[MS-BKUP]:
Microsoft NT Backup File Structure

Revision Summary

Date	Revision History	Revision Class	Comments
03/02/2007	1.0	Major	Updated and revised the technical content.
04/03/2007	1.1	Minor	Updated the technical content.
05/11/2007	1.2	Minor	Clarifications; Minor edits
06/01/2007	1.3	Minor	Updated the technical content.
07/03/2007	1.3.1	Editorial	Revised and edited the technical content.
08/10/2007	1.3.2	Editorial	Revised and edited the technical content.
09/28/2007	1.3.3	Editorial	Revised and edited the technical content.
10/23/2007	1.3.4	Editorial	Revised and edited the technical content.
01/25/2008	1.3.5	Editorial	Revised and edited the technical content.
03/14/2008	1.3.6	Editorial	Revised and edited the technical content.
06/20/2008	2.0	Major	Updated and revised the technical content.
07/25/2008	2.0.1	Editorial	Revised and edited the technical content.
08/29/2008	2.0.2	Editorial	Revised and edited the technical content.
10/24/2008	2.0.3	Editorial	Revised and edited the technical content.
12/05/2008	3.0	Major	Updated and revised the technical content.
01/16/2009	3.0.1	Editorial	Revised and edited the technical content.
02/27/2009	3.0.2	Editorial	Revised and edited the technical content.
04/10/2009	3.0.3	Editorial	Revised and edited the technical content.
05/22/2009	4.0	Major	Updated and revised the technical content.
07/02/2009	4.0.1	Editorial	Revised and edited the technical content.
08/14/2009	4.0.2	Editorial	Revised and edited the technical content.
09/25/2009	4.1	Minor	Updated the technical content.
11/06/2009	4.1.1	Editorial	Revised and edited the technical content.
12/18/2009	5.0	Major	Updated and revised the technical content.
01/29/2010	5.0.1	Editorial	Revised and edited the technical content.
03/12/2010	5.0.2	Editorial	Revised and edited the technical content.

Date	Revision History	Revision Class	Comments
04/23/2010	5.0.3	Editorial	Revised and edited the technical content.
06/04/2010	5.0.4	Editorial	Revised and edited the technical content.
07/16/2010	5.1	Minor	Clarified the meaning of the technical content.
08/27/2010	5.1	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	5.1	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	5.1	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	5.1	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	5.1	No change	No changes to the meaning, language, or formatting of the technical content.
03/25/2011	5.1	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	5.1	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	5.2	Minor	Clarified the meaning of the technical content.
09/23/2011	5.2	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	5.2	No change	No changes to the meaning, language, or formatting of the technical content.
03/30/2012	5.2	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	5.2	No change	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	5.2	No change	No changes to the meaning, language, or formatting of the technical content.
01/31/2013	5.2	No change	No changes to the meaning, language, or formatting of the technical content.
08/08/2013	6.0	Major	Significantly changed the technical content.

Contents

1 Introduction

This specification describes the network format of the Windows **NT backup file** format and its constituent structures that may be used in other protocols.

Sections 1.7 and 2 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. All other sections and examples in this specification are informative.

1.1 Glossary

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

access control list (ACL)
ASCII
backup stream
FAT file system
FAT32 file system
file stream
main stream
named stream
NTFS
NT backup file
Object ID
reparse point
security descriptor
serialize
sparse file
staging file
unnamed stream

The following terms are specific to this document:

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

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the 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".

[MS-FRS1] Microsoft Corporation, "File Replication Service Protocol".

[MS-FRS2] Microsoft Corporation, "Distributed File System Replication Protocol".

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

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

1.2.2 Informative References

[FFS] McKusick, M. K., Joy, W. N., Leffler, S. J., et al., "A Fast File System for UNIX", Computer Systems 2(3):181-197, 1984.

[MS-DLTCS] Microsoft Corporation, "Distributed Link Tracking Central Store Protocol".

[MS-DLTW] Microsoft Corporation, "Distributed Link Tracking: Workstation Protocol".

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

[MS-WSO] Microsoft Corporation, "Windows System Overview". (Archived)

[MSFT-NTFS] Microsoft Corporation, "NTFS Technical Reference", March 2003, http://technet2.microsoft.com/WindowsServer/en/Library/81cc8a8a-bd32-4786-a849-03245d68d8e41033.mspx

1.3 Overview

This document specifies the structure of NT backup files as they are used in over-the-wire protocols. This file format is not a protocol; however, it is used to describe the format of data that is sent across the wire as payloads of other protocols, in particular, the File Replication Service Protocol (as specified in [MS-FRS1]) and the SD Microsoft Distributed File System Replication Protocol, as specified in [MS-FRS2]. As such, the format is specified in this document as a reference that other protocols can use to ensure consistency and accuracy. Contained in this specification are the following:

- An overview about the structure of an NT backup file.
- The definition of the WIN32_STREAM_ID (section 2.2) backup stream header.
- The definition of a **backup stream** that may be found within an NT backup file.

While a simple model of a file is a stream of bytes, files have become more complicated over the years. They may contain large regions of zero data, more than one stream of bytes, and special attributes such as **reparse points**, as well as having **access control lists (ACLs)** attached to them. In some instances, such as making a backup copy of a file or transmitting a file over a network, it is helpful to **serialize** the file—that is, to efficiently express a file's full complexity as a simple stream of bytes. The NT Backup format is a particular format for this serialization.

As an example, consider **sparse files**. Many modern file systems, including NT File System, **NTFS**, (as specified in [MSFT-NTFS]) and many implementations based on Berkeley Fast File System support sparse files (for more information about Berkeley Fast File System, see [FFS]). In these file systems, unwritten portions of files have zero data, but they do not necessarily result in the allocation of disk space or the writing of zeros to the disk. When serializing these files, it is more efficient not to store the zero ranges in the serialize representation. Furthermore, in the case of NTFS, whether a range is allocated or unallocated is observable by the application via the FSCTL_QUERY_ALLOCATED_RANGES file system control (as specified in [MS-FSCC]), so replacing unallocated ranges with simple zeros alters the semantics of the file. The NT Backup format enables such a file to be transmitted over a network or a representation of it to be stored on a file system or backup medium that lacks the richness of the original file system.

As another example, consider **named streams**. NTFS supports a feature wherein a file may have a number of streams with associated names, in addition to the default (unnamed) **main stream**. For example, the main stream of a file named a.txt might contain the bytes **unnamed stream**, while named stream stream1, opened as a.txt:stream1, would contain "This is stream1." The NT Backup format enables serialization of any number of streams.

The content of an NT Backup serialize file is a set of backup streams. (The different uses of the word "stream" are specified in section 2.2.) Each backup stream represents one aspect of the original file, such as its ACL, a contiguous allocated section of a **file stream**, a reparse point, and so on.

1.4 Relationship to Protocols and Other Structures

The File Replication Service Protocol (as specified in [MS-FRS1]) and the Distributed File System Replication Protocol (as specified in [MS-FRS2]) rely on the structures and definitions in this document to create and interpret the contents of a **staging file** that are sent and received across the network during file replication.

1.5 Applicability Statement

The structures and classes that this document defines are useful for any lower-level protocol, such as the File Replication Service Protocol (as specified in [MS-FRS1]), that serializes and exchanges native Windows file formats, but does not require that those file formats be remapped into a protocol-specific representation.

1.6 Versioning and Localization

None.

1.7 Vendor-Extensible Fields

None.

2 Structures

The following sections specify the Microsoft NT Backup File Structure.

Unless otherwise specified, all numeric fields that this document specifies are little-endian.

Note that the word "stream" is used in two different contexts. It can indicate:

- A part of the NT backup file format.
- A place in a file where data is stored (or the actual data therein is stored, depending on the context).

To avoid confusion between these two usages, this document uses backup stream to indicate the portion of the NT backup file; file stream, main stream, named stream, unnamed stream, and **alternate stream** indicate the parts of a file in the file system.

The Windows NT backup file format and its constituent structures reference commonly used data types as defined in [MS-DTYP].

2.1 NT Backup File

An NT backup file is made up of zero or more backup streams that appear one right after the other. A backup stream is a logically related collection of data that is related to one file. For example, the file's contents are a backup stream; its security information is a backup stream, and so on. Each backup stream in an NT backup file consists of a **WIN32_STREAM_ID (section 2.2)** header, followed by the file-specific data for that backup stream.

An implementation that creates a backup stream for an NT backup file for use over-the-wire MUST use only the **dwStreamId** values that are specified in section 2.2 when creating a backup stream. An implementation reading an NT backup file, and then creating a file based on it, MUST fail if its **dwStreamId** value is not one of those specified in section 2.2.

If an unrecognized or unused **dwStreamId** value or an otherwise malformed NT backup file is encountered, an implementation MAY process the remainder of the NT backup file. An implementation MAY delete the portion of the file it has created upon finding a malformed NT backup file.<1>

2.2 WIN32_STREAM_ID

The **WIN32_STREAM_ID** structure is a header that precedes each backup stream in the NT backup file. This header identifies the type of backup stream, its size, and other attributes. The structure is as follows.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
dwStreamId																															
dwStreamAttributes																															
Size																															

...
dwStreamNameSize
cStreamName (variable)
...

dwStreamId (4 bytes): A 32-bit, unsigned integer that indicates the type of data in this backup stream. The value of this field MUST be one of the following.

Value	Meaning
ALTERNATE_DATA 0x00000004	Alternative data streams.
DATA 0x00000001	Standard data.
EA_DATA 0x00000002	Extended attribute data.
LINK 0x00000005	Hard link information.
OBJECT_ID 0x00000007	Object identifiers.
REPARSE_DATA 0x00000008	Reparse points.
SECURITY_DATA 0x00000003	Security descriptor data.
SPARSE_BLOCK 0x00000009	Data in a sparse file.
TXFS_DATA 0x0000000A	Transactional file system.

dwStreamAttributes (4 bytes): A 32-bit, unsigned long integer that indicates properties of the backup stream. The value of this field MUST be the bitwise OR of zero or more of the following. Other bits are unused and MUST be 0 and ignored on receipt.

Value	Meaning
STREAM_NORMAL_ATTRIBUTE 0x00000000	This backup stream has no special attributes.
STREAM_CONTAINS_SECURITY 0x00000002	The backup stream contains security information. This attribute applies only to backup stream of type SECURITY_DATA.
STREAM_SPARSE_ATTRIBUTE 0x00000008	The backup stream is part of a sparse file stream. This attribute applies only to backup stream of type DATA, ALTERNATE_DATA,

Value	Meaning
	and SPARSE_BLOCK.

Size (8 bytes): A 64-bit, unsigned integer that specifies the length of the data portion of the backup stream; this length MUST NOT include the length of the header. The next backup stream within the NT backup file, if any, MUST start at **Size** + **dwStreamNameSize** bytes beyond the end of this **WIN32_STREAM_ID** structure. Note that the alternate stream name, whose size is **dwStreamNameSize** bytes, is part of the header for the purposes of calculating the position of the next **WIN32_STREAM_ID** structure.

dwStreamNameSize (4 bytes): A 32-bit, unsigned integer that specifies the length of the alternate stream name, in bytes. The value of this field MUST be 0 for all **dwStreamId** values other than ALTERNATE_DATA. For StreamID ALTERNATE_DATA, the value of this field MUST be in the range 0–65536, and it MUST be an integral multiple of two.

cStreamName (variable): A Unicode string that specifies the name of the alternate stream. This string MUST NOT be null-terminated.

Size bytes of data MUST follow the header. The meaning of the data depends on the **dwStreamId** value and is specified in the following sections.

2.3 Alternate Data Backup Stream Structure

Some file systems support files that have named streams. The ALTERNATE_DATA **dwStreamId** field value MUST be used when recording these streams. Aside from the requirement that ALTERNATE_DATA MUST have a nonzero length name in the **cStreamName** field of the WIN32_STREAM_ID (section 2.2) structure, an ALTERNATE_DATA stream is identical to a DATA (section 2.4) stream. An underlying file system might constrain the naming of alternate streams. If the implementation does not support named streams or does not support the **cStreamName** that is specified in the ALTERNATE_DATA structure, the implementation MAY ignore the stream or MAY choose some other way to store the data.<2>

2.4 Data Backup Stream Structure

The data portion of a data backup stream structure is the data within the main stream of the file. When creating a backup file, the bytes of the main stream MUST be copied without modification to the data portion of a data backup stream.

2.5 Extended Attribute Data Backup Stream Structure

An implementation SHOULD NOT create an extended attribute data backup stream in an NT Backup format file and MUST ignore an extended attribute data backup stream, if received.<3>

2.6 Link Backup Stream Structure

An implementation SHOULD NOT create a LINK backup stream and MUST ignore one if encountered.

2.7 Object ID Backup Stream Structure

The Object ID Backup Stream contains a unique identifier for a file. The structure of the data portion of this backup stream is as follows:

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
ObjectID																															
...																															
...																															
...																															
Data																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															
(Data cont'd for 4 rows)																															

ObjectID (16 bytes): A 16-byte GUID, as specified in [MS-DTYP] section 2.3.4.2, assigned by the server on which the file is stored, that uniquely identifies the file or directory within the volume in which it is stored.

Data (48 bytes): This field contains 48 bytes of implementation-defined metadata associated with the file.<4>

2.8 Reparse Backup Stream Structure

A reparse backup stream contains information about a reparse point in the file system. The data portion of a reparse point backup stream is the contents of the reparse point, which MUST be a REPARSE_DATA_BUFFER or REPARSE_GUID_DATA_BUFFER structure, as specified in [MS-FSCC] sections 2.1.2.2 and 2.1.2.3, respectively.

2.9 Security Stream Structure

The data portion of a SECURITY_DATA backup stream MUST contain a SECURITY_DESCRIPTOR. For more information, see [MS-WSO] section 3.1.2.3.2.

2.10 Sparse Block Stream Structure

A sparse file (or a sparse named stream) is represented both by a DATA backup stream (or an ALTERNATE_DATA backup stream in the case of a named stream) followed by one or more SPARSE_BLOCK backup streams.

A SPARSE_BLOCK backup stream represents a portion of a sparse file that contains data; unallocated portions of a file are indicated by the absence of a SPARSE_BLOCK stream that describes that part of the file. The structure of the data portion of this backup stream is as follows:

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Offset																															
...																															
Data (variable)																															
...																															

Offset (8 bytes): An unsigned, 64-bit integer that specifies the offset within the file stream (not in the backup stream) of the data contained in this sparse block.

Data (variable): The data for this allocated region of the file stream. This field MUST be of length (Size - 8), where **Size** is the value of the WIN32_STREAM_ID header's **Size** field that is associated with this backup stream.

An implementation SHOULD use file system sparse file support, if available, to represent the reconstituted file. Failure to do so could waste large amounts of disk space.<5>

2.11 TXFS Stream Structure

A TXFS backup stream has no meaning on a system other than the system on which it was created. An implementation MUST NOT send a TXFS stream and MUST ignore a TXFS stream, if received.

2.12 Structure Usage

This specification describes only the format of individual structures.

This section specifies the process of creating an NT backup file and of reconstituting an ordinary file from an NT backup file.

2.12.1 Creating an NT Backup File

This section specifies a process for creating an NT backup file corresponding to a given file F.

The contents of the NT backup file MUST be a number of backup streams, with their formats specified as in section 2.2, with no padding or other bytes between them. Each backup stream MUST consist of a WIN32_STREAM_ID followed by the appropriate backup stream data.

The order of the backup streams within an NT backup file is arbitrary, except that SPARSE_BLOCK backup streams MUST follow the DATA or ALTERNATE_DATA backup streams for the file stream that contains the data in the SPARSE_BLOCK backup stream.<6>

The code creating the NT backup file SHOULD generate at most one backup stream of each of the following types: DATA, OBJECT_ID, REPARSE_DATA, and SECURITY_DATA.

If the file system supports it, the code creating the NT backup file SHOULD generate a SECURITY_DATA backup stream that contains the **security descriptor** for F.

If the file system supports it and F has an **object ID**, the code creating the NT backup file SHOULD generate an **Object ID Backup Stream** containing it.

If the file system supports it and F has a reparse point, then the code creating the NT backup file SHOULD generate a REPARSE_DATA backup stream containing it.

If the main stream of F is not empty, the code that creates the NT backup file MUST generate a DATA backup stream for it. The DATA backup stream SHOULD have STREAM_SPARSE_ATTRIBUTE set if, and only if, the main stream of F is sparse. If the main stream of F is sparse, the DATA backup stream SHOULD have a **Size** of 0. If the main stream of F is not sparse, the DATA backup stream MUST contain the contents of the main stream of F.

If the main stream of F is sparse, it has any nonzero data in it, and that data is not stored in the DATA stream, the code that creates the NT backup file MUST generate one or more SPARSE_BLOCK backup streams. The set of SPARSE_BLOCKs together with the DATA backup stream MUST contain all of the nonzero data of the file. The SPARSE_BLOCKs for the DATA backup stream MUST follow the DATA backup stream and MUST NOT follow any ALTERNATE_DATA backup streams.

If F has any named streams, the code that creates the NT backup file MUST generate one ALTERNATE_DATA backup stream for each named stream in F. The ALTERNATE_DATA stream MUST have its **cStreamName** equal to that of the corresponding named stream of F. If the named stream is sparse, it MUST be treated identically to a sparse unnamed stream, except that the appropriate SPARSE_BLOCKs MUST follow the ALTERNATE_DATA backup stream for this named stream and MUST precede the next ALTERNATE_DATA backup stream (if any) in the NT backup file.

2.12.2 Reconstituting a File from an NT Backup File

An implementation creating a file F given an NT backup file MUST process all of the backup streams contained within the NT backup file.

For OBJECT_ID, REPARSE_DATA, and SECURITY_DATA backup streams, it SHOULD create the corresponding object ID, reparse point, or security descriptor on F, if the file system in which F resides supports those features. If there is more than one backup stream of any of these particular types (or of type DATA), the implementation creating F can either:

- Select any one of the backup streams of that type, or <7>
- Fail without reconstituting that stream due to the presence of more than one backup stream of that type

If the NT backup file contains a DATA backup stream, the code that creates F MUST put the data from the DATA backup stream into the main stream of F. If any SPARSE_BLOCK backup streams occur in the NT backup file after the DATA backup stream and before any ALTERNATE_DATA backup stream, the code that creates F MUST put the data that is contained in the SPARSE_BLOCK at the specified place in F's main stream. If there are SPARSE_BLOCKs and the file system storing F has support for sparse files, it SHOULD use the sparse file support to avoid allocating disk space for the portions of the main stream that are not described by the DATA or SPARSE_BLOCK backup stream.

If the NT backup file contains one or more ALTERNATE_DATA backup streams and the file system holding F supports ALTERNATE_DATA streams, the code that creates F SHOULD generate the appropriate named stream, using as the name the contents of the **cStreamName** field in the

WIN32_STREAM_ID header. Processing of ALTERNATE_DATA streams MUST otherwise be identical to that of DATA streams, including the rules for SPARSE_BLOCKs.

3 Structure Examples

This section presents an example of serializing an input file that has two streams. The unnamed stream contains "unnamed stream" in **ASCII**, and the alternate stream "stream1" contains "This is stream 1". This section does not show a sparse region because the NTFS file system requires sparse regions to be integral multiples of 64-KB bytes aligned on 64-KB byte boundaries, which would result in too large of an example.

This section displays dumps of file contents. These are formatted with 16 bytes per line, each byte shown in hexadecimal without the leading "0x" prefix. Each line starts with the offset, in hexadecimal from the beginning of the file of the first byte in the line. The end of each line is the bytes in the line displayed in ASCII, with unprintable characters replaced with a period (.).

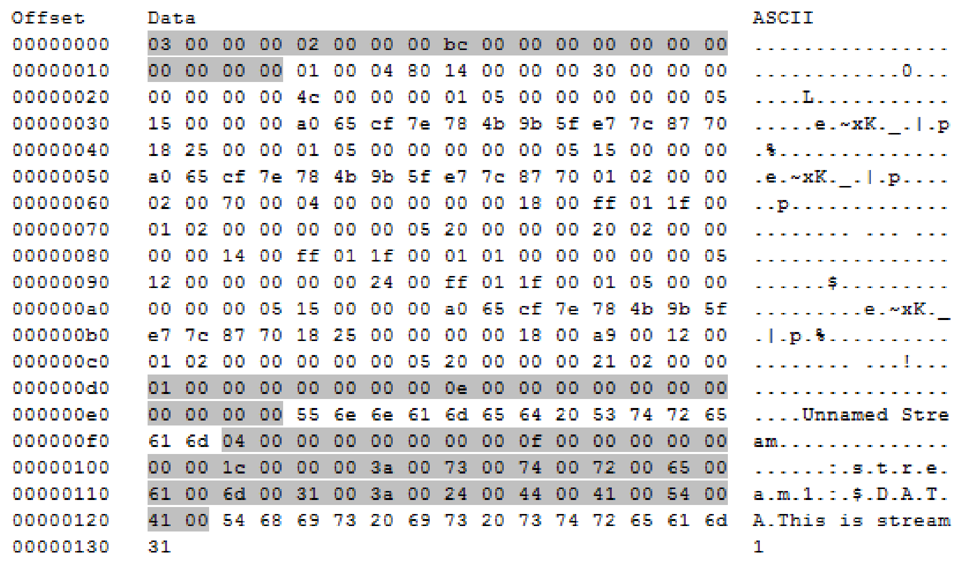
The raw contents of a.txt is as follows:

```
Offset      Data                                               ASCII
00000000    55 6e 6e 61 6d 65 64 20 53 74 72 65 61 6d          Unnamed Stream
```

The contents of a.txt:stream1 is:

```
Offset      Data                                               ASCII
00000000    54 68 69 73 20 69 73 20 73 74 72 65 61 6d 31       This is stream1
```

The serialized representation of a.txt is:



```
Offset      Data                                               ASCII
00000000    03 00 00 00 02 00 00 00 bc 00 00 00 00 00 00 00    ................
00000010    00 00 00 00 01 00 04 80 14 00 00 00 30 00 00 00    ............0...
00000020    00 00 00 00 4c 00 00 00 01 05 00 00 00 00 00 05    ....L...........
00000030    15 00 00 00 a0 65 cf 7e 78 4b 9b 5f e7 7c 87 70    .....e.~xK._.|.p
00000040    18 25 00 00 01 05 00 00 00 00 00 05 15 00 00 00    .%..............
00000050    a0 65 cf 7e 78 4b 9b 5f e7 7c 87 70 01 02 00 00    .e.~xK._.|.p....
00000060    02 00 70 00 04 00 00 00 00 00 18 00 ff 01 1f 00    ..p.............
00000070    01 02 00 00 00 00 00 05 20 00 00 00 20 02 00 00    ........ ... ...
00000080    00 00 14 00 ff 01 1f 00 01 01 00 00 00 00 00 05    ................
00000090    12 00 00 00 00 00 24 00 ff 01 1f 00 01 05 00 00    ......$.........
000000a0    00 00 00 05 15 00 00 00 a0 65 cf 7e 78 4b 9b 5f    .........e.~xK._
000000b0    e7 7c 87 70 18 25 00 00 00 00 18 00 a9 00 12 00    .|.p.%..........
000000c0    01 02 00 00 00 00 00 05 20 00 00 00 21 02 00 00    ........ ...!...
000000d0    01 00 00 00 00 00 00 00 0e 00 00 00 00 00 00 00    ................
000000e0    00 00 00 00 55 6e 6e 61 6d 65 64 20 53 74 72 65    ....Unnamed Stre
000000f0    61 6d 04 00 00 00 00 00 00 00 0f 00 00 00 00 00    am..............
00000100    00 00 1c 00 00 00 3a 00 73 00 74 00 72 00 65 00    ......:.s.t.r.e.
00000110    61 00 6d 00 31 00 3a 00 24 00 44 00 41 00 54 00    a.m.1.:.$.D.A.T.
00000120    41 00 54 68 69 73 20 69 73 20 73 74 72 65 61 6d    A.This is stream
00000130    31                                                 1
```

The **WIN32_STREAM_IDs** at the beginning of each backup stream are indicated by a gray background. There are three backup streams in this case: first, a SECURITY_DATA stream that contains the security descriptor for this file; second, the DATA stream that contains the contents of the main stream; and third, an ALTERNATE_DATA stream that contains the contents of stream1. The NTFS file system reports alternate stream names as ":streamName:$DATA." This is an implementation detail of the NTFS file system and does imply that other file systems necessarily follow this stream naming convention.

4 Security Considerations

4.1 Security Considerations for Implementers

Allowing the use of all possible NT Backup file stream types when this serialization format is used by a file transfer protocol could unexpectedly grant access to a wider range of functionality than the protocol author intended, such as creation of reparse points and assignment of security descriptors. Protocols that use these structures in a generic format should protect themselves appropriately by blocking stream types that their implementers do not want to support or stream types that they do not recognize. The latter is significant if the underlying file systems on the sending or receiving sides might be upgraded to support new functionality that was not there when the protocol was initially implemented.

Implementers of the NT Backup format should carefully design their protocols so that attackers may not modify the contents of the backup streams, because such modifications can lead to arbitrary changes in the restored file, including—but not limited to—changes in the contents, access control lists, or reparse information of the file.

It is also important to protect the NT Backup serialized files themselves from unauthorized modification, for the same reasons.

4.2 Index of Security Parameters

None.

5 Appendix A: Product Behavior

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

- Windows 2000 operating system
- Windows Server 2003 operating system
- Windows Server 2008 operating system
- Windows 7 operating system
- Windows Server 2008 R2 operating system
- Windows 8 operating system
- Windows Server 2012 operating system
- 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 2.1: Windows stops processing an NT backup file when it detects a malformation, but it does not delete any partially restored file information.

<2> Section 2.3: The NTFS file system supports named streams. All other Microsoft file systems such as the **FAT file system** and the **FAT32 file system** do not support named streams. The NTFS file system requires that the name of each ALTERNATE_DATA stream in an NT backup file begin with the colon character (:). NTFS appends ":$DATA" to the name of any alternate stream that does not already end that way, so it handles a.txt:AlternateStream and a.txt:AlternateStream:$DATA identically. When restoring an NT Backup format file onto the FAT file system, Windows ignores named streams.

<3> Section 2.5: The extended attribute stream exists only in files created by Windows versions Windows NT 3.1, Windows NT 3.5 and Windows NT 3.51.

<4> Section 2.7: This field contains Distributed Link Tracking information for the file, if any, if the file is stored in an NTFS file system. The first 16 bytes contain the file's birth volume ID, which is the volume on which the object resided when the object ID was created, or 0 if the volume had no object identifier at that time. The next 16 bytes contain the object ID at the time the object was created, which MUST be unique within the volume on which it was created. The next 16 bytes contain the domain ID, which is set to 0. The FILE_OBJECTID_BUFFER structure is specified in [MS-FSCC] section 2.1.3. For more information about the Distributed Link Tracking Service, see [MS-DLTCS] and [MS-DLTW].

<5> Section 2.10: Windows uses sparse file support when writing to the NTFS file system but not to others, such as the FAT file system and the FAT32 file system.

<6> Section 2.12.1: When creating an NT backup file for use with the File Replication Service Protocol, as specified in [MS-FRS1], Windows requires that the first backup stream in the NT backup file be of type BACKUP_SECURITY_DATA.

<7> Section 2.12.2: Windows always selects the last instance of the blocks in question for installation into F.

6 Change Tracking

This section identifies changes that were made to the [MS-BKUP] protocol document between the January 2013 and August 2013 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	**Tracking number (if applicable) and description**	**Major change (Y or N)**	**Change type**
5 Appendix A: Product Behavior	Modified this section to include references to Windows 8.1 operating system and Windows Server 2012 R2 operating system.	Y	Content updated.

7 Index