[MS-CER]:
Corporate Error Reporting Version 1.0 Protocol

Revision Summary

Date	Revision History	Revision Class	Comments
05/11/2007	0.1		MCPP Milestone 4 Initial Availability
08/10/2007	0.2	Minor	Updated the technical content.
09/28/2007	0.3	Minor	Updated the technical content.
10/23/2007	0.4	Minor	Updated the technical content.
11/30/2007	0.5	Minor	Updated the technical content.
01/25/2008	0.5.1	Editorial	Revised and edited the technical content.
03/14/2008	0.5.2	Editorial	Revised and edited the technical content.
04/25/2008	1.0	Major	Updated and revised the technical content.
05/16/2008	1.0.1	Editorial	Revised and edited the technical content.
06/20/2008	2.0	Major	Updated and revised the technical content.
07/25/2008	2.1	Minor	Updated the technical content.
08/29/2008	2.2	Minor	Updated the technical content.
10/24/2008	2.2.1	Editorial	Revised and edited the technical content.
12/05/2008	2.2.2	Editorial	Revised and edited the technical content.
01/16/2009	2.2.3	Editorial	Revised and edited the technical content.
02/27/2009	2.2.4	Editorial	Revised and edited the technical content.
04/10/2009	2.2.5	Editorial	Revised and edited the technical content.
05/22/2009	2.2.6	Editorial	Revised and edited the technical content.
07/02/2009	2.2.7	Editorial	Revised and edited the technical content.
08/14/2009	2.2.8	Editorial	Revised and edited the technical content.
09/25/2009	2.3	Minor	Updated the technical content.
11/06/2009	2.3.1	Editorial	Revised and edited the technical content.
12/18/2009	2.3.2	Editorial	Revised and edited the technical content.
01/29/2010	2.3.3	Editorial	Revised and edited the technical content.
03/12/2010	2.4	Minor	Updated the technical content.
04/23/2010	2.4.1	Editorial	Revised and edited the technical content.

Date	Revision History	Revision Class	Comments
06/04/2010	3.0	Major	Updated and revised the technical content.
07/16/2010	3.0	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	3.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	3.0	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	3.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	3.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	3.0	No change	No changes to the meaning, language, or formatting of the technical content.
03/25/2011	3.0	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	3.0	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	3.1	Minor	Clarified the meaning of the technical content.
09/23/2011	3.1	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	3.1	No change	No changes to the meaning, language, or formatting of the technical content.
03/30/2012	3.1	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	3.1	No change	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	3.1	No change	No changes to the meaning, language, or formatting of the technical content.
01/31/2013	3.1	No change	No changes to the meaning, language, or formatting of the technical content.
08/08/2013	3.1	No change	No changes to the meaning, language, or formatting of the technical content.
11/14/2013	3.1	No change	No changes to the meaning, language, or formatting of the technical content.
02/13/2014	3.1	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This document specifies the Corporate Error Reporting Version 1.0 Protocol. This protocol is designed to enable businesses to manage all **error reporting** information within the organization. Through use of this protocol, problem reports generated on a set of client machines can be directed to a local or remote **CER file share** for analysis.

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

1.1 Glossary

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

American National Standards Institute (ANSI) character set
ASCII
Augmented Backus-Naur Form (ABNF)
Universal Naming Convention (UNC)

The following terms are specific to this document:

bucket: A positive integer value that represents a mapping for a particular **error signature**.

CER client: A client configured to use the Corporate Error Reporting Version 1.0 Protocol.

CER file share: A designated folder that acts as the drop location for the **error reports** copied by the Corporate Error Reporting Version 1.0 Protocol.

error report: Information contained in a set of files that describes a problem event that has occurred on the system. The report is typically compressed into a single file for transmission.

error signature: An ordered collection of strings which represents an individual error or class of errors.

error subpath: A fragment of a Server Message Block (SMB) Protocol file server directory path which is composed from the strings in the **error signature** and is used to direct **error reports** on the **CER file share**. For more details, see section 2.2.3.

tracking: A **CER client** feature that adds information to logging files during the error reporting process.

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.

[ISO/IEC-8859-1] International Organization for Standardization, "Information Technology -- 8-Bit Single-Byte Coded Graphic Character Sets -- Part 1: Latin Alphabet No. 1", ISO/IEC 8859-1, 1998, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=28245

Note There is a charge to download the specification.

[LAVY-MEGGITT] Lavy, M. and Meggitt, A., "Windows Management Instrumentation (WMI)", Sams, 2001, ISBN: 1578702607.

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

[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

[RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, http://www.ietf.org/rfc/rfc3986.txt

[RFC4234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, http://www.ietf.org/rfc/rfc4234.txt

1.2.2 Informative References

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

[MSDN-CAB] Microsoft Corporation, "Microsoft Cabinet Format", http://msdn.microsoft.com/en-us/library/bb417343.aspx

[MSDN-FILE] Microsoft Corporation, "Naming a File", http://msdn.microsoft.com/en-us/library/aa365247.aspx

1.3 Overview

The Corporate Error Reporting Version 1.0 Protocol provides an organization with the ability to copy error reports from a set of client machines to a CER file share on a specified Server Message Block (SMB) Protocol file server with additional configuration options.

An error event, such as an application or kernel fault, causes the client system to collect information for an error report. The Corporate Error Reporting Version 1.0 Protocol does not create the original contents of the error report.

The **CER client** then performs a check to determine if a path to a CER file share has been specified for this client system. If the path has been specified, the Corporate Error Reporting Version 1.0 Protocol will be used.

If the Corporate Error Reporting Version 1.0 Protocol will be used, the CER client constructs several CER file share paths based on the specific error that has occurred, and it constructs one CER file share path that applies to all reports for that CER file share. The CER client then attempts to read

configuration files at that location. These configuration files, if present, indicate CER client behavior settings, whether the error report information should be copied to the CER file share, and additional data requests to be included in the error report.

If the report is to be copied, the CER client gathers the additional data requested, then compresses the report information into a single file. Then the CER client copies the error reporting file to the specified CER file share where it can be accessed by an administrator for **tracking** or analysis purposes.

Whether or not an error reporting file was copied, the CER client also updates the configuration file that indicates the number of reports for that particular error and, depending on configuration, may also update tracking files.

The server is simply an SMB Protocol file server with a specific set of files. All behavior specific to the Corporate Error Reporting Version 1.0 Protocol is in the CER client and helper processes which access the file server, but the CER client and these helper processes never interact directly.

1.4 Relationship to Other Protocols

The Corporate Error Reporting Version 1.0 Protocol uses the SMB Protocol to copy error reports from the client machine to the CER file share.

1.5 Prerequisites/Preconditions

1. The client system is able to create error reports.
2. An implementation-specific file compression algorithm is required to organize the error reporting information into one file.
3. The Corporate Error Reporting Version 1.0 Protocol assumes that the client is configured with the file path of the CER file share on the server.
4. The Corporate Error Reporting Version 1.0 Protocol assumes that the client has permission to copy files to the CER file share.
5. It is assumed that corporate administrators have the ability to interpret the files that the Corporate Error Reporting Version 1.0 Protocol copies to the CER file share on the SMB Protocol server.
6. The CER file share can have the following file: <fileshare>\policy.txt.
7. An external process, either manual or automated, can generate status.txt files at the appropriate paths described in section 2.2.3.

1.6 Applicability Statement

The Corporate Error Reporting Version 1.0 Protocol is appropriate for small, medium, or large organizations that want to manage and review all error reporting information within the organization.

In addition, the Corporate Error Reporting Version 1.0 Protocol is only applicable to environments where all client machines support a common (local) file path syntax, although it is applicable to any such file path syntax.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The Corporate Error Reporting Version 1.0 Protocol MUST use Server Message Block, as specified in [MS-SMB], to read and write files on the specified CER file share.

2.2 Message Syntax

The Corporate Error Reporting Version 1.0 Protocol transmits messages in the form of files.

2.2.1 Count.txt

The Count.txt file includes two text parameters that track aggregate information about how frequently certain problems occur on specific clients and how much report data has been collected. Its format is a CRLF-delimited list of name-value pairs. It MUST be encoded using the **ANSI character set** as specified in [ISO/IEC-8859-1], with an implementation-specific valid code page. It MUST conform to the following syntax, as specified in [RFC4234]:

```
Countfile =  Cabs Hits
Cabs      = %d67.97.98.115.32.71.97.116.104.101.114.101.100.61 (1DIGIT / ((%x31-39)
1*DIGIT)) CRLF ; the encoded characters spell case-sensitive "Cabs Gathered="
Hits      = %d84.111.116.97.108.32.72.105.116.115.61 (%x31-39)*DIGIT CRLF
; the encoded characters spell case-sensitive "Total Hits="
```

Cabs: The number of error reporting files that have been copied for this problem. This MUST be a positive integer or zero.

Hits: The total number of hits this problem has received. This MUST be a positive integer, and MUST NOT be zero.

2.2.2 Tracking Files

There are two tracking files with very similar formats. The hits.log file is specific to a given error signature, while a single crash.log file is used for all errors reported to the CER file share. Each of their formats is a CRLF-delimited list of errors where each error is represented by a list of tab-delimited items. They MUST be encoded using the ANSI character set. They share many aspects of their grammar, as specified in [RFC4234]:

```
Time     = Hours ":" Minutes ":" Seconds
Date     = Month "-" Day "-" Year
Hours    = 2DIGIT    ;    00-23
Minutes  = 2DIGIT    ;    00-59
Seconds  = 2DIGIT    ;    00-59
Month    = 2DIGIT    ;    01-12
Day      = 2DIGIT    ;    01-28, 01-29, 01-30, 01-31 based on Month/Year
Year     = 4DIGIT
Machine = (1*15CHAR) / "UNKNOWN"    ;    see below for delimiter handling
User = (1*256CHAR) / "unknown user" ;    see below for delimiter handling
```

Time: The local machine time when the report was generated (HH:MM:SS).

Date: The local machine date when the report was generated (MM-DD-YYYY).

Machine: The non-fully-qualified machine name of the machine that generated the error report. A CRLF pair MUST NOT appear in **Machine**, since that is reserved as the line delimiter. An HTAB MUST NOT appear in **Machine**, since that is reserved as the item delimiter. If the system is unable to determine the name of the machine, then "UNKNOWN" MUST be used in its place.

User: The user name of the user who was logged on when the report was generated. If the system is unable to determine the name of that user, then "unknown user" MUST be used in its place. A CRLF pair MUST NOT appear in **User**, since that is reserved as the line delimiter. An HTAB MUST NOT appear in **User**, since that is reserved as the item delimiter.

2.2.2.1 Hits.log

The hits.log file MUST conform to the following syntax, as specified in [RFC4234].

```
HitsLog  = HitEntry / (HitEntry HitsLog)
HitEntry = Time SP SP Date HTAB Machine HTAB User HTAB FileName CRLF
FileName = 1*260CHAR  / "No CAB"    ;    see below for delimiter handling
```

Time, **Date**, **Machine**, **User**: These elements MUST conform to section 2.2.2.

FileName: The name of the error reporting file. A CRLF pair MUST NOT appear in the **FileName**, since that is reserved as the line delimiter. An HTAB MUST NOT appear in the **FileName**, since that is reserved as the item delimiter. If an error reporting file was not written, the string "No CAB" MUST be used in its place.

2.2.2.2 Crash.log

The crash.log file MUST conform to the following syntax, as specified in [RFC4234].

```
CrashLog = CrashEntry / (CrashEntry CrashLog)
CrashEntry = Time SP SP Date HTAB Machine HTAB User HTAB ErrorInfo CRLF
ErrorInfo = BucketID / ErrorSubPath
BucketID    = (%x31-39)*DIGIT
ErrorSubPath = 1*CHAR     ;    see below for delimiter handling
```

Time, **Date**, **Machine**, **User**: These elements MUST conform to section 2.2.2.

ErrorInfo: The CER client MUST write the **BucketID** to the crash.log file if it found it in the status.txt file (section 2.2.5) for the error in question; otherwise it MUST write the **error subpath**.

BucketID: This MUST be a positive decimal integer, and MUST NOT be zero.

ErrorSubPath: This is the error subpath (directory structure fragment) for the reported error (as described in section 2.2.3). A CRLF pair MUST NOT appear in the **ErrorSubPath**, since that is reserved as the line delimiter. An HTAB MUST NOT appear in the **ErrorSubPath**, since that is reserved as the item delimiter.

2.2.3 CER File Share Folder Structure

The Corporate Error Reporting Version 1.0 Protocol specifies two globally unique file path locations per CER file share, the root-level location of policy.txt (section 2.2.4) and the root-level location of crash.log (section 2.2.2.2).

Every type of error report also uses individual count.txt (section 2.2.1), hits.log (section 2.2.2.1) and status.txt (section 2.2.5) files. Each specific instance of an error also has a single error report file, whose name is generated by the CER client. In the examples below, the term "<error reporting file>" is substituted for the actual file name. Every error report looks for the count.txt, hits.log (if the tracking feature is enabled), and status.txt files, and writes its error reporting file, using a standard scheme. The scheme uses the concept of the error subpath SMB file server directory path fragment, whose specifics will be discussed under each kind of report.

The terms below in angle brackets ("<" and ">") are placeholders, not literals. The actual values MUST be made up solely of **ASCII** characters. They MUST NOT contain prohibited characters in the filesystem environment in which they operate, and the directory names MUST NOT consist of prohibited names in the file system environment in which they operate. <1>

Once the CER client generates the **error signature** specific file names (having replaced the placeholders in the error subpath with the proper parameters for the particular error report), it MUST compute the length of these filenames. If any of them are longer than 260 characters, the report MUST be discarded by the CER client.

If the directory paths required for these files do not exist on the CER file share, the CER client MUST create them.

The form of the path names MUST be as follows, where the <UNC file share path> represents the path to the CER file share.

Global files

```
<UNC file share path>\policy.txt
<UNC file share path>\crash.log
```

Error signature-specific files

```
<UNC file share path>\cabs\<error subpath>\<error reporting file>
<UNC file share path>\cabs\<error subpath>\hits.log
<UNC file share path>\status\<error subpath>\status.txt
<UNC file share path>\counts\<error subpath>\count.txt
```

2.2.3.1 Application Fault or Hang Reports

For user-mode error reports, the CER client MUST obtain the following information from the system to create this error signature.

Parameter	Description
AppName	The file name of the faulting application binary. This MUST be between 1 and 64 characters long.
AppVer	The file version of the faulting application binary obtained from the file descriptor. This

Parameter	Description
	MUST be between 1 and 24 characters long.
ModName	The file name of the faulting module binary. This MUST be between 1 and 64 characters long.
ModVer	The file version of the faulting module binary obtained from the file descriptor. This MUST be between 1 and 24 characters long.
Offset	The code location in the faulting binary where the exception occurred (in hexadecimal without a leading 0x identifier). The offset value MUST be the full length specified by the architecture of the faulting process; for example, a valid 32-bit offset is "0000abcd", and a valid 64-bit offset is "0000abcd12345678".

The error subpath for these types of reports MUST be composed using the error signature as follows: "<AppName>\<AppVer>\<ModName>\<ModVer>\<Offset>".

The specific file paths used in making this type of report MUST be as follows:

- <UNC file share path>\cabs\<AppName>\<AppVer>\<ModName>\<ModVer>\<Offset>\<error reporting file>
- <UNC file share path>\cabs\<AppName>\<AppVer>\<ModName>\<ModVer>\<Offset>\hits.log
- <UNC file share path>\status\<AppName>\<AppVer>\<ModName>\<ModVer>\<Offset>\status.txt
- <UNC file share path>\counts\<AppName>\<AppVer>\<ModName>\<ModVer>\<Offset>\count.txt

2.2.3.2 Specialized Reporting Types

For other event types, the folder structure is dependent on the type of error event described in the report.

2.2.3.2.1 Kernel Fault Reports

For kernel fault reports, "blue" MUST be used as the error signature and error subpath. The specific file paths used in making this type of report MUST be as follows.

- <UNC file share path>\cabs\blue\<error reporting file>
- <UNC file share path>\cabs\blue\hits.log
- <UNC file share path>\status\blue\status.txt
- <UNC file share path>\counts\blue\count.txt

2.2.3.2.2 Shutdown Reports

For unplanned shutdown reports, "shutdown" MUST be used as the error signature and error subpath. The specific file paths used in making this type of report MUST be as follows.

- <UNC file share path>\cabs\shutdown\<error reporting file>
- <UNC file share path>\cabs\shutdown\hits.log

- <UNC file share path>\status\shutdown\status.txt
- <UNC file share path>\counts\shutdown\count.txt

2.2.4 Policy.txt

The Corporate Error Reporting Version 1.0 Protocol allows for a set of configuration parameters that describe which tracking or response options are preferred. These parameters are set in one of two text files located in specific locations on the CER file share, and instruct the CER client exactly what information to include in the error report. The configuration parameters MUST be contained in a policy.txt and status.txt file as specified in this section and section 2.2.5.

If present, policy.txt MUST be placed in the root folder of the CER file share. Its format is a CRLF-delimited list of name-value pairs. It MUST be encoded using the ANSI character set. The file MUST conform to the following **Augmented Backus-Naur Form (ABNF)**, as specified in [RFC4234]. Note that the terms in the "Policyrule" rule can appear in any order and all permutations are not illustrated in the ABNF for brevity and clarity.

```
Policyrule              = [Tracking] [CrashesPerBucket] [URLLaunch]
                          [NoSecondLevelCollection] [NoFileCollection]
                          [NoExternalURL] [FileTreeRoot]
                          ; terms may appear in any order
CERBooleanValue         = "YES" / "TRUE" / "1" / "NO" / "FALSE" / "0"
                          ; case insensitive
Tracking                = %d84.114.97.99.107.105.110.103.61 CERBooleanValue CRLF
                          ; the encoded characters spell case-sensitive "Tracking="
CrashesPerBucket        = %d67.114.97.115.104.101.115.32.112.101.114.32.98.117.99.1
                          07.101.116.61 (1DIGIT / ((%x31-39)1*DIGIT)) CRLF
                          ; the encoded characters spell case-sensitive "Crashes
                          per bucket="
URLLaunch               = %d85.82.76.76.97.117.110.99.104.61 Url CRLF
                          ; the encoded characters spell case-sensitive "URLLaunch="
Url                     = URI ; [RFC 3986], see below for delimiter handling
NoSecondLevelCollection = %d78.111.83.101.99.111.110.100.76.101.118.101.108.67.111.
                          108.108.101.99.116.105.111.110.61 CERBooleanValue CRLF
                          ; the encoded characters spell case-sensitive "NoSecond
                          LevelCollection="
NoFileCollection        = %d78.111.70.105.108.101.67.111.108.108.101.99.116.105.111.
                          110.61 CERBooleanValue CRLF
                          ; the encoded characters spell case-sensitive "NoFile
                          Collection="
NoExternalURL           = %d78.111.69.120.116.101.114.110.97.108.85.82.76.61
                          CERBooleanValue CRLF
                          ; the encoded characters spell case-sensitive "NoExternalURL="
FileTreeRoot            = %d70.105.108.101.84.114.101.101.82.111.111.116.61 Path CRLF
                          ; the encoded characters spell case-sensitive "FileTreeRoot="

Path                    = 1*260CHAR    ;    see below for delimiter handling
```

CERBooleanValue: The values "YES", "TRUE", and "1" represent True. The values "NO", "FALSE", and "0" represent False.

Tracking: A True **CERBooleanValue** instructs the CER client that it MUST enable internal tracking in the form of a hits.log (section 2.2.2.1) file and a crash.log (section 2.2.2.2) file. A False **CERBooleanValue** instructs the CER client that it MUST disable internal tracking. If this parameter is absent, the CER client SHOULD use a default value of "NO". <2>

Crashes per bucket: This MUST be a positive integer value or zero. This value indicates the maximum number of error reporting files to collect for each error. If this many error reporting files already exist for this error on the CER file share, then an additional error reporting file MUST NOT be submitted. If this parameter is absent, the CER client SHOULD use a default value of "5".

URLLaunch: This parameter informs the CER client that the specified URL references information to show the user.

Url: It MUST conform to the URL syntax, as specified in [RFC3986]. A CRLF pair MUST NOT appear in the **Url**, since that is reserved as the line delimiter.

NoSecondLevelCollection: A True CERBooleanValue instructs the CER client to ignore additional data requests specified in status.txt (section 2.2.5). A False CERBooleanValue instructs the CER client to honor additional data requests. If this parameter is absent, the CER client SHOULD use a default value of "NO".<3>

NoFileCollection: A True CERBooleanValue instructs the CER client to ignore file requests specified in status.txt (section 2.2.5). A False CERBooleanValue instructs the CER client to honor file requests. If this parameter is absent, the CER client SHOULD use a default value of "NO". File requests are the subset of data requests that gather files from the user's computer; they are specified in section 2.2.5 by the **fDoc** and **GetFile** rules.<4>

NoExternalURL: A True CERBooleanValue instructs the CER client to ignore URL information described in status.txt. A False CERBooleanValue instructs the CER client to honor URL information. If this parameter is absent, the CER client SHOULD use a default value of "NO".<5>

FileTreeRoot: The value MUST be a **Universal Naming Convention (UNC)** path. This parameter instructs the CER client to redirect the error reporting process to use this file tree location instead of the file tree location at which it found this file, starting from looking for a policy.txt file at that new file tree location. This new file tree location MUST meet the prerequisites/preconditions (section 1.5) and file tree paths (section 2.2.3) for the destination server. If this parameter is not indicated, the CER client MUST continue the error reporting process using the file tree root at which it found this file. To avoid an infinite loop, the CER client SHOULD keep track of how many times it has been redirected using this method, and if it is redirected more than 10 times, it SHOULD abort the error reporting process.

Path: A CRLF pair MUST NOT appear in the **Path** value, since that is reserved as the line delimiter.

2.2.5 Status.txt

If present, status.txt MUST be placed in a final subfolder of the "status" branch of the CER file share (For an example, see section 4.1). The presence of a status.txt file in the final subfolder for a specific error instructs the CER client to take on different behavior for that error. The status.txt file supports any of the parameters that can be specified in policy.txt, with the exception of FileTreeRoot. If there is a conflict, a status.txt parameter MUST override a corresponding policy.txt parameter.

The status.txt file format is a CRLF-delimited list of name-value pairs. It MUST be encoded using the ANSI character set. The file MUST conform to the following ABNF, as specified in [RFC4234]. Note that the terms in the "StatusRule" rule can appear in any order and all permutations are not illustrated in the ABNF for brevity and clarity.

```
StatusRule              = [Response] [BucketID] [iData]
                          [MemoryDump] [RegKeyValues] [fDoc]
                          [WQLKeyValues] [GetFileKeyValues]
                          [GetFileVersionKeyValues][Tracking]
```

```
                          [CrashesPerBucket] [URLLaunch]
                          [NoSecondLevelCollection] [NoFileCollection]
                          [NoExternalURL] ; terms may appear in any order
CERBooleanValue         = "YES" / "TRUE" / "1" / "NO" / "FALSE" / "0"
                          ; case insensitive
Response                = %d82.101.115.112.111.110.115.101.61 ResponseValue CRLF
                          ; the encoded characters spell case-sensitive "Response="
ResponseValue           = "1" / Url
BucketID                = %d66.117.99.107.101.116.61 (%x31-39)*DIGIT CRLF
                          ; the encoded characters spell case-sensitive "Bucket="
iData                   = %d105.68.97.116.97.61 CERBooleanValue CRLF
                          ; the encoded characters spell case-sensitive "iData="
MemoryDump              = %d77.101.109.111.114.121.68.117.109.112.61 CERBooleanValue CRLF
                          ; the encoded characters spell case-sensitive "MemoryDump="
RegKeyValues            = %d82.101.103.75.101.121.61 RegKeyList CRLF
                          ; the encoded characters spell case-sensitive "RegKey="
RegKeyList              = RegKey [";" [RegKeyList]]
RegKey                  = 1*CHAR    ;    see below for delimiter handling
fDoc                    = %d102.68.111.99.61 CERBooleanValue CRLF
                          ; the encoded characters spell case-sensitive "fDoc="
WQLKeyValues            = %d87.81.76.61 WQLList CRLF
                          ; the encoded characters spell case-sensitive "WQL="
WQLList                 = (WQL / WQL ";" WQLList)
WQL                     = 1*CHAR ; WMI query syntax specified in [LAVY-MEGGITT],
                          see below for delimiter handling
GetFileKeyValues        = %d71.101.116.70.105.108.101.61 GetFileList CRLF
                          ; the encoded characters spell case-sensitive "GetFile="
GetFileList             = GetFile [";" [GetFileList]]
GetFile                 = Path
GetFileVersionKeyValues = %d71.101.116.70.105.108.101.86.101.114.115.105.111.110.61
                          GetFileList CRLF
                          ; the encoded characters spell case-sensitive "GetFileVersion="
Path                    = 1*CHAR    ;    see below for delimiter handling
Tracking                = %d84.114.97.99.107.105.110.103.61 CERBooleanValue CRLF
                          ; the encoded characters spell case-sensitive "Tracking="
CrashesPerBucket        = %d67.114.97.115.104.101.115.32.112.101.114.32.98.117.99
                          .107.101.116.61 (1DIGIT / ((%x31-39)1*DIGIT)) CRLF
                          ; the encoded characters spell case-sensitive "Crashes per bucket="
URLLaunch               = %d85.82.76.76.97.117.110.99.104.61 [Url] CRLF
                          ; the encoded characters spell case-sensitive "URLLaunch="
Url                     = URI ; [RFC 3986], see below for delimiter handling
NoSecondLevelCollection = %d78.111.83.101.99.111.110.100.76.101.118.101.108.67.111.108
                          .108.101.99.116.105.111.110.61 CERBooleanValue CRLF
                          ; the encoded characters spell case-sensitive
"NoSecondLevelCollection="
NoFileCollection        = %d78.111.70.105.108.101.67.111.108.108.101.99.116.105
                          .111.110.61 CERBooleanValue CRLF
                          ; the encoded characters spell case-sensitive "NoFileCollection="
NoExternalURL           = %d78.111.69.120.116.101.114.110.97.108.85.82.76.61
                          CERBooleanValue CRLF
                          ; the encoded characters spell case-sensitive "NoExternalURL="
```

CERBooleanValue: The values "YES", "TRUE", and "1" represent True. The values "NO", "FALSE", and "0" represent False.

Response: The value "1" for this parameter indicates that the error reporting system has no additional information to display about this problem. If this parameter is a URL the CER client SHOULD display a response prompt pointing to the URL specified by this parameter.

BucketID: This MUST be a positive decimal integer, and MUST NOT be zero.

iData: A True **CERBooleanValue** (as well as the absence of the parameter) instructs the CER client that an error reporting file MUST be generated for this error signature. A False **CERBooleanValue** instructs the CER client that an error reporting file MUST NOT be made. This is one of several values consulted in determining whether to generate an error reporting file; see section 3.1.7 for more details.

MemoryDump: A True **CERBooleanValue** instructs the CER client to add sections of the memory address space of the affected process to the error report. A False **CERBooleanValue** (as well as the absence of the parameter) instructs the CER client to do nothing with respect to the memory address space.

RegKeyValues: This parameter lists any number of semicolon-delimited registry key names. The CER client MUST collect the values of these keys if they are present in the registry. The CER client MUST include this information in the error reporting file.

RegKey: A CRLF pair MUST NOT appear in the **RegKey** value, since that is reserved as the line delimiter. A semicolon MUST NOT appear in the **RegKey** value, since that is reserved as the list item delimiter.

fDoc: A True **CERBooleanValue** instructs the CER client that the contents of any currently open documents in the software generating the error report are requested to be added to the error report. A False **CERBooleanValue** (as well as the absence of the parameter) instructs the CER client to do nothing with respect to the open documents.

WQLKeyValues: A string value that instructs the CER client to collect the Windows Management Instrumentation (WMI) objects (as specified in [LAVY-MEGGITT]) that are specified by this parameter, and include them in this error report.

WQL: A CRLF pair MUST NOT appear in the **WQL** value, since that is reserved as the line delimiter. A semicolon MUST NOT appear in the **WQL** value, since that is reserved as the list item delimiter.

GetFile: This parameter lists any number of semicolon-delimited file names to collect and include in the error report. It MUST be in a file path notation supported by the client systems that are expected to encounter the type of error this file corresponds to. The notation MUST support environment variables.

GetFileVersion: This parameter lists any number of semicolon-delimited file names to collect version information from and include in the error report. It MUST be in a file path notation supported by the client systems that are expected to encounter the type of error this file corresponds to. The notation MUST support environment variables.

Path: A CRLF pair MUST NOT appear in the **Path** value, since that is reserved as the line delimiter. A semicolon MUST NOT appear in the **Path** value, since that is reserved as the list item delimiter.

Tracking, **Crashes Per Bucket**, **NoSecondLevelCollection**, **NoFileCollection**, **NoExternalURL**, **URL**: These elements MUST conform to section 2.2.4.

URLLaunch: This parameter SHOULD be as specified in section 2.2.4. <6>

3 Protocol Details

3.1 Client to Server Detail

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. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

DWFileTreeRoot

The *DWFileTreeRoot* parameter specifies the UNC path to the location of the CER file share. The existence of this parameter instructs the CER client that the Corporate Error Reporting Version 1.0 Protocol will be used.<7>

3.1.2 Timers

None.

3.1.3 Initialization

The CER client MUST check for the existence of the *DWFileTreeRoot* parameter. If there is no parameter, or if the parameter is not valid, the CER client MUST stop any further processing.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

None.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

When a system or application error occurs, if the CER client is configured to use the Corporate Error Reporting Version 1.0 Protocol as specified in section 3.1.1, the CER client MUST perform the following actions:

1. The CER client MUST first construct the required file paths depending on the type of error event, as specified in section 2.2.3. The CER client MUST then check for the existence of the relevant policy.txt and status.txt files, as specified in sections 2.2.4 and 2.2.5, and if they exist then read the files by using the SMB Protocol. It MUST verify that the files conform to the format specified in sections 2.2.4 and 2.2.5. If the files do not conform to the specified formats, or if individual entries in those files do not conform to the specified formats, the configuration options which are incorrectly specified MUST NOT be honored by the CER client.

2. If both a status.txt file and a policy.txt file exist for a particular error and contain the same parameter with different values, the CER client MUST use status.txt parameter over the policy.txt parameter. If neither exist, the CER client SHOULD refer to its default value for that particular parameter. <8>

3. The CER client MUST determine whether there are already a sufficient number of reports for this particular error. First, it must determine how many reports have already been made. If the count.txt file as specified in section 2.2.1 exists, the CER client MUST read the number of copied error reporting files from it. Otherwise the CER client MUST assume that the number of reports is zero. Next, the CER client MUST determine the maximum number of copied error reporting files for this error. It first checks for a CrashesPerBucket value set by status.txt, then by policy.txt, then falls back to its default value. Finally, the CER client compares these two numbers and as long as there are fewer copied reports than the CrashesPerBucket value instructs, then the CER client MUST continue the error reporting file generation process. If there are already a sufficient number of reports then the CER client MUST skip to step 8 below.

4. The CER client MUST check the status.txt file (section 2.2.5) for the existence of an iData parameter with an affirmative Boolean value to verify that an error reporting file is requested. If the parameter is absent or has a negative Boolean value, the CER client MUST skip to step 8 below.

5. The CER client MUST check the status.txt file (section 2.2.5) to determine whether any additional data is specified for the error reporting file. If so, the CER client MUST collect as much of the requested data as possible (for example, if a file is requested in the GetFile section of status.txt that is not present on the system, then the file cannot be included but the error report SHOULD still be made<9>). The method of data collection is implementation-specific.<10> Note that there is no requirement that two clients use the same method or format of error information.

6. The CER client MUST compress the complete report information into a single file by using any implementation-specific file compression.<11> Note that there is no requirement that two clients use the same file compression scheme. If the data cannot be compressed, the CER client MUST NOT continue creating an error reporting file, and MUST skip to step 8 below.

7. The CER client MUST attempt to copy the error report to the CER file share. If this attempt fails, the error report MUST NOT be copied.

8. If the count.txt file does not exist, the CER client MUST create the file and set its contents as follows. The value <cFilesCopied> is "1" if the CER successfully copied a file in step 7, and is otherwise "0".

```
Cabs Gathered=<cFilesCopied>
Total Hits=1
```

   If the count.txt file already exists, the CER client MUST attempt to increment the Total Hits value by one and the Cabs Gathered value by <cFilesCopied>.

9. This final step applies only if the policy.txt or status.txt file indicated that internal tracking is enabled. If crash.log does not yet exist, the CER client MUST create an empty crash.log file at that location. Then the CER client MUST attempt to append to the crash.log file following the format specified in section 2.2.2.2. If hits.log does not yet exist, the CER client MUST create an empty hits.log file at that location. Then the CER client MUST attempt to append to the hits.log file following the format specified in section 2.2.2.1.

4 Protocol Examples

4.1 Application Fault Example

1. An application fault occurs while running TestApplication.exe.
2. The system creates an error report.
3. The CER client checks to see whether a CER file share has been configured as specified in section 3.1.1. The following value is set:

```
DWFileTreeRoot = "\\MyCerServer\CERFileShare\"
```

4. The CER client checks for the existence of a policy.txt file at the location specified by DWFileTreeRoot. No policy.txt file exists.
5. The CER client constructs the following folder structure based on the information specified in section 2.2.3:

```
\\MyCerServer\CERFileShare\status\TestApplication\1.0.0.0\TestModule\
1.0.0.0\00000000\status.txt
```

6. A status.txt file exists at this location. The CER client parses the status.txt file, which includes the following parameters and values:

```
Tracking=YES
Response=http://www.microsoft.com/ms.htm
Crashes per bucket=100
NoSecondLevelCollection=NO
NoFileCollection=NO
RegKey=HKLM\Software\Microsoft\PCHealth\ErrorReporting;HKLM\Software\
Microsoft\PCHealth\Test
iData=1
fDoc=0
WQL=select * from Win32_logicaldisk
GetFile=%WINDIR%\system32\notepad.exe;%WINDIR%\system32\faultrep.dll
GetFileVersion=%WINDIR%\system32\notepad.exe;%WINDIR%\system32\
faultrep.dll
```

7. This status.txt file has specified a "Crashes per bucket" value of 100, so the CER client checks to make sure that 100 error reporting files have not already been collected for this problem. It does this by looking at the count.txt file for the error:

```
\\MyCerServer\CERFileShare\counts\TestApplication\1.0.0.0\TestModule\1.0.0.0\00000000\
count.txt
```

   The count.txt file has the following contents:

```
Cabs Gathered=5
Total Hits=10
```

Since 5 is fewer than 100, the CER client continues the data collection process.

8. This status.txt file has specified that additional data be added to the error report, in the form of two registry key values, a WMI query, two files, and version information for two files. The CER client collects this information and compresses all of the report files into a single file with the randomly generated name of "d5je031w.cab".

9. The CER client copies the error reporting files to the CER file share:

```
\\MyCerServer\CERFileShare\cabs\TestApplication\1.0.0.0\TestModule\
1.0.0.0\00000000\d5je031w.cab
```

10. The CER client updates the following file on the CER file share to increment the number of hits and the number of copied error reporting files:

```
\\MyCerServer\CERFileShare\counts\TestApplication\1.0.0.0\TestModule\
1.0.0.0\00000000\count.txt
```

The count.txt file now has the following contents:

```
Cabs Gathered=6
Total Hits=11
```

11. The status.txt file for this error signature has enabled internal tracking, so the CER client opens the crash.log file on the CER file share for this problem:

```
\\MyCerServer\CERFileShare\crash.log
```

The CER client appends the following text to the crash.log file:

```
"15:32:23  04-23-2007   TestMachine   TestUser
TestApplication\1.0.0.0\TestModule\1.0.0.0\00000000"
```

12. The CER client also opens the hits.log file on the CER file share:

```
\\MyCerServer\CERFileShare\cabs\TestApplication\1.0.0.0\TestModule\
1.0.0.0\00000000\hits.log
```

13. The CER client appends the following text to the hits.log file:

```
"15:32:23  04-23-2007   TestMachine   TestUser   d5je031w.cab"
```

4.2 Kernel Fault Example

1. Kernel-mode fault occurs.
2. The system creates an error report.
3. The CER client checks to see whether a CER file share has been configured as specified in section 3.1.1. The following value is set:

```
DWFileTreeRoot = \\MyCerServer\CERFileShare\
```

4. The CER client constructs the path for the policy.txt file:

```
\\MyCerServer\CERFileShare\policy.txt
```

5. The CER client attempts to read the policy.txt file, and finds that no policy.txt file exists.
6. The CER client constructs the path for the status.txt file. Because this is a kernel mode report, the path of status.txt is the following:

```
\\MyCerServer\CERFileShare\status\blue\status.txt
```

7. The CER client attempts to read the status.txt file, and finds that no status.txt file exists.
8. Because neither a policy.txt nor a status.txt file exists, this error would ordinarily be subject to the CER client's default value of 5 for "Crashes per bucket". However, the CER client determines that since this particular type of error does not have parameters, that it will not restrict the number of copied error reporting files.
9. The CER client creates an error reporting file and uses the SMB Protocol to copy it to the specified file share:

```
\\MyCerServer\CERFileShare\cabs\blue\d5JE031w.cab
```

10. The CER client opens the count.txt file named as follows:

```
\\MyCerServer\CERFileShare\counts\blue\count.txt
```

The CER client updates its contents to reflect the additional hit and copied error reporting file:

```
Cabs Gathered=12345
Total Hits=23456
```

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

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

- Windows XP operating system
- Windows Server 2003 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 2.2.3: Windows has a variety of prohibited characters and names; for more information, see [MSDN-FILE].

<2> Section 2.2.4: On Windows XP and Windows Server 2003, these configuration elements may also be set via group policy on the machine running the CER client, which will cause those values to override the defaults described here.

<3> Section 2.2.4: On Windows XP and Windows Server 2003, these configuration elements may also be set via group policy on the machine running the CER client, which will cause those values to override the defaults described here.

<4> Section 2.2.4: On Windows XP and Windows Server 2003, these configuration elements may also be set via group policy on the machine running the CER client, which will cause those values to override the defaults described here.

<5> Section 2.2.4: On Windows XP and Windows Server 2003, these configuration elements may also be set via group policy on the machine running the CER client, which will cause those values to override the defaults described here.

<6> Section 2.2.5: It is possible to configure a CER tool in such a way that the **URLLaunch** value is left empty.

<7> Section 3.1.1: On Windows XP and Windows Server 2003, the following registry key specifies the UNC path of the CER file share:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\PCHealth\
ErrorReporting\DW] "DWFileTreeRoot"
```

<8> Section 3.1.7: On Windows XP and Windows Server 2003, these configuration elements may also be set via group policy on the machine running the CER client, which will cause those values to override the defaults described in section 2.2.4.

<9> Section 3.1.7: On Windows XP and Windows Server 2003, if a MemoryDump is requested and the CER client cannot generate one, the CER client will skip to step 8 of this section.

<10> Section 3.1.7: Windows uses a new file for each piece of information collected (for example, .reg for a registry key or .mdmp for a minidump).

<11> Section 3.1.7: Windows uses .CAB files for this compression; for more information, see [MSDN-CAB].

7 Change Tracking

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

8 Index