[MS-HRL-Preview]:

Hyper-V Replica Log (HRL) File Format

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **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 can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation 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 might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are 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 as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does 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 documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Preliminary Documentation. This particular Open Specifications document provides documentation for past and current releases and/or for the pre-release version of this technology. This document provides final documentation for past and current releases and preliminary documentation, as applicable and specifically noted in this document, for the pre-release version. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. Because this documentation might change between the pre-release version and the final version of this technology, there are risks in relying on this preliminary documentation. To the extent

that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Comments
6/13/2016	Released Preview Document.

Table of Contents

1 Introduction

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

This specification defines the Hyper-V Replica Log (HRL) File Format, which provides a persistent backing store for files that track changes that have been made to the primary server. These files, called log files, record each write request; each entry provides information about the address range that is modified and new data in that range. Log files are written sequentially with the newest record appended to the end of the log file.

Sections 1.7 and 2 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

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.

[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

None.

1.3 Overview

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

This document covers the format of HRL file that a creator application needs to adhere to for the file to be parsed by Hyper-V. The HRL traversing algorithm is also covered in the latter section.

1.4 Relationship to Protocols and Other Structures

None.

1.5 Applicability Statement

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

This file format provides a persistent backing store for file changes that need to be tracked and have been made to the primary server.

1.6 Versioning and Localization

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

The version of the HRL File Format is determined by the value of the **LogFormatVersion** field in the header, as defined in section 2.2.

HRL Version	Value
Log Format Version 1	0x00010000

1.7 Vendor-Extensible Fields

None.

2 Structures

2.1 Log File Format

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

The following figure is a simplified representation of the log file. The log file header contains identification information, stores the size of the metadata field, and stores the location of the last valid log data to indicate the end of the log.

Figure 1: Log file structure

Each metadata item consists of a metadata header and metadata entries. The first entry of the metadata header stores the previous metadata location in the log file. This is used in traversing the metadata structures from the bottom of the log.

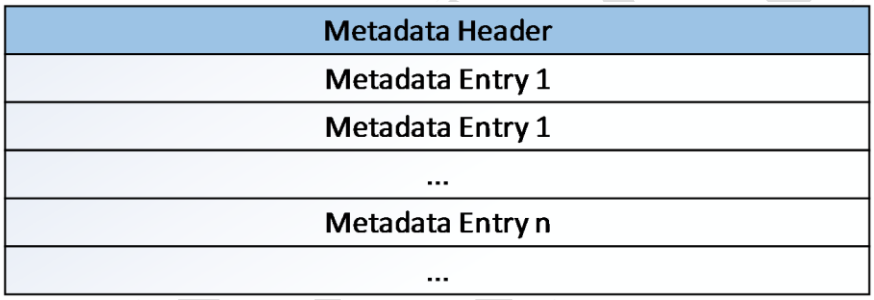

Figure 2: Metadata structure

The log file is optimized for writing sequentially. Therefore, the metadata describing each log entry is written after the data entries.

2.2 Log File Header Format

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

The log file header stores information about the log.

```
struct _CTLOG_HEADER_PACKED{
    UCHAR Cookie [8];
    ULONG LogFormatVersion;
    ULONG TimeStamp;
    UCHAR CreatorApplication[4];
```

```
    ULONG CreatorVersion;
    ULONG64 OriginalSize;
    ULONG64 CurrentSize;
    ULONG Checksum;
    ULONG64 EOLLocation;
    ULONG MetadataSize;
    UCHAR UniqueId[16];
    UCHAR PreviousUniqueId[16];
    ULONG FileType;
    UCHAR Flags[2];
    UCHAR Vhd2DataWriteGuid[16]
    UCHAR Reserved[3970];
}
```

Cookie: This field is used to uniquely identify the original creator of the log file. The values are case-sensitive.

This field MUST be set to "msctlog" to identify this file as a log file. The cookie is stored as an eight-character ASCII string with the "m" in the first byte, the "s" in the second byte, and so on.

LogFormatVersion: This field MUST be initialized to 0x00010000. It is divided into a major/minor version and matches the version of the specification used in creating the file. The most-significant two bytes are for the major version. The least-significant two bytes are the minor version.

TimeStamp: This field stores the creation time of the log file. Its value is the number of seconds since January 1, 2000, 12:00:00 AM in UTC/GMT.

CreatorApplication: This field is a left-justified text field used to identify which application created the log file. It uses a single-byte character set. If the log file is created by Failover Replication, "ct" is written in this field.

CreatorVersion: This field holds the major/minor version of the application that created the hard disk image. For Failover Replication, this value is set to winver. This is not verified as part of the file verification exercise.

OriginalSize: This field stores the size of the log file, in bytes, at the time it was created.

CurrentSize: This field stores the current size of the log file, in bytes.

This value is the same as the original size when the log file is created. This value can change depending on whether the log file is expanded.

Checksum: This field holds a checksum of the log file header. Its value is a one's complement of the sum of all the bytes in the header not including the checksum field.

If the checksum verification fails, then the log file is assumed to be corrupt.

EOLLocation: This field indicates the offset of the end of the log file. This field is reset to zero upon opening the file and set to the correct value on closing the file. This field can be used to ascertain whether the file was closed properly. For example, an EOL value of zero in the header indicates that the file did not close properly.

MetadataSize: This field indicates the size of the metadata used in the log file. Typical metadata sizes are multiples of 512 bytes. The default is 4,096 bytes.

UniqueId: This field is a unique ID that identifies the log file. It is a 128-bit universally unique identifier (UUID).

PreviousUniqueId: This field indicates the unique ID of the previous log file. It can be used to build a log file chain if needed. This field is a 128-bit universally unique identifier (UUID).

FileType: This field identifies the type of the log file. For an HRL file, this field's value is set to 0.

Flags: This field is not used and MUST be set to 0.

Vhd2DataWriteGuid: This field stores the GUID of VHD2. This helps in detecting offline Patch detection i.e. modifying the content of VHD when change tracking is not enabled.

Reserved: This field is reserved and MUST be set to 0. It is 3,970 bytes in size.

2.3 Metadata Header Format

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

Each metadata block has a small header to indicate the valid number of log entries in that metadata.

```
struct _CTLOG_METADATA_HEADER_PACKED{
    ULONG64 PreviousMetadataLocation;
    ULONG ValidMetadataEntries;
    ULONG Checksum
    UCHAR Reserved[16];
}
```

PreviousMetadataLocation: This field contains the relative location of the previous metadata in the log file. It is used when reading the metadata blocks quickly while replaying the log on the recovery.

ValidMetadataEntries: This field contains the count of valid metadata entries in the metadata. It is used to indicate the last metadata entry which might not be fully occupied.

Checksum: Contains the checksum of the header.

Reserved: This field MUST be set to 0. It is 16 bytes in size.

2.4 Log Metadata Entry Format

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

The following is the format of the metadata entry.

```
struct _CTLOG_METADATA_ENTRY_PACKED{
    ULONG64 ByteOffset;
    ULONG Checksum
    ULONG DataLength;
    ULONG TimeStamp;
    BYTE  MetaOperation
    ULONG DataChecksum
    UCHAR Location
    UCHAR Reserved[6];
}
```

ByteOffset: This field contains the byte offset on the logical disk where the data has to be written.

Checksum: This field contains the checksum of the metadata entry.

DataLength: This field indicates the length of the data to be written on the disk.

TimeStamp: This field stores the time of the writing of this particular log entry. This is the number of seconds since January 1, 2000 12:00:00 AM in UTC/GMT.

MetaOperation: This field contains the meta-operation identifier.

DataChecksum: This field contains the checksum of the data associated with this metadata entry.

Location: This field contains the tracing related information.

Reserved: This field MUST be set to 0. It is 6 bytes in size.

2.5 Log Traversing Algorithm

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

The log file is optimized for writing sequentially. Therefore, the metadata describing each log entry is written after the data entries. For the log file's data to be read, a two-pass traversal of the log needs to be performed. The first pass retrieves metadata locations. In the second pass, which is more comprehensive, each log entry pointed to by the metadata is read for applying to the recovery.

The following steps are used to traverse a log file:

1. Initialize a stack for storing metadata location offsets.
2. Read the log file header and retrieve the EOL location from the log file header. Also read the metadata size.
3. Using the value of the EOL location and the metadata size, calculate the location of the last metadata. This is equal to (EOL location – metadata size).
4. Call it the current metadata and push its value on to the stack.
5. Traverse to the location of the current metadata and read the first field of the metadata header from this location. This field points to the location of the previous metadata in the log file.
6. If this location is nonzero, then go to step 4.
7. At the end of the first pass, the stack contains the offsets of all the metadata structures in the log in the correct order.
8. Pop the value at the top of the stack. This is the location of the first metadata.
9. Traverse to this location and read the metadata structure.
10. Each entry of the metadata provides the details of a data field in the log that can be read from the log and applied to the recovery.
11. Each metadata entry provides the length of the data written in the log. Since data is written sequentially, the start of the data field will immediately follow the end of the last data field, the end of the log header, or the previous metadata header.
12. Read all the data entries pointed to by the metadata structure, and then go to step 8. Repeat steps 8-11 until the stack is empty.

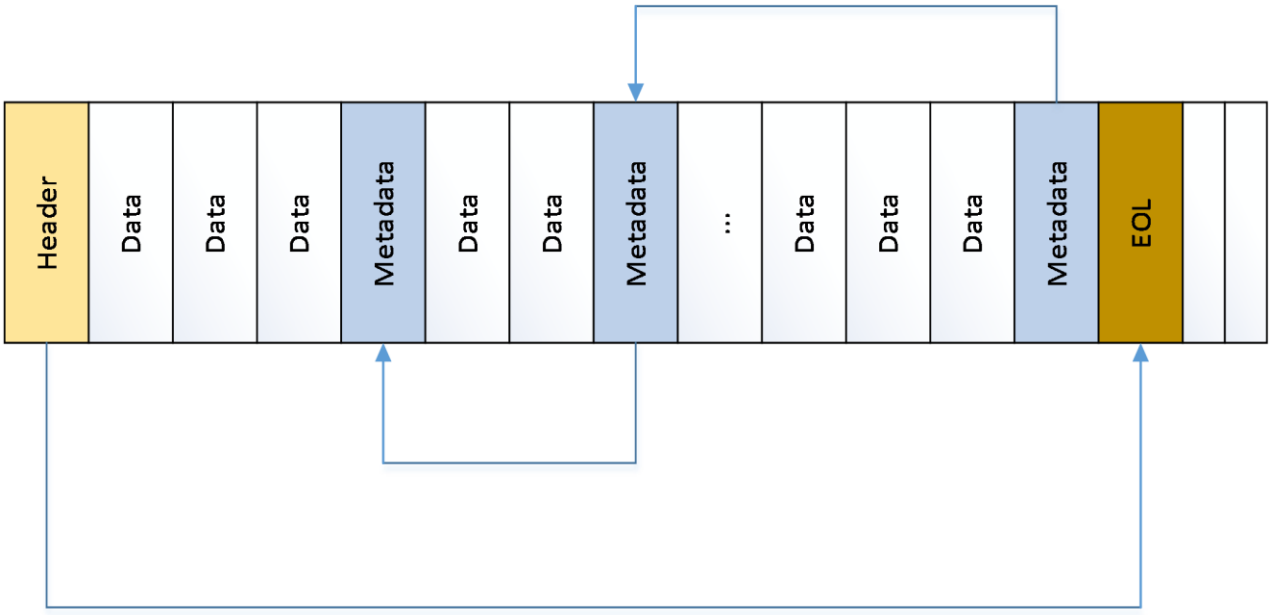

Figure 3: Log traversing algorithm

3 Structure Examples

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

The following are examples of a Log File Header and the first Metadata Header of the HRL file.

Log File Header

```
Cookie = "msctlog "
LogFormatVersion = 0x0010000
TimeStamp 516739283
CreatorApplication = "ct  "
CreatorVersion = 0x60003
OriginalSize = 0
CurrentSize = 99971072
Checksum = 4294959984
EOLLocation = 99971072
Errorcode = 0
MetadataSize = 4096
UniqueId = {15b98874-27d2-4a98-9a22-3f6f49c468a8}
PreviousUniqueId = {b3548aff-c3b7-4d27-bd6e-ca8a3cb80e5a}
LastModifiedTimeStamp 516739530
TotalMetadataEntries = 2768
 File Type = 0
 Flags = 0,0
 File Type = 0
 Flags = 0,0
```

Metadata Header (#1)(Offset:99966976)

```
PreviousMetadataLocation = 98274816
UsedMetadataEntries = 101
Actual Checksum = 4294966828
```

4 Security

4.1 Security Considerations for Implementers

None.

4.2 Index of Security Fields

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.

Note: Some of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

- Windows Server 2012 operating system
- Windows Server 2012 R2 operating system
- Windows Server 2016 Technical Preview 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.