[MS-DLTW]:
Distributed Link Tracking:
Workstation Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.

- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.

- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.

- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.

- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.

- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
10/22/2006	0.01		MCPP Milestone 1 Initial Availability
01/19/2007	1.0		MCPP Milestone 1
03/02/2007	1.1		Monthly release
04/03/2007	1.2		Monthly release
05/11/2007	1.3		Monthly release
06/01/2007	1.3.1	Editorial	Revised and edited the technical content.
07/03/2007	2.0	Major	New section, Receiving a FSCTL_LMR_SET_LINK_TRACKING_INFORMATION Request.
07/20/2007	2.1	Minor	Updated the technical content.
08/10/2007	3.0	Major	Updated and revised the technical content.
09/28/2007	4.0	Major	Updated and revised the technical content.
10/23/2007	4.1	Minor	Updated the IDL.
11/30/2007	4.1.1	Editorial	Revised and edited the technical content.
01/25/2008	4.1.2	Editorial	Revised and edited the technical content.
03/14/2008	4.1.3	Editorial	Revised and edited the technical content.
05/16/2008	4.1.4	Editorial	Revised and edited the technical content.
06/20/2008	5.0	Major	Updated and revised the technical content.
07/25/2008	6.0	Major	Updated and revised the technical content.
08/29/2008	6.0.1	Editorial	Revised and edited the technical content.
10/24/2008	6.0.2	Editorial	Revised and edited the technical content.
12/05/2008	6.1	Minor	Updated the technical content.
01/16/2009	6.1.1	Editorial	Revised and edited the technical content.
02/27/2009	6.1.2	Editorial	Revised and edited the technical content.
04/10/2009	6.1.3	Editorial	Revised and edited the technical content.
05/22/2009	6.1.4	Editorial	Revised and edited the technical content.
07/02/2009	6.2	Minor	Updated the technical content.
08/14/2009	6.2.1	Editorial	Revised and edited the technical content.

Release: Friday, February 4, 2011

Date	Revision History	Revision Class	Comments
09/25/2009	6.3	Minor	Updated the technical content.
11/06/2009	6.3.1	Editorial	Revised and edited the technical content.
12/18/2009	7.0	Major	Updated and revised the technical content.
01/29/2010	8.0	Major	Updated and revised the technical content.
03/12/2010	8.0.1	Editorial	Revised and edited the technical content.
04/23/2010	8.0.2	Editorial	Revised and edited the technical content.
06/04/2010	8.0.3	Editorial	Revised and edited the technical content.
07/16/2010	8.0.3	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	9.0	Major	Significantly changed the technical content.
10/08/2010	9.0	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	9.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	9.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	9.0	No change	No changes to the meaning, language, or formatting of the technical content.

Release: Friday, February 4, 2011

Contents

1   Introduction

This document specifies the Distributed Link Tracking: Workstation Protocol.

Distributed Link Tracking (DLT) consists of two protocols that work together to discover the new location of a file that has moved. DLT can determine if the file has moved on a mass-storage device, within a computer, or between computers in a network. The Distributed Link Tracking: Workstation Protocol helps a computer locate files that have been moved within a computer or between computers in a computer network.

In addition to the Distributed Link Tracking: Workstation Protocol, DLT includes the Distributed Link Tracking: Central Manager Protocol that keeps track of file and **volume** moves as well as other relevant information from participating computers so it can provide this information in response to **workstation** queries. Both DLT protocols are **remote procedure call (RPC)** interfaces.

1.1   Glossary

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

> **Distributed Link Tracking (DLT)**
> **dynamic endpoint**
> **endpoint**
> **Fid**
> **FileLinkInformation**
> **FileLocation**
> **globally unique identifier (GUID)**
> **Interface Definition Language (IDL)**
> **MachineID**
> **Microsoft Interface Definition Language (MIDL)**
> **named pipe**
> **NetBIOS Name**
> **ObjectID**
> **opnum**
> **remote procedure call (RPC)**
> **RPC protocol sequence**
> **security provider**
> **Server Message Block (SMB)**
> **Universal Naming Convention (UNC)**
> **universally unique identifier (UUID)**
> **volume**
> **VolumeID**
> **well-known endpoint**

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

> **FileID**

The following terms are specific to this document:

> **CrossVolumeMoveFlag:** A value of either zero or one, stored as an attribute on a file, which is used to indicate whether the file has been moved across volumes at any time in the past.

> **workstation:** A terminal or desktop computer in a network. In this context, **workstation** is just a generic term for a user's (client) machine, in contrast to a server.

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

1.2   References

1.2.1   Normative References

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

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, http://www.opengroup.org/public/pubs/catalog/c706.htm

[MS-DTYP] Microsoft Corporation, "Windows Data Types", January 2007.

[MS-ERREF] Microsoft Corporation, "Windows Error Codes", January 2007.

[MS-FSCC] Microsoft Corporation, "File System Control Codes", July 2007.

[MS-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions", January 2007.

[MS-SMB] Microsoft Corporation, "Server Message Block (SMB) Protocol Specification", July 2007.

[MS-SMB2] Microsoft Corporation, "Server Message Block (SMB) Version 2 Protocol Specification", July 2007.

[RFC1088] McLaughlin III, L., "A Standard for the Transmission of IP Datagrams over NetBIOS Networks", STD 48, RFC 1088, February 1989, http://www.ietf.org/rfc/rfc1088.txt

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

1.2.2   Informative References

[MS-DLTM] Microsoft Corporation, "Distributed Link Tracking: Central Manager Protocol Specification", January 2007.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

1.3   Overview

The Distributed Link Tracking: Workstation Protocol is based on the RPC runtime, as specified in [C706] and [MS-RPCE], and on the **server message block (SMB)** protocol and extensions, as specified in [MS-SMB] and [MS-SMB2].

This protocol is used by a client to get a file's identity and location on the server computer as a **MachineID**, **FileID**, **FileLocation**, and **Universal Naming Convention (UNC)** name. If a client contacts a server that previously stored the file, but the file has been moved to a new computer, the server may be able to return the MachineID of the computer to which the file was moved, so that the client can contact the DLT Workstation server on the new computer to get the file's current UNC and FileLocation, or another referral. This process of following referrals continues until a server returns the file's UNC name and FileLocation, or an error.

Rather than following referrals in this manner, a client may use the Distributed Link Tracking: Central Manager Protocol to determine a file's current MachineID and FileLocation, and then use that information to initiate a call to the DLT Workstation server on the computer indicated by that MachineID. For more information on using the Distributed Link Tracking: Central Manager Protocol in combination with the Distributed Link Tracking: Workstation Protocol, see [MS-DLTM].

The following is a scenario that describes the DLT protocols working together:

1. A file is created on computer M1. M1 assigns identifiers, specifically FileID and FileLocation, to the file.

2. Computer M0 takes note of the file, locally storing its identifiers.

3. The file is moved from computer M1 to M2 and from there to M3. In concert with these moves, the file maintains its FileID but gets a new FileLocation assigned.

4. If the Distributed Link Tracking: Central Manager Protocol is used, clients on computers M1 and M2 notify the server that the file has been moved, indicating the file's FileID and its old and new FileLocation values.

5. Computer M0 finds the file in its new location in one of two ways:

   1. Using only the Distributed Link Tracking: Workstation Protocol:

      ▪ M0 contacts M1, using the identifiers stored previously, and learns that the file was moved to M2.

      ▪ M0 contacts M2 and learns that the file was moved to M3.

      ▪ M0 contacts M3 and learns the file's new name and location.

   2. Using both the Distributed Link Tracking: Workstation Protocol and the Distributed Link Tracking: Central Manager Protocol:

      ▪ M0 contacts a DLT Central Manager server to query the current location of the file.

      ▪ The server queries its tables and determines that the file is currently on computer M3.

      ▪ M0 contacts the DLT Workstation client on M3 and learns the file's new name and location.

The following is an example of a file being moved between computers, and shows in more detail how to use the Distributed Link Tracking: Workstation Protocol to determine the file's new location. In this example, only the Distributed Link Tracking: Workstation Protocol is used, without using the Distributed Link Tracking: Central Manager Protocol.

1. In the initial state, a file is located on a computer named M1. Assume that the file is named "F1.txt", and can be located via the UNC "\\M1\share1\F1.txt".

2. Before the file is moved, a user on computer M0 requests that information about the file be saved,<1> so that its location can be determined after it has been moved. As a result, M0 stores the UNC, the MachineID, the FileLocation, and the FileID.

3. The file is moved from machine M1 to machine M2; for example, to the UNC "\\M2\share2\F2.txt". M1 stores the file's old FileLocation and FileID, as well as the file's new FileLocation and MachineID.

4. When M0 attempts to open the file<2> by using the UNC "\\M1\share1\F1.txt", it receives a file-not-found error message. M0 then initiates a call to the DLT Workstation server on M1 with the previously stored FileID and FileLocation of the file.

5. The DLT Workstation server on M1 returns to the DLT Workstation client on M0 that the file was moved, and specifies the MachineID of the file's new location (M2), as well as the file's new FileLocation value.

6. M0 then repeats the call, but this time to the DLT Workstation server on M2 with the new FileLocation value.

7. The DLT Workstation server on M2 returns to the DLT Workstation client on M0 the file's new UNC, "\\M2\share2\F2.txt".

8. The DLT Workstation client on M0 then updates its stored values with the updated UNC and FileLocation values.

1.4   Relationship to Other Protocols

The Distributed Link Tracking: Workstation Protocol is dependent on the RPC runtime, as specified in [C706] and [MS-RPCE], which is one of its transport protocols.

The other transport protocol used is the Server Message Block Protocol, as specified in [MS-SMB] and [MS-SMB2]. The messages sent on this protocol are specified in [MS-FSCC].

The Distributed Link Tracking: Central Manager Protocol [MS-DLTM] includes a means for the server to return to the client information about the new location of a file that has been moved. If such a server is available, it can be used to determine the correct computer on which to initiate a call to a DLT Workstation server.

1.5   Prerequisites/Preconditions

The Distributed Link Tracking: Workstation Protocol defines an RPC interface and, as a result, has the prerequisites as specified in [C706] and [MS-RPCE] as those common to RPC interfaces.

For security purposes, this protocol also assumes that security credentials usable with RPC already exist between the client and the server. That is, the client possesses credentials that the server will treat as authorized.

1.6   Applicability Statement

The Distributed Link Tracking: Workstation Protocol is applicable to computers in a network that share files, as specified in the Server Message Block (SMB) Protocol, as specified in [MS-SMB] and [MS-SMB2], when each of the following is true:

▪ The computer holds files.

▪ References are made to files.

▪ Files are moved to new locations within the computer or to a new computer.

▪ It is expected that file references are to be found after the referent file has been moved in this way.

This protocol is less likely to be useful in situations where large numbers of files are being moved between volumes. The server maintains a table of files, as specified in section 3.1.1, that have been linked to and have been moved off a volume. This table is limited to 10,000 entries. When it

exceeds this limit, the oldest entries are reused. If an entry in the table for a file has been overwritten, it might not be possible for a server to return information about the location to which a file has been moved. If the Distributed Link Tracking: Central Manager Protocol is being used in conjunction with this protocol, it might not be necessary to use the tables for this protocol. However, the tables of the DLT Central Manager also have size limits, as specified in [MS-DLTM] section 3.1.1.

1.7   Versioning and Capability Negotiation

This section covers versioning issues in the area of supported transports. The protocol is defined over multiple **named pipes**. The mechanism for determining the named pipe to use is specified in section 2.1.

1.8   Vendor-Extensible Fields

The Distributed Link Tracking: Workstation Protocol does not define any vendor-extensible fields other than fields that contain **globally unique identifiers (GUIDs)**, as specified in [MS-DTYP] (section 2.3.2.2), and UNC names. The methods for generating **GUIDs** are as specified in [C706].

This protocol uses HRESULTs, as specified in [MS-ERREF] section 2.1 and discussed in HRESULT (section 2.2.1). Vendors are free to choose their own values for this field, so long as the C bit (0x20000000) is set, indicating that it is a customer code.

1.9   Standards Assignments

Parameter	Value	Reference
RPC interface UUID, as specified in **[MS-DTYP] (section 2.3.2.1)**, for Distributed Link Tracking: Workstation Protocol	300f3532-38cc-11d0-a3f0-0020af6b0add	[C706]. See also section 2.1.
Named pipe	\\pipe\ntsvcs	[MS-RPCE]. See also section 2.1.
Named pipe	\\pipe\trkwks	[MS-RPCE]. See also section 2.1.

2 Messages

2.1 Transport

The Distributed Link Tracking: Workstation Protocol MUST use the **RPC protocol sequence** "ncacn_np", as specified in [MS-RPCE] section 2.1.1.2.

This protocol MUST use the following named pipes, which are **well-known endpoints**:

- \\pipe\ntsvcs

- \\pipe\trkwks

There is no functional difference between these two pipes. The DLT Workstation server MAY<3> listen on the **endpoint** named "\\pipe\ntsvcs" and SHOULD listen on the endpoint named "\\pipe\trkwks". The DLT Workstation client MAY use either endpoint name in its calls, but SHOULD attempt to use the "\\pipe\trkwks" endpoint first, and fall back to "\\pipe\ntsvcs" if that fails, as specified in section 3.2.4.1.

This protocol MUST use the following parameters:

- **UUID**: The RPC interface UUID value defined in section 1.9 MUST be used.

- **Version number**: 1.2.

2.2 Common Data Types

In addition to RPC base types, the following sections use the definition of a GUID, as specified in [C706] Appendix A.

2.2.1 HRESULT

HRESULT is a 32-bit, signed integer that is returned by RPC method calls. A negative value indicates an error, while a nonnegative value denotes success.

HRESULT is specified in [MS-ERREF] section 2.1. In the Distributed Link Tracking: Workstation Protocol, HRESULT is returned by the **LnkSearchMachine** method, which is specified in section 3.1.4.1.

2.2.2 CMachineId

The **CMachineId** structure is used to represent a MachineID, which is a unique identifier that represents the identity of a computer.

```
typedef struct CMachineId {
  char _szMachine[16];
} CMachineId;
```

_szMachine: This member MUST be a **NetBIOS name**, as specified in [RFC1088]. This name MUST be terminated with a zero byte, and any remaining bytes MUST also be zero.

2.2.3   CDomainRelativeObjId

The **CDomainRelativeObjId** structure is used to represent a file's FileLocation or its FileID, which is the FileLocation of a file at the time it was originally created.

```
typedef struct CDomainRelativeObjId {
  CVolumeId _volume;
  CObjId _object;
} CDomainRelativeObjId;
```

_volume: A volume identifier for the volume that contains the file, as specified in section 2.2.4.

_object: A file identifier for the file, as specified in section 2.2.5.

2.2.4   CVolumeId

The **CVolumeId** type is used to represent a **VolumeID**, which is a unique identifier that represents the identity of a file system volume.

```
typedef struct CVolumeId {
  GUID _volume;
} CVolumeId;
```

_volume: This field MUST contain a GUID for a volume. The lowest-order bit of this value MUST be zero. Further restrictions on the value of a VolumeID are defined in section 3.1.1.

2.2.5   CObjId

The **CObjId** type is used to represent an **ObjectID**, which is a unique identifier that represents the identity of a file within a file system volume.

```
typedef struct CObjId {
  GUID _object;
} CObjId;
```

_object: This field MUST contain an identifier for a file. It is unique within the volume on which the file was created. Restrictions on the value of a VolumeID are defined in section 3.1.1.

3   Protocol Details

The Distributed Link Tracking: Workstation Protocol is in the form of an RPC interface. The server of this interface MUST respond to **LnkSearchMachine** calls.

3.1   DLT Workstation 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 this protocol. This description is provided to explain 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 specified in this document.

The DLT Workstation server maintains the following information:

Volumes: The volumes on the computer on which the server is running. Each volume has a VolumeID, and each volume stores a **MoveTable** (described later). Each VolumeID MUST be unique within the server computer. A protocol server MAY have VolumeID values generated for it by using the Distributed Link Tracking: Central Manager Protocol, as specified in [MS-DLTM], section 3.2.5.3.<4>

Files: Each file on a volume that is to be tracked across moves MUST have an ObjectID, a FileID, and a **CrossVolumeMoveFlag** associated with it. The ObjectID MUST be created for the file before a link is made to the file, and MUST be unique within a volume. If the CrossVolumeMoveFlag is zero, the ObjectID component of the FileID MUST be the same as the file's ObjectID, and the VolumeID component of the FileID MUST be the same as the volume's VolumeID. File locations can be specified with a UNC. <5> <6>

MoveTable: A **MoveTable** is stored on each volume. The **MoveTable** has an entry for each file that has been moved off the volume, where the file had an ObjectID. An entry in the table maps the file's ObjectID to the MachineID of the computer to which the file was moved, along with the file's new FileLocation at that location. The **MoveTable** MUST NOT hold more than the most recent 10,000 entries.

Note The preceding conceptual data can be implemented by using a variety of techniques. Any data structure that stores the preceding conceptual data may be used in the implementation.

3.1.2   Timers

No timers are associated with this protocol.

3.1.3   Initialization

The server initializes the RPC protocol as specified in section 2.1.

The server MAY<7> listen on the "\\pipe\ntsvcs"named pipe endpoint. The server SHOULD also listen on the "\\pipe\trkwks"named pipe endpoint.

There MUST NOT be more than one server instance running on a computer.

3.1.4   Message Processing Events and Sequencing Rules

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use Opnum: 2
Opnum3NotUsedOnWire	Reserved for local use Opnum: 3
Opnum4NotUsedOnWire	Reserved for local use Opnum: 4
Opnum5NotUsedOnWire	Reserved for local use Opnum: 5
Opnum6NotUsedOnWire	Reserved for local use Opnum: 6
Opnum7NotUsedOnWire	Reserved for local use Opnum: 7
Opnum8NotUsedOnWire	Reserved for local use Opnum: 8
Opnum9NotUsedOnWire	Reserved for local use Opnum: 9
Opnum10NotUsedOnWire	Reserved for local use Opnum: 10
Opnum11NotUsedOnWire	Reserved for local use Opnum: 11
LnkSearchMachine	Searches for a file object on the specified computer. If information on the file is found, the method attempts to return it. If the file has been moved, the method returns information about its new location. Opnum: 12

In the preceding table, the term "Reserved for local use" means that the client MUST NOT send the **opnum**, and the server behavior is undefined <8> because it does not affect interoperability.

The methods MUST NOT throw an exception.

3.1.4.1   Receiving a LnkSearchMachine Call (Opnum 12)

The **LnkSearchMachine** method searches for a file object on the specified computer. If information on the file is found, the method attempts to return it. If the file has been moved, the method returns information about its new location.

```
HRESULT LnkSearchMachine(
  [in] unsigned long Restrictions,
  [in] const CDomainRelativeObjId* pdroidBirthLast,
  [in] const CDomainRelativeObjId* pdroidLast,
  [out] CDomainRelativeObjId* pdroidBirthNext,
  [out] CDomainRelativeObjId* pdroidNext,
  [out] CMachineId* pmcidNext,
  [out, max_is(261), string] wchar_t* ptszPath
);
```

Restrictions: SHOULD be set to zero.<9>

pdroidBirthLast: FileID of the file for which the server is to search.

pdroidLast: FileLocation of the file for which the server is to search. This is the last known FileLocation by the client.

pdroidBirthNext: A new FileID returned from the server. This indicates that the server could not find the file, but that a different file that might be correct has been found. This is associated with the **TRK_E_POTENTIAL_FILE_FOUND** return value, which is defined below. How the server responds to that error is specified later in this section.

pdroidNext: New FileLocation for the file returned from the server.

pmcidNext: New MachineID for the computer that holds the file.

ptszPath: New UNC for file.

Return Values: A 32-bit integer that indicates success or failure: A value of 0 or any positive value indicates success; a negative value indicates failure. Some of the possible return codes are listed in the following table.

Return value/code	Description
0x8DEAD106 TRK_E_POTENTIAL_FILE_FOUND	A file was found by a DLT Workstation server processing a **LnkSearchMachine** call, and the file meets some—but not all—of the necessary criteria to be considered the correct file. More details are presented later in this section.
0x8DEAD101 TRK_E_REFERRAL	A file could not be found by a DLT Workstation server processing a **LnkSearchMachine** call, and the file meets some—but not all—of the necessary criteria to be considered the correct file. More details are presented later in this section.

Exceptions Thrown: None.

The server responds to this call by attempting to find the requested file in one of its volumes, and either returns the file's new location (in the form of a UNC), or returns information about the location to which the file has moved (see description of **TRK_E_REFERRAL** in the following bulleted list).

The server SHOULD<10> return a failure value, if the file is found, but the length of the path to be returned in the *pstszPath* argument exceeds 261 characters (where the 261 characters does not include a string terminator character).

For each of the possible return values, the output parameters of this call MUST be set as specified in the following list. In these specifications, the requested ObjectID MUST correspond to the **Object** field of the *pDroidLast* parameter, the requested VolumeID MUST correspond to the **Volume** field of the *pDroidLast* parameter, and the FileID requested MUST correspond to the value of the *pDroidBirthLast* parameter. The numeric values corresponding to these returns are defined in section 3.1.1:

- Success: A success value MUST be returned if the server finds the file, and the client is authorized to get information about the file via a UNC path. The file MUST be found by searching all volumes on the server computer for a file whose ObjectID is equal to the ObjectID of the request, and whose FileID is equal to the FileID of the request. To perform the authorization check, the server MUST use the client's identity (obtained as specified in [MS-RPCE] section 3.3.3.4.3) to determine, based on local policy, whether or not the client is authorized to get the UNC of the file.

  If there is more than one file on the server computer that satisfies these conditions, the file MUST be selected as follows:

  - If one of the matching files is on the volume whose VolumeID equals the VolumeID of the request, that file is selected.

  - Otherwise, the behavior is arbitrary.

  The server MUST set the output parameters as follows:

  pdroidBirthNext: The value of the *pdroidBirthLast* that is specified by the client.

  pdroidNext: The current FileLocation value for the file.

  pmcidNext: The MachineID of the server computer.

  ptszPath: The UNC of the file on the server computer. If the file can be located under multiple UNC paths, one of those UNC paths MUST be returned, but the server MAY return any of those paths.<11>

- TRK_E_REFERRAL: This value MUST be returned if the file is no longer stored on any of the volumes of the server computer, but if the MoveTable of the last volume has an entry for the file. (Note that there is no additional per-file access check in this case.) The last volume MUST be determined by using the volume on the server whose VolumeID equals the VolumeID of the request. The entry in the MoveTable for the file, as defined in section 3.1.1, MUST be that entry whose ObjectID equals the ObjectID of the request.

  The server MUST set the output parameters as follows:

  pdroidBirthNext: The value of the FileID specified by the client in the **pdroidBirthLast** field of the call.

  pdroidNext: The value of the FileLocation field for the file's entry in the **MoveTable**.

  pmcidNext: The value of the MachineID field for the file's entry in the **MoveTable**.

  ptszPath: The server MUST NOT modify this value.

- TRK_E_POTENTIAL_FILE_FOUND: This value MUST be returned if there is no file on any of the server's volumes with the requested FileID and ObjectID, and there is no entry for this file's ObjectID in the MoveTable, but if there is a file with the requested ObjectID on one of the server's volumes, and the FileID on that file is all zeros.<12>

The server MUST set the output parameters as follows:

pdroidBirthNext: The value of the FileID for the found file.

pdroidNext: The FileLocation of the file found by the server.

pmcidNext: The MachineID of the server computer.

ptszPath: A UNC of the file on the server computer.

- Other negative return values: A negative value other than those mentioned above MUST be returned if none of the preceding cases is met. The server MUST NOT modify any of the output parameters.

3.1.4.2   Receiving a FSCTL_LMR_SET_LINK_TRACKING_INFORMATION Request

The FSCTL_LMR_SET_LINK_TRACKING_INFORMATION request is defined in [MS-FSCC], section 2.3.23. If the connection to the remote machine is using the [MS-SMB] protocol, this request is received via an NT_TRANSACT_IOCTL subcommand of an SMB_COM_NT_TRANSACTION request, as specified in [MS-SMB] section 2.2.7.2.1. Otherwise, this request is received in an SMB2 IOCTL request, as specified in [MS-SMB2] section 2.2.31.

This request is sent when a file has been moved by a remote machine between volumes within the local machine, or from the local machine to another machine, as defined in section 3.1.6.4.

Requests can be sent with or without a **TargetFileObject** value set in the REMOTE_LINK_TRACKING_INFORMATION data element, as specified in [MS-FSCC], section 2.3.23.

1. If the **TargetFileObject** field is not zero:

   - The file was moved by a remote machine, but from one volume on the local machine to another volume on the local machine.

   - This request MUST be processed by following the steps of section 3.1.6.1.

2. If the **TargetFileObject** field is zero:

   - The file was moved by a remote machine, from the local machine, to another machine.

   - An entry MUST be added to the **MoveTable** for the source file's ObjectID. The MachineID field of that entry MUST be the value from the NetBIOSName field of the TargetLinkTrackingInformationBuffer field in the request. For the FileLocation field of the new **MoveTable** entry, the VolumeID MUST be the value of the VolumeID field of the TargetLinkTrackingInformationBuffer field in the request, and the ObjectID MUST be the value of the ObjectID field of the TargetLinkTrackingInformationBuffer field in the request.

3.1.5   Timer Events

There are no timer events.

3.1.6   Other Local Events

When a file that has an ObjectID is moved to another volume, either on the same computer or on a different computer, an entry is added to the **MoveTable**. That entry indicates the file's ObjectID before it was moved, the MachineID of the computer to which the file was moved, and the file's new FileLocation value.

Specifically:

- If a file is moved between volumes within the local machine, the procedure defined in section 3.1.6.1 MUST be followed.

- If a file is moved from the local machine to a remote machine, the procedure defined in section 3.1.6.2 MUST be followed.

- If a file is moved from a remote machine to the local machine, the procedure defined in section 3.1.6.3 MUST be followed.

- If a file is moved by the local machine from one remote machine to another remote machine, the procedure in section 3.1.6.4 MUST be followed.

The remainder of this section describes these procedures. In these descriptions, a file move operation is modeled as a copy of a file from a source location to a new location, followed by the deletion of the source file.

The remainder of this section also refers to the following FSCTL structures, which are defined in [MS-FSCC]. If the connection to the remote machine is using the [MS-SMB] protocol, these requests MUST be sent for a file using the NT_TRANSACT_IOCTL subcommand of an SMB_COM_NT_TRANSACTION request, as specified in [MS-SMB] section 2.2.7.2.1. Otherwise, these requests MUST be sent in an SMB2_IOCTL request, as specified in [MS-SMB2] section 2.2.31.

- FSCTL_CREATE_OR_GET_OBJECT_ID ([MS-FSCC], section 2.3.1). This request returns a file's ObjectID, FileID, and CrossVolumeMoveFlag values.<13>

- FSCTL_GET_OBJECT_ID ([MS-FSCC], section 2.3.15). This request returns a file's ObjectID, FileID, and CrossVolumeMoveFlag values.<14>

- FSCTL_LMR_SET_LINK_TRACKING_INFORMATION ([MS-FSCC], section 2.3.23). This request notifies a server of a moved file.

- FSCTL_SET_OBJECT_ID_EXTENDED ([MS-FSCC], section 2.3.51). This request is used to set the FileID and CrossVolumeMoveFlag values on a file after it has been moved. Specifically, the EXTENDED_INFO element of this request MUST be set as follows:

  - **ObjectID**: This 16-byte field MUST be the value of the ObjectID component of the FileID that is to be set on a target file.

  - **C**: This single bit represents the CrossVolumeMoveFlag. If the flag is set this value MUST be 1. If the flag is cleared this value MUST be 0.

  - **VolumeID**: This 16-byte field (which starts with the C bit, described previously) MUST be the value of the VolumeID component of the FileID that is to be set on a target file.

  - **Unused**: The remaining 16 bytes of the element MUST be set to zero.

3.1.6.1   File Is Moved Between Volumes on Local Machine

If a file is moved between volumes on the local machine, the following updates to the abstract data model MUST occur:

- If the target volume does not already have a file with this ObjectID, the ObjectID MUST be set on the new file. Otherwise, a new, unique ObjectID MUST be created for the new file.

- The CrossVolumeMoveFlag MUST be set to 1 on the target file.

- The FileID of the source file MUST be set as the FileID of the target file.

- An entry MUST be added to the **MoveTable** (section 3.1.1) for the source file's ObjectID. The MachineID field of that entry MUST be the local machine's MachineID. The FileLocation field of that entry MUST be set to a value such that the VolumeID field is the target volume's VolumeID, and the ObjectID field is the target file's ObjectID.

3.1.6.2 File Is Moved from Local Machine to Remote Machine

If a file is moved from the local machine to a remote machine, the following procedure SHOULD be followed:

- The ObjectID of the target file SHOULD be retrieved by sending an FSCTL_CREATE_OR_GET_OBJECT_ID request.<15>

- If this step is successful, the VolumeID of the target file's volume MUST be retrieved by sending a request for an instance of the **FileFsObjectIdInformation** class, as specified in [MS-FSCC] section 2.5.6, and by interpreting the **ObjectId** field of the returned **FILE_FS_OBJECTID_INFORMATION** structure as the VolumeID. If the connection to the remote machine is using the [MS-SMB] protocol, this request MUST be sent by specifying **FileFsObjectIdInformation** as the **InformationLevel** of a **TRANS2_QUERY_FS_INFORMATION** request, specified in [MS-SMB] section 2.2.6.3.1. Otherwise, the connection to the remote machine is using the [MS-SMB2] protocol, and this request MUST be sent by specifying **FileFsObjectidInformation** as the **FileInfoClass** of an **SMB2_QUERY_INFO** request, specified in [MS-SMB2] section 2.2.37.<16>

- If the preceding step is successful, an entry MUST be added to the **MoveTable** (section 3.1.1) for the source file's ObjectID. The MachineID field of that entry MUST be the target machine's MachineID. The FileLocation field of that entry MUST be composed of the target volume's VolumeID and the target file's ObjectID.

- If the preceding step is successful, the FileID of the source file MUST be set on the target file by sending an FSCTL_SET_OBJECT_ID_EXTENDED request for the target file. In this request, the CrossVolumeMoveFlag MUST be set to 1, and the VolumeID and ObjectID fields MUST be those of the corresponding components of the source file's FileID.

3.1.6.3 File Is Moved from Remote Machine to Local Machine

If a file is moved from a remote machine to the local machine, the following procedure SHOULD<17> be followed:

- The FileID of the source file MUST be obtained by sending two FSCTL_GET_OBJECT_ID requests for the source file.<18> If either FSCTL_GET_OBJECT_ID request fails, no further processing SHOULD be performed. The source file is not considered to have an ObjectID, and as specified earlier, only files with ObjectIDs are processed.

- If this step is successful, the VolumeID of the target file's volume MUST be retrieved by sending a request for an instance of the **FileFsObjectIdInformation** class, as specified in [MS-FSCC] section 2.5.6, and by interpreting the **ObjectId** field of the returned **FILE_FS_OBJECTID_INFORMATION** structure as the VolumeID. If the connection to the remote machine is using the [MS-SMB] protocol, this request MUST be sent by specifying **FileFsObjectIdInformation** as the **InformationLevel** of a **TRANS2_QUERY_FS_INFORMATION** request, specified in [MS-SMB] section 2.2.6.3.1. Otherwise, the connection to the remote machine is using the [MS-SMB2] protocol, and this request MUST be sent by specifying **FileFsObjectidInformation** as the **FileInfoClass** of an **SMB2_QUERY_INFO** request, specified in [MS-SMB2] section 2.2.37.<19>

- If the preceding step is successful, the source machine MUST be notified of the move by sending an FSCTL_LMR_SET_LINK_TRACKING_INFORMATION request ([MS-FSCC] section 2.3.23). In this request, the **TargetFileObject** data element MUST be zero. In the TARGET_LINK_TRACKING_INFORMATION_Buffer data element ([MS-FSCC] section 2.3.23.3), the **Type** MUST be zero, the **VolumeID** field MUST be the VolumeID of the target file, the **ObjectID** field MUST be the ObjectID of the target file, and the **NetBIOSName** field MUST be the MachineID of the local machine. For the **NetBIOSName** field, if the MachineID ends with more than one zero byte, it MUST be truncated such that only one zero byte is included in the value.

- If that step is successful, the ObjectID and FileID of the source file MUST be deleted by sending an FSCTL_DELETE_OBJECT_ID request ([MS-FSCC] section 2.3.3) for the source file.

- If that step is successful, the target file MUST have its FileID set to the FileID of the source file, and the CrossVolumeMoveFlag MUST be set to 1.

3.1.6.4 File Is Moved by the Local Machine from Remote Machine to Remote Machine

This section specifies the following two procedures for moving a file by the local machine:

1. From one remote machine to another remote machine.

2. From one volume on a remote machine to another volume on the same remote machine.

Examples of these procedures are described in FSCTLs (section 4.2).

If the source MachineID and target MachineID are different, the following procedure SHOULD<20> be followed:<21>

- The FileID of the source file MUST be obtained by sending two FSCTL_GET_OBJECT_ID requests for the source file. If either request fails, no further processing is performed. The source file is not considered to have an ObjectID, and as specified earlier, only files with ObjectIDs are processed.

- If this step is successful, the VolumeID of the target file's volume MUST be retrieved by sending a request for an instance of the **FileFsObjectIdInformation** class, as specified in [MS-FSCC] section 2.5.6, and by interpreting the **ObjectId** field of the returned **FILE_FS_OBJECTID_INFORMATION** structure as the VolumeID. If the connection to the remote machine is using the [MS-SMB] protocol, this request MUST be sent by specifying **FileFsObjectIdInformation** as the **InformationLevel** of a **TRANS2_QUERY_FS_INFORMATION** request, specified in [MS-SMB] section 2.2.6.3.1. Otherwise, the connection to the remote machine is using the [MS-SMB2] protocol, and this request MUST be sent by specifying **FileFsObjectidInformation** as the **FileInfoClass** of an **SMB2_QUERY_INFO** request, specified in [MS-SMB2] section 2.2.37.<22>

- If that step is successful, the ObjectID and FileID of the target file MUST be determined by sending an FSCTL_CREATE_OR_GET_OBJECT_ID request for the target file.

- If that step is successful, the source machine MUST be notified of the move by sending an FSCTL_LMR_SET_LINK_TRACKING_INFORMATION request for the source file. In this request, the **TargetFileObject** data element MUST be set to zero. For the **TargetLinkTrackingInformation** data element, the format for a **TargetLinkTrackingInformationLength** greater than or equal to 36 MUST be used, the **Type** field MUST be set to zero, the **VolumeID** field MUST be set as the VolumeID of the target file, the **ObjectID** field MUST be set as the ObjectID of the target file, and the **NetBIOSName** field MUST be set as the MachineID of the target machine. For the

NetBIOSName field, if the MachineID ends with more than one zero byte, it MUST be truncated such that only one zero byte is included in the value.

- If that step is successful, an FSCTL_SET_OBJECT_ID_EXTENDED request MUST be sent for the target file with the FileID set to be that of the source file and with the CrossVolumeMoveFlag value set to 1.

If the source MachineID and target MachineID are the same, but the source and destination volumes are different, the following procedure SHOULD be followed:

- The FileID of the source file SHOULD be obtained by sending an FSCTL_GET_OBJECT_ID request for the source file.<23>

- If the preceding step is successful, the source machine MUST be notified of the move by sending an FSCTL_LMR_SET_LINK_TRACKING_INFORMATION request for the source file. In this request, the **TargetFileObject** data element MUST be set to be the **Fid** of the open target file if the connection to the remote machine is using the [MS-SMB] protocol (see [MS-FSCC] section 2.3.23.1), or set to zero otherwise (see [MS-FSCC] section 2.3.23.2)<24> (Note that in this latter case, because zero is passed as the **TargetFileObject**, the source machine will treat this notification as that of a move by a remote machine from the local machine to another machine, as described in section 3.1.4.2.)

  For the **TargetLinkTrackingInformation** data element, the format for a **TargetLinkTrackingInformationLength** less than 36 MUST be used (consequently, the **Type**, **VolumeId**, and **ObjectId** fields are not sent). The **NetBIOSName** field MUST be set to the MachineID of the remote machine. If that MachineID value ends with more than one zero byte, it MUST be truncated such that only one zero byte is included in the value. See section 3.1.4.2 for more information on this request.

3.2 DLT Workstation Client Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. This description is provided to explain 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 specified in this document.

A client of the Distributed Link Tracking: Workstation Protocol maintains information about a file it wants to track. This information consists of the following:

FileLinkInformation: Information about a file necessary to identify and locate it, including the file's last known UNC name, the MachineID of the computer on which the file was last known to be located, the last known FileLocation of the file, and the file's permanent FileID.

Maintaining this information allows the client to call the **LnkSearchMachine (section 3.1.4.1)** interface.

Note The preceding conceptual data can be implemented by using a variety of techniques. Any data structure that stores the preceding conceptual data may be used in the implementation.

3.2.2 Timers

No timers are associated with this protocol.

3.2.3 Initialization

The client initializes the RPC protocol as specified in section 2.1.

The client SHOULD<25> perform method calls by using the named pipe endpoint of "\\pipe\trkwks", and MAY call by using the named pipe endpoint of "\\pipe\ntsvcs", as noted in section 2.1.

3.2.4 Message Processing Events and Sequencing Rules

3.2.4.1 Completing a LnkSearchMachine Call

This section prescribes the actions that are necessary when completing a call to the **LnkSearchMachine** method.

When a client initiates a **LnkSearchMachine** call, the return value and output parameters are calculated as specified in section 3.1.4.1. If the return value indicates success (a value of zero or higher), the client has completed the operation, and a new UNC has been found. The client updates the FileLinkInformation that it maintains for the file:

- The UNC MUST be updated to the value returned in the *ptszPath* parameter of the call.

- The MachineID MUST be updated to be the MachineID of the server to which the call was made.

- The FileID MUST be updated to be that which is returned in the *pdroidLast* parameter of the call.

If an error value of 0xC0020017 is returned, indicating that the requested endpoint was not found, and the client had used the named pipe endpoint "\\pipe\trkwks", then the client MUST repeat the call by using the named pipe endpoint "\\pipe\ntsvcs".

If an error value of TRK_E_REFERRAL is returned, and the client has not already made a request for this file to the DLT Workstation server on the MachineID indicated in the *pmcidNext* parameter returned by the server in this call, then the client MAY<26> initiate another call to a DLT Workstation server on that computer. If it makes this call, it MUST call **LnkSearchMachine** with input parameters set as follows:

pdroidBirthLast: The same value as in the completed call.

pdroidLast: The value returned by the server in the **pdroidNext** field.

Restrictions: This parameter is unused and MUST be set to zero.

If the preceding algorithm does not produce a successful return value, the client MUST return an error indication to the upper layers that triggered the call. If the return value specifically is TRK_E_POTENTIAL_FILE_FOUND, the client MUST also return an indication of such, as well as the updated MachineID, FileLocation, FileID, and UNC, to the upper layers that triggered the call.<27>

3.2.5 Timer Events

There are no timer events.

3.2.6 Other Local Events

The DLT Workstation client calls the **LnkSearchMachine** method when it wants to find a file's new location. For example, if a file has been moved (if its UNC has changed), and a higher layer wants to open the file again, the DLT Workstation calls **LnkSearchMachine** in an attempt to determine the file's new path.

The client MUST call **LnkSearchMachine**, with input parameters set as follows:

pdroidBirthLast: The FileID from the FileLinkInformation maintained by the client.

pdroidLast: The value from the FileLinkInformation maintained by the client.

Restrictions: Unused and MUST be set to zero.

The client MUST send the **LnkSearchMachine** method call to the DLT Workstation server that runs on the computer identified by the MachineID of the FileLinkInformation.

4 Protocol Examples

4.1 LnkSearchMachine

The following example shows an example **LnkSearchMachine** call, similar to the example in section 1.3 with the IDs set as follows:

On machine M1:

- VolumeID: 8e7e9c15f59b4cf9952b03616aa51ebe

- ObjectID: 6479f083cfb245c29c713f586d6e038f

On machine M2:

- VolumeID: 20aaf9f7e0f0154f7681dd8a7a8872f5

- ObjectID: 73c7a25fbb1cdc1189ad00123f7ad5f3

In this example, the DLT Workstation client successfully contacts the named pipe "\\pipe\trkwks" with the following **LnkSearchMachine** call.

```
HRESULT
LnkSearchMachine (
    [in] unsigned long Restrictions = 0,
    [in] const CDomainRelativeObjId *pdroidBirthLast= {
        8e7e9c15f59b4cf9952b03616aa51ebe
        6479f083cfb245c29c713f586d6e038f },
    [in] const CDomainRelativeObjId *pdroidLast = {
        8e7e9c15f59b4cf9952b03616aa51ebe,
        6479f083cfb245c29c713f586d6e038f },
    [out] CDomainRelativeObjId *pdroidBirthNext= {
        irrelevant, filled in by server},
    [out] CDomainRelativeObjId *pdroidNext= {
        irrelevant, filled in by server},
    [out] CMachineId *pmcidNext = {
        irrelevant, filled in by server},
    [out, max_is(261), string] WCHAR* ptszPath = {
        irrelevant, filled in by server}
);
```

The server receives this call, verifies that the client is authorized to send it requests, consults its list of file objects, and finds that the file is indeed at that location, with a UNC of "\\M2\share2\F2.txt", and that the client is authorized to obtain its information.

The server responds with the following.

```
HRESULT = S_OK
LnkSearchMachine (
    [in] unsigned long Restrictions = {unmodified},
    [in] const CDomainRelativeObjId *pdroidBirthLast =
        {unmodified},
    [in] const CDomainRelativeObjId *pdroidLast= {unmodified],
    [out] CDomainRelativeObjId *pdroidBirthNext = {
```

```
            8e7e9c15f59b4cf9952b03616aa51ebe,
            6479f083cfb245c29c713f586d6e038f },
    [out] CDomainRelativeObjId *pdroidNext = {
            20aaf9f7e0f0154f7681dd8a7a8872f5,
            73c7a25fbb1cdc1189ad00123f7ad5f3 },
    [out] CMachineId *pmcidNext = "M2",
    [out, max_is(261), string] WCHAR* ptszPath =
            "\\M2\share2\F2.txt"
);
```

The client sees the successful completion and updates its FileLinkInformation state with the new location.

4.2 FSCTLs

The following is an example of an FSCTL_LMR_SET_LINK_TRACKING_INFORMATION request, which notifies a server of a moved file. This example demonstrates a file being moved from a remote machine to the local machine, as in section 3.1.6.3, or from a remote machine to a different remote machine, as in the first procedure in section 3.1.6.4.

```
TargetFileObject = 0

TargetLinkTrackingInformationLength = 32 // Decimal 50

Type = 0

VolumeId = 8e7e9c15f59b4cf9952b03616aa51ebe

ObjectId = 6479f083cfb245c29c713f586d6e038f

NetBIOSName = 54 65 73 74 4D 61 63 68 69 6E 65 0D 0A 00 // "TestMachine"
```

The following is a related example, also of an FSCTL_LMR_SET_LINK_TRACKING_INFORMATION request but for a file that is moved from a remote machine to the same remote machine, as in the second procedure in section 3.1.6.4, in the case where the connection to the remote machine is using the [MS-SMB] protocol.

```
TargetFileObject = 108

TargetLinkTrackingInformationLength = 0

Type = 0

VolumeId = 0

ObjectId = 0

NetBIOSName = 54 65 73 74 4D 61 63 68 69 6E 65 0D 0A 00 // "TestMachine"
```

The following is an example of an FSCTL_SET_OBJECT_ID_EXTENDED request, which sets the FileID to a file (this buffer is a VolumeID, followed by an ObjectID, followed by 16 bytes of zeros).

```
ExtendedInfo =
    8e7e9c15f59b4cf9952b03616aa51ebe \
    6479f083cfb245c29c713f586d6e038f \
    00000000000000000000000000000000
```

The following is an example of both an FSCTL_CREATE_OR_GET_OBJECT_ID reply and an FSCTL_GET_OBJECT_ID reply, which returns a file's ObjectID and FileID (the FileID is composed of the **BirthVolumeId** and **BirthObjectId** fields). In this example, the file has never been moved. As a consequence, the ObjectID and **BirthObjectId** fields are the same value. This also means that the **CrossVolumeMoveFlag**, stored as the low order bit of the first byte of the **BirthVolumeId** field, is zero.

```
ObjectId = 6479f083cfb245c29c713f586d6e038f

BirthVolumeId = 8e7e9c15f59b4cf9952b03616aa51ebe

BirthObjectId = 6479f083cfb245c29c713f586d6e038f

DomainId = 0
```

5 Security

5.1 Security Considerations for Implementers

There are no special security considerations for implementers to consider.

5.2 Index of Security Parameters

Security parameter	Section
Authenticated users group	3.1.4.1

6   Appendix A: Full IDL

For ease of implementation, the full **Interface Definition Language (IDL)** is provided as follows, where "ms-rpce.idl" is the IDL found in [MS-DTYP], Appendix A.

```
import "ms-dtyp.idl";

typedef signed long SequenceNumber;

typedef struct CObjId {
    GUID    _object;
} CObjId;

typedef struct CVolumeId {
    GUID _volume;
} CVolumeId;

typedef struct CMachineId {
    char _szMachine[ 16 ];
} CMachineId;

typedef struct CDomainRelativeObjId {
    CVolumeId _volume;
    CObjId _object;
} CDomainRelativeObjId;

[
    uuid(300f3532-38cc-11d0-a3f0-0020af6b0add),
    version(1.2),
    pointer_default(unique)
]

interface trkwks {

// Local only
void Opnum0NotUsedOnWire(void);

// Local only
void Opnum1NotUsedOnWire(void);

// Local only
void Opnum2NotUsedOnWire(void);

// Local only
void Opnum3NotUsedOnWire(void);

// Local only
void Opnum4NotUsedOnWire(void);

// Local only
void Opnum5NotUsedOnWire(void);

// Local only
void Opnum6NotUsedOnWire(void);

// Local only
void Opnum7NotUsedOnWire(void);
```

```
// Local only
void Opnum8NotUsedOnWire(void);

// Local only
void Opnum9NotUsedOnWire(void);

// Local only
void Opnum10NotUsedOnWire(void);

// Local only
void Opnum11NotUsedOnWire(void);

HRESULT LnkSearchMachine (
    [in] unsigned long Restrictions,
    [in] const CDomainRelativeObjId*  pdroidBirthLast,
    [in] const CDomainRelativeObjId*  pdroidLast,
    [out] CDomainRelativeObjId*        pdroidBirthNext,
    [out] CDomainRelativeObjId*        pdroidNext,
    [out] CMachineId*                  pmcidNext,
    [out, max_is(261), string] WCHAR* ptszPath
);
}
```

7   Appendix B: 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 Windows® 2000 operating system

- Windows® XP operating system

- Windows Server® 2003 operating system

- Windows Vista® operating system

- Windows Server® 2008 operating system

- Windows® 7 operating system

- Windows Server® 2008 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: All versions of Windows listed in the supported products list in Appendix B: Product Behavior: The user on M0 indicates an expectation to keep information about the file by creating a Shell shortcut for it. The Shell shortcut is a file stored on M0 that holds the UNC, MachineID, FileLocation, and FileID for F"1.txt".

<2> Section 1.3: All versions of Windows listed in the supported products list in Appendix B: Product Behavior: These calls are triggered when the user activates the Shell shortcut that was created to store information about the file "F1.txt". When the Shell shortcut is activated, it attempts to open the file. M1 returns an error indicating that the file does not exist, so the Shell shortcut implementation initiates the call to the DLT workstation on M1 to get the updated UNC. After storing the new FileLocation and UNC in the Shortcut file, the Shell shortcut implementation opens the destination file "F2.txt", and loads it into, for example, the Notepad program. None of this processing is visible to end users. End users simply activate the Shell shortcut by double-clicking it, and then view the contents of the file in Notepad.

<3> Section 2.1: Windows 2000: The DLT Workstation server listens only on the "\\pipe\ntsvcs" endpoint, and it makes calls only to that endpoint.

Windows XP and Windows Server 2003: The server listens on both the "\\pipe\ntsvcs" and "\\pipe\trkwks" endpoint, and they make calls to "\\pipe\trkwks" first. If such a call results in a failure return value of 0xC0020017, which indicates that the pipe name was not found, these implementations make calls to the "\\pipe\ntsvcs" endpoint instead.

Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2: The server listens only on the "\\pipe\trkwks" endpoint.

<4> Section 3.1.1: In a configuration where a DLT Central Manager server is available, the VolumeID is generated by using that protocol. Otherwise, an arbitrary VolumeID is generated.

<5> Section 3.1.1: All versions of Windows listed in the supported products list in Appendix B: Product Behavior: The CrossVolumeMoveFlag is stored as the low-order bit of the first byte of the **CVolumeID**, as specified in section 3.1.1.

<6> Section 3.1.1: Not all files on a volume have an ObjectID. Those files without an ObjectID are not considered part of this abstract data model.

<7> Section 3.1.3: Windows 2000: The DLT Workstation server listens only on the "\\pipe\ntsvcs"endpoint.

Windows XP and Windows Server 2003: The server also listens on the "\\pipe\trkwks"endpoint.

Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2: The server listens only on the "\\pipe\trkwks"endpoint.

<8> Section 3.1.4: Opnums reserved for local use apply to Windows as follows.

Opnum	Description
0-3	Not used by Windows.
4	Returns ERROR_NOT_IMPLEMENTED. It is never used.
5-8	Not used by Windows.
9	Only used locally by Windows, never remotely.
10-11	Not used by Windows.

<9> Section 3.1.4.1: All versions of Windows listed in the supported products list in Appendix B: Product Behavior: This field is a set of flags, but it is unused by the client. Servers alter their behavior in response to these flags, which are summarized as follows:

- 0x02: Prevents move notification information stored in the MoveTable from being used during a search request.

- 0x10: Prevents the service from searching any volume except the volume that is specified by the VolumeID in the request.

- 0x20: Prevents the search operation from giving preference to the volume specified by the VolumeID in the request, for the case where the requested file is located on multiple volumes. This flag also prevents the search operation from searching any machine other than the last known machine.

<10> Section 3.1.4.1: Windows 2000, Windows XP without any service packs, and Windows XP SP1 follow this behavior. In Windows XP SP2, Windows XP SP3, and Windows Server 2003, a failure value is returned if the path to be returned exceeds 130 characters. In Windows Vista and Windows 7, if the path to be returned exceeds 259 characters, a success code is returned, but the returned *ptszPath* argument is truncated to return only the machine and share names, not the file path.

<11> Section 3.1.4.1: If there are multiple valid UNC paths, the path to be returned is chosen as follows:

- A UNC path with read/write access privileges is preferred to a UNC path with only read access privileges.

- If two paths have the same access privileges, a UNC path with greater coverage of the volume is preferred to a UNC path with less coverage. For example, a UNC path of C:\DOCUMENTS has more coverage than a path of C:\DOCUMENTS\LETTERS.

- If two paths have the same access privileges and coverage, a visible path is preferred over a hidden path. A path is considered hidden if the share in the path ends in a '$' character, such as C$ or D$.

- If, after the preceding checks, no preference can be made between two paths, one is chosen arbitrarily.

<12> Section 3.1.4.1: All versions of Windows listed in the supported products list in Appendix B: Product Behavior: This situation occurs in a backup/restore sequence of operations. When a file is backed up, its ObjectID and FileID are stored with the backup data. When the file is restored:

- If the file is being restored over an existing file, and the file already has an ObjectID set, it is left unmodified.

- If another file on the volume to which the file is being restored already has the ObjectID, then no ObjectID is set on the restored file.

- Otherwise, the ObjectID is restored onto the file, but the FileID is not. This is because of the ambiguity of a restore operation, which could be a true restore operation for a corrupted disk, or could be a copy operation.

<13> Section 3.1.6: If a file does not already have an ObjectID and FileID, one is created for it.

<14> Section 3.1.6: If a file does not already have an ObjectID value and a FileID value, this request returns an error result.

<15> Section 3.1.6.2: All versions of Windows listed in the supported products list in Appendix B: Product Behavior follow this recommendation except when the local and target machines are both running Windows Vista or Windows 7. On those versions, the request is not sent, and consequently this procedure is not followed.

<16> Section 3.1.6.2: If the local and remote machines are both running Windows Vista, Windows Server 2008, Windows 7, or Windows Server 2008 R2, then the [MS-SMB2] protocol is used. Otherwise, the [MS-SMB] protocol is used.

<17> Section 3.1.6.3: As noted in section 3.1.1 a file's moves are not tracked if it does not have an ObjectID and FileID. To determine whether the source file is to be tracked, all Windows implementations send an FSCTL_GET_OBJECT_ID request ([MS-FSCC] section 2.3.15) for the source file. If that request succeeds, and the **BirthObjectId** field of the FILE_OBJECTID_BUFFER structure ([MS-FSCC] section 2.1.3) in the response is not all zeros, then the file is tracked as described in this section.

<18> Section 3.1.6.3: All versions of Windows listed in the supported products list in Appendix B: Product Behavior follow this recommendation except when the local and target machines are both running Windows Vista or Windows 7. On those versions, this request is not sent, and consequently this procedure is not followed.

<19> Section 3.1.6.3: If the local and remote machines are both running Windows Vista, Windows Server 2008, Windows 7, or Windows Server 2008 R2, then the [MS-SMB2] protocol is used. Otherwise, the [MS-SMB] protocol is used.

<20> Section 3.1.6.4: As noted in section 3.1.1 a file's moves are not tracked if it does not have an ObjectID and FileID. To determine whether the source file is to be tracked, all Windows implementations send a FSCTL_GET_OBJECT_ID request for the source file. If that request succeeds, and the **BirthObjectId** field of the FILE_OBJECTID_BUFFER structure ([MS-FSCC] section 2.1.3) in the response is not all zeros, then the file is tracked as described in this section.

<21> Section 3.1.6.4: All versions of Windows listed in the supported products list in Appendix B: Product Behavior follow this behavior, except if the local machine is running Windows Vista or Windows 7 and the source machine is not running Windows Server 2008 or Windows Server 2008 R2. In these cases, the following steps under "If the source MachineID and target MachineID are the same" are followed. Note that in those steps, the FSCTL_LMR_SET_LINK_TRACKING_INFORMATION request fails, because the Fid of the target file represents a file on the target machine, and the request is being sent to the source machine.

<22> Section 3.1.6.4: If the local and remote machines are both running Windows Vista, Windows Server 2008, Windows 7, or Windows Server 2008 R2, then the [MS-SMB2] protocol is used. Otherwise, the [MS-SMB] protocol is used.

<23> Section 3.1.6.4: All versions of Windows listed in the supported products list in Appendix B: Product Behavior follow this recommendation except when either of the following is true:

- The local and source machines are both running Windows Vista or Windows 7.

- The local machine is running Windows Vista without any service packs and the source machine is running Windows Server 2008 or Windows Server 2008 R2.

In these cases, this request is not sent, and consequently the following step is not followed.

<24> Section 3.1.6.4: If the local and remote machines are both running Windows Vista, Windows Server 2008, Windows 7, or Windows Server 2008 R2, then the [MS-SMB2] protocol is used. Otherwise, the [MS-SMB] protocol is used.

<25> Section 3.2.3: Windows 2000: The DLT Workstation client sends only on the "\\pipe\ntsvcs"endpoint.

Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2: The client initially sends on the "\\pipe\trkwks"endpoint.

<26> Section 3.2.4.1: All versions of Windows listed in the supported products list in Appendix B: Product Behavior: After receiving this error from a DLT Workstation server, a new SEARCH call is initiated to the DLT Central Manager, as specified in [MS-DLTM] section 3.2.6.3. If this SEARCH call fails with a TRK_E_NOT_FOUND error, the client uses the VolumeID component of the FileLocation returned with referral in a FIND_VOLUME request to the DLT Central Manager, as specified in [MS-DLTM] section 3.2.6.4. If both of these calls to the DLT Central Manager fail, the processing specified earlier in this section for TRK_E_REFERRAL is also followed. If this SEARCH call succeeds, the same processing is followed, only using the MachineID and FileLocation values returned from the DLT Central Manager call. If the SEARCH call returns a TRK_E_NOT_FOUND error, but the subsequent FIND_VOLUME succeeds, the same processing is followed, only using the FileLocation from the referral and the MachineID returned from the FIND_VOLUME request. In each of these cases, if another TRK_E_REFERRAL return value is subsequently received, the preceding processing specified for TRK_E_REFERRAL is followed; no new call is made to the DLT Central Manager.

<27> Section 3.2.4.1: All versions of Windows listed in the supported products list in Appendix B: Product Behavior: This typically causes the user to be prompted with a dialog box. The dialog box states that the file could not be found, but a new file was found that might be correct. The user is prompted either to use that file or to abort the action.

8   Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

9 Index