and is no longer being maintained or was replaced with a new document that provides current implementation details. We archive our documents online [Windows Protocol].

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.

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992, http://www.ietf.org/rfc/rfc1321.txt

[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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.ietf.org/rfc/rfc2818.txt

1.2.2 Informative References

[MSDN-FSA] Microsoft Corporation, "File Attribute Constants", http://msdn.microsoft.com/en-us/library/windows/desktop/gg258117(v=vs.85).aspx

[MSDN-ISK2-SerializeWithOptions] Microsoft Corporation, "ISyncKnowledge2_SerializeWithOptions", http://msdn.microsoft.com/en-us/library/dd937105.aspx

[MSDN-MSF] Microsoft Corporation, "Microsoft Sync Framework", http://msdn.microsoft.com/en-us/library/bb902854.aspx

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

1.3 Overview

The Enterprise Client Synchronization Protocol is used to access REST-based file sync services over the HTTPS transport.

The protocol is used for uploading and downloading file data between a client and server participating in the synchronization of a namespace. Both the upload and download scenarios are driven by the client role of this protocol.

Before creating any sessions for the data transfer, the client initially queries the server for information on the sync target share and the server's capabilities.

In the upload scenario, the client is notified of local changes to the file data from an underlying **Sync Framework**<1> and takes the following steps:

1. Create the server session.
2. Get the server's sync knowledge.
3. Prepare the server for upload.
4. Perform data transfer (upload).
5. Commit change.
6. Delete session.

In the download scenario, the protocol provides a mechanism for the client to receive notifications about changes on the server that will need to be synchronized. When the client receives a notification for server-side changes, the client takes the following steps:

1. Create the server session.
2. Update the server on the client’s sync knowledge.
3. Get the change batch from the server.
4. Perform data transfer (download).
5. Delete session.

1.4 Relationship to Other Protocols

None.

1.5 Prerequisites/Preconditions

The Enterprise Client Synchronization Protocol does not provide a mechanism for a client to discover the existence and location of a global sync **URI** for syncing a namespace. It is a prerequisite that the client's local configuration include a global sync URI that can be used to enumerate all servers exposing that namespace for synchronization.

The Enterprise Client Synchronization Protocol does not define an authentication or authorization scheme. Implementers of this protocol should review the recommended security prerequisites in Security Considerations for Implementers (section 5.1) of this document.

1.6 Applicability Statement

This protocol is applicable where file data is required to be synchronized across multiple devices in an enterprise network.

1.7 Versioning and Capability Negotiation

This protocol currently supports one version (1.0). The server returns the supported versions in the response for Server Capabilities as specified in section 3.1.5.3. The protocol does not provide any mechanism for capability negotiation based on versions. <2>

Version	Value
ECS Protocol version 1	"1.0"

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The Enterprise Client Synchronization Protocol uses secure HTTP 1.1 as transport, as specified in [RFC2616] and [RFC2818].

2.2 Common Data Types

The following table summarizes the server-side resources used by the Enterprise Client Synchronization Protocol.

Resource type	Description
Server Discovery	The resource used to enumerate all servers exposing the namespace for sync.
Share Discovery	The resource used to discover the sync share on the server.
Server Capabilities	The resource used to query the capabilities of the sync service.
Detect Server Changes	The resource used to do polling for the server-side changes.
User Configuration	The resource used to query the user configuration information for a sync target share.
Create Session	The resource used to create a new session on the server.
Sync Batch Parameters	The resource used to retrieve or update synchronization batch parameters.
Prepare Batch	The resource used to send information to the server to prepare for the change batch.
Upload Batch	The resource used to commit a change batch on the server.
Delete Session	The resource used to delete a sync session on the server.
Download Batch	The resource used to receive a change batch from the server to the client.
Upload Data	The resource used to send file data to the server.
Download Data	The resource used to receive file data from the server.

2.2.1 HTTP Headers

The following table summarizes the set of HTTP headers defined by this specification.

Header	Description
x-ecs-admin-contact	A string representing the email contact of the server administrator.
x-ecs-change-detection	When present, this header indicates that the server is currently in the process of enumerating changes done outside of sync to the user's data set.
x-ecs-changes-URL	The **URL** to be used by the client to poll for changes on the server.

Header	Description
x-ecs-continue	The opaque continuation token received from the server in a download batch operation.
x-ecs-devicename	A string representing the name and operating system of the device issuing the request.
x-ecs-metadata-version	A string containing the version of the server metadata for the current sync replica.
x-ecs-partnershipID	The PartnershipID string returned by the server in Share Discovery, as specified in section 3.1.5.2.1.2
x-ecs-request-error	A string describing request failure information.
x-ecs-session-id	The identifier of sync session.
x-ecs-session-location-url	The URL to access a session.
x-ecs-share-type	A string representing the type of the share to discover. Allowed values for this string are "UserData".
x-ecs-supported-versions	A comma-separated list of protocol versions supported by the server.

2.2.1.1 x-ecs-request-error

This header describes request failure information. The value for this header MUST be a string representation of the HRESULT in hexadecimal format.

Example:

x-ecs-request-error: "0x8007005"

```
hexdigit = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9" / "a" / "b" / "c" / "d"
/ "e" / "f" / "A" / "B" / "C" / "D" / "E" / "F" / "x"
x-ecs-request-error = 8 hexdigit
```

2.2.1.2 x-ecs-devicename

A string representing the name and operating system of the device issuing the request in the following format:

{Device Name, OS Family, OS Major Version, OS Minor Version, Sync App Version}

Example:

x-ecs-devicename: { homelp, Windows, 6, 2, WinApp1}

```
String = *(%x20-7E)
DeviceName = String
OSFamily = String
MajorVersion = String
MinorVersion = String
```

```
SyncAppVersion = String
x-ecs-devicename = "{" DeviceName "," OSFamily "," MajorVersion "," MinorVersion ","
SyncAppVersion "}"
```

2.2.1.3 x-ecs-share-type

A string representing the type of the share to discover.

```
String = *(%x20-7E)
x-ecs-share-type = String
```

2.2.1.4 x-ecs-changes-URL

This header provides the URL to be used by the client to poll for changes on the server.

```
String = *(%x20-7E)
x-ecs-changes-URL = String
```

2.2.1.5 x-ecs-partnershipID

A string representing the PartnershipID for a sync target. This value MUST be Base64-encoded.

```
String = *(%x20-7E)
x-ecs-partnershipID = String
```

2.2.1.6 x-ecs-admin-contact

A string representing the email contact of the server admin.

```
String = *(%x20-7E)
x-ecs-admin-contact = String
```

2.2.1.7 x-ecs-session-location-url

The URL prefix to access a session.

```
String = *(%x20-7E)
x-ecs-session-location-url = String
```

2.2.1.8 x-ecs-session-id

An implementation-specific string that identifies the session.

```
String = *(%x20-7E)
x-ecs-session-id = String
```

2.2.1.9 x-ecs-continue

The opaque continuation token received from the server in a download batch operation.

```
String = *(%x20-7E)
x-ecs-continue = String
```

2.2.1.10 x-ecs-metadata-version

A string containing the version of the server metadata for the current sync replica.

```
String = *(%x20-7E)
x-ecs-metadata-version = String
```

2.2.1.11 x-ecs-supported-versions

A comma-separated list of protocol versions supported by the server. The values for this header MUST be one of the version strings listed in section 1.7.

```
String = *(%x20-7E)
ProtocolVersion = String
ProtocolVersionWithComma = ProtocolVersion ","
x-ecs-supported-versions = "{" 0 * ProtocolVersionWithComma ProtocolVersion "}"
```

2.2.1.12 x-ecs-change-detection

When present, this header indicates that the server is currently in the process of enumerating changes done outside of sync to the user's data set. The value for this header MUST be an implementation-specific string.

Example:

x-ecs-change-detection: running

```
String = *(%x20-7E)
x-ecs-change-detection = String
```

2.2.2 Common Data Structures

The following table summarizes the set of common data structures defined by this specification. Common data types are specified in [MS-DTYP].

Structure Name	Section	Description
SYNC_GID	2.2.2.1	The structure used to represent an identifier for a file that is part of a sync batch.
SYNC_VERSION	2.2.2.2	The structure used to represent a version for the file in the sync batch.
SYNC_BLOB	2.2.2.3	The structure used to specify a binary stream of data.
QUOATA_USAGE_ENTRY	2.2.2.4	The structure used to specify the current data usage of an user on the sync target share.
POLICY_ENTRY	2.2.2.5	The structure used to specify the policies on how the client is expected to set up its target directory.
BATCH_LIMITS_ENTRY	2.2.2.6	The structure used to specify the parameters that describe sync batch characteristics.
FILE_METADATA_ENTRY	2.2.2.7	The structure used to specify the metadata information for a file in a sync batch.
FILE_INFO_INPUT_ENTRY	2.2.2.8	The structure used to specify the sync preparation information of a file that the client sends to a server before an upload.
FILE_INFO_ENTRY	2.2.2.9	The structure used to specify the sync preparation information of a file that the server sends to the client before an upload.
FILE_STATUS_ENTRY	2.2.2.10	The structure used to specify the status of a file commit in a sync process.
UPLOAD_ENTRY	2.2.2.11	The structure used to specify information of the data for a file that is being uploaded to the server.
UPLOAD_RESPONSE_ENTRY	2.2.2.12	The structure used to specify the server response data for a file being uploaded.
DOWNLOAD_ENTRY	2.2.2.13	The structure used to specify the information of the data for a file that is being downloaded from the server.
DOWNLOAD_RESPONSE_ENTRY	2.2.2.14	The structure used to specify the server response data for a file being uploaded.
FILE_DOWNLOAD_INFO_ENTRY	2.2.2.15	The structure that specifies the information on how the file content needs to be downloaded by the client.
SYNC_CHANGE_BATCH	2.2.2.16	The structure that defines the metadata describing the changes to be synchronized.
SYNC_MD5HASH	2.2.2.17	The structure that defines the serialization format used by this protocol for **MD5 hash** data.
VECTOR_POLICY_ENTRY	2.2.2.18	The structure representing a collection of POLICY_ENTRY structures.

Structure Name	Section	Description
VECTOR_FILE_METADATA_ENTRY	2.2.2.19	The structure representing a collection of FILE_METADATA_ENTRY structures.
VECTOR_FILE_INFO_INPUT_ENTRY	2.2.2.20	The structure representing a collection of FILE_INFO_INPUT_ENTRY structures.
VECTOR_FILE_INFO_ENTRY	2.2.2.21	The structure representing a collection of FILE_INFO_ENTRY structures.
VECTOR_FILE_STATUS_ENTRY	2.2.2.22	The structure representing a collection of FILE_STATUS_ENTRY structures.
VECTOR_UPLOAD_ENTRY	2.2.2.23	The structure representing a collection of UPLOAD_ENTRY structures.
VECTOR_UPLOAD_RESPONSE_ENTRY	2.2.2.24	The structure representing a collection of UPLOAD_RESPONSE_ENTRY structures.
VECTOR_DOWNLOAD_ENTRY	2.2.2.25	The structure representing a collection of DOWNLOAD_ENTRY structures.
VECTOR_DOWNLOAD_RESPONSE_ENTRY	2.2.2.26	The structure representing a collection of DOWNLOAD_RESPONSE_ENTRY structures.
VECTOR_FILE_DOWNLOAD_INFO_ENTRY	2.2.2.27	The structure representing a collection of FILE_DOWNLOAD_INFO_ENTRY structures.
VECTOR_STRING	2.2.2.28	The structure representing a collection of null-terminated strings.

2.2.2.1 SYNC_GID

The SYNC_GID structure represents an identifier for a file that is part of a sync batch.

```
typedef struct _SYNC_GID {
  ULONGLONG GidPrefix;
  GUID UniqueId;
} SYNC_GID;
```

GidPrefix: The prefix component of the identifier.

UniqueId: The GUID component of the identifier.

2.2.2.2 SYNC_VERSION

The SYNC_VERSION structure represents a version for the file in the sync batch.

```
typedef struct _SYNC_VERSION {
  DWORD ReplicaKey;
  ULONGLONG TickCount;
} SYNC_VERSION;
```

ReplicaKey: The replica key associated with the version.

TickCount: The tick count associated with the version

2.2.2.3 SYNC_BLOB

The SYNC_BLOB structure represents a binary stream of data.

```
typedef struct _SYNC_BLOB {
  ULONG BlobSize;
  BYTE BlobData[];
} SYNC_BLOB;
```

BlobSize: The size of the blob data in bytes.

BlobData: A stream of bytes containing the blob data.

2.2.2.4 QUOTA_USAGE_ENTRY

The QUOTA_USAGE_ENTRY structure specifies the current data usage of user on the sync target share.

```
typedef struct _QUOTA_USAGE_ENTRY {
    ULONG64 UserUsage;
    ULONG64 UserDataFreeSpace;
} QUOTA_USAGE_ENTRY;
```

UserUsage: The amount of data, in bytes, in the user's share.

UserDataFreeSpace: The amount of available free space, in bytes, in the user's share.

2.2.2.5 POLICY_ENTRY

The POLICY_ENTRY structure specifies the policies on how the client is expected to set up its target directory.

```
typedef struct _POLICY_ENTRY {
    BYTE PolicyName;
    BOOL PolicyType;
} POLICY_ENTRY;
```

PolicyName: An unsigned 8-bit value that identifies the policy. This MUST be one of the following values.

Value	Description
0x01	Encryption
0x02	Password
0x03	AutoLock

PolicyType: A Boolean that when set to TRUE indicates that this policy is enforced.

2.2.2.6 BATCH_LIMITS_ENTRY

The BATCH_LIMITS_ENTRY structure specifies the parameters that describe sync batch characteristics.

```
typedef struct _BATCH_LIMITS_ENTRY {
    ULONG MaxFileDataSize;
    ULONG MaxFileCount;
} BATCH_LIMITS_ENTRY;
```

MaxFileDataSize: Maximum size, as a multiple of 1048576 bytes, of file data in one batch.

MaxFileCount: Maximum count of files in one batch.

2.2.2.7 FILE_METADATA_ENTRY

The FILE_METADATA_ENTRY structure specifies the metadata information for a file in a sync batch.

```
typedef struct _BATCH_LIMITS_ENTRY {
    SYNC_GID FileId;
    SYNC_VERSION SyncVersion;
    GUID FileStreamVersion;
    SYNC_GID ParentId;
    DWORD Attributes;
    FILETIME NamespaceChangeTime;
    FILETIME AttributeChangeTime;
    FILETIME CreatedTime;
    FILETIME ModifiedTime;
    ULONGLONG ContentSize;
    WCHAR Name[];
    USHORT OriginatingDeviceNameIndex;
} FILE_METADATA_ENTRY;
```

FileId: An identifier that uniquely identifies the file or directory within the sync batch.

SyncVersion: The version of the file entry.

FileStreamVersion: A GUID representing the version of the file content. This MUST not be used for directories.

ParentId: An identifier for the metadata of the parent item. The parent item MUST be a directory.

FileAttributes: The attributes of the file allowed by the underlying object store.<3>

NamespaceChangeTime: The time at which the name of the file or its **ParentId** was changed. This field is used for resolving conflicts during synchronization. This MUST be in the format as specified in [MS-DTYP] section 2.3.3.

AttributeChangeTime: The time at which the attributes of the file were changed. This field is used for resolving conflicts during synchronization. This MUST be in the format as specified in [MS-DTYP] section 2.3.3.

CreatedTime: The time at which the file was created. This MUST be in the format as specified in [MS-DTYP] section 2.3.1.

ModifiedTime: The time at which the file was last modified. This MUST be in the format as specified in [MS-DTYP] section 2.3.1.

ContentSize: Size of the file in bytes. This MUST be zero for a directory entry.

Name: A null-terminated string representing the name of the file or directory. This string MUST not be empty.

OriginatingDevideNameIndex: The index of the string representing the device in the **DeviceNames** field of the SYNC_CHANGE_BATCH structure, as specified in section 2.2.2.16.

2.2.2.8 FILE_INFO_INPUT_ENTRY

The FILE_INFO_INPUT_ENTRY structure specifies the sync preparation information of a file that the client sends to the server before an upload.

```
typedef struct _FILE_INFO_INPUT_ENTRY {
    SYNC_GID SyncItemId;
    ULONGLONG FileSize;
    WCHAR FileExtension[];
    GUID StreamId;
} FILE_INFO_INPUT_ENTRY;
```

SyncItemId: An identifier that is used to correlate the file metadata with the Sync Framework change item.

FileSize: Size of the file in bytes.

FileExtension: A null-terminated string containing the extension of the file.

StreamId: The stream identifier of the file.

2.2.2.9 FILE_INFO_ENTRY

The FILE_INFO_ENTRY structure specifies the sync preparation information of a file that the server sends to the client before an upload.

```
typedef struct _FILE_INFO_ENTRY {
    SYNC_GID SyncItemId;
    WCHAR Uri[];
    UCHAR ProtocolType;
    HRESULT PrepareResult;
} FILE_INFO_ENTRY;
```

SyncItemId: An identifier that is used to correlate the file metadata with the Sync Framework change item.

Uri: A null-terminated string containing the URI used to perform data transfer for this file. This MUST be empty if the **ProtocolType** field is 1.

ProtocolType: Indicates the type of data transfer that the server supports for this file. This value MUST be one of the following.

Value	Meaning
0x00	No upload is required for this file.
0x01	File batching data transfer is required to transfer this file.

PrepareResult: When **ProtocolType** is zero, this value indicates the reason for not uploading the file. Otherwise, this value indicates any error that occurred during the preparation process.

2.2.2.10 FILE_STATUS_ENTRY

The FILE_STATUS_ENTRY structure specifies the information of a file commit in a sync process.

```
typedef struct _FILE_STATUS_ENTRY {
    SYNC_GID SyncItemId;
    HRESULT Status
} FILE_STATUS_ENTRY;
```

SyncItemId: An identifier that is used to correlate the file metadata with the Sync Framework change item.

Status: Indicates the result of applying changes to the file.

2.2.2.11 UPLOAD_ENTRY

The UPLOAD_ENTRY structure specifies the information of the data for a file that is being uploaded to the server.

```
typedef struct _UPLOAD_ENTRY {
    SYNC_GID SyncItemId;
    SYNC_MD5HASH FileHash;
    ULONG64 Offset;
    ULONG Length;
    ULONGLONG Reserved;
    SYNC_BLOB UploadData;
} UPLOAD_ENTRY;
```

SyncItemId: A value that uniquely identifies the file replica within the change batch.

FileHash: The MD5 hash, as specified in [RFC1321] of the file contents. The hash is serialized in SYNC_MD5HASH format as specified in section 2.2.2.17.

Offset: The offset, in bytes, in the file for the data to be uploaded.

Length: The length, in bytes, of the file data to be uploaded.

UploadData: The payload data to be uploaded.

2.2.2.12 UPLOAD_RESPONSE_ENTRY

The UPLOAD_RESPONSE_ENTRY structure specifies the server response data for a file being uploaded.

```
typedef struct _UPLOAD_RESPONSE_ENTRY {
```

```
    SYNC_GID SyncItemId;
    HRESULT Result;
} UPLOAD_RESPONSE_ENTRY;
```

SyncItemId: A value that uniquely identifies the file replica within the change batch.

Result: The result of the upload process for the file. If the upload is successful, this value MUST be set to S_OK.

2.2.2.13 DOWNLOAD_ENTRY

The DOWNLOAD_ENTRY structure specifies the information of the data for a file that is being downloaded from the server.

```
typedef struct _DOWNLOAD_ENTRY {
    SYNC_GID SyncItemId;
    SYNC_BLOB FileVersion;
} DOWNLOAD_ENTRY;
```

SyncItemId: An identifier that is used to correlate the file metadata with the Sync Framework change item.

FileVersion: An opaque stream of bytes (ETag) that identifies the file version.

2.2.2.14 DOWNLOAD_RESPONSE_ENTRY

The DOWNLOAD_RESPONSE_ENTRY structure specifies the server response data for a file being uploaded.

```
typedef struct _DOWNLOAD_RESPONSE_ENTRY {
    SYNC_GID SyncItemId;
    ULONGLONG DataLength;
    SYNC_BLOB Data;
    HRESULT Result;
    SYNC_MD5HASH Hash;
} DOWNLOAD_RESPONSE_ENTRY;
```

SyncItemId: An identifier that is used to correlate the file metadata with the Sync Framework change item.

DataLength: The length in bytes of the file data sent by the server.

Data: The contents of the file being downloaded from the server.

Result: The result of the file download operation. When the result indicates an error, the **DataLength** field MUST be zero and the **Data** field MUST be empty.

Hash: The MD5 hash, as specified in [RFC1321], of the file contents. The hash is serialized in SYNC_MD5HASH format as specified in section 2.2.2.17.

2.2.2.15 FILE_DOWNLOAD_INFO_ENTRY

The FILE_INFO_ENTRY structure specifies information that is required for the client to download the file content.

```
typedef struct _FILE_INFO_ENTRY {
    SYNC_GID SyncId;
    WCHAR Uri[];
    UCHAR ProtocolType;
} FILE_DOWNLOAD_INFO_ENTRY;
```

SyncId: A value that uniquely identified the file item within the sync batch.

Uri: The URI used to perform data transfer of the file stream. This MUST be empty.

ProtocolType: Indicates the type of data transfer that the server supports for this file. This value MUST be one of the following.

Value	Meaning
0x01	File batching data transfer is required to transfer this file.

2.2.2.16 SYNC_CHANGE_BATCH

The SYNC_CHANGE_BATCH structure defines the metadata that describes the changes to be synchronized.

```
typedef struct _SYNC_CHANGE_BATCH {
  VECTOR_FILE_METADATA_ENTRY Files;
  VECTOR_STRING DeviceNames;
  SYNC_BLOB SyncMetadata;
} SYNC_CHANGE_BATCH;
```

Files: A VECTOR_FILE_METADATA_ENTRY structure containing metadata information for all files in the sync batch.

DeviceNames: A VECTOR_STRING structure that contains a list of the names of the devices that generated the changes to the files in this batch.

SyncMetadata: An implementation-specific<4> stream of bytes in SYNC_BLOB format that represents the serialized server metadata.

2.2.2.17 SYNC_MD5HASH

The SYNC_MD5HASH structure defines the serialization format used by this protocol for MD5 hash data. The calculation of the MD5 hash is specified in [RFC1321].

```
typedef struct _SYNC_MD5HASH {
  ULONGLONG High;
  ULONGLONG Low;
} SYNC_MD5HASH;
```

High: The first 64 bytes of the MD5 hash data.

Low: The last 64 bytes of the MD5 hash data.

2.2.2.18 VECTOR_POLICY_ENTRY

The VECTOR_POLICY_ENTRY structure represents a collection of POLICY_ENTRY structures, as specified in section 2.2.2.5, in the following format:

```
typedef struct _VECTOR_POLICY_ENTRY {
  ULONG NumEntries;
  POLICY_ENTRY EntryStream[];
} VECTOR_POLICY_ENTRY;
```

NumEntries: The number of entries in the vector.

EntryStream: A stream of bytes containing zero or more POLICY_ENTRY structures.

2.2.2.19 VECTOR_FILE_METADATA_ENTRY

The VECTOR_FILE_METADATA structure represents a collection of FILE_METADATA_ENTRY structures, as specified in section 2.2.2.7, in the following format:

```
typedef struct _VECTOR_ FILE_METADATA _ENTRY {
  ULONG NumEntries;
  BYTE EntryStreamBytes[];
} VECTOR_ FILE_METADATA _ENTRY;
```

NumEntries: The number of entries in the vector.

EntryStreamBytes: A stream of bytes representing a sequence of zero or more FILE_METADATA_ENTRY structures.

2.2.2.20 VECTOR_FILE_INFO_INPUT_ENTRY

The VECTOR_FILE_INFO_INPUT structure represents a collection of FILE_INFO_INPUT_ENTRY structures, as specified in section 2.2.2.8, in the following format:

```
typedef struct _VECTOR_ FILE_INFO_INPUT_ENTRY {
  ULONG NumEntries;
  BYTE EntryStreamBytes[];
} VECTOR_ FILE_INFO_INPUT_ENTRY;
```

NumEntries: The number of entries in the vector.

EntryStreamBytes: A stream of bytes representing a sequence of zero or more FILE_INFO_INPUT_ENTRY structures.

2.2.2.21 VECTOR_FILE_INFO_ENTRY

The VECTOR_FILE_INFO structure represents a collection of FILE_INFO _ENTRY structures, as specified in section 2.2.2.9, in the following format:

```
typedef struct _VECTOR_FILE_INFO_ENTRY {
  ULONG NumEntries;
  BYTE EntryStreamBytes[];
} VECTOR_FILE_INFO_ENTRY;
```

NumEntries: The number of entries in the vector.

EntryStreamBytes: A stream of bytes representing a sequence of zero or more FILE_INFO_ENTRY structures.

2.2.2.22 VECTOR_FILE_STATUS_ENTRY

The VECTOR_FILE_STATUS_ENTRY structure represents a collection of FILE_STATUS_ENTRY structures, as specified in section 2.2.2.10, in the following format:

```
typedef struct _VECTOR_FILE_STATUS_ENTRY {
  ULONG NumEntries;
  FILE_STATUS_ENTRY EntryStream[];
} VECTOR_FILE_STATUS_ENTRY;
```

NumEntries: The number of entries in the vector.

EntryStream: A stream of bytes containing zero or more FILE_STATUS_ENTRY structures.

2.2.2.23 VECTOR_UPLOAD_ENTRY

The VECTOR_UPLOAD_ENTRY structure represents a collection of UPLOAD_ENTRY structures, as specified in section 2.2.2.11, in the following format:

```
typedef struct _VECTOR_UPLOAD_ENTRY {
  ULONG NumEntries;
  BYTE EntryStreamBytes[];
} VECTOR_UPLOAD_ENTRY;
```

NumEntries: The number of entries in the vector.

EntryStreamBytes: A stream of bytes representing a sequence of zero or more UPLOAD_ENTRY structures.

2.2.2.24 VECTOR_UPLOAD_RESPONSE_ENTRY

The VECTOR_UPLOAD_RESPONSE_ENTRY structure represents a collection of UPLOAD_RESPONSE_ENTRY structures, as specified in section 2.2.2.12, in the following format:

```
typedef struct _VECTOR_UPLOAD_RESPONSE_ENTRY {
  ULONG NumEntries;
  UPLOAD_RESPONSE_ENTRY EntryStream[];
} VECTOR_UPLOAD_RESPONSE_ENTRY;
```

NumEntries: The number of entries in the vector.

EntryStream: A stream of bytes containing zero or more UPLOAD_ RESPONSE_ENTRY structures.

2.2.2.25 VECTOR_DOWNLOAD_ENTRY

The VECTOR_DOWNLOAD_ENTRY structure represents a collection of DOWNLOAD_ENTRY structures, as specified in section 2.2.2.13, in the following format:

```
typedef struct _VECTOR_DOWNLOAD_ENTRY {
  ULONG NumEntries;
  BYTE EntryStreamBytes[];
} VECTOR_DOWNLOAD_ENTRY;
```

NumEntries: The number of entries in the vector.

EntryStream: A stream of bytes representing a sequence of zero or more DOWNLOAD_ENTRY structures.

2.2.2.26 VECTOR_DOWNLOAD_RESPONSE_ENTRY

The VECTOR_DOWNLOAD_RESPONSE_ENTRY structure represents a collection of DOWNLOAD_RESPONSE_ENTRY structures, as specified in section 2.2.2.14, in the following format:

```
typedef struct _VECTOR_DOWNLOAD_RESPONSE_ENTRY {
  ULONG NumEntries;
  BYTE EntryStreamBytes[];
} VECTOR_DOWNLOAD_RESPONSE_ENTRY;
```

NumEntries: The number of entries in the vector.

EntryStreamBytes: A stream of bytes representing a sequence of zero or more DOWNLOAD_RESPONSE_ENTRY structures.

2.2.2.27 VECTOR_FILE_DOWNLOAD_INFO_ENTRY

The VECTOR_FILE_DOWNLOAD_INFO_ENTRY structure represents a collection of FILE_DOWNLOAD_INFO_ENTRY structures, as specified in section 2.2.2.15, in the following format:

```
typedef struct VECTOR_FILE_DOWNLOAD_INFO_ENTRY {
  ULONG NumEntries;
  BYTE EntryStreamBytes[];
} VECTOR_FILE_DOWNLOAD_INFO_ENTRY;
```

NumEntries: The number of entries in the vector.

EntryStream: A stream of bytes representing a sequence of zero or more FILE_DOWNLOAD_INFO_ENTRY structures.

2.2.2.28 VECTOR_STRING

The VECTOR_STRING structure represents a collection of strings in the following format:

```
typedef struct _VECTOR_STRING {
  ULONG NumEntries;
  WCHAR EntryStreamBytes[];
} VECTOR_STRING;
```

NumEntries: The number of strings in the vector.

EntryStreamBytes: A stream of bytes containing zero or more null-terminated strings.

3 Protocol Details

The resources defined by this protocol are categorized as follows:

- Service Discovery Resources
- Server Changes Resources
- User Configuration Resources
- Session Resources
- Data Transfer Server Resources

The following sections describe the processing of each resource in the above mentioned categories.

3.1 ServiceDiscovery Server Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in the Enterprise Client Synchronization 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 server implements the following:

HostURLPrefixList: A list of URLs for the servers that expose the synchronization namespace.

SupportedVersions: A list of strings that represent the ECS Protocol versions supported by the server. The allowed values for the version strings are listed in section 1.7.

PartnershipIdList: A list of identifiers in the string format that relate a client with a specific sync share.

3.1.2 Timers

None.

3.1.3 Initialization

The server MUST initialize the following:

HostURLPrefixList MUST be set to a list of server URLs in an implementation-specific manner.

SupportedVersions MUST be set to the string "1.0".

PartnershipIdList MUST be set to empty.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

Resource	Description
Server Discovery	The resource used to enumerate all servers exposing the namespace for sync.
Share Discovery	The resource uses to discover the sync share on the server.
Server Capabilities	The resource used to query the capabilities of the sync service.

The responses to all the operations can result in the following status codes.

Status code	Reason phrase	Description
200	OK	Server is discovered and URL is sent back in response body.
404	Not Found	Cannot find the sync server URL for the given user. Either the user doesn't have a value for the attribute or the AD schema does not support the attribute.
500	Internal Server Error	Unexpected server error. Specific error code in the request error header.

3.1.5.1 Server Discovery

The Server Discovery resource represents a list of servers that expose the synchronization namespace.

The following operations are allowed to be performed on this resource.

HTTP method	Description
GET	Enumerate a list of servers that expose the synchronization namespace.

3.1.5.1.1 GET

This operation is transported by an HTTP GET request.

The operation can be invoked through the following URI suffix:

```
Sync/{version}/Discover/ServerUrl
```

The following is an example of a complete URI for this operation:

http://sync.contoso.com/Sync/1.0/Discover/ServerUrl

The request message for this operation contains the following HTTP headers.

Request header	Usage	Value
x-ecs-devicename	Optional	A string representing the name and operating system of the device issuing the request in the following format: {Device Name, OS Family, OS Major Version, OS Minor Version, Sync App

Request header	Usage	Value
		Version} Example: x-ecs-devicename: { homelp, Windows, 6.2, 9200, 1.0}

The response message for this operation contains the following HTTP headers.

Response header	Usage	Value
x-ecs-request-error	Optional	A string describing request failure information. This MUST be a string representation of the HRESULT in hexadecimal format. Example: x-ecs-request-error: "0x8007005"

The response message for this operation can result in the following status codes.

Status code	Description
200	Server is discovered and URL is sent back in response body.
404	Cannot find sync server URL for the given user. Either the user doesn't have a value for the attribute or the AD schema does not support the attribute.
500	Unexpected server error. Specific error code in the request error header.

3.1.5.1.1.1 Request Body

None.

3.1.5.1.1.2 Response Body

The response for this method MUST contain the following:

Entry	Type
ServerUrls	VECTOR_STRING

ServerUrls: A list of host server URL prefixes that the client can connect to for all subsequent sync operations.

3.1.5.1.1.3 Processing Details

The server MUST fill the response body with the URLs from **HostURLPrefixList**.

3.1.5.2 Share Discovery

This resource represents the sync share on the server for a user.

The following operations are allowed to be performed on this resource.

HTTP method	Description
GET	Query the properties of the sync share

3.1.5.2.1 GET

This operation is transported by an HTTP GET request.

The operation can be invoked through the following URI suffix:

```
Sync/{version}/Discover/Share
```

The following is an example of a complete URI for this operation:

http://www.contoso.sync.com/Sync/1.0/Discover/Share

The request message for this operation contains the following HTTP headers.

Request header	Usage	Value
x-ecs-share-type	Optional	A string representing the type of the share to discover. If this header is present in the request, its value MUST be set to "UserData".

The response message for this operation contains the following HTTP headers.

Response header	Usage	Value
x-ecs-request-error	Optional	A string describing request failure information. This MUST be a string representation of the HRESULT in hexadecimal format. Example: x-ecs-request-error: "0x8007005"

The response message for this operation can result in the following status codes.

Status code	Description
200	Success
404	There is no sync share for the user (possible reasons include the type not existing or the user not having access).
500	Unexpected server error.

3.1.5.2.1.1 Request Body

None.

3.1.5.2.1.2 Response Body

The response for this method MUST contain the following:

Entry	Type
PartnershipId	String
EnterpriseId	String
DataSize	ULONG64

PartnershipId: A unique ID on the server that represents a combination of the client and the sync share. This ID will be used by the client as an x-ecs-partnershipID header in subsequent requests.

EnterpriseId: A unique ID on the server that is used for encrypting the data on the user's device in a way that associates it with the user's company.

DataSize: Total size of the user data on the sync share for this user in bytes.

3.1.5.2.1.3 Processing Details

If the request contains x-ecs-share-type header and its value is not "UserData", the server MUST fail the request with a status code of 404.

The server MUST generate a unique ID in the form of a string and set it to **PartnershipId** in the response body. The server MUST add the generated ID to **PartnershipIdList**.

If the server cannot evaluate the size of the user data, it MUST set DataSize in the response body to the decimal value of 18446744073709551615, that is, the maximum value allowed by ULONG64.

3.1.5.3 Server Capabilities

The Server Capabilities resource represents information about the protocol support on the server.

The following operations are allowed to be performed on this resource.

HTTP method	Description
GET	Query protocol support information from the server.

3.1.5.3.1 GET

This operation is transported by an HTTP GET request.

The operation can be invoked through the following URI suffix:

```
Sync/{version}/Capabilities
```

The following is an example of a complete URI for this operation:

https://www.contoso.sync.com/Sync/1.0/Capabilities

The request message for this operation contains the following HTTP headers.

Request header	Usage	Value
x-ecs-devicename	Optional	A string representing the name and operating system of the device issuing the request in the following format: {Device Name, OS Family, OS Major Version, OS Minor Version, Sync App Version} Example: x-ecs-devicename: {homelp, Windows, 6.2, 9200, 1.0}

The response message for this operation contains the following HTTP headers.

Response header	Usage	Value
x-ecs-request-error	Optional	A string describing request failure information. This MUST be a string representation of the HRESULT in hexadecimal format. Example: x-ecs-request-error: "0x8007005"
x-ecs-supported-versions	Optional	A comma-separated list of strings indicating the protocol versions supported by the server. Example: x-ecs-supported-versions: {1.0,2.0}

The response message for this operation can result in the following status codes.

Status code	Description
200	Success
500	Unexpected server error.

3.1.5.3.1.1 Request Body

None.

3.1.5.3.1.2 Response Body

The response for this method MUST contain the following:

Entry	Type
ProtocolType	UCHAR

ProtocolType: The data transfer type that the server supports. This MUST be set to the following value:

Value	Description
0x01	File batching data transfer protocol.

3.1.5.3.1.3 Processing Details

The server SHOULD set x-ecs-supported-versions in the response header to the **SupportedVersions**.

If the server supports file batching for data transfer, the server MUST set the bit to 0x01 in ProtocolType in the response body.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 DetectServerChanges Server Details

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

Resource	Description
Detect Server Changes	The resource used to poll for the server-side changes.

The responses to all the operations can result in the following status codes.

Status code	Reason phrase	Description
200	OK	Server knowledge has changed.
304	Not Modified	Server knowledge did not change.

3.2.5.1 Detect Server Changes

The Detect Server Changes resource allows the client to poll for the changes on the server.

HTTP method	Description
GET	Query the changes on the server

3.2.5.1.1 GET

This operation is transported by an HTTP GET request.

The operation can be invoked through the following URI suffix:

```
Sync/{version}/Changes
```

The following is an example of a complete URI for this operation:

http://contoso.com/Sync/1.0/Changes

The request message for this operation contains the following HTTP headers.

Request header	Usage	Value
If-None-Match	Optional	The ETag from a previous response
x-ecs-partnershipID	Required	The PartnershipID for the sync target share.

The response message for this operation contains the following HTTP headers.

Response header	Usage	Value
ETag	Optional	A value identifying the version of the data for this PartnershipID.
x-ecs-change-detection	Optional	If this header is present, it indicates that the server is currently in the process of enumerating changes done outside of sync to the user's data set. The client can choose to use a shorter polling interval in case there are any changes detected.

The response message for this operation can result in the following status codes.

Status code	Description
200	Server knowledge has changed; at least one of the two ETags is different.
304	Server knowledge did not change.

3.2.5.1.1.1 Request Body

None.

3.2.5.1.1.2 Response Body

None.

3.2.5.1.1.3 Processing Details

None.

3.2.6 Timer Events

None

3.2.7 Other Local Events

None

3.3 UserConfiguration Server Details

3.3.1 Abstract Data Model

None.

3.3.2 Timers

None

3.3.3 Initialization

None

3.3.4 Higher-Layer Triggered Events

None

3.3.5 Message Processing Events and Sequencing Rules

Resource	Description
User Configuration	The resource that represents user configuration information for a sync target share.

3.3.5.1 User Configuration

HTTP method	Description
GET	Query user configuration information for a sync target share.

3.3.5.1.1 GET

This operation is transported by an HTTP GET request.

The operation can be invoked through the following URI suffix:

```
Sync/{version}/Configuration
```

The following is an example of a complete URI for this operation:

http://host.com/Sync/1.0/Configuration

The request message for this operation contains the following HTTP headers.

Request header	Usage	Value
x-ecs-partnershipID	Optional	The PartnershipID for the sync target that was received in the GET response for the Share Discovery resource.

3.3.5.1.1.1 Request Body

None.

3.3.5.1.1.2 Response Body

The response for this method MUST contain the following:

Entry	Type
Quota Usage	QUOTA_USAGE_ENTRY structure
Policy List	VECTOR_POLICY_ENTRY

QuotaUsage: Describes the current data usage for the user.

PolicyList: A list of policy entries that describe how the client needs to set up its target directory. The policies processed first take precedence.

3.3.5.1.1.3 Processing Details

The server MUST query the quota usage on the sync share in an implementation-specific manner, construct the QUOTA_USAGE_ENTRY structure as specified in section 2.2.2.4, and set the Quota Usage field in the response body to this structure. If the server cannot determine the current quota usage for the user, it MUST set the **UserUsage** field of the QUOTA_USAGE_ENTRY structure to the decimal value of 18446744073709551615, that is, the maximum value allowed by ULONG64.

The server MUST query all the policies that apply to the sync share and MUST construct a VECTOR_POLICY_ENTRY structure using each policy being applied. The server MUST set the Policy List in the response body to the VECTOR_POLICY_ENTRY structure.

3.3.6 Timer Events

None

3.3.7 Other Local Events

None

3.4 PeerSynchronizationSession Server Details

3.4.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in the Enterprise Client Synchronization 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.

3.4.1.1 Global

The server implements the following:

SessionList: A list of **Sessions** defined in section 3.4.1.2.

3.4.1.2 Per Session

SessionLocationURL: The URL prefix used to access this session.

SessionId: A string that identifies the session.

3.4.2 Timers

None

3.4.3 Initialization

The server MUST set **SessionList** to empty.

3.4.4 Higher-Layer Triggered Events

None

3.4.5 Message Processing Events and Sequencing Rules

Resource	Description
Create Session	The resource used to create a new session on the server.
Sync Batch Parameters	The resource used to retrieve or update synchronization batch parameters.
Prepare Batch	The resource used to send information to the server to prepare for the change batch.
Upload Batch	The resource used to commit a change batch on the server.
Delete Session	The resource used to delete a sync session on the server.
Download Batch	The resource used to receive a change batch from the server to the client.

The responses to all the operations can result in the following status codes.

Status code	Reason phrase	Description
200	OK	Success. See individual methods on each resource in the subsequent sections for more details.
201	Created	Session created.
404	Not Found	No session exists with the specified ID.

Status code	Reason phrase	Description
202	Accepted	An asynchronous operation for retrieving the sync batch parameters has started.
400	Bad Request	Batch is out of sequence, or the last batch has already been received.
409	Conflict	Batch already received.

3.4.5.1 Create Session

The Create Session resource facilitates a client to start a sync session on the server.

HTTP method	Description
PUT	Create a new sync session.

3.4.5.1.1 PUT

This operation is transported by an HTTP PUT request.

The operation can be invoked through the following URI suffix:

```
Sync/{version}/Session
```

The following is an example of a complete URI for this operation:

http://contoso.com/Sync/1.0/Session

The request message for this operation contains the following HTTP headers.

Request header	Usage	Value
x-ecs-partnershipID	Required	The PartnershipID for the sync target.
x-ecs-devicename	Optional	A string representing the name and operating system of the device issuing the request in the following format: {Device Name, OS Family, OS Major Version, OS Minor Version, Sync App Version} Example: x-ecs-devicename: { homelp, Windows, 6.2, 9200, 1.0}

The response message for this operation contains the following HTTP headers.

Response header	Usage	Value
x-ecs-request-error	Optional	A string describing request failure information. This MUST be a string representation of the HRESULT in hexadecimal format. Example: x-ecs-request-error: "0x8007005"

Response header	Usage	Value
x-ecs-admin-contact	Optional	A string representing the email contact of the server administrator.
x-ecs-metadata-version		A string containing the version of the server metadata for the current sync replica.
x-ecs-session-id	Optional	A GUID in string format or an implementation-specific string that identifies the session. The client MUST use this value to build the complete URL in the subsequent requests on a session.

The response message for this operation can result in the following status codes.

Status code	Description
200	The response contains the URI prefix for a session that already exists.
201	The response contains the URI prefix for a newly created session.

3.4.5.1.1.1 Request Body

The request body for this method MUST contain the following.

Entry	Type
Type	UCHAR
ReplicaId	GUID

Type: The type of data transfer from the client's perspective. This MUST be one of the following values.

Value	Meaning
0x01	Upload
0x02	Download
0x03	Upload with full enumeration
0x04	Download with full enumeration

ReplicaId: Identifies the client replica.

3.4.5.1.1.2 Response Body

None.

3.4.5.1.1.3 Processing Details

The server MUST create a new **Session** and initialize the following values:

SessionLocationURL MUST be set to an implementation-specific URL that provides access to this session.

SessionId MUST be set to an implementation-specific string that identifies the session.

The server MUST insert the **Session** in **SessionList.**

3.4.5.2 Sync Batch Parameters

The Sync Batch Parameters resource represents synchronization batch parameters that are used to calculate the changes that need to be transmitted in an upload or download scenario.

HTTP method	Description
GET	Retrieve the synchronization batch parameters (that is, server knowledge) from the server.
PUT	Update the client's synchronization batch parameters (that is, client knowledge) to the server.

3.4.5.2.1 GET

The GET method on the Sync Batch Parameters resource is issued by the client to obtain the server's sync knowledge in an upload scenario.

This operation is transported by an HTTP GET request.

The operation can be invoked through the following URI suffix on the *x-ecs-session-location-url* returned in the response header of the PUT method on the Creation Session resource:

```
/SyncBatchParameters
```

The following is an example of a complete URI for this operation:

https://contoso.com/Sync/1.0/Session/{60b8ca1d-59f3-4495-b299-f6cad89ada3b}/SyncBatchParameters

The request message for this operation contains the following HTTP headers.

Request header	Usage	Value
x-ecs-partnershipID	Required	The PartnershipID for the sync target share returned by the server in the GET response on Share Discovery resource.

The response message for this operation contains the following HTTP headers.

Response header	Usage	Value
x-ecs-request-error	Optional	A string describing request failure information. This MUST be a string representation of the HRESULT in hexadecimal format. Example: x-ecs-request-error: "0x8007005"

The response message for this operation can result in the following status codes.

Status code	Description
200	Batch parameters are returned in the response body.
404	No session exists with the specified ID.

3.4.5.2.1.1 Request Body

None.

3.4.5.2.1.2 Response Body

The response body for this method MUST contain the following:

Entry	Type
SyncKnowledge	Array of BYTEs
BatchLimits	BATCH_LIMITS_ENTRY structure

SyncKnowledge: An implementation-specific<5> stream of bytes that represents the serialized server sync knowledge.

BatchLimits: Parameters describing batch characteristics.

3.4.5.2.1.3 Processing Details

The server MUST obtain the synchronization batch parameters in an implementation-specific manner. The server MUST set SyncKnowledge and BatchLimits entries in the response and set the status code to 200.

3.4.5.2.2 PUT

The PUT method on the Sync Batch Parameters resource is issued by the client to update the client's sync knowledge to the server in an upload scenario.

This operation is transported by an HTTP PUT request.

The operation can be invoked through the following URI suffix:

```
/SyncBatchParameters
```

The following is an example of a complete URI for this operation:

https://contoso.com/Sync/1.0/Session/{60b8ca1d-59f3-4495-b299-f6cad89ada3b}/SyncBatchParameters

The request message for this operation contains the following HTTP headers.

Request header	Usage	Value
x-ecs-	Required	The PartnershipID for the sync target share returned by the server in the

Request header	Usage	Value
partnershipID		GET response on Share Discovery resource.

The response message for this operation can result in the following status codes.

Status code	Description
200	Success
202	Background processing initiated.
404	No session exists with the specified ID.
409	Client knowledge already received.

3.4.5.2.2.1 Request Body

The request body for this method MUST contain the following.

Entry	Type
SyncKnowledge	Array of BYTEs
BatchLimits	BATCH_LIMITS_ENTRY structure

SyncKnowledge: An implantation-specific<6> stream of bytes that represents the serialized server sync knowledge.

BatchLimits: Parameters describing batch characteristics.

3.4.5.2.2.2 Response Body

The response body for this method MUST contain the following.

Entry	Type
TotalFileCount	ULONG
TotalFileSize	ULONGLONG

TotalFileCount: Total number of files that have changed on the server.

TotalFileSize: Sum of the sizes of all the files that have changed on the server.

3.4.5.2.2.3 Processing Details

Upon receiving this request, the server MUST generate a change batch corresponding to the files to be downloaded to the client.

3.4.5.3 Prepare Batch

The Prepare Batch resource is used by the client to request server-side preparation for a file upload in a sync process.

HTTP method	Description
PUT	Update server with the file information necessary to prepare for the upload.

3.4.5.3.1 PUT

This operation is transported by an HTTP PUT request.

The operation can be invoked through the following URI suffix:

```
/PrepareBatch/{BatchIndex}
```

The BatchIndex in the URI suffix is an unsigned integer representing a specific batch that the client wants to upload.

The following is an example of a complete URI for this operation:

https://contoso.com/Sync/1.0/Session/{60b8ca1d-59f3-4495-b299-f6cad89ada3b}/PrepareBatch/3

The request message for this operation contains the following HTTP headers.

Request header	Usage	Value
x-ecs-partnershipID	Required	The PartnershipID for the sync target share returned by the server in the GET response on Share Discovery resource.

The response message for this operation contains the following HTTP headers.

Response header	Usage	Value
x-ecs-request-error	Optional	A string describing request failure information. This MUST be a string representation of the HRESULT in hexadecimal format. Example: x-ecs-request-error: "0x8007005"

The response message for this operation can result in the following status codes.

Status code	Description
200	Batch parameters are returned in the response body.
404	No session exists with the specified ID.

3.4.5.3.1.1 Request Body

The request body for this method MUST contain the following.

Entry	Type
FileVector	VECTOR_FILE_INFO_INPUT_ENTRY

FileVector: A list of file info input entries that need to be prepared for the synchronization process.

3.4.5.3.1.2 Response Body

The response body for this method MUST contain the following:

Entry	Type
FileList	VECTOR_FILE_INFO_ENTRY

FileList: A list of file status entries indicating the status of the change batch preparation.

3.4.5.3.1.3 Processing Details

None.

3.4.5.4 Upload Batch

The Upload Batch resource is used by the client to commit a change batch on the server.

HTTP method	Description
PUT	Commit change batch.

3.4.5.4.1 PUT

This operation is transported by an HTTP PUT request.

The operation can be invoked through the following URI suffix:

```
/UploadBatch/{BatchIndex}
```

The BatchIndex in the URI suffix is an unsigned integer representing a specific batch that the client wants to upload.

The following is an example of a complete URI for this operation:

https://contoso.com/Sync/1.0/Session/{60b8ca1d-59f3-4495-b299-f6cad89ada3b}/ UploadBatch/3

The request message for this operation contains the following HTTP headers.

Request header	Usage	Value
x-ecs-partnershipID	Required	The PartnershipID for the sync target share returned by the server in the GET response on Share Discovery resource.

The response message for this operation contains the following HTTP headers.

Response header	Usage	Value
x-ecs-request-error	Optional	A string describing request failure information. This MUST be a string representation of the HRESULT in hexadecimal format. Example: x-ecs-request-error: "0x8007005"

The response message for this operation can result in the following status codes.

Status code	Description
200	Batch has been processed synchronously.
202	Batch has been accepted and will be processed in the background.
400	Batch is out of sequence, or the last batch has already been received.
404	Session not found.
409	Batch already received.

3.4.5.4.1.1 Request Body

The request body for this method MUST contain the following.

Entry	Type
SerializedSyncBatch	SYNC_CHANGE_BATCH

SerializedSyncBatch: The metadata describing the changes to be synchronized.

3.4.5.4.1.2 Response Body

The response body for this method MUST contain the following.

Entry	Type
FileStatusList	VECTOR_FILE_STATUS_ENTRY

FileStatusList: A list of file status entries indicating the status of change batch commit operation.

3.4.5.4.1.3 Processing Details

3.4.5.5 Delete Session

The Delete Session resource facilitates a client to remove a server session after a synchronization process is completed or aborted.

HTTP method	Description
DELETE	Delete server session.

3.4.5.5.1 DELETE

This operation is transported by an HTTP DELETE request.

The client MUST use the *x-ecs-session-location-url* returned by the server in the Creation Session response (see section 3.4.5.1.1) as the URI for this operation.

The following is an example of a complete URI for this operation:

https://contoso.com/Sync/1.0/Session/{60b8ca1d-59f3-4495-b299-f6cad89ada3b}

The request message for this operation contains the following HTTP headers.

Request header	Usage	Value
x-ecs-partnershipID	Required	The PartnershipID for the sync target share returned by the server in the GET response on Share Discovery resource.

The response message for this operation can result in the following status codes.

Status code	Description
200	The server will delete the session.
404	Session not found.

3.4.5.5.1.1 Request Body

None.

3.4.5.5.1.2 Response Body

None.

3.4.5.5.1.3 Processing Details

The server MUST extract the session ID from the URI in an implementation-specific manner and verify whether there is a **Session** that exists in the global **SessionList** with a matching session ID. If no session is found, the server MUST return a status code of 404. Otherwise, the server MUST delete the **Session** entry from **SessionList** and return a status code of 200.

3.4.5.6 Download Batch

The Download Batch resource is used by the client to obtain change batch information from the server in an upload scenario.

HTTP method	Description
GET	Query server knowledge on a change batch.

3.4.5.6.1 GET

The GET method on the Download Batch resource is issued by the client to obtain metadata information for all files in a change batch.

This operation is transported by an HTTP GET request.

The operation can be invoked through the following URI on the *x-ecs-session-location-url* that is returned in the response header of the PUT method on the Creation Session resource:

```
/DownloadBatch
```

The following is an example of a complete URI for this operation:

https://contoso.com/Sync/1.0/Session/{60b8ca1d-59f3-4495-b299-f6cad89ada3b}/DownloadBatch

The request message for this operation contains the following HTTP headers.

Request header	Usage	Value
x-ecs-partnershipID	Required	The PartnershipID for the sync target share returned by the server in the GET response on Share Discovery resource.
x-ecs-continue	Optional	The opaque continuation token received from the server in a download batch operation. This MUST NOT be present in the first batch of a sync process.

The response message for this operation contains the following HTTP headers.

Response header	Usage	Value
x-ecs-continue	Optional	The opaque continuation token that the client needs to use in the subsequent download batch requests on this session.

The response message for this operation can result in the following status codes.

Status code	Description
200	Success
400	Invalid continuation token.
404	No session exists with the specified ID.

3.4.5.6.1.1 Request Body

None.

3.4.5.6.1.2 Response Body

The response body for this method MUST contain the following.

Entry	Type
BatchMetadata	SYNC_CHANGE_BATCH
DownloadInfo	VECTOR_FILE_DOWNLOAD_INFO_ENTRY

BatchMetadata: The metadata describing the changes to be synchronized.

DownloadInfo: A collection of FILE_DOWNLOAD_INFO_ENTRY structures as specified in section 2.2.2.15.

3.4.5.6.1.3 Processing Details

None.

3.4.6 Timer Events

None.

3.4.7 Other Local Events

None.

3.5 ServerAPI Server Details

3.5.1 Abstract Data Model

None.

3.5.2 Timers

None.

3.5.3 Initialization

None.

3.5.4 Higher-Layer Triggered Events

None.

3.5.5 Message Processing Events and Sequencing Rules

Resource	Description
Upload Data	The resource used to send file data to the server.
Download Data	The resource used to receive file data from the server.

The responses to all the operations can result in the following status codes.

Status code	Reason phrase	Description
200	OK	Request completed.
409	Conflict	This is returned when a request is received to upload data that is out of order or is the wrong version of the batch.
416	Requested Range Not Satisfiable	This is returned when a request is received to upload data that is out of order or is the wrong version of the batch.

3.5.5.1 Upload Data

The Upload Data resource facilitates the uploading of file data from client to server.

HTTP method	Description
PUT	Upload file data to server.

3.5.5.1.1 PUT

This operation is transported by an HTTP PUT request.

The operation can be invoked through the following URI suffix on the session location URL constructed by the client as specified in section 3.6.5.1.

```
/UploadData/
```

The following is an example of a complete URI for this operation:

http://contoso.com/Sync/1.0/Session/{60b8ca1d-59f3-4495-b299-f6cad89ada3b}/UploadData

The request message for this operation contains the following HTTP headers.

Request header	Usage	Value
x-ecs-partnershipID	Required	The PartnershipID for the sync target share returned by the server in the GET response on the Share Discovery resource.

The response message for this operation can result in the following status codes.

Status code	Description
200	Request completed.
409	This is returned when a request is received to upload data that is out of order or is the wrong version of the batch.

3.5.5.1.1.1 Request Body

The request body for this method MUST contain the following:

Entry	Type
UploadEntryVector	VECTOR_UPLOAD_ENTRY

UploadEntryVector: A list of UPLOAD_ENTRY structures, as specified in section 2.2.2.11, for each file being uploaded.

3.5.5.1.1.2 Response Body

The response body for this method MUST contain the following:

Entry	Type
UploadResponseEntryVector	VECTOR_UPLOAD_RESPONSE_ENTRY

UploadResponseEntryVector: A list of UPLOAD_RESPONSE_ENTRY structures, as specified in section 2.2.2.12, indicating the server's response for each file being uploaded by the client.

3.5.5.1.1.3 Processing Details

The server MUST extract the session ID from the URI in an implementation-specific manner and verify whether a **Session** exists in a global **SessionList** with a matching session ID. If no session is found, the server MUST return a status code of 404.

3.5.5.2 Download Data

The Download Data resource facilitates downloading file data from server to client.

HTTP method	Description
PUT	Download file data from server.

3.5.5.2.1 PUT

This operation is transported by an HTTP PUT request.

The operation MUST be invoked through the following URI suffix on the session location URL constructed by the client as specified in section 3.6.5.2:

```
/DownloadData/
```

The following is an example of a complete URI for this operation:

http://contoso.com/Sync/1.0/Session/{60b8ca1d-59f3-4495-b299-f6cad89ada3b}/DownloadData

The request message for this operation contains the following HTTP headers.

Request header	Usage	Value
x-ecs-partnershipID	Required	The PartnershipID for the sync target share returned by the server in the GET response on Share Discovery resource.

The response message for this operation can result in the following status codes.

Status code	Description
200	Request completed.
416	This is returned when a request is received to download data that is out of order or is the wrong version of the batch.

3.5.5.2.1.1 Request Body

The request body for this method MUST contain the following.

Entry	Type
DownloadEntryVector	VECTOR_DOWNLOAD_ENTRY

DownloadEntryVector:

A list of DOWNLOAD_ENTRY structures, as specified in section 2.2.2.13, for each file to be downloaded from the server.

3.5.5.2.1.2 Response Body

The response body for this method MUST contain the following.

Entry	Type
DownloadResponseEntryVector	VECTOR_DOWNLOAD_RESPONSE_ENTRY

UploadResponseEntryVector: A list of DOWNLOAD_RESPONSE_ENTRY structures, as specified in section 2.2.2.14, indicating the server's response for each file being downloaded by the client.

3.5.5.2.1.3 Processing Details

For each file represented by a DOWNLOAD_ENTRY structure in DownloadEntryVector, the server MUST construct the DOWNLOAD_RESPONSE_ENTRY and add it to VECTOR_DOWNLOAD_RESPONSE_ENTRY. The VECTOR_DOWNLOAD_RESPONSE_ENTRY MUST be filled as the DownloadResponseEntryVector in the response body.

3.5.6 Timer Events

None.

3.5.7 Other Local Events

None.

3.6 ECS Client Details

3.6.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in the Enterprise Client Synchronization 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.

3.6.1.1 Global

SyncClientVersion: The highest version of the Enterprise Client Synchronization Protocol that the client supports. This MUST be one of the version values specified in section 1.7.

GlobalSyncURI: The URI to be used to enumerate all servers exposing a namespace for file synchronization.

SyncTargetHostList: A list of server URIs that expose the sync namespace.

HostURLPrefix: The URL prefix that the client currently uses to connect to for all sync operations.

SyncTargetPartnershipId: An identifier for the sync target share for a specific user.

SyncServerVersions: An array of strings representing the protocol versions supported by the server.

SyncServerDataTransferType: The type of data transfer that the server supports for file uploading or downloading.

SyncChangesURL: The URL to be used by the client to poll for changes on the server.

SessionLocationURL: The URL to access the session on the server.

UserDataUsed: The amount of data, in bytes, used by the user in the sync target share.

UserDataFree: The amount of data, in bytes, available for the user in the sync target share.

SyncBatchServerKnowledge: An opaque serialized binary stream representing the version of the file data on the server.

SyncBatchClientKnowledge: An opaque serialized binary stream representing the version of the file data on the client.

UploadFileList: A list of UploadFile objects, as specified in section 3.6.1.2.

DownloadFileList: A list of DownloadFile objects, as specified in section 3.6.1.3.

UploadBatchLimits: The parameters that describe batch characteristics in an upload scenario. This MUST be in the format of the BATCH_LIMITS structure as specified in section 2.2.2.6

DownloadBatchLimits: The parameters that describe batch characteristics in a download scenario. This MUST be in the format of the BATCH_LIMITS structure as specified in section 2.2.2.6.

3.6.1.2 Per UploadFile

SyncId: An identifier of the file that is used to correlate the change batch information from the underlying Sync Framework.

DataUploadURL: The URI to be used for performing data transfer for this file.

CommitStatus: The status, as an HRESULT value, of the commit operation after file data has been uploaded.

3.6.1.3 Per DownloadFile

SyncId: An identifier of the file that is used to correlate the with the change batch information from the underlying Sync Framework.

ConcurrencyInfo: An opaque stream of bytes that identifies the file version.

3.6.2 Timers

None.

3.6.3 Initialization

When the ECS client is started, it MUST do the following:

The client MUST initialize **SyncClientVersion** in an implementation-specific<7> manner.

The client MUST initialize **GlobalSyncURI** based on local configuration.

The client MUST initialize **SyncTargetHostList** to empty.

The client MUST initialize **HostURLPrefix** to **GlobalSyncURI**.

The client MUST then synchronize with the data on the server by performing the following steps:

Server Discovery:

The client MUST send a GET request on Discover Server resource, as specified in section 3.1.5.1, using **GlobalSyncURI** as the URI.

If the server responds with a status of 200, the client MUST set **SyncTargetHostList** to the ServerUrls entry that the server returned in the response body.

Share Discovery:

For each host URL Prefix in the **SyncTargetHostList**:

The client MUST attempt to send a GET request on Share Discovery resource, as specified in section 3.1.5.2.1, using the format "<host-url-prefix>/ Sync/**ClientSyncVersion**/Discover/Share".

If the server responds with a status of 200, the client MUST store the *PartnershipId* entry from the response body as **SyncTargetPartnershipId**. The client MUST set **HostURLPrefix** to the URL prefix of the server that returned a status code of 200 and skip the remaining entries in the **SyncTargetHostList** for share discovery.

If the server responds with a status code other than 200, the client MUST move to the next entry in the **SyncTargetHostList** and attempt the share discovery. The server MUST repeat this process until it gets a success response (200) or all the entries in the **SyncTargetHostList** are exhausted. If all the entries in **SyncTargetHostList** get exhausted before the client gets a success response, the client MUST wait for an implementation-specific period of time and retry the Server Discovery and Share Discovery steps mentioned earlier.

The client MUST set **SyncChangesURL** to **HostURLPrefix**.

Server Capabilities:

The client MUST send a GET request on the Server Capabilities resource, as specified in section 3.1.5.3.1, using the URI format "**HostURLPrefix**/Sync/**ClientSyncVersion**/Capabilities".

If the server responds with a status of 200, the client MUST store the entries in the response body as follows:

- SyncServerVersions is set to x-ecs-supported-versions in the response header
- SyncServerDataTransferType is set to ProtocolType in the response body

Obtain User Configuration:

The client MUST send a GET request on the User Configuration resource, as specified in section 3.3.5.1.1, by using the URI format "**HostURLPrefix**/Sync/SyncClientVersion/UserConfiguration"

If the server responds with a status code of 200, the client MUST store the entries in the response body as follows:

- *UserUsage* member of *QuotaUsage* entry is set to **UserDataUsed**.
- *UserDataFreeSpace* member of *QuotaUsage* entry is set to **UserDataFree**.

Poll for Server Changes:

The client MUST poll for the server changes by sending a GET request on the Detect Server Changes resource, as specified in section 3.2.5.1.1, using the URI format "**SyncChangesURL**/Sync/**SyncClientVersion**/Changes".

If the server responds with a status code of 200, the client MUST initiate file download process as outlined in section 3.6.5.2. Once the download process is completed, the client MUST continue to poll for the server changes by issuing a new request using the ETag returned by the server in the previous response as the *If-None-Match* header in the new request.

If the server responds with a status code of 304, the client SHOULD<8> wait for an implementation-specific period of time and reissue the same request.

If the server does not respond within an implementation-specific timeout, the client MUST reissue the same request.

3.6.4 Higher-Layer Triggering Events

The following sections describe the operations performed by the ECS client in response to events triggered by higher-layer applications.

3.6.4.1 Application Requests Uploading Data To Sync Target

The application provides the following:

- An opaque binary stream representing the version of the files changed
- Size of the file data, in bytes, to be uploaded to the server

If the size of the file data to be uploaded is greater than the value of **UserDataFree**, the client MUST return an implementation-specific error to the calling application.

The client MUST perform the steps as specified in section 3.6.5.1 to upload the file data to the sync server.

3.6.5 Message Processing Events and Sequencing Rules

The following sections describe the sequence of operations performed by the client in upload and download scenarios.

3.6.5.1 Upload Scenario

The client MUST do the following:

Create Session:

The client MUST send a PUT request on Create Session resource, as specified in section 3.4.5.1.1, by using the URI format "**HostURLPrefix**/Sync/**SyncClientVersion**/Session". The x-ecs-partnershipId header must be set to **SyncTargetPartnershipId**. The x-ecs-devicename header MUST be set to an implementation-specific string.

If the server responds with a status code of 200, the client MUST construct the SessionLocationURL using the x-ecs-session-id header as follows:

"**HostUrlPreifx**/Sync/**SyncClientVersion**/Session/<x-ecs-session-id>/"

Get Sync Batch Parameters from Server:

The client MUST send a GET request on the Sync Batch Parameters resource, as specified in section 3.4.5.2.1, by using the URI format "**SessionLocationURL**/SyncBatchParameters". The *x-ecs-partnershipId* header must be set to **SyncTargetPartnershipId**.

If the server responds with a status code of 200, the client MUST store the entries in the response body as follows:

- SyncBatchServerKnowledge is set to **SyncKnowledge**.
- UploadBatchLimits is set to **BatchLimits**.

Inform Server to Prepare Batch:

The client MUST send a PUT request on the Prepare Batch resource, as specified in section 3.4.5.3.1, by using the URI format "**SessionLocationURL**/PrepareBatch/<BatchIndex>". The BatchIndex in the URI MUST be set to a value that uniquely identifies the batch within the session. Using the **SyncBatchServerKnowledge** and the application-supplied version information, the client MUST construct a VECTOR_FILE_INFO_INPUT_ENTRY structure in an implementation-specific manner and insert the same as the FileVector entry in the request body.

If the server responds with a status code of 200, the client MUST store the information in FILE_INFO_ENTRY structures in FileList in the response body as follows:

- If the ProtocolType is 1, create a new **UploadFile** object and initialize it as follows:
  - **UploadFile.SyncId** MUST be set to *SyncItemId*.
- Insert the UploadFile object into the global **UploadFileList**.

Data Transfer:

From each entry in UploadFileList, the client MUST construct a VECTOR_UPLOAD_ENTRY structure and MUST send a PUT request on Upload Data resource, as specified in section 3.5.5.1.1, by using the URI format "SessionLocationURL/UploadData".

Commit Change Batch:

The client MUST send a PUT request on the Upload Batch resource, as specified in section 3.4.5.4.1, by using the URI format "**SessionLocationURL**/UploadBatch/<BatchIndex>". The BatchIndex in the URI MUST be set to an implementation-specific value that identifies the batch to be uploaded. The SerializedSyncBatch in the request body MUST be set to an opaque binary stream representing the change batch.

For each **UploadFile** in **UploadFileList**, the client MUST update **UploadFile.CommitStatus** to the Status entry returned in the FILE_STATUS structure.

Delete Session:

The client MUST sent a DELETE request on Delete Session resource, as specified in section 3.4.5.5.1, by using the URI format "**SessionLocationURL**".

The client MUST clear all entries in the **UploadFileList**.

3.6.5.2 Download Scenario

The client MUST do the following:

Create Session:

The client MUST send a PUT request on the Create Session resource, as specified in section 3.4.5.1.1, by using the URI format "**HostURLPrefix**/Sync/**SyncClientVersion**/Session". The x-ecs-partnershipId header must be set to **SyncTargetPartnershipId**. The x-ecs-devicename header MUST be set to an implementation-specific string.

If the server responds with a status code of 200, the client MUST construct the SessionLocationURL using the x-ecs-session-id header as follows:

"**HostUrlPreifx**/Sync/**SyncClientVersion**/Session/<x-ecs-session-id>/"

Update Server with Sync Batch Parameters:

The client MUST obtain the information on the version of files in an implementation-specific manner and set it to **SyncBatchClientKnowledge**. The client MUST set **DownloadBatchLimits** to implementation-specific values.

The client MUST send a PUT request on the Sync Batch Parameters resource, as specified in section 3.4.5.2.2, by using the URI format "**SessionLocationURL**/SyncBatchParameters". The *SyncKnowledge* and *BatchLimits* in the request body MUST be set to **SyncBatchClientKnowledge** and **DownloadBatchLimits**, respectively.

Obtain Change Batch details:

The client MUST obtain server knowledge on the change batch by sending a GET request on the Download Batch resource, as specified in section 3.4.5.6.1, by using the URI format "**SessionLocationURL**/DownloadBatch". If the server responds with a status code of 200, the client MUST create a list of **DownlodFile** objects using the ChangeBatch content in the response body and insert the **DownloadFile** objects into **DownloadFileList**.

Data Transfer:

The client MUST send a PUT request on the Download Data resource, as specified in section 3.5.5.2.1. The DownloadEntryVector in the request body MUST be set to a VECTOR_DOWNLOAD_ENTRY constructed in an implementation-specific manner using each **DownloadFile** object in **DownloadFileList**.

If the server responds with a status code of 200, the client MUST update the local file data corresponding to SyncItemId member of each DOWNLOAD_RESPONSE_ENTRY structure in the response body.

Delete Session:

The client MUST send a DELETE request on the Delete Session resource, as specified in section 3.4.5.5.1, by using the URI format "**SessionLocationURL**".

3.6.6 Timer Events

None.

3.6.7 Other Local Events

None.

4 Protocol Examples

None.

5 Security

5.1 Security Considerations for Implementers

The Enterprise Client Synchronization Protocol requires that all the requests from the client be authenticated. The client is expected to use an implementation-dependent authentication mechanism to obtain a security token and include that token in the standard HTTP Authorization header. The server will validate the token and use it to authorize the request.

5.2 Index of Security Parameters

None.

6 Appendix C: 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:

- Windows 8.1 operating system
- Windows Server 2012 R2 operating system

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.

<1> Section 1.3: The Windows ECS Clients depend on the Microsoft Sync Framework as described in [MSDN-MSF] to receive notifications of local file changes.

<2> Section 1.7: The following table illustrates the Windows operating system versions that support the ECS client and server.

ECS client	ECS Server
Windows 8.1	Windows Server 2012 R2

<3> Section 2.2.2.7: Windows uses the File Attribute Constants as defined in [MSDN-FSA].

<4> Section 2.2.2.16: Windows uses the format as described in [MSDN-CBB2-SerializeWithOptions].

<5> Section 3.4.5.2.1.2: Windows servers use the format as described in [MSDN-ISK2-SerializeWithOptions] for this field.

<6> Section 3.4.5.2.2.1: Windows clients use the format as described in [MSDN-ISK2-SerializeWithOptions] for this field.

<7> Section 3.6.3: Windows clients set this to "1.0".

<8> Section 3.6.3: If x-ecs-change-detection header is not present in the response, Windows 8.1 client uses a default wait time of 10 minutes, which is configurable. If x-ecs-change-detection header is present in the response, Windows 8.1 client uses a wait time of 30 seconds.

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