[MS-GPEF]: Group Policy: Encrypting File System Extension

Revision Summary

Date	Revision History	Revision Class	Comments
12/18/2006	0.01		MCPP Milestone 2 Initial Availability
03/02/2007	1.0		MCPP Milestone 2
04/03/2007	1.1		Monthly release
05/11/2007	1.2		Monthly release
07/03/2007	1.2.1	Editorial	Revised and edited the technical content.
07/20/2007	1.2.2	Editorial	Revised and edited the technical content.
08/10/2007	2.0	Major	Updated and revised the technical content.
09/28/2007	3.0	Major	Updated and revised the technical content.
10/23/2007	3.0.1	Editorial	Revised and edited the technical content.
11/30/2007	3.0.2	Editorial	Revised and edited the technical content.
01/25/2008	3.0.3	Editorial	Revised and edited the technical content.
03/14/2008	4.0	Major	Updated and revised the technical content.
05/16/2008	4.0.1	Editorial	Revised and edited the technical content.
06/20/2008	4.0.2	Editorial	Revised and edited the technical content.
07/25/2008	4.0.3	Editorial	Revised and edited the technical content.
08/29/2008	4.0.4	Editorial	Revised and edited the technical content.
10/24/2008	4.0.5	Editorial	Revised and edited the technical content.
12/05/2008	5.0	Major	Updated and revised the technical content.
01/16/2009	5.0.1	Editorial	Revised and edited the technical content.
02/27/2009	5.0.2	Editorial	Revised and edited the technical content.
04/10/2009	5.0.3	Editorial	Revised and edited the technical content.
05/22/2009	6.0	Major	Updated and revised the technical content.
07/02/2009	6.0.1	Editorial	Revised and edited the technical content.
08/14/2009	6.0.2	Editorial	Revised and edited the technical content.
09/25/2009	7.0	Major	Updated and revised the technical content.
11/06/2009	8.0	Major	Updated and revised the technical content.

Date	Revision History	Revision Class	Comments
12/18/2009	8.1	Minor	Updated the technical content.
01/29/2010	9.0	Major	Updated and revised the technical content.
03/12/2010	10.0	Major	Updated and revised the technical content.
04/23/2010	10.0.1	Editorial	Revised and edited the technical content.
06/04/2010	11.0	Major	Updated and revised the technical content.
07/16/2010	12.0	Major	Significantly changed the technical content.
08/27/2010	13.0	Major	Significantly changed the technical content.
10/08/2010	13.1	Minor	Clarified the meaning of the technical content.
11/19/2010	13.1	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	13.1	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	14.0	Major	Significantly changed the technical content.
03/25/2011	15.0	Major	Significantly changed the technical content.
05/06/2011	16.0	Major	Significantly changed the technical content.
06/17/2011	16.1	Minor	Clarified the meaning of the technical content.
09/23/2011	16.1	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	17.0	Major	Significantly changed the technical content.
03/30/2012	17.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	17.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	17.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/31/2013	18.0	Major	Significantly changed the technical content.
08/08/2013	19.0	Major	Significantly changed the technical content.
11/14/2013	19.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/13/2014	19.0	No change	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
05/15/2014	19.1	Minor	Clarified the meaning of the technical content.

Contents

1 Introduction

The Group Policy: Encrypting File System Extension uses Group Policy: Core Protocol , specified in [MS-GPOL], to allow remote administrative configuration of the **Encrypting File System (EFS)**.

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]:

Active Directory
certificate
certification authority (CA) or certificate authority
certificate template
client-side extension GUID (CSE GUID)
common name (CN)
computer-scoped Group Policy Object path
data recovery agent (DRA)
domain
domain controller (DC)
elliptic curve cryptography (ECC)
Encrypting File System (EFS)
globally unique identifier (GUID)
Group Policy Object (GPO)
Group Policy server
Lightweight Directory Access Protocol (LDAP)
NT file system (NTFS)
page file or paging file
registry
registry policy file
Rivest-Shamir-Adleman (RSA)
security identifier (SID)
self-signed certificate
share
smart card
tool extension GUID or administrative plug-in GUID
Unicode
universally unique identifier (UUID)
X.509

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.

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.

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

[MS-EFSR] Microsoft Corporation, "Encrypting File System Remote (EFSRPC) Protocol".

[MS-FASOD] Microsoft Corporation, "File Access Services Protocols Overview".

[MS-GPOL] Microsoft Corporation, "Group Policy: Core Protocol".

[MS-GPREG] Microsoft Corporation, "Group Policy: Registry Extension Encoding".

[NSA] National Security Agency "NSA Suite B Cryptography", November 2009, http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml

[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

[RFC5280] Cooper, D., Santesson, S., Farrell, S., et al., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008, http://www.ietf.org/rfc/rfc5280.txt

1.2.2 Informative References

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

[MS-LSAD] Microsoft Corporation, "Local Security Authority (Domain Policy) Remote Protocol".

1.3 Overview

Encrypting File System (EFS) is a capability of the **New Technology File System (NTFS)**. It encrypts files stored on disk in a manner transparent to users and applications. Each user of EFS is associated with a key pair of a public key cryptography system. In addition, administrators may configure **data recovery agents (DRAs)**, which are logical entities, each associated with its own key pair. When EFS encrypts a file, it randomly generates a symmetric key that is used to encrypt the file data. It then encrypts a copy of this key with the public key of each user and with the public key of each DRA that is authorized to access the file, and stores these in the file metadata. When a user with access to one of the corresponding private keys tries to open such a file, NTFS automatically invokes EFS functionality to extract the symmetric key from the file metadata and decrypt the file data on the fly.

The behavior of EFS can be controlled through Group Policy. This mechanism may be used by an administrator to enable or disable EFS, or to enforce policies such as those related to key management or data recovery. All EFS policies are machine-specific, meaning that all users on a given machine will have the same policy applied to them.

1.3.1 Background

The Group Policy: Core Protocol, specified in [MS-GPOL], allows clients to discover and retrieve policy settings created by administrators of a **domain**. These settings are persisted within Group Policy Objects (GPOs), which are assigned to policy target accounts in **Active Directory**. Policy target accounts are either computer accounts or user accounts in Active Directory. Each client uses the **Lightweight Directory Access Protocol (LDAP)** to determine which GPOs are applicable to it by consulting the Active Directory objects corresponding to its computer account and the user accounts of any users logging on to the client computer.

On each client, each GPO is interpreted and acted upon by software components known as client-side plug-ins. Each client-side plug-in is associated with a specific class of settings. The client-side plug-ins that are responsible for a given GPO are specified by using an attribute on the GPO. This attribute specifies a list of **GUID** pairs. The first GUID of each pair is referred to as a **client-side extension (CSE GUID)**. The second GUID of each pair is referred to as a **tool extension GUID**.

For each GPO that is applicable to a client, the client consults the CSE GUIDs listed in the GPO to determine which client-side plug-ins on the client should handle the GPO. The client then invokes the client-side plug-ins to handle the GPO.

A client-side plug-in uses the contents of the GPO to retrieve and process settings specific to its class, in a manner specific to the plug-in.

1.3.2 EFS Group Policy Extension Overview

EFS Group Policy settings are accessible from a GPO through the Group Policy: Encrypting File System Extension to the Group Policy: Core Protocol specified in [MS-GPOL]. The extension provides a mechanism for administrative tools to obtain metadata about registry-based settings.

The process of configuring and applying the EFS Group Policy settings consists of the following steps:

1. An administrator invokes a Group Policy administrative tool to administer a GPO through the Group Policy: Core Protocol using the Policy Administration Protocol, as specified in [MS-GPOL] section 2.2.8. Through this protocol, the presence of the tool extension GUID for computer policy settings for the Group Policy: Encrypting File System Extension in is retrieved, which indicates that the GPO contains policy settings that should be administered through the Policy Administration portion of the Group Policy: Encrypting File System Extension.

   The administrative tool invokes a plug-in specific to the Group Policy: Encrypting File System Extension so that the administrator can administer the EFS settings, which results in the storage and retrieval of metadata inside a GPO on a **Group Policy server**. This metadata describes configuration settings to be applied to a generic settings database (the **registry** in Windows) on a client that is affected by the GPO.

   The administrator views the data and updates it as desired.

2. A client computer affected by that GPO is started (or is connected to the network, if this happens after the client starts), and the Group Policy: Core Protocol is invoked by the client to retrieve policy settings from the Group Policy server. As part of this processing, two GUIDs are read from the GPO: the registry extension's CSE GUID, as specified in [MS-GPREG] section 1.9, and the EFS extension's CSE GUID.

3. The presence of the registry extension's CSE GUID, as specified in [MS-GPREG] section 1.9, in the GPO instructs the client to invoke a registry extension plug-in component for policy

application. This component parses the file of settings and saves them in the generic settings database (registry) on the local machine.

4. The presence of the EFS extension's CSE GUID in the GPO instructs the client to invoke an EFS extension plug-in component for policy application. This component is not required for the protocol and does not affect the operation of the protocol. Specifically, this component is intended to adjust the internal state of the EFS on the client, and it is not intended to participate in any network communication.

5. The EFS on the client recognizes that its configuration has been updated and takes the appropriate actions.

This document specifies the behavior of the administrative plug-in mentioned in step 1. The operation of the Group Policy: Core Protocol in step 2 is specified in [MS-GPOL] section 3.2. The process of retrieving the settings in step 3 is specified in [MS-GPREG] section 3.2. Step 4 and step 5 are specific to an implementation of EFS and are not specified.

1.4 Relationship to Other Protocols

Group Policy: Encrypting File System Extension is invoked as an extension of the Group Policy: Core Protocol. Group Policy: Encrypting File System Extension is only initiated as part of the Group Policy: Core Protocol . The invocation of this (and of all Group Policy protocol extensions) is specified in [MS-GPOL] section 3. Group Policy: Encrypting File System Extension explicitly depends on all the protocols upon which the Group Policy: Core Protocol depends.

Group Policy: Encrypting File System Extension also depends on the Group Policy: Registry Extension Encoding specified in [MS-GPREG] to retrieve settings from a GPO and to populate settings in the client registry. Group Policy: Encrypting File System Extension uses file access protocols described in [MS-FASOD] as its underlying transport.

Group Policy: Encrypting File System Extension configures settings that are used by the EFSRPC protocol. These settings are defined in [MS-EFSR] section 3.1.1.1.

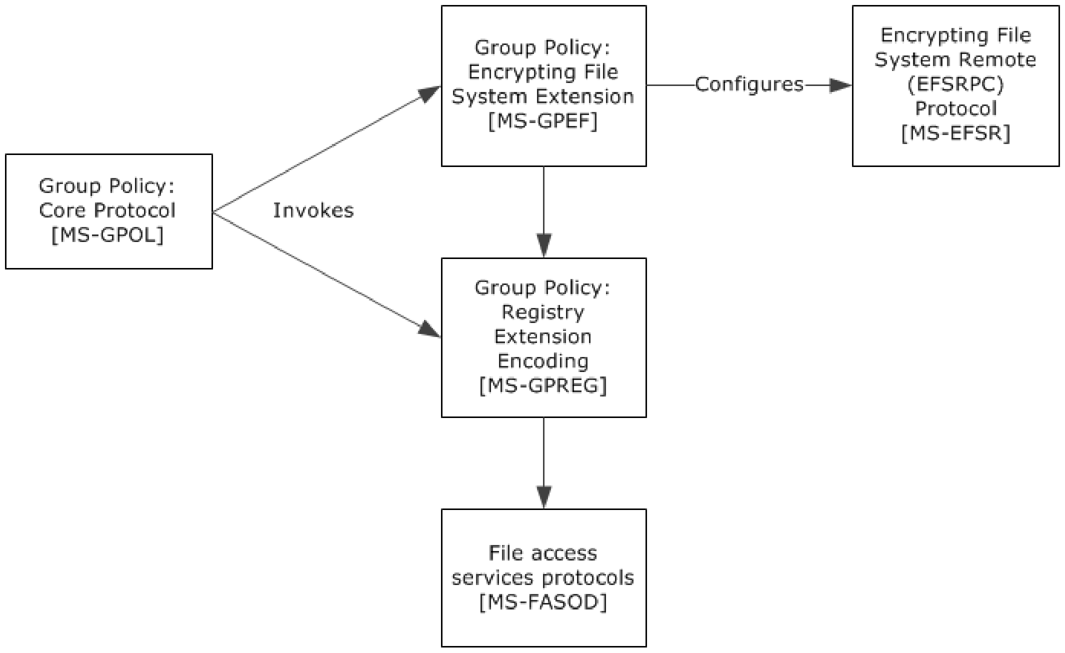
Group Policy: Core Protocol [MS-GPOL]
Invokes
Group Policy: Encrypting File System Extension [MS-GPEF]
Configures
Encrypting File System Remote (EFSRPC) Protocol [MS-EFSR]
Group Policy: Registry Extension Encoding [MS-GPREG]
File access services protocols [MS-FASOD]

Figure 1: Protocol relationship diagram

1.5 Prerequisites/Preconditions

The prerequisites for this extension are the same as those for the Group Policy: Core Protocol.

In addition, the Group Policy: Registry Extension Encoding must be present on the client to retrieve the settings stored by the EFS Group Policy administrative plug-in.

1.6 Applicability Statement

The Group Policy: Encrypting File System Extension is only applicable within the Group Policy framework and only when a number of client computers in a domain support EFS. The Group Policy: Encrypting File System Extension is intended to be used to configure certain aspects of EFS behavior on such clients.

The Group Policy: Encrypting File System Extension is in a certain class of extensions that have only an administrative-side extension and no client-side extension. These extensions are data structures and are documented here for informative purposes only.

1.7 Versioning and Capability Negotiation

The Group Policy: Encrypting File System Extension does not provide versioning or capability negotiations.

All EFS Group Policy settings have unique definitions, and all implementations are required to support a base set of options. Thus, the only differences between implementations are in the sets of additional options supported. Due to the nature of the Group Policy configuration store used for EFS settings, a **domain controller (DC)** can store and maintain even those settings which its Group Policy: Encrypting File System Extension do not support.

In a heterogeneous environment, it is expected that some client computers will not recognize some settings specified here. A client will not be able to use the administrative plug-in to configure those settings that it does not support. However, the EFS administrative plug-in will neither modify nor destroy any settings it does not support.

1.8 Vendor-Extensible Fields

The Group Policy: Encrypting File System Extension does not define any vendor-extensible fields.

1.9 Standards Assignments

This protocol defines CSE GUID and tool extension GUID values as specified in [MS-GPOL] section 1.8. The assignments are as follows.

Parameter	Value
CSE GUID	{B1BE8D72-6EAC-11D2-A4EA-00C04F79F83A}
Tool extension GUID (computer policy settings)	{53D6AB1D-2488-11D1-A28C-00C04FB94F17}
Tool extension GUID (default domain policy settings)	{53D6AB1B-2488-11D1-A28C-00C04FB94F17}

2 Messages

2.1 Transport

The EFS Group Policy administrative plug-in uses the transport specified in [MS-GPOL] to read and modify settings in the central policy store. Specifically, it uses the file access services protocols described in [MS-FASOD] as the underlying transport for reading, updating, creating, and deleting EFS group policy settings. Information is retrieved from the policy store and written to the client's registry by the Group Policy: Registry Extension Encoding ([MS-GPREG] section 3.2) using file access protocols as the underlying transport. The file access version, capabilities, and authentication used for this connection are negotiated between the client and the server when the connection is established.

2.2 Message Syntax

The Group Policy: Encrypting File System Extension MUST use the message syntaxes as specified in [MS-GPOL] section 2.2 and [MS-GPREG] section 2.2. EFS Group Policy options are implemented as entries in the machine-specific Registry Policy file used by the Group Policy: Registry Extension Encoding. To support a given Group Policy option, the EFS administrative plug-in MUST provide a method to write and query the corresponding entry in the machine-specific Registry Policy file of the relevant GPO.

The following EFS Group Policy options are defined:

- EFS recovery policy
- EFS enabled status
- EFS additional options
- EFS user template name
- EFS self-signed **certificate** key length or algorithm identifier.

These are described in more detail in the following sections. Because all message processing is performed by the Group Policy: Core Protocol, the following sections merely specify the format of the corresponding entries in the machine-specific Registry Policy file. The intent of various settings is also described in the following sections; however, these settings are processed by the EFS in the client, and their descriptions here are only for informative purposes, not for normative purposes.

2.2.1 EFS Recovery Policy

This option MUST be supported by all implementations of the Group Policy: Encrypting File System Extensions.

When writing the EFS recovery policy, the administrative plug-in MUST configure the machine-specific Registry Policy file to create a registry key named Software\Policies\Microsoft\SystemCertificates\EFS. This key MUST contain three subkeys, named Certificates, CRLs, and CTLs, respectively. The Certificates subkey MUST in turn contain zero or more subkeys, each of which represents the **X.509** certificate (as specified in [RFC5280]) of an EFS recovery agent. The format of these entries is specified in section 2.2.1.1. The CRLs and CTLs subkeys MUST be empty.

In addition to the previous information, the administrative plug-in MUST create an additional entry in the machine-specific Registry Policy file, which contains all the applicable EFS recovery agent certificates marshaled into a single value, as specified in section 2.2.1.2.

2.2.1.1 Recovery Agent Certificate

A separate registry key MUST be created for each EFS recovery agent under the path Software\Policies\Microsoft\SystemCertificates\EFS\Certificates. The name of this key MUST be the 40-character string corresponding to the hexadecimal representation of the SHA-1 hash of the certificate (this quantity is sometimes referred to as the certificate "thumbprint"). This key MUST contain a single value of type REG_BINARY. The name of this value MUST be "Blob". The format of this value (hereafter referred to as the certificate binary large object (BLOB)) is described in the following sections.

2.2.1.1.1 Certificate BLOB

The certificate BLOB MUST consist of zero or more certificate properties, followed by the encoded certificate. The format of the properties is specified in section 2.2.1.1.1.1. The format of the encoded certificate is specified in section 2.2.1.1.1.2.

2.2.1.1.1.1 Certificate BLOB Properties

Each property in the certificate BLOB structure MUST be formatted 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
PropertyID																															
Reserved																															
Length																															
Value (variable)																															
...																															

PropertyID (4 bytes): This field MUST identify the property whose value is contained in the **Value** field. It MUST be an unsigned 32-bit integer in little-endian format. Valid integer values are shown in the following table.

Value	Meaning
KEY_PROV_INFO 2	This property is used to provide hints regarding the handling of the certificate. Its format is specified later in this section.
SHA1_HASH 3	20-byte array representing the SHA-1 hash of the certificate.
MD5_HASH 4	16-byte array representing the MD5 hash of the certificate.

Value	Meaning
KEY_SPEC 6	Unsigned 32-bit integer in little-endian format. The only valid value is 1 (also referred to as AT_KEYEXCHANGE).
ENHKEY_USAGE 9	The value of the extended key usage extension on the certificate, in ASN.1 DER encoding. For more information, see [RFC5280] section 4.2.1.12.
FRIENDLY_NAME 11	A null-terminated **Unicode** string representing the display name for the certificate.
DESCRIPTION 13	A null-terminated Unicode string representing a brief description of the certificate.
SIGNATURE_HASH 15	A 20-byte array containing the SHA-1 hash of the certificate signature, or a 16-byte array containing the MD5 hash of the certificate signature.
KEY_IDENTIFIER 20	A 20-byte array containing the SHA-1 hash of the certificate subject public key.
AUTO_ENROLL 21	A null-terminated Unicode string that contains the name or object identifier used for autoenrollment. This is present when the certificate was obtained through autoenrollment.
PUBKEY_ALG_PARA 22	The algorithm identifier for the public key contained in the certificate, in Distinguished Encoding Rules (DER) encoding. The structure of an X.509 certificate is defined by [RFC5280].
ISSUER_PUBLIC_KEY_MD5_HASH 24	A 16-byte array containing the MD5 hash of the public key associated with the private key used to sign the certificate.
SUBJECT_PUBLIC_KEY_MD5_HASH 25	A 16-byte array containing the MD5 hash of the public key contained in the certificate.
DATE_STAMP 27	A date stamp, in the form of an unsigned 64-bit integer in little-endian format representing the number of 100-nanosecond intervals since January 1, 1601.
ISSUER_SERIAL_NUMBER_MD5_HASH 28	A 16-byte array containing the MD5 hash of the **CA** signing certificate serial number.
SUBJECT_NAME_MD5_HASH 29	A 16-byte array containing the MD5 hash of the subject name in the certificate.

Reserved (4 bytes): Reserved. MUST be set to 0x01 0x00 0x00 0x00.

Length (4 bytes): This field MUST contain the length of the **Value** field in bytes. It MUST be an unsigned 32-bit number in little-endian format.

Value (variable): This field MUST contain the value of the specified property, in the format specified for the property associated with the table of possible values for PropertyID.

2.2.1.1.1.1.1 KEY_PROV_INFO

The value for the KEY_PROV_INFO property (if this property is present) MUST be in the following format.

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 to Container Name																															
Offset to Provider Name																															
Provider Type																															
Flags																															
Reserved																															
...																															
Key Specification																															
Name Data (variable)																															
...																															

Offset to Container Name (4 bytes): This MUST be set to the offset, in bytes, of the Container Name subfield of the **Name Data** field from the beginning of this structure. It MUST be an unsigned 32-bit integer in little-endian format.

Offset to Provider Name (4 bytes): This MUST be set to the offset, in bytes, of the Provider Name subfield of the **Name Data** field from the beginning of this structure. It MUST be an unsigned 32-bit integer in little-endian format.

Provider Type (4 bytes): This field indicates the class of cryptographic algorithm associated with the public key in the certificate. It MUST be set to the unsigned 32-bit number 0x00000001 (in little-endian format) to signify an RSA public key.

Flags (4 bytes): This field SHOULD be set to zero, and its value MUST be ignored by the client.

Reserved (8 bytes): This field is two rows total in the previous diagram and MUST be set to zero.

Key Specification (4 bytes): This field indicates the cryptographic capabilities associated with the public key in the certificate. It MUST be set to the unsigned 32-bit number 0x00000001 (in little-endian format) to signify that the key is usable for both signature and encryption operations.

Name Data (variable): This field MUST contain the following items, in any order, at the locations indicated by the respective Offset fields described earlier in this section. These items MUST be completely contained inside this field and MUST NOT overlap each other. There MUST be no unused areas within this field that span more than eight contiguous bytes. All unused bytes within this field SHOULD be set to zero. Unused bytes MUST be ignored by the implementation.

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
Container Name (variable)																															
...																															
Provider Name (variable)																															
...																															

Container Name (variable): This MUST be a null-terminated Unicode string representing a specific key container in the cryptographic service provider (CSP) referred to by the provider name.

Provider Name (variable): This MUST be a null-terminated Unicode string representing the CSP associated with the public key contained in the certificate.

2.2.1.1.1.2 Certificate BLOB Encoding

The encoded certificate structure MUST be formatted 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
Reserved																															
...																															
Length																															
Value (variable)																															
...																															

Reserved (8 bytes): This field MUST be set to the following bytes, in the following order: 0x20 0x00 0x00 0x00 0x01 0x00 0x00 0x00.

Length (4 bytes): This field MUST contain the length of the **Value** field in bytes. It MUST be an unsigned 32-bit number in little-endian format.

Value (variable): This field MUST contain the ASN.1 DER encoding of the X.509 certificate of the EFS Recovery Agent. The certificate MUST contain a public key for use with the RSA or ECC encryption algorithm. For more information, see [RFC5280].

2.2.1.2 EfsBlob Value

The EfsBlob entry MUST be represented in the machine-specific Registry Policy file as follows:

Key: **Software\Policies\Microsoft\SystemCertificates\EFS \EfsBlob**

Value: "EfsBlob" or one of the special values as described in [MS-GPREG] section 3.2.5.1.

Type: REG_BINARY.

Size: Equal to size of the **Data** field.

Data: Specified in the next section.

The format of the EfsBlob entry is specified in section 2.2.1.2.1.

2.2.1.2.1 EfsBlob

The EfsBlob packet is a data structure that contains EFS Recovery Keys.

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
Reserved																															
Key count																															
Keys (variable)																															
...																															

Reserved (4 bytes): Reserved array of bytes. MUST be set to 0x01 0x00 0x01 0x00, in that order.

Key count (4 bytes): Number of recovery keys included. This field MUST be greater than zero. This field MUST contain a 32-bit integer in little-endian format.

Keys (variable): This field MUST consist of one or more entries as specified in the **Key count** field, with each entry being formatted as described in section 2.2.1.2.2.

2.2.1.2.2 EfsKey

The EfsKey packet contains an EFS key.

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
Length1																															
Length2																															
SID offset																															
Reserved1																															

Certificate length
Certificate offset
Reserved2
...
SID (variable)
...
Certificate

Length1 (4 bytes): This field MUST be equal to the length of the key structure in bytes, as measured from the beginning of the **Length1** field to the end of the **Certificate** field. This field MUST be a 32-bit unsigned integer in little-endian format.

Length2 (4 bytes): This field MUST be equal to the length of the key structure in bytes, as measured from the beginning of the **Length2** field to the end of the **Certificate** field. This field MUST be a 32-bit unsigned integer in little-endian format. Note that the value of **Length2** is always four bytes less than the value of **Length1**. This redundancy is due to historical reasons.

SID offset (4 bytes): This field MUST be equal to the offset of the **SID** field in bytes, starting from the beginning of the **Length2** field. This field MUST be a 32-bit unsigned integer in little-endian format.

Reserved1 (4 bytes): This field MUST be set to 0x02 0x00 0x00 0x00. This field MUST be a 32-bit unsigned integer in little-endian format.

Certificate length (4 bytes): This field MUST be equal to the length of the **Certificate** field in bytes. This field MUST be a 32-bit unsigned integer in little-endian format.

Certificate offset (4 bytes): This field MUST be equal to the offset of the **Certificate** field in bytes, starting from the beginning of the **Length2** field. This field MUST be a 32-bit unsigned integer in little-endian format.

Reserved2 (8 bytes): All bits within this field SHOULD be set to zero. The client MUST ignore any nonzero values.

SID (variable): (Optional field.) This field MAY<1> contain the **security identifier (SID)** of a valid user within the domain. When set to a nonzero value, this field is intended to be used as a hint indicating which user created the key, and it does not affect the protocol processing at either the client or the server, as specified in [MS-DTYP] section 2.4.2.

Certificate (4 bytes): This field MUST contain the ASN.1 representation, in DER encoding, of an X.509 certificate from among the EFS recovery agent certificates described earlier.

2.2.2 EFS Enabled Status

Key: **Software\Policies\Microsoft\Windows NT\CurrentVersion\EFS**

Value: "EfsConfiguration" or one of the special values in [MS-GPREG] section 3.2.5.1.

Type: REG_DWORD.

Size: Equal to size of the **Data** field.

Data: 0x00000000 (to enable EFS on the client) or 0x00000001 (to disable EFS on the client).

This option SHOULD<2> be supported by implementations of the Group Policy: Encrypting File System Extension. If an implementation chooses not to support this option, the administrative plug-in MUST NOT modify the Registry Policy entry described earlier.

If the client supports this option but the option is not present, the client SHOULD use a default value of 0x00000000.

2.2.3 EFS Additional Options

Key: **Software\Policies\Microsoft\Windows NT\CurrentVersion\EFS**

Value: "EfsOptions" or one of the special values in [MS-GPREG] section 3.2.5.1.

Type: REG_DWORD.

Size: Equal to size of the **Data** field.

The registry value name "EfsOptions" may be replaced with one of the special values in [MS-GPREG] section 3.2.5.1.

Data: A 32-bit value consisting of the bitwise OR of zero or more of the following flags.

Value	Meaning
0x00000001	EFS should attempt to encrypt the user's Documents folder and its contents.
0x00000002	When using a **smart card** to store the user's private key, EFS should derive a symmetric key from the private key, cache it in memory, and perform symmetric key operations with it instead of asymmetric key operations with the private and public keys on the smart card.
0x00000004	EFS should permit users to use public keys associated with **self-signed certificates** for encryption.
0x00000010	EFS should flush all per-user secrets and keying material from memory after an idle interval as specified in the EFS cache timeout option (see more later in this section). If this flag is supported by an implementation, that implementation MUST also support the cache timeout option described later.
0x00000020	For users that are logged on to the client interactively, EFS should flush all per-user secrets and keying material from memory whenever the user temporarily locks the session.
0x00000100	EFS should reject attempts by users to create encrypted files or to encrypt existing files using keys not stored on a smart card.
0x00000200	This setting is intended to be used as a hint to the client to enable encryption of the system **page file**.
0x00000400	EFS should remind users to back up their keys each time they change their EFS key.
0x00001000	EFS should disallow the use of **ECC** keys for user and recovery keys. This flag MUST NOT be specified in combination with 0x00002000. If neither 0x00001000 nor 0x00002000 is

Value	Meaning
	specified, then both ECC and **RSA** keys are permitted.
0x00002000	EFS should require the use of ECC keys for user and recovery keys. This flag MUST NOT be specified in combination with 0x00001000. If neither 0x00001000 nor 0x00002000 is specified, then both ECC and RSA keys are permitted.

With the exception of flag 0x00000200, an implementation SHOULD<3> support all the flags described in this section. An implementation MAY <4> support flag 0x00000200.

If the client supports this option but the option is not present, the client SHOULD use a default value of 0x00000002 | 0x00000004 | 0x00000010.

2.2.4 EFS Cache Timeout

The EFS cache timeout field is specified as follows:

Key: **Software\Policies\Microsoft\Windows NT\CurrentVersion\EFS**

Value: "CacheTimeout" or one of the special values in [MS-GPREG] section 3.2.5.1.

Type: REG_DWORD.

Size: Equal to size of the **Data** field.

Data: This field is intended to be used in conjunction with a flag to flush the cache on timeout (specified earlier as 0x00000010). The implementation MUST either support both the **CacheTimeout** field AND the flag to flush the cache on timeout, or neither the **CacheTimeout** field nor the flag to flush the cache on timeout. This value MUST be expressed as a number of minutes, and the value SHOULD be no less than 5 and no greater than 10080 (that is, one week).<5>

If the client supports this option but the option is not present, the client SHOULD use a default value of 480 (that is, 8 hours).

2.2.5 EFS User Template Name

Key: Software\Policies\Microsoft\Windows NT\CurrentVersion\EFS

Value: "TemplateName" or one of the special values in [MS-GPREG] section 3.2.5.1.

Type: REG_SZ.

Size: Equal to size of the **Data** field.

Data: A variable-length, null-terminated Unicode string. This setting specifies the **common name (CN)** of a **certificate template**, and is used by the EFS subsystem on the client for certificate enrollment requests. See section 3.2.5.1 and the corresponding section 3.1.4.1 of [MS-EFSR] for details of how and when this field is used by the client.

Implementations MAY<6> choose to support this option.

If the client supports this option but the option is not present, the client SHOULD use the default value "EFS".

2.2.6 EFS RSA Self-Signed Certificate Key Length

Key:**Software\Policies\Microsoft\Windows NT\CurrentVersion\EFS**

Value: "RSAKeyLength" or one of the special values in [MS-GPREG] section 3.2.5.1.

Type: REG_DWORD.

Size: Equal to size of the **Data** field.

Data: A 32-bit multiple of 8, representing the key length, in bits. This value SHOULD be no less than 1024 and no greater than 16384.<7>

This setting specifies the key length, in bits, that EFS must use when generating an RSA self-signed certificate. Such a certificate is generated when a user with no existing EFS keys attempts to create a new encrypted file or to convert an existing plain text file to encrypted form, and EFS fails to enroll the user for a suitable certificate from a certificate authority (CA).

Implementations MAY<8> choose to support this option. If this option is supported, the flag to disable self-signed certificates (defined as 0x00000004 in section 2.2.3) MUST be supported.

If the client supports this option but the option is not present, the client SHOULD use a default value of 2048.

2.2.7 EFS ECC Self-Signed Certificate Algorithm Identifier

Key: **Software\Policies\Microsoft\Windows NT\CurrentVersion\EFS**

Value: "SuiteBAlgorithm" or one of the special values in [MS-GPREG] section 3.2.5.1.

Type: REG_SZ.

Size: Equal to size of the **Data** field.

Data: A variable-length, null-terminated Unicode string. This setting specifies the algorithm that EFS must use when generating an ECC self-signed certificate. Such a certificate is generated when a user with no existing EFS keys attempts to create a new encrypted file or to convert an existing plain text file to encrypted form, and EFS fails to enroll the user for a suitable certificate from a certificate authority (CA).

Implementations MAY<9> choose to support this option. If this option is supported, the flag to disable self-signed certificates (defined as 0x00000004 in section 2.2.3) MUST be supported.

An implementation that supports this option MUST support the following identifiers.

Algorithm Identifier	Description
"ECDH_P256"	The 256-bit prime elliptic curve Diffie-Hellman key exchange algorithm.
"ECDH_P384"	The 384-bit prime elliptic curve Diffie-Hellman key exchange algorithm.
"ECDH_P521"	The 521-bit prime elliptic curve Diffie-Hellman Key exchange algorithm.

If the client supports this option but the option is not present, the client SHOULD use the default value "ECDH_P256".

3 Protocol Details

3.1 Administrative Plug-in Details

The administrative plug-in mediates between the user interface (UI) and a remote data store that contains the EFS policy settings. Its purpose is to receive EFS policy information from a UI and to write the EFS policy information to a remote data store.

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.

The EFS Group Policy administrative plug-in relies on a collection of settings specified in section 2.2 and stored as a Unicode configuration file ([MS-GPREG] section 2.2) at a remote storage location using the Group Policy: Core Protocol. The administrative plug-in treats these settings merely as a collection of name-value pairs.

The EFS Group Policy administrative plug-in reads in these settings from the remote storage location and displays them to an administrator through a UI.

An administrator can then use the UI to make further configuration changes and the EFS Group Policy administrative plug-in will make corresponding changes to the name-value pairs stored in the aforementioned Unicode configuration file.

This conceptual data can be implemented using a variety of techniques. An implementation can implement such data using any method. <10>

This protocol also includes one ADM element, **Administered GPO (Public)**, which is directly accessed from the Group Policy: Core Protocol, as specified in [MS-GPOL] section 3.3.1.3.

3.1.2 Timers

This protocol does not introduce any new timers.

3.1.3 Initialization

No additional initialization steps are needed.

3.1.4 Higher-Layer Triggered Events

The EFS Group Policy administrative plug-in is invoked when an administrator launches the user interface for editing Group Policy settings. The plug-in displays the current settings to the administrator, and when the administrator requests a change in settings, it updates the stored configuration appropriately as specified in section 2.2, after performing additional checks and actions as noted in this section.

When the administrator requests an update of the EFS recovery policy, the administrative plug-in MUST also create or update the EfsBlob entry as specified in section 2.2.1.

The administrative plug-in SHOULD<11> take measures in its UI to ensure that the user cannot unknowingly set the EFS user template name to an invalid value. If the implementation supports the flag requiring smart cards for EFS (specified in section 2.2.3) and if that option is configured to require smart cards for EFS, the administrative plug-in SHOULD ensure that this template is compatible with smart cards.

Implementations SHOULD<12> prevent users from configuring very low values for the EFS self-signed certificate key length (as specified in section 2.2.6), as short keys are insecure. Implementations MAY also restrict the maximum key length permitted.

3.1.5 Processing Events and Sequencing Rules

The EFS Group Policy administrative plug-in reads extension-specific data from the remote storage location and will then pass that information to a UI to display the current settings to an administrator.

It will also write the extension-specific configuration data to the remote storage location if the administrator makes any changes to the existing configuration.

Any additional entries in the configuration data that do not pertain to the configuration options specified in section 2.2, or that are not supported by the particular implementation, MUST be ignored by the plug-in. The plug-in MUST NOT overwrite, delete, or otherwise modify any settings that it does not support.

The EFS Group Policy administrative plug-in queries and persists these settings in the "registry.pol" **registry policy file** under the **computer-scoped Group Policy Object path**.

The EFS Group Policy administrative plug-in MUST invoke the following event to load the "registry.pol" file:

- Load Policy Settings event specified in [MS-GPREG] section 3.1.4.1

The EFS Group Policy administrative plug-in MUST invoke the following events to update the "registry.pol" file:

1. Update Policy Settings event specified in [MS-GPREG] section 3.1.4.2
2. Group Policy Extension Update event described in [MS-GPOL] section 3.3.4.4 with the following parameters:
   - "*GPO DN*" is set to the distinguished name of the **Administered GPO**
   - "*Is User Policy*" is set to FALSE
   - "*CSE GUID*" is set to the Group Policy: Encrypting File System *CSE GUID* (defined in [MS-GPEF] section 1.9)
   - "*TOOL GUID*" is set to the Group Policy: Encrypting File System *Tool extension GUID (computer policy settings)* (defined in [MS-GPEF] section 1.9)
3. Group Policy Extension Update event described in [MS-GPOL] section 3.3.4.4 with the following parameters:
   - "*GPO DN*" is set to the distinguished name of the **Administered GPO**
   - "*Is User Policy*" is set to FALSE
   - "*CSE GUID*" is set to the Group Policy: Registry Extension Encoding *CSE GUID* (defined in [MS-GPREG] section 1.9)

- "*TOOL GUID*" is set to the Group Policy: Encrypting File System *Tool extension GUID (computer policy settings)* (defined in [MS-GPEF] section 1.9)

In all cases, the <gpo path> is set to computer-scoped Group Policy Object path, and the settings contained in the "registry.pol" file are used for the Policy Setting State. No other policy files are accessed by this plug-in. The plug-in MUST use the registry policy file format specified in [MS-GPREG] section 2.2.1 to query and update the policy entries described in section 2.2 in the "registry.pol" file.

3.1.6 Timer Events

This protocol does not introduce any new timers.

3.1.7 Other Local Events

The administrative plug-in MAY<13> use other events to populate the EFS policy, especially the recovery policy, to minimize the probability of data loss.

3.2 Client Details

Clients of this protocol consume the settings specified using the administrative plug-in (section 3.1). These settings specify behavior for the EFS subsystem on the client. The client also provides a facility for higher-layer applications to bind a user to a certificate suitable for EFS.

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. 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.

This protocol uses the model outlined in section 3.2.1.1 of [MS-GPREG] to store and retrieve settings on the client. Settings defined by the administrative plug-in 3.1 are populated to a persistent generic database on the client by methods described in [MS-GPREG]. The client then queries the database using the key and value names outlined in sections 2.2.1 - 2.2.7 to retrieve the settings. Based on the data retrieved for these settings, the client modifies the internal state of the EFS subsystem to conform to the specified settings.

In addition to the collection of settings described above, public elements of the User-Certificate Binding ADM (exposed by [MS-EFSR] section 3.1.1.1) are directly accessed. The following public ADM elements are configured by the clients of [MS-GPEF]:

- **RequireV3Template** (exposed by [MS-EFSR] section 3.1.1.1)
- **DisallowV3Template** (exposed by [MS-EFSR] section 3.1.1.1)
- **RequireSmartCard** (exposed by [MS-EFSR] section 3.1.1.1)
- **TemplateName** (exposed by [MS-EFSR] section 3.1.1.1)

The listed elements are exposed by the ADM of [MS-EFSR], as specified in [MS-EFSR] section 3.1.1.1.

Public elements of the EFSRPC Server Control (exposed by [MS-EFSR] section 3.1.1.2) are also directly accessed. The following public ADM elements are configured by clients of [MS-GPEF]:

- **EfsDisabled** (exposed by [MS-EFSR] 3.1.1.2)

The listed element is exposed by the ADM of [MS-EFSR], as specified in [MS-EFSR] section 3.1.1.2.

3.2.2 Timers

This protocol does not introduce any new timers.

3.2.3 Initialization

No additional initialization steps are needed.

3.2.4 Higher-Layer Triggered Events

3.2.4.1 Process Group Policy

A client of [MS-GPEF] MAY<14> register an EFS extension plug-in component using the EFS extension's **CSE GUID**. If the client registers an EFS extension plug-in component, the Group Policy: Core Protocol launches the component by invoking the Process Group Policy event. The abstract interface for the Process Group Policy event is specified in [MS-GPOL] section 3.2.4.1.

3.2.5 Processing Events and Sequencing Rules

If a client of Group Policy: Encrypting File System Extension registers an EFS extension plug-in component, the client plug-in component that receives an updated collection of settings MUST only adjust the internal state of the EFS subsystem on the client, as specified in section 3.2.5.1. It MUST NOT participate in any network communication.

3.2.5.1 Receiving Updated Policy

When a client of [MS-GPEF] receives an updated collection of settings via the procedure in [MS-GPREG] section 3.2.4.1, it directly accesses the public User-Certificate Binding ADM elements ([MS-EFSR] section 3.1.1.1) and configures them in the following way:

- **RequireV3Template (Public)**: If the value of the **EfsOptions** field (section 2.2.3) is present in the client database and flag 0x00002000 is set, the client MUST set this value to True. Otherwise, this element is not modified.
- **DisallowV3Template (Public)**: If the value of the **EfsOptions** field (section 2.2.3) is present in the client database and flag 0x00001000 is set, the client MUST set this value to True. Otherwise, this element is not modified.
- **RequireSmartCard (Public)**: If the value of the **EfsOptions** field (section 2.2.3) is present in the client database and flag 0x00000100 is set, the client MUST set this value to True. Otherwise, this element is not modified.
- **TemplateName (Public)**: If the value of the **TemplateName** field (section 2.2.5) is present in the client database, the client MUST use the value from the database to set the **TemplateName (Public)** User-Certificate Binding ADM element ([MS-EFSR] section 3.1.1.1). Otherwise, this element is not modified.

A client of [MS-GPEF] also directly accesses the public EFSRPC Server Control ADM elements ([MS-EFSR] section 3.1.1.2) and configures them in the following way:

EfsDisabled (Public): If the value of the **EfsConfiguration** field (section 2.2.2) is present in the client database and equal to 0x00000001, the client SHOULD set this value to true. A client implementation MAY<15> use an alternative mechanism for configuring the **EfsDisabled** public ADM element.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

In the following example, an administrator sets up a new domain and wants to enable EFS use on the computers in the domain. The client computers run an operating system whose EFS implementation contains a system process that is initialized at startup and terminated at shutdown.

First, the administrator installs and configures an operating system on a computer that is intended to function as the DC. After taking the necessary steps to designate the computer as a DC and creating a user account with administrative privileges over the new domain, the administrator restarts the machine and logs on as the newly created user. At this point, the administrative plug-in is triggered: a new public-private key pair is generated, a self-signed X.509 certificate is created containing the public key, with its enhanced key usage extension set to the value denoting File Recovery. This certificate is written to the Group Policy configuration store as specified in section 2.2.1. An EfsBlob entry is also created that contains this certificate.

The administrator then launches the user interface for the administrative plug-in, and sets the status of EFS to Enabled. This causes the following entry to be written to the machine-specific Registry Policy file of the relevant GPO.

Key: **Software\Policies\Microsoft\Windows NT\CurrentVersion\EFS**

Value: "EfsConfiguration".

Type: REG_DWORD.

Size: Equal to size of the **Data** field.

Data: 0x00000000.

The administrator then adds client computers to this domain. The operating system used on these computers incorporates a long-running system process as part of its EFS implementation. This process monitors updates to Group Policy and reconfigures EFS accordingly when such an update is received. When a client computer is restarted for the first time after being added to the domain, it contacts the domain controller and reads Group Policy information as specified in [MS-GPOL]. As part of this process, a machine-specific registry policy file containing the following items is also downloaded:

- A set of values under the registry key, Software\Policies\Microsoft\SystemCertificates\EFS\Certificates, which represent the certificate created by the administrative plug-in as described earlier.
- The value EfsBlob under the registry key, Software\Policies\Microsoft\SystemCertificates, consisting of the certificate described earlier represented in the format specified in section 2.2.1.
- The registry value EfsConfiguration described earlier.

The Group Policy: Registry Extension Encoding on the client parses this file and adds the configuration information to the machine's registry.

The EFS client-side extension plug-in is then invoked. This plug-in signals the long-running EFS system process that its Group Policy settings have changed. The EFS process then reads the EfsBlob, verifies that it is consistent with the values stored under the HKLM\Software\Policies\Microsoft\SystemCertificates\EFS\Certificates registry key, and copies the EfsBlob value into an internal buffer so it will be used from that point on by the EFS routines manipulating the EFS file metadata. It also sets an internal variable to signify that EFS is enabled.

When a user logs on to the client, the desktop environment is configured to expose user interface elements that allow them to use EFS functionality. The user creates a new directory, marks it as encrypted, and creates a new file within it. An EFS routine is called to generate the metadata for this new file. It generates a symmetric key for encrypting the file contents, encrypts one copy of it with the user's public key, and another copy with the recovery certificate contained in the EfsBlob value that it has stored internally. These two encrypted copies of the key are stored in the file's EFS metadata, which is then written to disk.

5 Security

5.1 Security Considerations for Implementers

The Group Policy: Encrypting File System Extension sets the EFS recovery policy on the client computer. This policy consists of one or more public keys contained in X.509 certificates. Anyone who possesses any one of the associated private keys has the ability to decrypt all files that are encrypted or modified by any user on the client while the policy is in effect. Therefore, it is extremely important that implementers provide a means of protecting the integrity of the recovery policy against tampering, especially during its transfer from server to client. Ideally, this should be provided as part of the transport for the Group Policy: Core Protocol. The security method used is implementation-specific.

The Microsoft implementation of EFS uses RSA for public key cryptography. As of this writing, key sizes of 2,048 bits and higher are thought to provide adequate security for most applications. The EFS self-signed certificate key length option should support a large enough range of key sizes. Implementations should impose minimum limits on key length to ensure security.

The National Security Agency (NSA) has defined a set of cryptographic algorithms that are to be used for secure sharing of information. These algorithms are collectively referred to as "Suite B" ([NSA]). The Group Policy: Encrypting File System Extension includes settings that express use of Elliptic Curve Cryptography (ECC). Specifically, when option 0x00002000 is enabled in EFS Addition Options (section 2.2.3), an implementation that supports this option is expected to enforce the use of ECC for user and recovery certificates. When this setting is enabled, the Windows implementation of EFS restricts the algorithms allowed for new user and recovery certificates to ECC algorithms. Using this setting in conjunction with an appropriate EFS Recovery Policy (section 2.2.1), EFS User Template Name (section 2.2.5), and EFS ECC Self-Signed Certificate Algorithm Identifier (section 2.2.7), an administrator can configure a Windows implementation of EFS to use only algorithms allowed by Suite B for EFS certificates. If an administrator wants to configure EFS certificates in a manner conformant with Suite B by using the Windows implementation, it is the responsibility of the administrator to understand the conformant algorithms and to correspondingly configure the set of algorithms used, with the settings described above. Other implementations are not required to support algorithms included in Suite B.

5.2 Index of Security Parameters

There are no security parameters used by this extension.

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 2000 operating system
- Windows XP operating system
- Windows Server 2003 operating system
- Windows Server 2003 R2 operating system
- Windows Vista 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 2.2.1.2.2: Windows adds the SID whenever a user manually creates a certificate and key. When the DRA is created automatically as specified in section 3.1.7, no SID is added.

<2> Section 2.2.2: This setting is not supported in Windows 2000.

<3> Section 2.2.3: Windows Vista and Windows Server 2008 support only flags 0x00000001 through 0x00000400.

<4> Section 2.2.3: The Windows Vista and Windows Server 2008 implementations use flag 0x00000200 to enable encryption of the system page file by NTFS to avoid the security implications of unintended information transfer through old page file contents. The symmetric key for the encrypted page file is kept in memory at all times, effectively ensuring that the page file becomes unreadable when the system is powered off.

<5> Section 2.2.4: If **CacheTimeout** is set to a value less than 5 minutes, Windows behaves as though it were set to 5 minutes. If **CacheTimeout** is set to a value greater than 10080 minutes (1 week), Windows behaves as though it were set to 10080 minutes (1 week).

<6> Section 2.2.5: This field is supported by Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2.

<7> Section 2.2.6: If **RSAKeyLength** is set to a value less than 1024 or greater than 16384, Windows ignores that value and behaves as if **RSAKeyLength** were set to the default of 2048.

<8> Section 2.2.6: This field is supported by Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2.

<9> Section 2.2.7: This field is supported by Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2.

<10> Section 3.1.1:

The EFS configuration data is stored in registry keys of the managed computer as described in section 2.2.1 and its subsections. The EFS implementations in Windows 2000, Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, and Windows Server 2008 used code within LSA to handle this configuration data, and that code left a copy of the configuration data in the LSA policy database from which it is available to be accessed using LSAD [MS-LSAD]. Windows code does not use LSAD over the wire to access this data. Specifically, in these operating system versions:

- The "Encrypting File System (EFS) Policy Information" specified in [MS-LSAD] section 3.1.1.1 is updated to match the EfsBlob value (specified in section 2.2.1.2).
- The Windows EFS reads the "Encrypting File System (EFS) Policy Information" from the Local Security Authority store (as specified in [MS-LSAD]) on the local machine.

This use of LSA was never necessary and was removed to simplify the Windows implementation in Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2.

<11> Section 3.1.4: The Windows implementation queries Active Directory to obtain a list of available certificate templates and filters out those that are not suitable for use with EFS. The user is then asked to choose among the remaining templates.

<12> Section 3.1.4: The Windows implementation requires this value to be a power of 2 between 1024 and 16384, inclusive.

<13> Section 3.1.7: The first time an administrator logs on to a domain controller after domain creation, the Microsoft implementation creates a DRA by generating a certificate and key for that administrator. It then updates the default domain policy **Group Policy Object (GPO)** by writing the certificate into the EFS recovery policy as specified in section 2.2.1. This implementation-specific update to the default domain policy is identified by the tool extension GUID (specified in section 1.9) in the GPO attribute **gPCMachineExtensionNames**.

This update to the Default Domain Policy follows a similar sequence of events as defined in the "registry.pol" update sequence in section 3.1.5 except that in Steps 2 and 3 the **Administered GPO** is set to the Default Domain Policy GPO and the "*TOOL GUID*" is set to the *Tool extension GUID (default domain policy settings)* specified in section 1.9.

<14> Section 3.2.4.1: The implementations of Windows 2000, Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, and Windows Server 2008 register an EFS extension plug-in component.

<15> Section 3.2.5.1: The Windows 2000 implementation configures the **EfsDisabled** ADM element in the following way:

If the EFS Recovery Policy (section 2.2.1) is not present (that is, there is no recovery policy defined) in the client database or is present but the number of keys under the Certificates subkey defined in section 2.2.1.1 is zero (that is, the recovery policy is empty), then the client sets the **EfsDisabled** ADM element value to true.

7 Change Tracking

This section identifies changes that were made to the [MS-GPEF] protocol document between the February 2014 and May 2014 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.
- The removal of a document from the documentation set.

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 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 changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

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.

- 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 dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1.4 Relationship to Other Protocols	57588 Replaced references to SMB and SMB2 with file access services protocols.	N	Content updated.
2.1 Transport	57588 Replaced references to SMB/SMB2 with references to file access protocols.	N	Content updated.

8 Index