[MS-OVBA]:
Office VBA File Format Structure

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
06/27/2008	1.0		Initial Availability
08/15/2008	1.01	Editorial	Revised and edited the technical content
01/16/2009	1.02	Editorial	Revised and edited the technical content
07/13/2009	1.03	Major	Changes made for template compliance
08/28/2009	1.04	Editorial	Revised and edited the technical content
11/06/2009	1.05	Editorial	Revised and edited the technical content
02/19/2010	2.0	Minor	Updated the technical content
03/31/2010	2.01	Editorial	Revised and edited the technical content
04/30/2010	2.02	Editorial	Revised and edited the technical content
06/07/2010	2.03	Editorial	Revised and edited the technical content
06/29/2010	2.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.05	Editorial	Changed language and formatting in the technical content.
12/17/2010	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	2.6	Minor	Clarified the meaning of the technical content.
04/11/2012	2.6	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	2.6	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2012	2.7	Minor	Clarified the meaning of the technical content.
02/11/2013	2.7.1	Editorial	Changed language and formatting in the technical content.

Date	Revision History	Revision Class	Comments
07/30/2013	2.7.1	No change	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	2.7.1	No change	No changes to the meaning, language, or formatting of the technical content.
02/10/2014	2.7.1	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction 8
1.1 Glossary 8
1.2 References 9
1.2.1 Normative References 9
1.2.2 Informative References 10
1.3 Structure Overview (Synopsis) 10
1.3.1 Project Information 10
1.3.2 Project References 10
1.3.3 Project Items 11
1.3.4 Byte Ordering 11
1.4 Relationship to Protocols and Other Structures 12
1.5 Applicability Statement 12
1.6 Versioning and Localization 12
1.7 Vendor-Extensible Fields 13

2 Structures 14
2.1 Conventions 14
2.1.1 ABNF Rules 14
2.1.1.1 Common ABNF Rules 14
2.1.1.2 ANYCHAR 14
2.1.1.3 EQ 14
2.1.1.4 FLOAT 15
2.1.1.5 GUID 15
2.1.1.6 HEXINT32 15
2.1.1.7 INT32 15
2.1.1.8 LibidReference 15
2.1.1.9 ModuleIdentifier 16
2.1.1.10 NWLN 16
2.1.1.11 PATH 16
2.1.1.12 ProjectReference 17
2.1.1.13 QUOTEDCHAR 17
2.1.1.14 VBABOOL 17
2.1.1.15 VbaIdentifier 18
2.1.2 Pseudocode 18
2.2 File Structure 18
2.2.1 Project Root Storage 19
2.2.2 VBA Storage 19
2.2.3 _VBA_PROJECT Stream 19
2.2.4 dir Stream 19
2.2.5 Module Stream 19
2.2.6 SRP Streams 19
2.2.7 PROJECT Stream 19
2.2.8 PROJECTwm Stream 19
2.2.9 PROJECTlk Stream 20
2.2.10 Designer Storages 20
2.2.11 VBFrame Stream 20
2.3 Record Types 20
2.3.1 PROJECT Stream: Project Information 20
2.3.1.1 ProjectProperties 20
2.3.1.2 ProjectId 21

2.3.1.3 ProjectModule 21
2.3.1.4 ProjectDocModule 21
2.3.1.5 ProjectStdModule 21
2.3.1.6 ProjectClassModule 21
2.3.1.7 ProjectDesignerModule 22
2.3.1.8 ProjectPackage 22
2.3.1.9 ProjectHelpFile 22
2.3.1.10 ProjectExeName32 22
2.3.1.11 ProjectName 22
2.3.1.12 ProjectHelpId 23
2.3.1.13 ProjectDescription 23
2.3.1.14 ProjectVersionCompat32 23
2.3.1.15 ProjectProtectionState 23
2.3.1.16 ProjectPassword 24
2.3.1.17 ProjectVisibilityState 25
2.3.1.18 HostExtenders 25
2.3.1.19 ProjectWorkspace 26
2.3.1.20 ProjectWindowRecord 26
2.3.2 PROJECTlk Stream: ActiveX Control Information 27
2.3.2.1 LICENSEINFO Record 27
2.3.3 PROJECTwm Stream: Module Name Information 28
2.3.3.1 NAMEMAP Record 29
2.3.4 VBA Storage: Visual Basic for Applications Project Information 29
2.3.4.1 _VBA_PROJECT Stream: Version Dependent Project Information 29
2.3.4.2 dir Stream: Version Independent Project Information 30
2.3.4.2.1 PROJECTINFORMATION Record 30
2.3.4.2.1.1 PROJECTSYSKIND Record 32
2.3.4.2.1.2 PROJECTLCID Record 33
2.3.4.2.1.3 PROJECTLCIDINVOKE Record 33
2.3.4.2.1.4 PROJECTCODEPAGE Record 34
2.3.4.2.1.5 PROJECTNAME Record 34
2.3.4.2.1.6 PROJECTDOCSTRING Record 34
2.3.4.2.1.7 PROJECTHELPFILEPATH Record 35
2.3.4.2.1.8 PROJECTHELPCONTEXT Record 36
2.3.4.2.1.9 PROJECTLIBFLAGS Record 36
2.3.4.2.1.10 PROJECTVERSION Record 37
2.3.4.2.1.11 PROJECTCONSTANTS Record 37
2.3.4.2.2 PROJECTREFERENCES Record 38
2.3.4.2.2.1 REFERENCE Record 38
2.3.4.2.2.2 REFERENCENAME Record 39
2.3.4.2.2.3 REFERENCECONTROL Record 40
2.3.4.2.2.4 REFERENCEORIGINAL Record 42
2.3.4.2.2.5 REFERENCEREGISTERED Record 42
2.3.4.2.2.6 REFERENCEPROJECT Record 43
2.3.4.2.3 PROJECTMODULES Record 44
2.3.4.2.3.1 PROJECTCOOKIE Record 45
2.3.4.2.3.2 MODULE Record 45
2.3.4.2.3.2.1 MODULENAME Record 47
2.3.4.2.3.2.2 MODULENAMEUNICODE Record 47
2.3.4.2.3.2.3 MODULESTREAMNAME Record 47
2.3.4.2.3.2.4 MODULEDOCSTRING Record 48
2.3.4.2.3.2.5 MODULEOFFSET Record 49
2.3.4.2.3.2.6 MODULEHELPCONTEXT Record 49

2.3.4.2.3.2.7 MODULECOOKIE Record 50
2.3.4.2.3.2.8 MODULETYPE Record 50
2.3.4.2.3.2.9 MODULEREADONLY Record 51
2.3.4.2.3.2.10 MODULEPRIVATE Record 51
2.3.4.3 Module Stream: Visual Basic Modules 51
2.3.5 VBFrame Stream: Designer Information 52
2.3.5.1 DesignerProperties 52
2.3.5.2 DesignerCaption 53
2.3.5.3 DesignerHeight 53
2.3.5.4 DesignerLeft 53
2.3.5.5 DesignerTop 53
2.3.5.6 DesignerWidth 53
2.3.5.7 DesignerEnabled 54
2.3.5.8 DesignerHelpContextId 54
2.3.5.9 DesignerRTL 54
2.3.5.10 DesignerShowModal 54
2.3.5.11 DesignerStartupPosition 54
2.3.5.12 DesignerTag 55
2.3.5.13 DesignerTypeInfoVer 55
2.3.5.14 DesignerVisible 55
2.3.5.15 DesignerWhatsThisButton 55
2.3.5.16 DesignerWhatsThisHelp 55
2.4 Algorithms 56
2.4.1 Compression and Decompression 56
2.4.1.1 Structures 56
2.4.1.1.1 CompressedContainer 56
2.4.1.1.2 DecompressedBuffer 57
2.4.1.1.3 DecompressedChunk 57
2.4.1.1.4 CompressedChunk 58
2.4.1.1.5 CompressedChunkHeader 58
2.4.1.1.6 CompressedChunkData 59
2.4.1.1.7 TokenSequence 59
2.4.1.1.8 CopyToken 60
2.4.1.2 State Variables 60
2.4.1.3 Algorithms 61
2.4.1.3.1 Decompression Algorithm 61
2.4.1.3.2 Decompressing a CompressedChunk 61
2.4.1.3.3 Decompressing a RawChunk 62
2.4.1.3.4 Decompressing a TokenSequence 63
2.4.1.3.5 Decompressing a Token 63
2.4.1.3.6 Compression algorithm 64
2.4.1.3.7 Compressing a DecompressedChunk 64
2.4.1.3.8 Compressing a TokenSequence 65
2.4.1.3.9 Compressing a Token 66
2.4.1.3.10 Compressing a RawChunk 67
2.4.1.3.11 Byte Copy 68
2.4.1.3.12 Extract CompressedChunkSize 68
2.4.1.3.13 Pack CompressedChunkSize 68
2.4.1.3.14 Pack CompressedChunkSignature 69
2.4.1.3.15 Extract CompressedChunkFlag 69
2.4.1.3.16 Pack CompressedChunkFlag 69
2.4.1.3.17 Extract FlagBit 70
2.4.1.3.18 Set FlagBit 70

2.4.1.3.19 CopyToken Algorithms 70
2.4.1.3.19.1 CopyToken Help 71
2.4.1.3.19.2 Unpack CopyToken 72
2.4.1.3.19.3 Pack CopyToken 72
2.4.1.3.19.4 Matching 73
2.4.2 Contents Hash 74
2.4.3 Data Encryption 76
2.4.3.1 Encrypted Data Structure 76
2.4.3.2 Encryption 77
2.4.3.3 Decryption 78
2.4.4 Password Hash 80
2.4.4.1 Password Hash Data Structure 80
2.4.4.2 Encode Nulls 81
2.4.4.3 Decode Nulls 82
2.4.4.4 Password Hash Algorithm 82
2.4.4.5 Password Hash Validation 83

3 Structure Examples 84
3.1 VBA Storage Information Example 84
3.1.1 _VBA_PROJECT Example 84
3.1.2 dir Stream Example 84
3.1.2.1 Project Information Example 84
3.1.2.2 Project Reference Information Example 87
3.1.2.3 Module Information Example 95
3.1.2.3.1 PROJECT MODULES Example 95
3.1.2.3.2 Module Record Examples 96
3.1.2.3.2.1 ThisWorkbook Document Module Record Example 96
3.1.2.3.2.2 Sheet1 Document Module Record Example 98
3.1.2.3.2.3 UserForm1 Designer Module Record Example 100
3.1.3 ThisWorkbook Decompressed Module Stream Example 101
3.1.4 Sheet1 Decompressed Module Stream Example 102
3.1.5 UserForm1 Decompressed Module Stream Example 103
3.1.6 PROJECT Stream Example 103
3.1.7 VBFrame Stream Example 107
3.2 Compression/Decompression Examples 108
3.2.1 No Compression Example 108
3.2.2 Normal Compression Example 108
3.2.3 Maximum Compression Example 109

4 Security Considerations 110
4.1 Project Integrity Verification 110
4.2 Encryption Method 110

5 Appendix A: Product Behavior 111

6 Change Tracking 113

7 Index 114

1 Introduction

This document specifies the Office VBA File Format Structure. This file format applies to VBA projects. VBA projects are a collection of embedded macros and custom forms for use in Office documents that can be used to extend a host application to provide custom behavior. This specification describes a storage that contains a VBA project.

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

1.1 Glossary

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

big-endian
code page
GUID
language code identifier (LCID)
little-endian
Unicode
UTF-16

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

absolute path
ActiveX control
ActiveX control library
Automation server
Automation type library
class identifier (CLSID)
class module
CLSID
compilation constant
designer
designer module
digest
document module
embedded macro
floating-point number
hash
Help topic identifier
license key
MD5
module
multibyte character set (MBCS)
OLE compound file
procedural module
project package
reference
relative path
right-to-left
SHA-1
storage
stream

twiddled type library
twip
VBA
VBA environment
VBA host application
VBA identifier
VBA project

The following terms are specific to this document:

aggregatable server: A Component Object Model (COM) server that can be contained by another COM server and that allows its interfaces to be used as if they were defined by the server that contains it.

extended type library: A component that contains Automation standard descriptions of exposed objects, properties, and methods that are implemented by an aggregatable server and supplemented by another Automation server.

Help file: A file that contains the documentation for a specific product or technology.

host extender: An Automation type that is provided by a host application to extend the functionality of an Automation server.

parent window: A primary window that provides window management functionality for a set of child windows.

run length encoding: A lossless compression method that replaces a contiguous series (run) of identical values in a data stream with a pair of values that represent the length of the series and the value itself. For example, a data stream that contains 57 consecutive entries with the value "10" could replace them all with the shorter pair of values "57", "10".

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.

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, https://www2.opengroup.org/ogsys/catalog/c706

[MS-CFB] Microsoft Corporation, "Compound File Binary File Format".

[MS-OAUT] Microsoft Corporation, "OLE Automation Protocol".

[MS-OFORMS] Microsoft Corporation, "Office Forms Binary File Format(s)".

[MS-VBAL] Microsoft Corporation, "VBA Language Specification".

[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

[RFC3174] Eastlake III, D., and Jones, P., "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001, http://www.ietf.org/rfc/rfc3174.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

[MC-CPB] Microsoft Corporation, "Code Page Bitfields", http://msdn.microsoft.com/en-us/library/dd317754.aspx

[MS-DOC] Microsoft Corporation, "Word Binary File Format (.doc) Structure Specification".

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

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary".

[MS-OSHARED] Microsoft Corporation, "Office Common Data Types and Objects Structures".

[MS-XLS] Microsoft Corporation, "Excel Binary File Format (.xls) Structure".

[MS-XLSB] Microsoft Corporation, "Excel Binary File Format (.xlsb) Structure Specification".

1.3 Structure Overview (Synopsis)

This file format defines an instance of a **VBA project**. The file format structure is a collection of records that define the VBA project. Each record defines part of one of three aspects of the project: project information, project **references (1)**, and project items.

1.3.1 Project Information

Records providing project information about the VBA project itself are contained within the following five **streams (1)**:

- The _VBA_PROJECT Stream (section 2.3.4.1) provides basic information about the VBA project, including the version information required to load the remainder of the structure.
- Project Information (section 2.3.4.2.1) in the dir Stream (section 2.3.4.2) contains information such as the name of the VBA project and help information.
- Project Properties (section 2.3.1.1) in the PROJECT Stream (section 2.3.1) contain additional information about the VBA project.
- The PROJECTwm Stream (section 2.3.3) contains information for mapping **module** names between **multibyte character set (MBCS)** and **UTF-16**.
- The PROJECTlk Stream (section 2.3.2) contains information about **ActiveX controls** used throughout the VBA project.

1.3.2 Project References

Records within Reference Information (section 2.3.4.2.2) in the dir Stream (section 2.3.4.2) define references (1) to external resources that are used by the VBA project. Each **REFERENCE** (section

2.3.4.2.2.1) in Reference Information (section 2.3.4.2.2) corresponds to a reference (1) to an external resource that can interact via OLE Automation as described in [MS-OAUT].

The three types of external references are as follows:

- A **REFERENCECONTROL** (section 2.3.4.2.2.3) specifies a reference (1) to external ActiveX controls that are used by the VBA project.
- A **REFERENCEREGISTERED** (section 2.3.4.2.2.5) specifies a reference (1) to external **Automation type libraries** that are used by the VBA project.
- A **REFERENCEPROJECT** (section 2.3.4.2.2.6) specifies a reference (1) to external VBA projects that are used by the VBA project.

1.3.3 Project Items

The VBA project contains a series of project items for **embedded macros**. Each project item is defined by a combination of records. The five types of project items are as follows:

- A **project package** specifies a **designer** class that can be extended in a **designer module**.
- A **document module** specifies a module for embedded macros and programmatic access associated with a document.
- A **procedural module** specifies a module for embedded macros.
- A **class module** that specifies a module that defines a class.
- A designer module specifies a module for extending a designer.

The **PROJECT** Stream (section 2.3.1) provides the type of every project item.

Document modules, procedural modules, class modules, and designer modules are items that can contain source code as described in [MS-VBAL] and other user-configurable settings. Within the dir Stream (section 2.3.4.2), a **MODULE** Record (section 2.3.4.2.3.2) exists for each such project item, where the **MODULENAME** (section 2.3.4.2.3.2.1) is the same as each **<ModuleIdentifier>** in the **PROJECT** Stream (section 2.3.1).

ProjectDesignerModule (section 2.3.1.7) specifies a project item that extends a designer. In addition to the source code, a Designer Storage (section 2.2.10) named **MODULESTREAMNAME** (section 2.3.4.2.3.2.3) will be present in the Project Root Storage (section 2.2.1) which contains additional, designer-specific information about the project item. The **VBFrame** Stream (section 2.3.5) specifies the **VBA**-specific information about the designer.

1.3.4 Byte Ordering

Some computer architectures number bytes in a binary word from left to right, which is referred to as **big-endian**. The byte numbering used for packet diagrams in this specification is big-endian. Other architectures number the bytes in a binary word from right to left, which is referred to as **little-endian**. The byte numbering used for enumerations, objects, and records in this specification is little-endian.

Using big-endian and little-endian methods, the number 0x12345678 would be stored as shown in the following table:

Byte order	Byte 0	Byte 1	Byte 2	Byte 3
Big-endian	0x12	0x34	0x56	0x78
Little-endian	0x78	0x56	0x34	0x12

1.4 Relationship to Protocols and Other Structures

This file format specifies several streams and storages in an **OLE compound file** as described in [MS-CFB]. It is related to the structures defined in the following references:

- [MS-DOC] includes an application of Microsoft® Visual Basic® for Applications (VBA) for embedded macros.
- [MS-XLS] includes an application of VBA for embedded macros.
- [MS-XLSB] includes an application of VBA for embedded macros.
- [MS-OSHARED] contains an application of the hashing algorithm specified in section 2.4.2 for securing VBA for embedded macros.
- [MS-OFORMS] specifies ActiveX controls that can be embedded in VBA as designers.
- OLE Automation Protocol, as described in [MS-OAUT], that can be used to execute embedded macros in VBA.
- The VBA language, as described in [MS-VBAL], specifies the source code language that is used for embedded macros in this format.

1.5 Applicability Statement

This document specifies a persistence format for embedded macros within a host document, and is not appropriate for stand-alone use. Embedded macros permit programmatic customization for the applications that use this structure.

This persistence format provides interoperability with applications that create or read documents conforming to this structure<1>.

1.6 Versioning and Localization

This document covers versioning issues in the following areas:

- **Structure Versions:** There is only one version of the Office VBA File Format Structure.
- **Localization:** This structure defines no locale-specific processes or data.

This file format contains performance caches that are not interoperable between versions. A version identifier (_VBA_PROJECT_Stream.Version, section 2.2.3) is defined to keep track of the exact version that saved a VBA project. When this version number matches the version used by Office, performance caches, specified by _VBA_PROJECT.PerformanceCache (section 2.2.3), SRP Streams (section 2.2.6), and Module Stream.PeformanceCache (section 2.3.4.3), will be used instead of the interoperable representation within the file. To be interoperable, this version number must be set to 0xFFFF so that performance caches are ignored.

1.7 Vendor-Extensible Fields

This file format provides a mechanism for vendor extension through custom designers. For details on using designers, see VBFrame Stream (section 2.3.5). No mechanism is provided for generating a unique **class identifier (CLSID)** for a designer.

2 Structures

2.1 Conventions

This section uses the following conventions and common definitions for pseudocode and ABNF rule definitions.

2.1.1 ABNF Rules

This section specifies ABNF rules common throughout section 2.

2.1.1.1 Common ABNF Rules

The following ABNF rules are used by section 2 and are included for reference. For more information, see [RFC4234] Appendix B.

ABNF Syntax:

```
CR     = %x0D

DIGIT  = %x30-39

DQUOTE = %x22

HEXDIG = DIGIT / "A" / "B" / "C" / "D" / "E" / "F"

HTAB   = %x09

LF     = %x0A

SP     = %x20

VCHAR  = %x21-7E

WSP    = SP / HTAB
```

2.1.1.2 ANYCHAR

Specifies any character value that is not a carriage-return, line-feed, or null.

ABNF syntax:

```
ANYCHAR = %x01-09 / %x0B / %x0C / %x0E-FF
```

2.1.1.3 EQ

Defines syntax for separating a property name from a value.

ABNF syntax:

```
EQ = *WSP "=" *WSP
```

2.1.1.4 FLOAT

Specifies a **floating-point number**.

ABNF syntax:

```
FLOAT = [SIGN] ( ( 1*DIGIT "."  1*DIGIT [EXP] ) /
        ( "."  1*DIGIT [EXP] ) /
        ( 1*DIGIT ["."] [EXP] ) )

EXP   = "e"  [SIGN]  1*DIGIT

SIGN  = "+"  / "-"
```

2.1.1.5 GUID

Specifies a **GUID**.

ABNF syntax:

```
GUID = "{" 8HEXDIG "-" 4HEXDIG "-" 4HEXDIG "-" 4HEXDIG "-" 12HEXDIG "}"
```

2.1.1.6 HEXINT32

Specifies a hexadecimal-encoded signed integer. MUST be between −2147483648 and 2147483647.

ABNF syntax:

```
HEXINT32 = "&H" 8HEXDIG
```

2.1.1.7 INT32

Specifies a signed integer. MUST be between −2147483648 and 2147483647.

ABNF syntax:

```
INT32 = ["-"] 1*DIGIT
```

2.1.1.8 LibidReference

Specifies the identifier of an Automation type library.

ABNF syntax:

```
LibidReference     = "*\" LibidReferenceKind LibidGuid
                     "#" LibidMajorVersion "." LibidMinorVersion
                     "#" LibidLcid
                     "#" LibidPath
                     "#" LibidRegName

LibidReferenceKind = %x47 / %x48
```

```
LibidGuid          = GUID

LibidMajorVersion  = 1*4HEXDIG

LibidMinorVersion  = 1*4HEXDIG

LibidLcid          = 1*8HEXDIG

LibidPath          = *(%x01-22 / %x24-FF)

LibidRegName       = *255(%x01-FF)
```

<LibidReferenceKind>:

Value	Meaning
%x47	<LibidPath> specifies a Windows file path.
%x48	<LibidPath> specifies a Macintosh path.

<LibidGuid>: The GUID of the Automation type library.

<MajorVersion>: An unsigned integer that specifies the major version of the Automation type library.

<LibidMinorVersion>: An unsigned integer that specifies the minor version of the Automation type library.

<LibidLcid>: The **LCID** of the Automation type library.

<LibidPath>: The path to the Automation type library.

<LibidRegName>: The Automation type library's display name.

2.1.1.9 ModuleIdentifier

Specifies the name of a module. SHOULD be an identifier as specified by [MS-VBAL]. MAY<2> be any string of characters. MUST be less than or equal to 31 characters long.

2.1.1.10 NWLN

Specifies a new line.

ABNF syntax:

```
NWLN = (CR LF) / (LF CR)
```

2.1.1.11 PATH

An array of characters that specifies a path to a file. MUST be less than 260 characters.

ABNF syntax:

```
PATH = DQUOTE *259QUOTEDCHAR DQUOTE
```

2.1.1.12 ProjectReference

Specifies the identifier of a VBA project.

ABNF syntax:

```
ProjectReference = "*\" ProjectKind ProjectPath

ProjectKind      = %x41-44

ProjectPath      = *(%x01-FF)
```

<ProjectKind>:

Value	Meaning
%x41	The referenced VBA project is standalone and <ProjectPath> specifies a Windows file path.
%x42	The referenced VBA project is standalone and <ProjectPath> specifies a Macintosh path.
%x43	The referenced VBA project is embedded and <ProjectPath> specifies a Windows file path.
%x44	The referenced VBA project is embedded and <ProjectPath> specifies a Macintosh path.

<ProjectPath>: The path to the VBA project.

2.1.1.13 QUOTEDCHAR

Specifies a single character.

ABNF syntax:

```
QUOTEDCHAR = WSP / NQCHAR / ( DQUOTE DQUOTE )

NQCHAR     = %x21 / %x23-FF
```

<DQUOTE DQUOTE>: Specifies a single double-quotation (") character.

2.1.1.14 VBABOOL

Specifies a Boolean value.

Value	Meaning
"0"	FALSE
"-1"	TRUE

ABNF syntax:

```
VBABOOL = "0" / "-1"
```

2.1.1.15 VbaIdentifier

Specifies a VBA Language identifier as specified by [MS-VBAL].

2.1.2 Pseudocode

All array indexing in pseudocode in this document is zero-based.

2.2 File Structure

Specifies a VBA project and contained project items. All data is stored in a structured storage as specified in [MS-CFB]. The **storages** and streams MUST be organized according to a hierarchy rooted at the Project Root Storage (section 2.2.1) as depicted in the following figure.

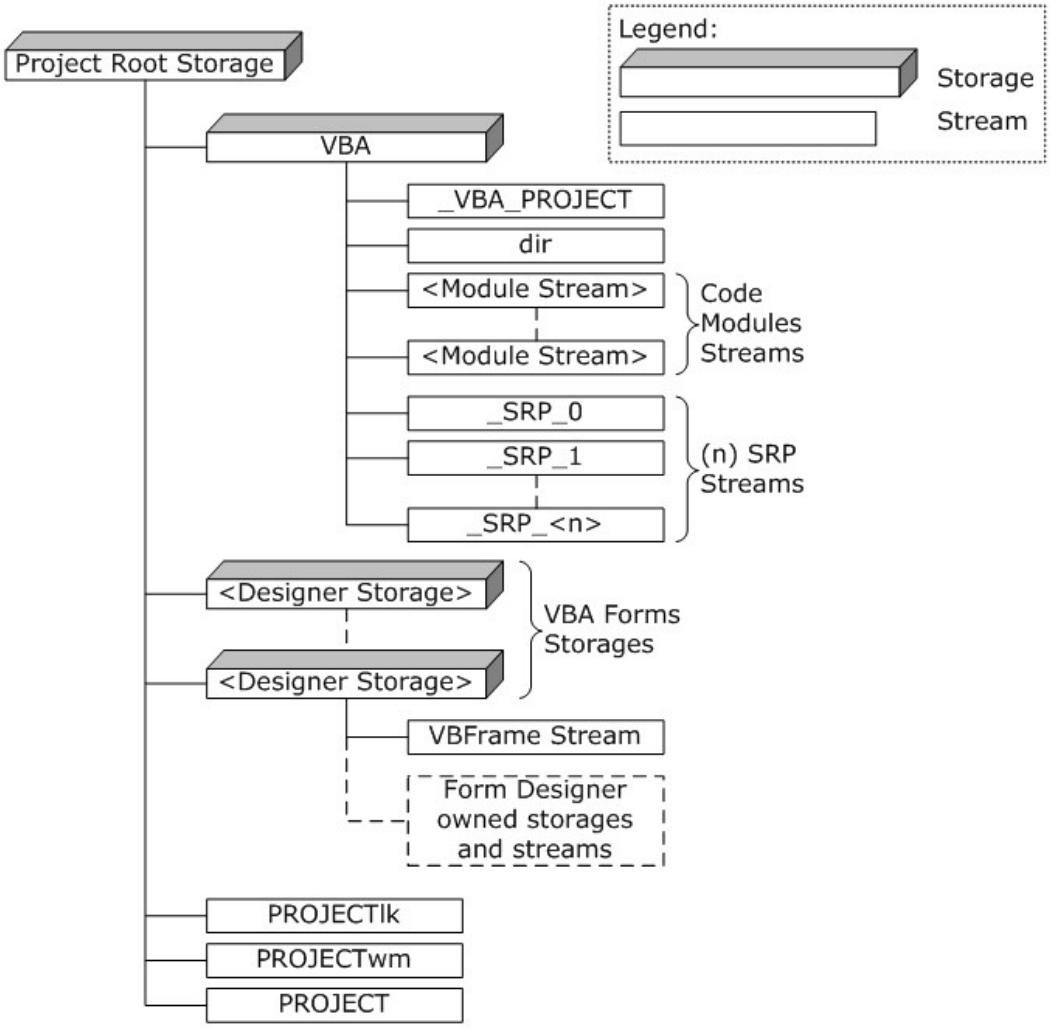


Figure 1: VBA storage hierarchy

2.2.1 Project Root Storage

A single root storage. MUST contain VBA Storage (section 2.2.2) and PROJECT Stream (section 2.2.7). Optionally contains **PROJECTwm** Stream (section 2.2.8), **PROJECTlk** Stream (section 2.2.9), and Designer Storages (section 2.2.10).

2.2.2 VBA Storage

A storage that specifies VBA project and module information. MUST have the name "VBA" (case-insensitive). MUST contain **_VBA_PROJECT** Stream (section 2.3.4.1) and dir Stream (section 2.3.4.2). MUST contain a Module Stream (section 2.2.5) for each module in the VBA project. Optionally contains **SRP** Streams (section 2.2.6).

2.2.3 _VBA_PROJECT Stream

A stream (1) that specifies the version-dependent project information. MUST have the name "_VBA_PROJECT" (case-insensitive). MUST contain data as specified by **_VBA_PROJECT** Stream (section 2.3.4.1).

2.2.4 dir Stream

A stream (1) that specifies VBA project properties, project references (1), and module properties. MUST have the name "dir" (case-insensitive). MUST contain data as specified by **dir** Stream (section 2.3.4.2).

2.2.5 Module Stream

A stream (1) that specifies the source code of modules in the VBA project. The name of this stream is specified by **MODULESTREAMNAME** (section 2.3.4.2.3.2.3). MUST contain data as specified by Module Stream (section 2.3.4.3).

2.2.6 SRP Streams

Streams (1) that specify an implementation-specific and version-dependent performance cache. MUST be ignored on read. MUST NOT be present on write.

The name of each of these streams is specified by the following ABNF grammar:

```
SRPStreamName = "__SRP_" 1*25DIGIT
```

2.2.7 PROJECT Stream

A stream (1) that specifies VBA project properties. MUST have the name "PROJECT" (case-insensitive). MUST contain data as specified by **PROJECT** Stream (section 2.3.1).

2.2.8 PROJECTwm Stream

A stream (1) that specifies names of modules represented in both MBCS and UTF-16 encoding. MUST have the name "PROJECTwm" (case-insensitive). MUST contain data as specified by **PROJECTwm** Stream (section 2.3.3).

2.2.9 PROJECTlk Stream

A stream (1) that specifies license information for ActiveX controls used in the VBA project. MUST have the name "PROJECTlk" (case-insensitive). MUST contain data as specified by **PROJECTlk** Stream (section 2.3.2).

2.2.10 Designer Storages

A designer storage MUST be present for each designer module in the VBA project. The name is specified by **MODULESTREAMNAME** (section 2.3.4.2.3.2.3). MUST contain **VBFrame** Stream (section 2.3.5). If the designer is an Office Form ActiveX control, then this storage MUST contain storages and streams (1) as specified by [MS-OFORMS] section 2.

2.2.11 VBFrame Stream

A stream (1) that specifies designer module properties. MUST contain data as specified by **VBFrame** Stream (section 2.3.5). Name of this stream MUST start with the UTF-16 character 0x0003 followed by the UTF-16 string "VBFrame" (case-insensitive).

2.3 Record Types

2.3.1 PROJECT Stream: Project Information

The PROJECT stream (1) specifies properties of the VBA project.

This stream is an array of bytes that specifies properties of the VBA project. MUST contain MBCS characters encoded using the **code page** specified in **PROJECTCODEPAGE** (section 2.3.4.2.1.4).

ABNF syntax:

```
VBAPROJECTText     = ProjectProperties NWLN
                     HostExtenders
                    [NWLN ProjectWorkspace]
```

2.3.1.1 ProjectProperties

Specifies project-wide properties.

ABNF syntax:

```
ProjectProperties  = ProjectId
                    *ProjectItem
                    [ProjectHelpFile]
                    [ProjectExeName32]
                     ProjectName
                     ProjectHelpId
                    [ProjectDescription]
                    [ProjectVersionCompat32]
                     ProjectProtectionState
                     ProjectPassword
                     ProjectVisibilityState

ProjectItem        = ( ProjectModule /
                       ProjectPackage ) NWLN
```

2.3.1.2 ProjectId

Specifies the class identifier (CLSID) for the VBA project.

ABNF syntax:

```
ProjectId          = "ID=" DQUOTE ProjectCLSID DQUOTE NWLN
ProjectCLSID       = GUID
```

<ProjectCLSID>: Specifies the class identifier (CLSID) of the VBA project's Automation type library. MUST be "{00000000-0000-0000-0000-000000000000}" when **ProjectPassword** (section 2.3.1.16) specifies a password **hash**.

2.3.1.3 ProjectModule

Specifies a module that contains VBA language source code as specified in [MS-VBAL].

ABNF syntax:

```
ProjectModule      = ( ProjectDocModule /
                       ProjectStdModule /
                       ProjectClassModule /
                       ProjectDesignerModule )
```

<ProjectModule>: Specifies the name and type of a specific module. MUST have a corresponding MODULE Record (section 2.3.4.2.3.2) in the dir Stream (section 2.3.4.2).

2.3.1.4 ProjectDocModule

Specifies a module that extends a document module.

ABNF syntax:

```
ProjectDocModule   = "Document=" ModuleIdentifier %x2f DocTlibVer

DocTlibVer         = HEXINT32
```

<DocTlibVer>: Specifies the document module's **Automation server** version as specified by [MS-OAUT].

2.3.1.5 ProjectStdModule

Specifies a procedural module.

ABNF syntax:

```
ProjectStdModule   = "Module=" ModuleIdentifier
```

2.3.1.6 ProjectClassModule

Specifies a class module.

ABNF syntax:

```
ProjectClassModule = "Class=" ModuleIdentifier
```

2.3.1.7 ProjectDesignerModule

Specifies a designer module.

ABNF syntax:

```
ProjectDesignerModule = "BaseClass=" ModuleIdentifier
```

2.3.1.8 ProjectPackage

Specifies the class identifier (CLSID) for a designer extended by one or more modules.

ABNF syntax:

```
ProjectPackage     = "Package=" GUID
```

2.3.1.9 ProjectHelpFile

Specifies a path to a **Help file** associated with this VBA project. MUST be the same value as specified in **PROJECTHELPFILEPATH** (section 2.3.4.2.1.7). MUST be present if **PROJECTHELPFILEPATH** specifies a value.

ABNF syntax:

```
ProjectHelpFile    = "HelpFile=" PATH NWLN
```

2.3.1.10 ProjectExeName32

Specifies a path. MUST be ignored.

ABNF syntax:

```
ProjectExeName32   = "ExeName32=" PATH NWLN
```

2.3.1.11 ProjectName

Specifies the short name of the VBA project.

ABNF syntax:

```
ProjectName        = "Name=" DQUOTE ProjectIdentifier DQUOTE NWLN

ProjectIdentifier  = 1*128QUOTEDCHAR
```

<ProjectIdentifier>: Specifies the name of the VBA project. MUST be less than or equal to 128 characters long. MUST be the same value as specified in **PROJECTNAME** (section 2.3.4.2.1.5). SHOULD be an identifier as specified by [MS-VBAL]. MAY<3> be any string of characters.

2.3.1.12 ProjectHelpId

Specifies a **Help topic identifier** in **ProjectHelpFile** (section 2.3.1.9) associated with this VBA project.

ABNF syntax:

```
ProjectHelpId      = "HelpContextID=" DQUOTE TopicId DQUOTE NWLN

TopicId            = INT32
```

<TopicId>: Specifies a Help topic identifier. MUST be the same value as specified in **PROJECTHELPCONTEXT** (section 2.3.4.2.1.8).

2.3.1.13 ProjectDescription

Specifies the description of the VBA project.

ABNF syntax:

```
ProjectDescription = "Description=" DQUOTE DescriptionText DQUOTE NWLN

DescriptionText    = *2000QUOTEDCHAR
```

<DescriptionText>: MUST be the same value as specified in **PROJECTDOCSTRING** (section 2.3.4.2.1.6).

2.3.1.14 ProjectVersionCompat32

Specifies the storage format version of the VBA project. MAY be missing<4>.

ABNF syntax:

```
ProjectVersionCompat32 = "VersionCompatible32=" DQUOTE "393222000" DQUOTE
                         NWLN
```

2.3.1.15 ProjectProtectionState

Specifies whether access to the VBA project was restricted by the user, the **VBA host application**, or the VBA project editor.

ABNF syntax:

```
ProjectProtectionState   = "CMG=" DQUOTE EncryptedState DQUOTE NWLN

EncryptedState           = 22*28HEXDIG
```

<EncryptedState>: Specifies whether access to the VBA project was restricted by the user, the VBA host application, or the VBA project editor, obfuscated by Data Encryption (section 2.4.3.2).

The **Data** parameter for Data Encryption (section 2.4.3.2) SHOULD be four bytes that specify the protection state of the VBA project. MAY<5> be 0x00000000. The **Length** parameter for Data Encryption (section 2.4.3.2) MUST be 4.

Values for **Data** are defined by the following bits:

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
A	B	C	Reserved																												

A - fUserProtected (1 bit): Specifies whether the user elected to protect the VBA project.

B - fHostProtected (1 bit): Specifies whether the VBA host application elected to protect the VBA project.

C - fVBEProtected (1 bit): Specifies whether the VBA project editor elected to protect the VBA project.

Reserved (29 bits): MUST be 0. MUST be ignored.

2.3.1.16 ProjectPassword

Specifies the password hash of the VBA project.

The syntax of ProjectPassword is defined as follows.

```
ProjectPassword     = "DPB=" DQUOTE EncryptedPassword DQUOTE NWLN

EncryptedPassword   = 16*HEXDIG
```

<EncryptedPassword>: Specifies the password protection for the VBA project.

A VBA project without a password MUST use 0x00 for the **Data** parameter for Data Encryption (section 2.4.3.2) and the **Length** parameter MUST be 1.

A VBA project with a password SHOULD specify the password hash of the VBA project, obfuscated by Data Encryption (section 2.4.3.2). The **Data** parameter for Data Encryption (section 2.4.3.2) MUST be an array of bytes that specifies a Hash Data Structure (section 2.4.4.1) and the **Length** parameter for Data Encryption MUST be 29. The Hash Data Structure (section 2.4.4.1) specifies a hash key and password hash encoded to remove null bytes as specified by section 2.4.4.

A VBA project with a password MAY<6> specify the plain text password of the VBA project, obfuscated by Data Encryption (section 2.4.3.2). In this case, the **Data** parameter Data Encryption (section 2.4.3.2) MUST be an array of bytes that specifies a null terminated password string encoded using MBCS using the code page specified by **PROJECTCODEPAGE** (section 2.3.4.2.1.4), and a **Length** parameter equal to the number of bytes in the password string including the terminating null character.

When the data specified by **<EncryptpedPassword>** is a password hash, ProjectId**.ProjectCLSID** (section 2.3.1.2) MUST be "{00000000-0000-0000-0000-000000000000}".

2.3.1.17 ProjectVisibilityState

Specifies whether the VBA project is visible.

ABNF syntax:

```
ProjectVisibilityState      = "GC=" DQUOTE
                              EncryptedProjectVisibility
                              DQUOTE NWLN

EncryptedProjectVisibility  = 16*22HEXDIG
```

<EncryptedProjectVisibility>: Specifies whether the VBA project is visible, obfuscated by Data Encryption (section 2.4.3.2).

The **Data** parameter for Data Encryption (section 2.4.3.2) is one byte that specifies the visibility state of the VBA project. The **Length** parameter for Data Encryption (section 2.4.3.2) MUST be 1.

Values for **Data** are:

Value	Meaning
0x00	VBA project is NOT visible. **<ProjectProtectionState>**.**fVBEProtected** (section 2.3.1.15) MUST be TRUE.
0xFF	VBA project is visible.

The default is 0xFF.

2.3.1.18 HostExtenders

Specifies a list of **host extenders**.

ABNF syntax:

```
HostExtenders    =  "[Host Extender Info]" NWLN
                        *HostExtenderRef

HostExtenderRef =  ExtenderIndex  "=" ExtenderGuid ";"
                      LibName ";" CreationFlags NWLN

ExtenderIndex    = HEXINT32

ExtenderGuid     = GUID

LibName          = "VBE" / *(%x21-3A / %x3C-FF)

CreationFlags    = HEXINT32
```

<HostExtenderRef>: Specifies a reference (1) to an **aggregatable server**'s Automation type library.

<ExtenderIndex>: Specifies the index of the host extender entry. MUST be unique to the list of HostExtenders.

<ExtenderGuid>: Specifies the GUID of the Automation type library to extend.

<LibName>: Specifies a host-provided Automation type library name. "VBE" specifies a built in name for the VBA Automation type library.

<CreationFlags>: Specifies a host-provided flag as follows:

Value	Meaning
0x00000000	MUST NOT create a new **extended type library** for the aggregatable server if one is already available to the **VBA environment**.
0x00000001	MUST create a new extended type library for the aggregatable server.

2.3.1.19 ProjectWorkspace

Specifies a list of module editor window states.

ABNF syntax:

```
ProjectWorkspace = "[Workspace]" NWLN
                   *ProjectWindowRecord
```

2.3.1.20 ProjectWindowRecord

Specifies the coordinates and state of a module editor window.

ABNF syntax:

```
ProjectWindowRecord = ModuleIdentifier "=" ProjectWindowState NWLN

ProjectWindowState  = CodeWindow [ ", " DesignerWindow ]

CodeWindow          = ProjectWindow

DesignerWindow      = ProjectWindow

ProjectWindow       = WindowLeft ", "
                      WindowTop ", "
                      WindowRight ", "
                      WindowBottom ", "
                      WindowState

WindowLeft          = INT32

WindowTop           = INT32

WindowRight         = INT32

WindowBottom        = INT32

WindowState         = ["C"] ["Z"] ["I"]
```

<ModuleIdentifier>: Specifies the name of the module. MUST have a corresponding **ProjectModule** (section 2.3.1.3).

<CodeWindow>: Specifies the coordinates and the state of a window used to edit the source code of a module.

<DesignerWindow>: Specifies the coordinates and the state of a window used to edit the designer associated with a module.

<WindowLeft>: Specifies the distance of the left edge of a window relative to a **parent window**.

<WindowTop>: Specifies the distance of the top edge of a window relative to a parent window.

<WindowRight>: Specifies the distance of the right edge of a window relative to a parent window.

<WindowBottom>: Specifies the distance of the bottom edge of a window relative to a parent window.

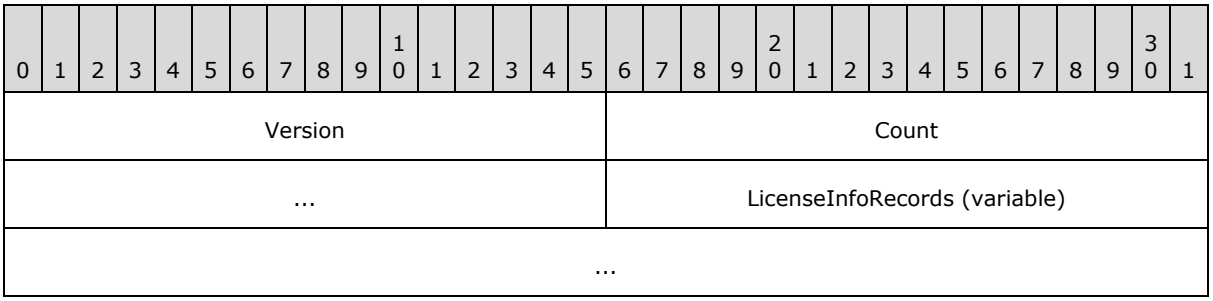
<WindowState>: Specifies the window state.

Values are defined as follows:

Value	Meaning
C	Closed.
Z	Zoomed to fill the available viewing area.
I	Minimized to an icon.

2.3.2 PROJECTlk Stream: ActiveX Control Information

Specifies license information for ActiveX controls.



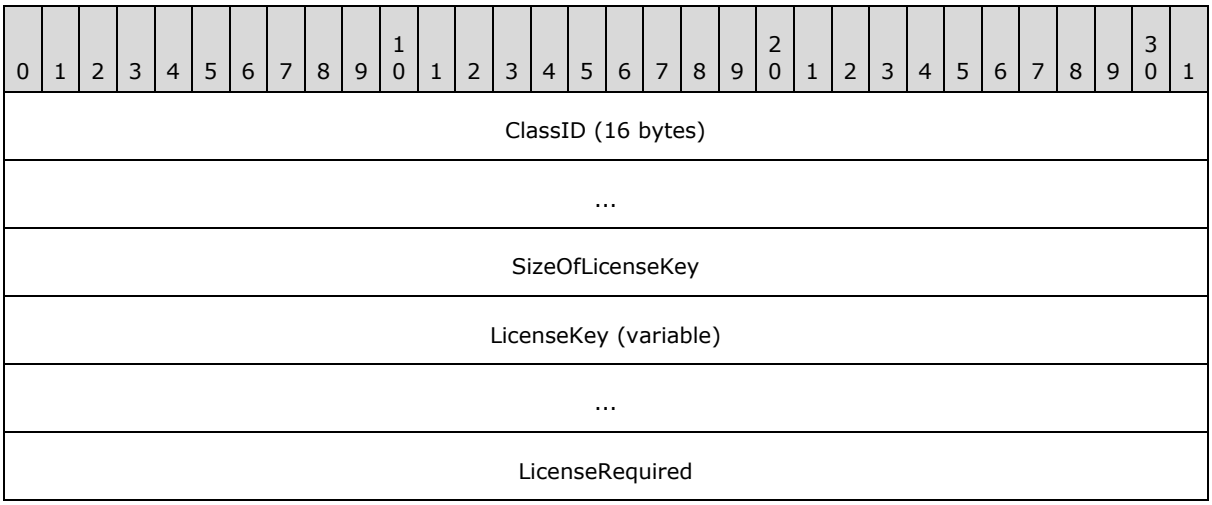
Version (2 bytes): An unsigned integer that specifies the version of this structure. MUST be 0x0001.

Count (4 bytes): An unsigned integer that specifies the number of elements in **LicenseInfoRecords**.

LicenseInfoRecords (variable): An array of **LICENSEINFO** (section 2.3.2.1).

2.3.2.1 LICENSEINFO Record

Specifies the information saved for each ActiveX control in the VBA project.



ClassID (16 bytes): A GUID that specifies the class identifier (CLSID) of an ActiveX control.

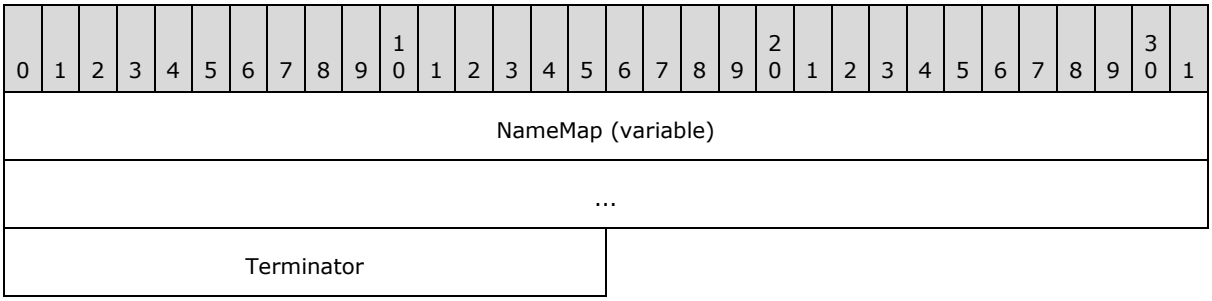
SizeOfLicenseKey (4 bytes): An unsigned integer that specifies the length of **LicenseKey** in bytes.

LicenseKey (variable): An array of **SizeOfLicenseKey** bytes that specifies the **license key** for the ActiveX control.

LicenseRequired (4 bytes): An unsigned integer that specifies a Boolean value. Specifies that the ActiveX control can be instantiated only by using a license-aware object creation method. SHOULD be 0x00000001 when the value of **SizeOfLicenseKey** is not zero. Otherwise SHOULD be 0x00000000<7>.

2.3.3 PROJECTwm Stream: Module Name Information

Specifies a map from MBCS module names to **Unicode** module names.

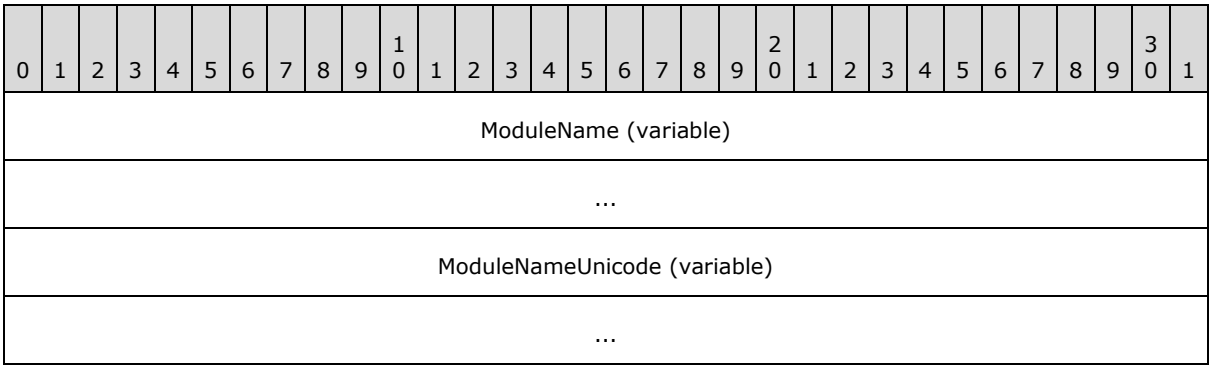


NameMap (variable): An array of **NAMEMAP** Record (section 2.3.3.1). The length of **NameMap** MUST be two bytes less than the size of the **PROJECTwm** Stream (section 2.2.8). Array items MUST appear in the same order as they appear in the **PROJECTMODULES** Record (section 2.3.4.2.3).

Terminator (2 bytes): An unsigned integer that specifies the end of the stream. MUST be 0x0000.

2.3.3.1 NAMEMAP Record

Maps a MBCS module name to a Unicode module name.



ModuleName (variable): A null-terminated string that specifies a module name. MUST contain MBCS characters encoded using the code page specified by **PROJECTCODEPAGE** (section 2.3.4.2.1.4). MUST match a module name specified by **MODULENAME** (section 2.3.4.2.3.2.1). The first byte MUST NOT be 0x00.

ModuleNameUnicode (variable): A null-terminated string that specifies a module name. MUST contain UTF-16 encoded characters. The first two bytes MUST NOT be 0x0000. MUST contain the UTF-16 encoding of **ModuleName**.

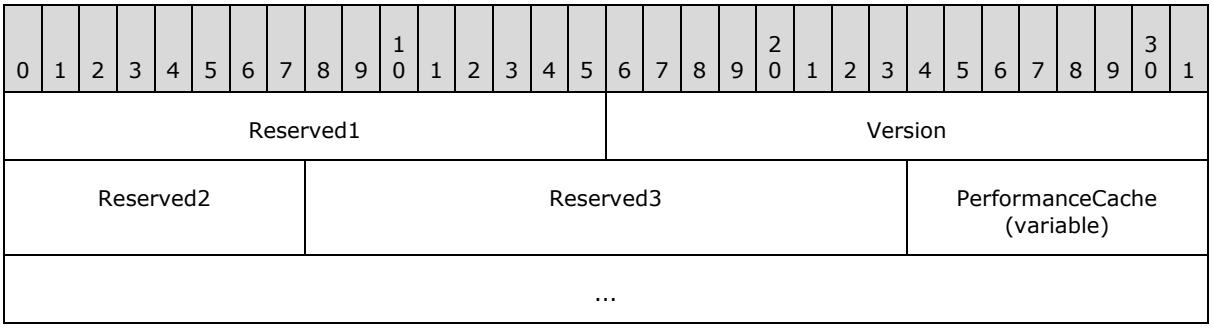
2.3.4 VBA Storage: Visual Basic for Applications Project Information

The VBA storage contains the **_VBA_PROJECT** Stream (section 2.3.4.1), the **dir** Stream (section 2.3.4.2), and **Module** Streams (section 2.3.4.3) for the VBA project. It also contains optional **SRP** Streams (section 2.2.6) that MUST be ignored.

2.3.4.1 _VBA_PROJECT Stream: Version Dependent Project Information

The _VBA_PROJECT stream contains the version-dependent description of a VBA project.

The first seven bytes of the stream are version-independent and therefore can be read by any version.



Reserved1 (2 bytes): MUST be 0x61CC. MUST be ignored.

Version (2 bytes): An unsigned integer that specifies the version of VBA used to create the VBA project. MUST be ignored on read. MUST be 0xFFFF on write.

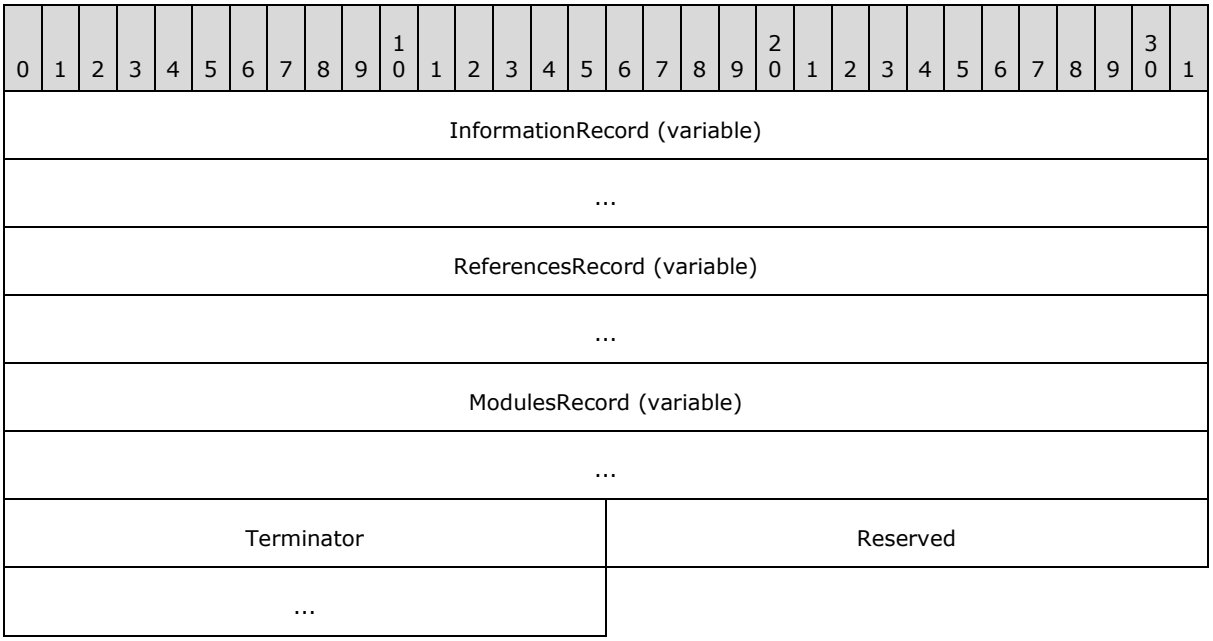
Reserved2 (1 byte): MUST be 0x00. MUST be ignored.

Reserved3 (2 bytes): Undefined. MUST be ignored.

PerformanceCache (variable): An array of bytes that forms an implementation-specific and version-dependent performance cache for the VBA project. The length of **PerformanceCache** MUST be seven bytes less than the size of **_VBA_PROJECT** Stream (section 2.3.4.1). MUST be ignored on read. MUST not be present on write.

2.3.4.2 dir Stream: Version Independent Project Information

The dir stream contains a series of bytes that specifies information for the VBA project, including project information, project references (1), and modules. The entire stream MUST be compressed as specified in Compression (section 2.4.1).



InformationRecord (variable): A **PROJECTINFORMATION** Record (section 2.3.4.2.1).

ReferencesRecord (variable): A **PROJECTREFERENCES** Record (section 2.3.4.2.2).

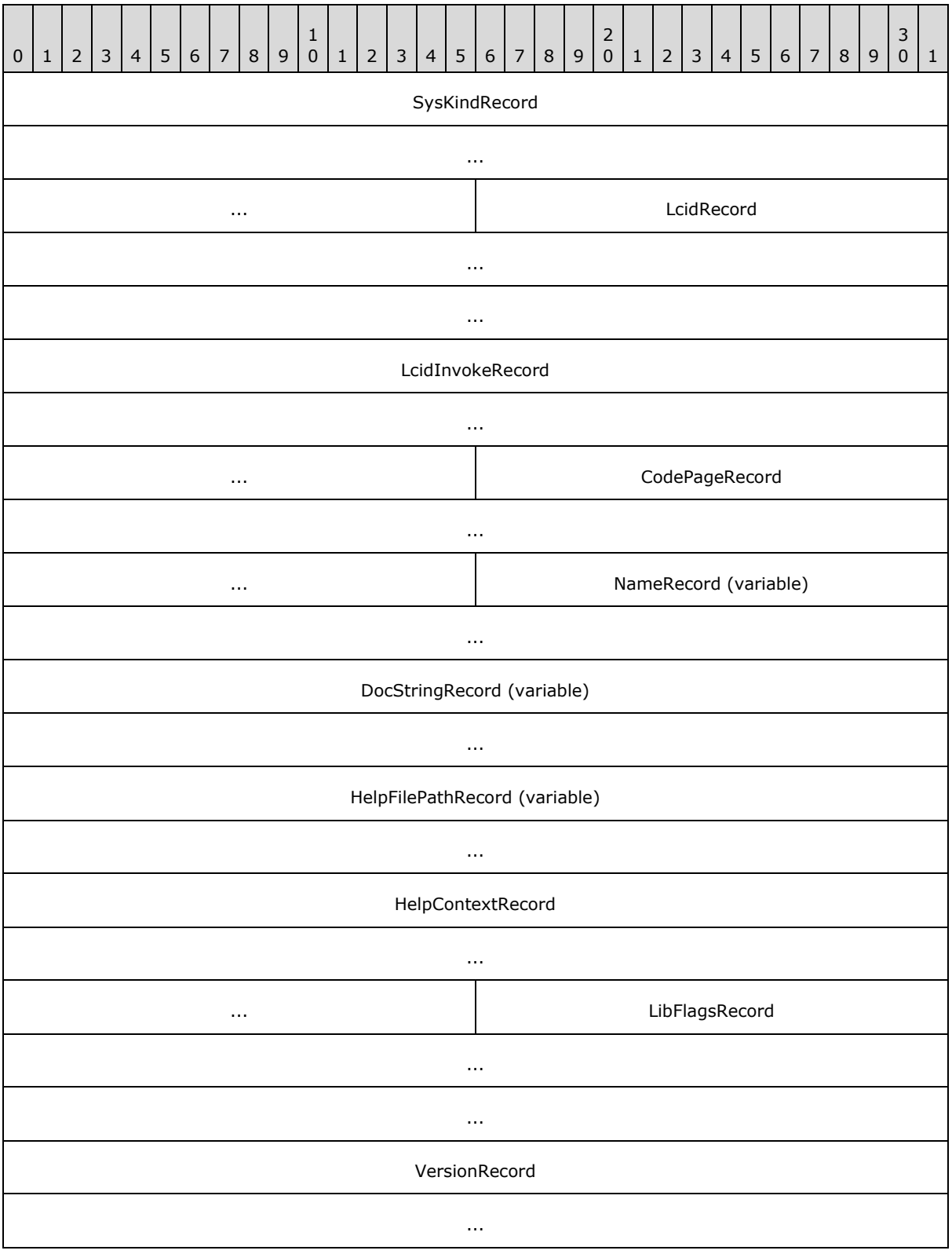
ModulesRecord (variable): A **PROJECTMODULES** Record (section 2.3.4.2.3).

Terminator (2 bytes): An unsigned integer that specifies the end of the version-independent information in this stream. MUST be 0x0010.

Reserved (4 bytes): MUST be 0x00000000. MUST be ignored.

2.3.4.2.1 PROJECTINFORMATION Record

Specifies version-independent information for the VBA project.



…
ConstantsRecord (variable)
…

SysKindRecord (10 bytes): A **PROJECTSYSKIND** Record (section 2.3.4.2.1.1).

LcidRecord (10 bytes): A **PROJECTLCID** Record (section 2.3.4.2.1.2).

LcidInvokeRecord (10 bytes): A **PROJECTLCIDINVOKE** Record (section 2.3.4.2.1.3).

CodePageRecord (8 bytes): A **PROJECTCODEPAGE** Record (section 2.3.4.2.1.4).

NameRecord (variable): A **PROJECTNAME** Record (section 2.3.4.2.1.5).

DocStringRecord (variable): A **PROJECTDOCSTRING** Record (section 2.3.4.2.1.6).

HelpFilePathRecord (variable): A **PROJECTHELPFILEPATH** Record (section 2.3.4.2.1.7).

HelpContextRecord (10 bytes): A **PROJECTHELPCONTEXT** Record (section 2.3.4.2.1.8).

LibFlagsRecord (10 bytes): A **PROJECTLIBFLAGS** Record (section 2.3.4.2.1.9).

VersionRecord (12 bytes): A **PROJECTVERSION** Record (section 2.3.4.2.1.10).

ConstantsRecord (variable): A **PROJECTCONSTANTS** Record (section 2.3.4.2.1.11). This field is optional.

2.3.4.2.1.1 PROJECTSYSKIND Record

Specifies the platform for which the VBA project is created.

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
Id																Size															
…																SysKind															
…																															

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0001.

Size (4 bytes): An unsigned integer that specifies the size of **SysKind**. MUST be 0x00000004.

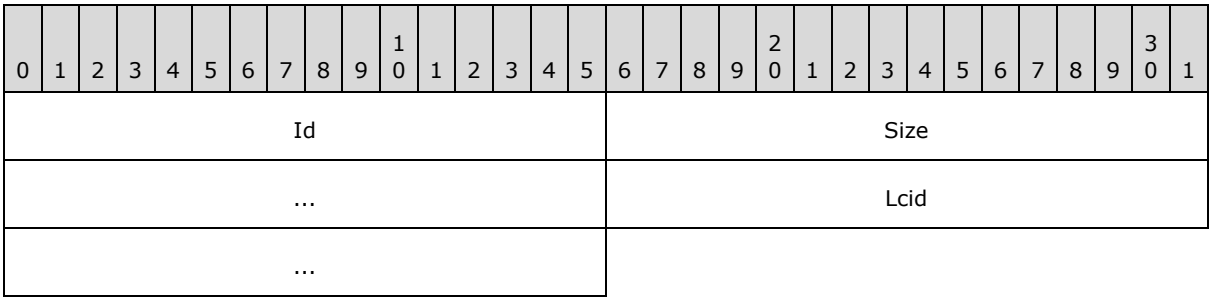
SysKind (4 bytes): An unsigned integer that specifies the platform for which the VBA project is created. MUST have one of the following values:

Value	Meaning
0x00000000	For 16-bit Windows Platforms.

Value	Meaning
0x00000001	For 32-bit Windows Platforms.
0x00000002	For Macintosh Platforms.
0x00000003	For 64-bit Windows Platforms.

2.3.4.2.1.2 PROJECTLCID Record

Specifies the VBA project's LCID.



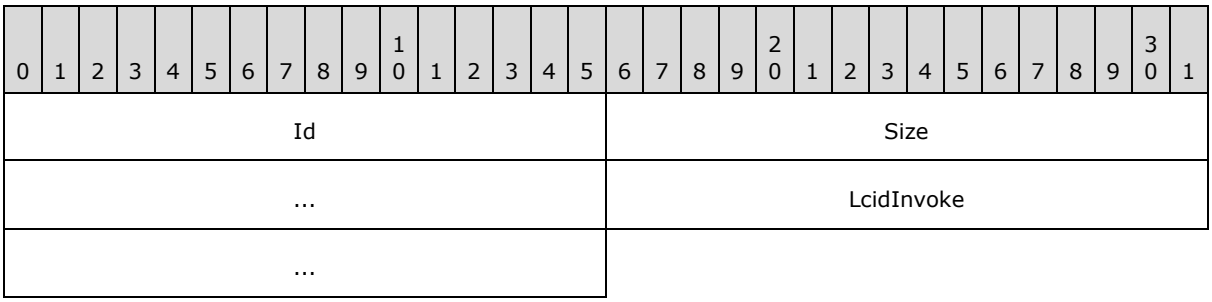
Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0002.

Size (4 bytes): An unsigned integer that specifies the size of **Lcid**. MUST be 0x00000004.

Lcid (4 bytes): An unsigned integer that specifies the LCID value for the VBA project. MUST be 0x00000409.

2.3.4.2.1.3 PROJECTLCIDINVOKE Record

Specifies an LCID value used for Invoke calls on an Automation server as specified in [MS-OAUT] section 3.1.4.4.



Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0014.

Size (4 bytes): An unsigned integer that specifies the size of **LcidInvoke**. MUST be 0x00000004.

LcidInvoke (4 bytes): An unsigned integer that specifies the LCID value used for Invoke calls. MUST be 0x00000409.

2.3.4.2.1.4 PROJECTCODEPAGE Record

Specifies the VBA project's code page.

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
Id																Size															
...																CodePage															

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0003.

Size (4 bytes): An unsigned integer that specifies the size of **CodePage**. MUST be 0x00000002.

CodePage (2 bytes): An unsigned integer that specifies the code page for the VBA project.

2.3.4.2.1.5 PROJECTNAME Record

Specifies a unique **VBA identifier** as the name of the VBA project.

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
Id																SizeOfProjectName															
...																ProjectName (variable)															
...																															

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0004.

SizeOfProjectName (4 bytes): An unsigned integer that specifies the size in bytes of **ProjectName**. MUST be greater than or equal to 1. MUST be less than or equal to 128.

ProjectName (variable): An array of **SizeOfProjectName** bytes that specifies the VBA identifier name for the VBA project. MUST contain MBCS characters encoded using the code page specified in **PROJECTCODEPAGE** (section 2.3.4.2.1.4). MUST NOT contain null characters.

2.3.4.2.1.6 PROJECTDOCSTRING Record

Specifies the description for the VBA project.

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
Id																SizeOfDocString															
...																DocString (variable)															

...	
Reserved	SizeOfDocStringUnicode
...	DocStringUnicode (variable)
...	

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0005.

SizeOfDocString (4 bytes): An unsigned integer that specifies the size in bytes of **DocString**. MUST be less than or equal to 2000.

DocString (variable): An array of **SizeOfDocString** bytes that specifies the description for the VBA project. MUST contain MBCS characters encoded using the code page specified in **PROJECTCODEPAGE** (section 2.3.4.2.1.4). MUST NOT contain null characters.

Reserved (2 bytes): MUST be 0x0040. MUST be ignored.

SizeOfDocStringUnicode (4 bytes): An unsigned integer that specifies the size in bytes of **DocStringUnicode**. MUST be even.

DocStringUnicode (variable): An array of **SizeOfDocStringUnicode** bytes that specifies the description for the VBA project. MUST contain UTF-16 characters. MUST NOT contain null characters. MUST contain the UTF-16 encoding of **DocString**.

2.3.4.2.1.7 PROJECTHELPFILEPATH Record

Specifies the path to the Help file for the VBA project. **<ProjectHelpFile>** MUST be defined in **PROJECT** Stream (section 2.3.1) if **SizeOfHelpFile1** is greater than zero.

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
Id																SizeOfHelpFile1															
...																HelpFile1 (variable)															
...																															
Reserved																SizeOfHelpFile2															
...																HelpFile2 (variable)															
...																															

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0006.

SizeOfHelpFile1 (4 bytes): An unsigned integer that specifies the size in bytes of **HelpFile1**. MUST be less than or equal to 260.

HelpFile1 (variable): An array of **SizeOfHelpFile1** bytes that specifies the path to the Help file for the VBA project. MUST contain MBCS characters encoded using the code page specified in **PROJECTCODEPAGE** (section 2.3.4.2.1.4). MUST NOT contain null characters.

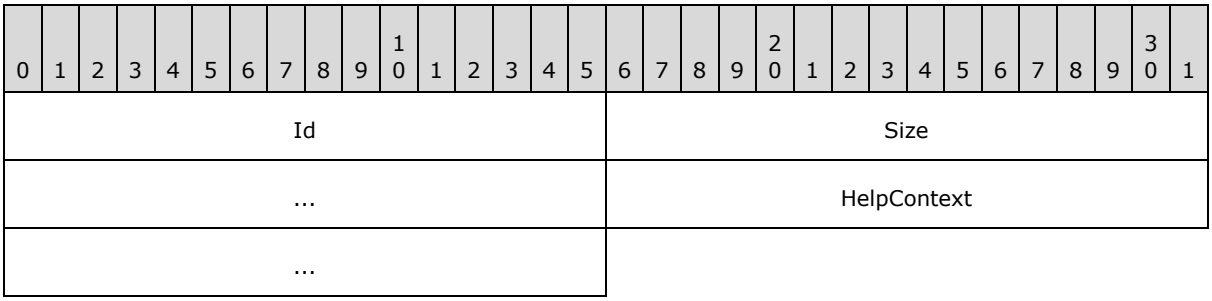
Reserved (2 bytes): MUST be 0x0049. MUST be ignored.

SizeOfHelpFile2 (4 bytes): An unsigned integer that specifies the size in bytes of **HelpFile2**. MUST be equal to **SizeOfHelpFile1**.

HelpFile2 (variable): An array of **SizeOfHelpFile2** bytes that specifies the path to the Help file for the VBA project. MUST contain MBCS characters encoded using the code page specified in **PROJECTCODEPAGE** (section 2.3.4.2.1.4). MUST NOT contain null characters. MUST contain the same bytes as **HelpFile1**.

2.3.4.2.1.8 PROJECTHELPCONTEXT Record

Specifies the Help topic identifier for the VBA project.



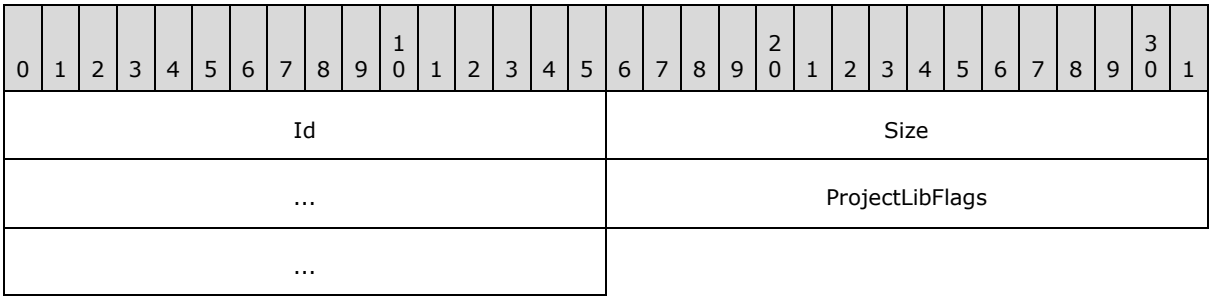
Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0007.

Size (4 bytes): An unsigned integer that specifies the size of **HelpContext**. MUST be 0x00000004.

HelpContext (4 bytes): An unsigned integer that specifies the Help topic identifier in the Help file specified by **PROJECTHELPFILEPATH** (section 2.3.4.2.1.7).

2.3.4.2.1.9 PROJECTLIBFLAGS Record

Specifies the LIBFLAGS for the VBA project's Automation type library as specified in [MS-OAUT] section 2.2.20.



Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0008.

Size (4 bytes): An unsigned integer that specifies the size of **ProjectLibFlags**. MUST be 0x00000004.

ProjectLibFlags (4 bytes): An unsigned integer that specifies LIBFLAGS for the VBA project's Automation type library as specified in [MS-OAUT] section 2.2.20. MUST be 0x00000000.

2.3.4.2.1.10 PROJECTVERSION Record

Specifies the version of the VBA project.

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
Id																Reserved															
...																VersionMajor															
...																VersionMinor															

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0009.

Reserved (4 bytes): MUST be 0x00000004. MUST be ignored.

VersionMajor (4 bytes): An unsigned integer specifying the major version of the VBA project.

VersionMinor (2 bytes): An unsigned integer specifying the minor version of the VBA project.

2.3.4.2.1.11 PROJECTCONSTANTS Record

Specifies the **compilation constants** for the VBA project.

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
Id																SizeOfConstants															
...																Constants (variable)															
...																															
Reserved																SizeOfConstantsUnicode															
...																ConstantsUnicode (variable)															
...																															

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x000C.

SizeOfConstants (4 bytes): An unsigned integer that specifies the size in bytes of **Constants**. MUST be less than or equal to 1015.

Constants (variable): An array of **SizeOfConstants** bytes that specifies the compilation constants for the VBA project. MUST contain MBCS characters encoded using the code page specified in **PROJECTCODEPAGE** (section 2.3.4.2.1.4). MUST NOT contain null characters.

MUST conform to the following ABNF grammar:

```
Constants     = Constant *( " : " Constant )

Constant      = ConstantName " = " ConstantValue

ConstantName  = VbaIdentifier

ConstantValue = ["-"] 1*5DIGIT
```

<ConstantName>: Specifies a unique VBA identifier for the constant.

<ConstantValue>: Specifies the numeric value for the constant. SHOULD be between −9999 and 32767. MAY be between −32768 and 32767 on read.<8>

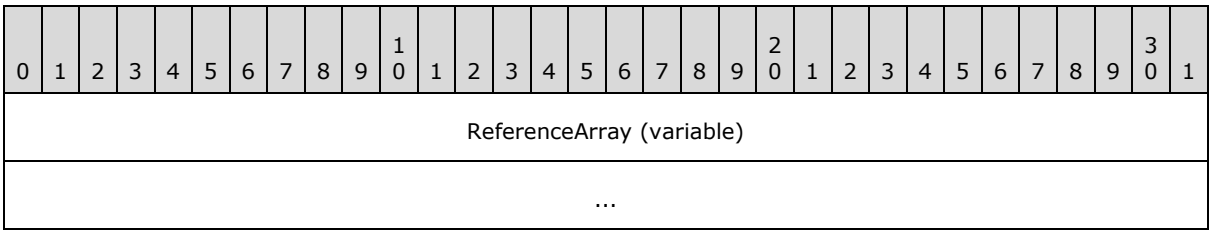
Reserved (2 bytes): MUST be 0x003C. MUST be ignored.

SizeOfConstantsUnicode (4 bytes): An unsigned integer that specifies the size in bytes of **ConstantsUnicode**. MUST be even.

ConstantsUnicode (variable): An array of **SizeOfConstantsUnicode** bytes that specifies the compilation constants for the VBA project. MUST contain UTF-16 characters. MUST NOT contain null characters. MUST contain the UTF-16 encoding of **Constants**.

2.3.4.2.2 PROJECTREFERENCES Record

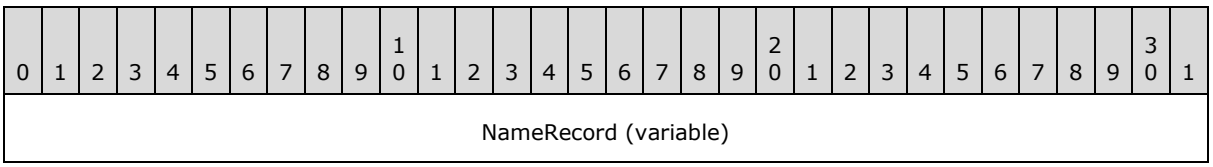
Specifies the external references (1) of the VBA project as a variably sized array of **REFERENCE** (section 2.3.4.2.2.1). The termination of the array is indicated by the beginning of **PROJECTMODULES** (section 2.3.4.2.3), which is indicated by a **REFERENCE** (section 2.3.4.2.2.1) being followed by an unsigned 16-bit integer with a value of 0x000F.



ReferenceArray (variable): An array of **REFERENCE** Records (section 2.3.4.2.2.1).

2.3.4.2.2.1 REFERENCE Record

Specifies a reference (1) to an Automation type library or VBA project.



...
ReferenceRecord (variable)
...

NameRecord (variable): A **REFERENCENAME** Record (section 2.3.4.2.2.2) that specifies the name of the referenced VBA project or Automation type library. This field is optional.

ReferenceRecord (variable): The type of **ReferenceRecord** is determined by the unsigned 16-bit integer beginning this field. The meanings of the possible values are listed in the following table:

Value	Meaning
0x002F	**ReferenceRecord** is a **REFERENCECONTROL** (section 2.3.4.2.2.3).
0x0033	**ReferenceRecord** is a **REFERENCECONTROL** (section 2.3.4.2.2.3).
0x000D	**ReferenceRecord** is a **REFERENCEREGISTERED** (section 2.3.4.2.2.5).
0x000E	**ReferenceRecord** is a **REFERENCEPROJECT** (section 2.3.4.2.2.6).

2.3.4.2.2.2 REFERENCENAME Record

Specifies the name of a referenced VBA project or Automation type library.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Id																SizeOfName															
...																Name (variable)															
...																															
Reserved																SizeOfNameUnicode															
...																NameUnicode (variable)															
...																															

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0016.

SizeOfName (4 bytes): An unsigned integer that specifies the size in bytes of **Name**.

Name (variable): An array of **SizeOfName** bytes that specifies the name of the referenced VBA project or Automation type library. MUST contain MBCS characters encoded using the code page specified in **PROJECTCODEPAGE** Record (section 2.3.4.2.1.4). MUST NOT contain null characters. MUST conform to the following ABNF grammar:

```
ReferenceName = RefProjectName / RefLibraryName

RefProjectName   = VbaIdentifier

RefLibraryName   = Identifier
```

<RefProjectName>: The name of a referenced project. **<ReferenceName>** MUST use the **<RefProjectName>** rule when the **ReferenceRecord** of the parent **REFERENCE** (section 2.3.4.2.2.1) is a **REFERENCEPROJECT** (section 2.3.4.2.2.6).

<RefLibraryName>: The name of a referenced Automation type library. **<ReferenceName>** MUST use the **<RefLibraryName>** rule when the **ReferenceRecord** of the parent **REFERENCE** (section 2.3.4.2.2.1) is a **REFERENCECONTROL** (section 2.3.4.2.2.3) or **REFERENCEREGISTERED** (section 2.3.4.2.2.5). **<Identifier>** is defined in [C706].

Reserved (2 bytes): MUST be 0x003E. MUST be ignored.

SizeOfNameUnicode (4 bytes): An unsigned integer that specifies the size in bytes of **NameUnicode**.

NameUnicode (variable): An array of **SizeOfNameUnicode** bytes that specifies the name of the referenced VBA project or Automation type library. MUST contain UTF-16 characters. MUST NOT contain null characters. MUST contain the UTF-16 encoding of **Name**.

2.3.4.2.2.3 REFERENCECONTROL Record

Specifies a reference (1) to a **twiddled type library** and its extended type library.

<table>
<tr><td colspan="32">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</td></tr>
<tr><td colspan="32">OriginalRecord (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="16">Id</td><td colspan="16">SizeTwiddled</td></tr>
<tr><td colspan="16">...</td><td colspan="16">SizeOfLibidTwiddled</td></tr>
<tr><td colspan="16">...</td><td colspan="16">LibidTwiddled (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">Reserved1</td></tr>
<tr><td colspan="16">Reserved2</td><td colspan="16">NameRecordExtended (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="16">Reserved3</td><td colspan="16">SizeExtended</td></tr>
</table>

...	SizeOfLibidExtended
...	LibidExtended (variable)
...	
Reserved4	
Reserved5	OriginalTypeLib (16 bytes)
...	
...	Cookie
...	

OriginalRecord (variable): A **REFERENCEORIGINAL** Record (section 2.3.4.2.2.4) that specifies the Automation type library the twiddled type library was generated from. SHOULD exist. MAY<9> not exist.

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x002F.

SizeTwiddled (4 bytes): An unsigned integer that specifies the sum of the size in bytes of **SizeOfLibidTwiddled**, **LibidTwiddled**, **Reserved1**, and **Reserved2**. MUST be ignored on read.

SizeOfLibidTwiddled (4 bytes): An unsigned integer that specifies the size in bytes of **LibidTwiddled**.

LibidTwiddled (variable): An array of **SizeOfLibidTwiddled** bytes. SHOULD be "*\G{00000000-0000-0000-0000-000000000000}#0.0#0##" (case-sensitive). MAY<10> specify a twiddled type library's identifier. The identifier MUST conform to the ABNF grammar **LibidReference** (section 2.1.1.8). MUST contain MBCS characters encoded using the code page specified in **PROJECTCODEPAGE** (section 2.3.4.2.1.4). MUST NOT contain null characters.

Reserved1 (4 bytes): MUST be 0x00000000. MUST be ignored.

Reserved2 (2 bytes): MUST be 0x00000000. MUST be ignored.

NameRecordExtended (variable): A **REFERENCENAME** Record (section 2.3.4.2.2.2) that specifies the name of the extended type library. This field is optional.

Reserved3 (2 bytes): MUST be 0x0030. MUST be ignored.

SizeExtended (4 bytes): An unsigned integer that specifies the sum of the size in bytes of **SizeOfLibidExtended**, **LibidExtended**, **Reserved4**, **Reserved5**, **OriginalTypeLib**, and **Cookie**. MUST be ignored on read.

SizeOfLibidExtended (4 bytes): An unsigned integer that specifies the size in bytes of **LibidExtended**.

LibidExtended (variable): An array of **SizeOfLibidExtended** bytes that specifies the extended type library's identifier. MUST contain MBCS characters encoded using the code page specified in **PROJECTCODEPAGE** (section 2.3.4.2.1.4). MUST NOT contain null characters. MUST conform to the ABNF grammar in **LibidReference** (section 2.1.1.8).

Reserved4 (4 bytes): MUST be 0x00000000. MUST be ignored.

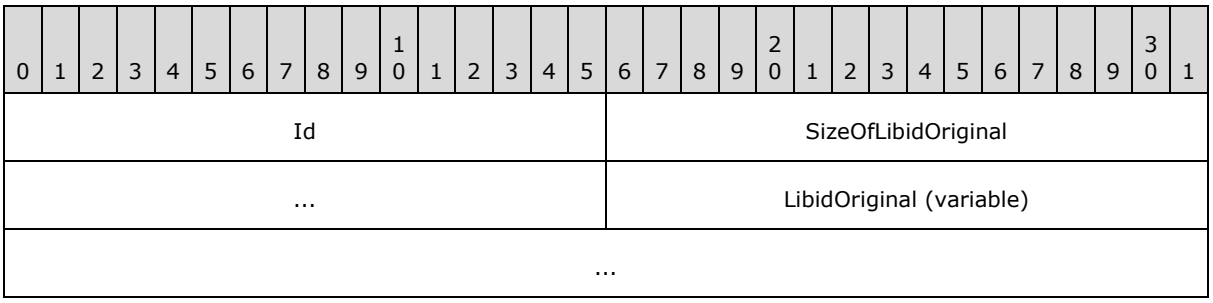
Reserved5 (2 bytes): MUST be 0x00000000. MUST be ignored.

OriginalTypeLib (16 bytes): A GUID that specifies the Automation type library the extended type library was generated from.

Cookie (4 bytes): An unsigned integer that specifies the extended type library's cookie. MUST be unique for each **REFERENCECONTROL** (section 2.3.4.2.2.3) in the VBA project with the same **OriginalTypeLib**.

2.3.4.2.2.4 REFERENCEORIGINAL Record

Specifies the identifier of the Automation type library the containing **REFERENCECONTROL**'s (section 2.3.4.2.2.3) twiddled type library was generated from.



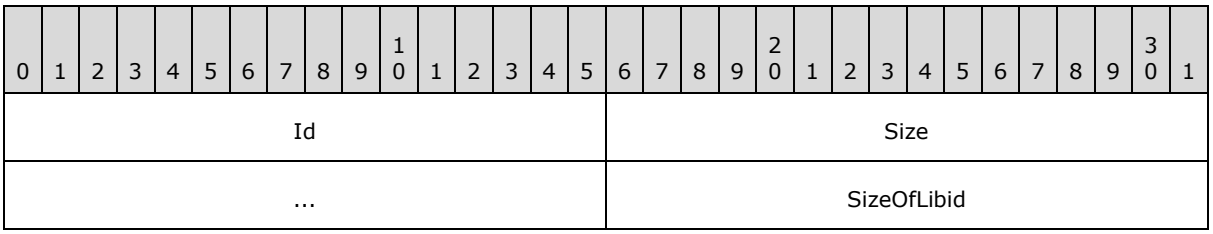
Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0033.

SizeOfLibidOriginal (4 bytes): An unsigned integer that specifies the size in bytes of **LibidOriginal**.

LibidOriginal (variable): An array of **SizeOfLibidOriginal** bytes that specifies the identifier of the Automation type library a **REFERENCECONTROL** (section 2.3.4.2.2.3) was generated from. MUST contain MBCS characters encoded using the code page specified in **PROJECTCODEPAGE** (section 2.3.4.2.1.4). MUST NOT contain null characters. MUST conform to the ABNF grammar in **LibidReference** (section 2.1.1.8).

2.3.4.2.2.5 REFERENCEREGISTERED Record

Specifies a reference (1) to an Automation type library.



...	Libid (variable)
...	
Reserved1	
Reserved2	

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x000D

Size (4 bytes): An unsigned integer that specifies the total size in bytes of **SizeOfLibid**, **Libid**, **Reserved1**, and **Reserved2**. MUST be ignored on read.

SizeOfLibid (4 bytes): An unsigned integer that specifies the size in bytes of **Libid**.

Libid (variable): An array of **SizeOfLibid** bytes that specifies an Automation type library's identifier. MUST contain MBCS characters encoded using the code page specified in **PROJECTCODEPAGE** (section 2.3.4.2.1.4). MUST NOT contain null characters. MUST conform to the ABNF grammar in **LibidReference** (section 2.1.1.8).

Reserved1 (4 bytes): MUST be 0x00000000. MUST be ignored.

Reserved2 (2 bytes): MUST be 0x00000000. MUST be ignored.

2.3.4.2.2.6 REFERENCEPROJECT Record

Specifies a reference (1) to an external VBA project.

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
Id																Size															
...																SizeOfLibidAbsolute															
...																LibidAbsolute (variable)															
...																															
SizeOfLibidRelative																															
LibidRelative (variable)																															
...																															
MajorVersion																															
MinorVersion																															

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x000E.

Size (4 bytes): An unsigned integer that specifies the total size in bytes of **SizeOfLibidAbsolute**, **LibidAbsolute**, **SizeOfLibidRelative**. **LibidRelative**, **MajorVersion**, and **MinorVersion**. MUST be ignored on read.

SizeOfLibidAbsolute (4 bytes): An unsigned integer that specifies the size in bytes of **LibidAbsolute**.

LibidAbsolute (variable): An array of **SizeOfLibidAbsolute** bytes that specifies the referenced VBA project's identifier with an **absolute path**, **<ProjectPath>**. MUST contain MBCS characters encoded using the code page specified in **PROJECTCODEPAGE** (section 2.3.4.2.1.4). MUST NOT contain null characters. MUST conform to the ABNF grammar **ProjectReference** (section 2.1.1.12).

SizeOfLibidRelative (4 bytes): An unsigned integer that specifies the size in bytes of **LibidRelative**.

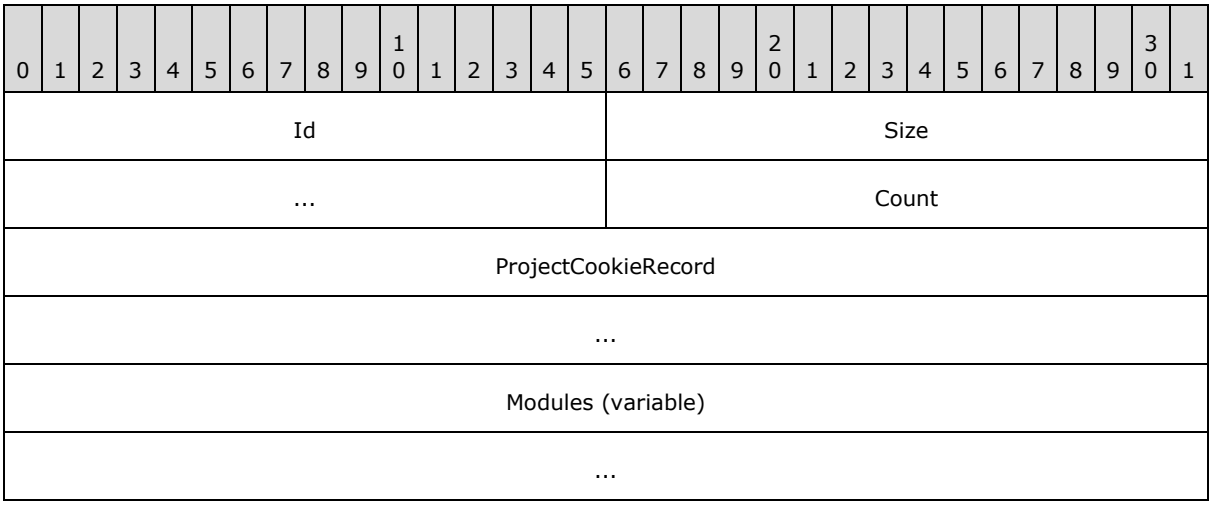
LibidRelative (variable): An array of **SizeOfLibidRelative** bytes that specifies the referenced VBA project's identifier with a **relative path**, **<ProjectPath>,** that is relative to the current VBA project. MUST contain MBCS characters encoded using the code page specified in **PROJECTCODEPAGE** (section 2.3.4.2.1.4). MUST NOT contain null characters. MUST conform to the ABNF grammar **ProjectReference** (section 2.1.1.12).

MajorVersion (4 bytes): An unsigned integer that specifies the major version of the referenced VBA project. On write MUST be the **PROJECTVERSION.VersionMajor** (section 2.3.4.2.1.10) of the referenced VBA project.

MinorVersion (2 bytes): An unsigned integer that specifies the minor version of the external VBA project. On write MUST be the **PROJECTVERSION.VersionMinor** (section 2.3.4.2.1.10) of the referenced VBA project.

2.3.4.2.3 PROJECTMODULES Record

Specifies data for the modules in the project.



Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x000F.

Size (4 bytes): An unsigned integer that specifies the size of **Count**. MUST be 0x00000002.

Count (2 bytes): An unsigned integer that specifies the number of elements in **Modules**.

ProjectCookieRecord (8 bytes): A **PROJECTCOOKIE** Record (section 2.3.4.2.3.1).

Modules (variable): An array of **MODULE** Records (section 2.3.4.2.3.2).

2.3.4.2.3.1 PROJECTCOOKIE Record

Specifies data that is ignored.

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
Id																Size															
...																Cookie															

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0013.

Size (4 bytes): An unsigned integer that specifies the size of **Cookie**. MUST be 0x00000002.

Cookie (2 bytes): MUST be ignored on read. MUST be 0xFFFF on write.

2.3.4.2.3.2 MODULE Record

Specifies data for a module. Source code for the module can be found in the **ModuleStream** (section 2.3.4.3) named as specified in **StreamNameRecord**. Every **MODULE** (section 2.3.4.2.3.2) MUST have a corresponding **<ProjectModule>** specified in **PROJECT** Stream (section 2.3.1).

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
NameRecord (variable)																															
...																															
NameUnicodeRecord (variable)																															
...																															
StreamNameRecord (variable)																															
...																															
DocStringRecord (variable)																															
...																															
OffsetRecord																															

...	
...	HelpContextRecord
...	
...	
CookieRecord	
...	
TypeRecord	
...	ReadOnlyRecord (optional)
...	
PrivateRecord (optional)	
...	Terminator
Reserved	

NameRecord (variable): A **MODULENAME** Record (section 2.3.4.2.3.2.1).

NameUnicodeRecord (variable): A **MODULENAMEUNICODE** Record (section 2.3.4.2.3.2.2). This field is optional.

StreamNameRecord (variable): A **MODULESTREAMNAME** Record (section 2.3.4.2.3.2.3).

DocStringRecord (variable): A **MODULEDOCSTRING** Record (section 2.3.4.2.3.2.4).

OffsetRecord (10 bytes): A **MODULEOFFSET** Record (section 2.3.4.2.3.2.5).

HelpContextRecord (10 bytes): A **MODULEHELPCONTEXT** Record (section 2.3.4.2.3.2.6).

CookieRecord (8 bytes): A **MODULECOOKIE** Record (section 2.3.4.2.3.2.7).

TypeRecord (6 bytes): A **MODULETYPE** Record (section 2.3.4.2.3.2.8).

ReadOnlyRecord (6 bytes): A **MODULEREADONLY** Record (section 2.3.4.2.3.2.9). This field is optional.

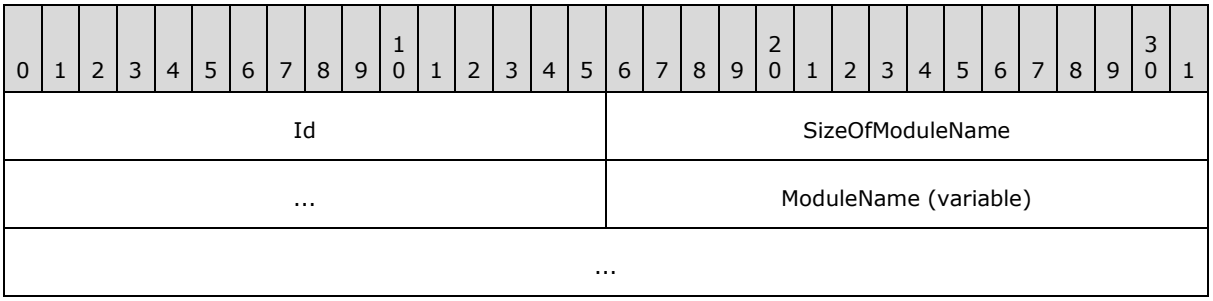
PrivateRecord (6 bytes): A **MODULEPRIVATE** Record (section 2.3.4.2.3.2.10). This field is optional.

Terminator (2 bytes): An unsigned integer that specifies the end of this record. MUST be 0x002B.

Reserved (4 bytes): MUST be 0x00000000. MUST be ignored.

2.3.4.2.3.2.1 MODULENAME Record

Specifies a VBA identifier as the name of the containing **MODULE Record** (section 2.3.4.2.3.2).



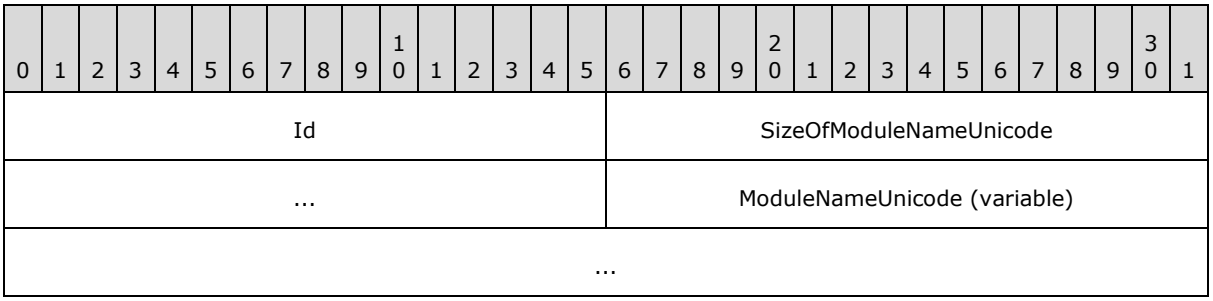
Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0019.

SizeOfModuleName (4 bytes): An unsigned integer that specifies the size in bytes of **ModuleName**.

ModuleName (variable): An array of **SizeOfModuleName** bytes that specifies the VBA identifier for the containing **MODULE Record**. MUST contain MBCS characters encoded using the code page specified in the **PROJECTCODEPAGE Record** (section 2.3.4.2.1.4). MUST NOT contain null characters.

2.3.4.2.3.2.2 MODULENAMEUNICODE Record

Specifies a VBA identifier as the name of the containing MODULE Record (section 2.3.4.2.3.2). MUST contain the UTF-16 encoding of **MODULENAME** Record (section 2.3.4.2.3.2.1).



Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0047.

SizeOfModuleNameUnicode (4 bytes): An unsigned integer that specifies the size in bytes of **ModuleNameUnicode**. MUST be even.

ModuleNameUnicode (variable): An array of **SizeOfModuleNameUnicode** bytes that specifies the VBA identifier for the containing MODULE Record (section 2.3.4.2.3.2). MUST contain UTF-16 characters. MUST NOT contain null characters. MUST contain the UTF-16 encoding of **MODULENAME** Record (section 2.3.4.2.3.2.1) **ModuleName**.

2.3.4.2.3.2.3 MODULESTREAMNAME Record

Specifies the stream name of the ModuleStream (section 2.3.4.3) in the VBA Storage (section 2.3.4) corresponding to the containing MODULE Record (section 2.3.4.2.3.2).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Id																SizeOfStreamName															
...																StreamName (variable)															
...																															
Reserved																SizeOfStreamNameUnicode															
...																StreamNameUnicode (variable)															
...																															

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x001A.

SizeOfStreamName (4 bytes): An unsigned integer that specifies the size in bytes of **StreamName**.

StreamName (variable): An array of **SizeOfStreamName** bytes that specifies the stream name of the ModuleStream (section 2.3.4.3). MUST contain MBCS characters encoded using the code page specified in PROJECTCODEPAGE (section 2.3.4.2.1.4). MUST NOT contain null characters.

Reserved (2 bytes): MUST be 0x0032. MUST be ignored.

SizeOfStreamNameUnicode (4 bytes): An unsigned integer that specifies the size in bytes of **StreamNameUnicode**. MUST be even.

StreamNameUnicode (variable): An array of **SizeOfStreamNameUnicode** bytes that specifies the stream name of the ModuleStream (section 2.3.4.3). MUST contain UTF-16 characters. MUST NOT contain null characters. MUST contain the UTF-16 encoding of **StreamName**.

2.3.4.2.3.2.4 MODULEDOCSTRING Record

Specifies the description for the containing **MODULE** Record (section 2.3.4.2.3.2).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Id																SizeOfDocString															
...																DocString (variable)															
...																															
Reserved																SizeOfDocStringUnicode															

...	DocStringUnicode (variable)
...	

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x001C.

SizeOfDocString (4 bytes): An unsigned integer that specifies the size in bytes of **DocString**.

DocString (variable): An array of **SizeOfDocString** bytes that specifies the description for the containing **MODULE** Record (section 2.3.4.2.3.2). MUST contain MBCS characters encoded using the code page specified in **PROJECTCODEPAGE** (section 2.3.4.2.1.4). MUST NOT contain null characters.

Reserved (2 bytes): MUST be 0x0048. MUST be ignored.

SizeOfDocStringUnicode (4 bytes): An unsigned integer that specifies the size in bytes of **DocStringUnicode**. MUST be even.

DocStringUnicode (variable): An array of **SizeOfDocStringUnicode** bytes that specifies the description for the containing **MODULE** Record (section 2.3.4.2.3.2). MUST contain UTF-16 characters. MUST NOT contain null characters. MUST contain the UTF-16 encoding of **DocString**.

2.3.4.2.3.2.5 MODULEOFFSET Record

Specifies the location of the source code within the ModuleStream (section 2.3.4.3) that corresponds to the containing MODULE Record (section 2.3.4.2.3.2).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Id																Size															
...																TextOffset															
...																															

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0031.

Size (4 bytes): An unsigned integer that specifies the size of **TextOffset**. MUST be 0x00000004.

TextOffset (4 bytes): An unsigned integer that specifies the byte offset of the source code in the ModuleStream (section 2.3.4.3) named by **MODULESTREAMNAME** Record (section 2.3.4.2.3.2.3).

2.3.4.2.3.2.6 MODULEHELPCONTEXT Record

Specifies the Help topic identifier for the containing MODULE Record (section 2.3.4.2.3.2).

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
Id																Size															
...																HelpContext															
...																															

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x001E.

Size (4 bytes): An unsigned integer that specifies the size of **HelpContext**. MUST be 0x00000004.

HelpContext (4 bytes): An unsigned integer that specifies the Help topic identifier in the Help file specified by PROJECTHELPFILEPATH Record (section 2.3.4.2.1.7).

2.3.4.2.3.2.7 MODULECOOKIE Record

Specifies ignored data.

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
Id																Size															
...																Cookie															

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x002C.

Size (4 bytes): An unsigned integer that specifies the size of **Cookie**. MUST be 0x00000002.

Cookie (2 bytes): MUST be ignored on read. MUST be 0xFFFF on write.

2.3.4.2.3.2.8 MODULETYPE Record

Specifies whether the containing MODULE Record (section 2.3.4.2.3.2) is a procedural module, document module, class module, or designer module.

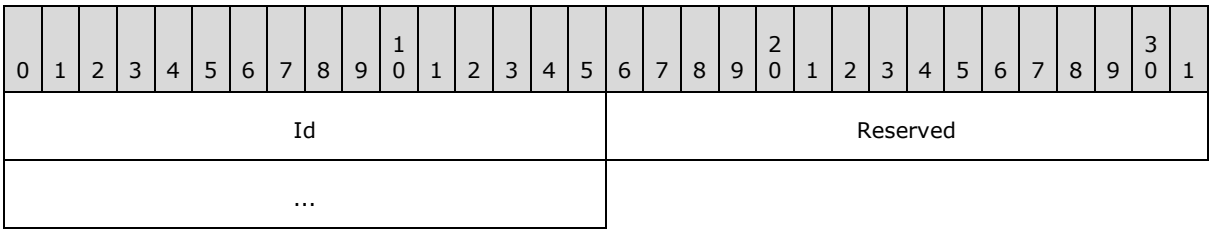
0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Id																Reserved															
...																															

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0021 when the containing MODULE Record (section 2.3.4.2.3.2) is a procedural module. MUST be 0x0022 when the containing MODULE Record (section 2.3.4.2.3.2) is a document module, class module, or designer module.

Reserved (4 bytes): MUST be 0x00000000. MUST be ignored.

2.3.4.2.3.2.9 MODULEREADONLY Record

Specifies that the containing MODULE Record (section 2.3.4.2.3.2) is read-only.

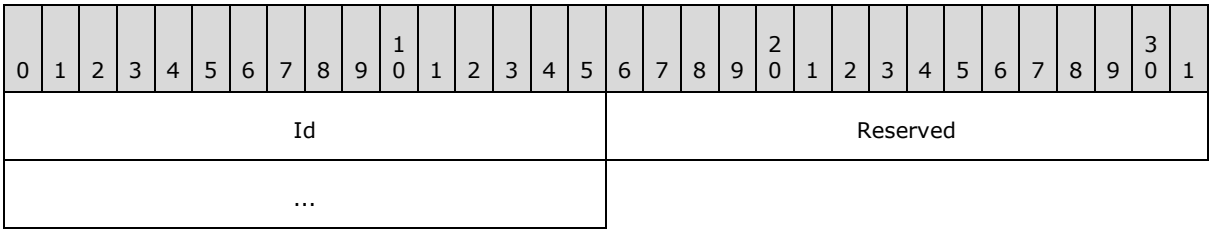


Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0025.

Reserved (4 bytes): MUST be 0x00000000. MUST be ignored.

2.3.4.2.3.2.10 MODULEPRIVATE Record

Specifies that the containing MODULE Record (section 2.3.4.2.3.2) is only usable from within the current VBA project.

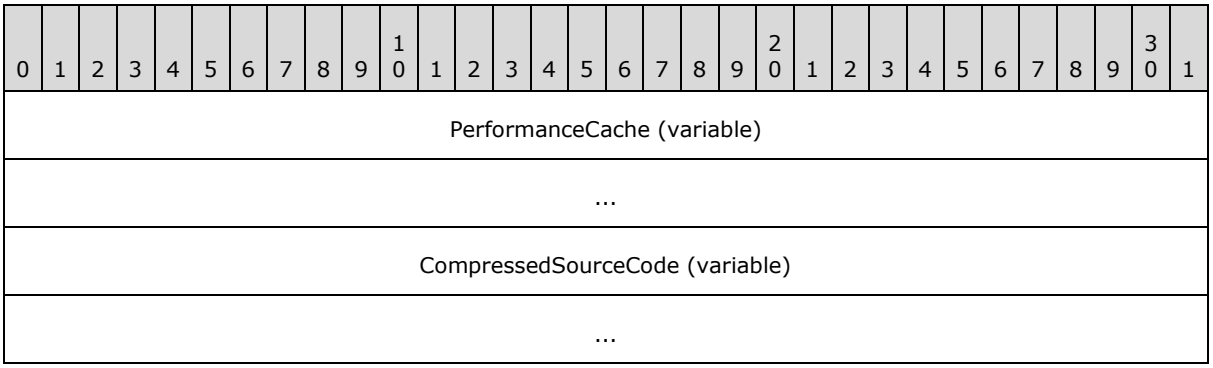


Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0028.

Reserved (4 bytes): MUST be 0x00000000. MUST be ignored.

2.3.4.3 Module Stream: Visual Basic Modules

Specifies the source code for a module.



PerformanceCache (variable): An array of bytes that forms an implementation-specific and version-dependent performance cache for the module. MUST be MODULEOFFSET (section 2.3.4.2.3.2.5) bytes in size. MUST be ignored on read.

CompressedSourceCode (variable): An array of bytes compressed as specified in Compression (section 2.4.1). When decompressed yields an array of bytes that specifies the textual representation of VBA language source code as specified in [MS-VBAL]. MUST contain MBCS characters encoded using the code page specified in PROJECTCODEPAGE (section 2.3.4.2.1.4).

2.3.5 VBFrame Stream: Designer Information

The VBFrame stream (1) specifies the extended property values of a designer.

This stream is an array of bytes that specifies the extended property values of a designer module. MUST contain MBCS characters encoded using the code page specified in PROJECTCODEPAGE (section 2.3.4.2.1.4).

Property values of the designer are set at design-time. Property values are used at run-time as specified to initialize the designer. For example, a designer can be used at run time to display data to and accept data from a user and the following properties could be used to determine the location of the designer.

ABNF syntax:

```
VBFrameText       = "VERSION 5.00" NWLN
                    "Begin" 1*WSP DesignerCLSID 1*WSP DesignerName *WSP NWLN
                     DesignerProperties "End" NWLN

DesignerCLSID     = GUID

DesignerName      = ModuleIdentifier
```

<DesignerCLSID>: Specifies the class identifier (CLSID) of the designer. The Automation type library that contains the designer specified MUST be referenced with a REFERENCECONTROL (section 2.3.4.2.2.3). The value "{C62A69F0-16DC-11CE-9E98-00AA00574A4F}" specifies the designer is an Office Form ActiveX control specified in [MS-OFORMS].

<DesignerName>: Specifies the name of the designer module associated with the properties.

2.3.5.1 DesignerProperties

Specifies the VBA-specific extended properties of a designer.

ABNF syntax:

```
DesignerProperties = [ *WSP DesignerCaption *WSP [ Comment ] NWLN ]
                     [ *WSP DesignerHeight *WSP [ Comment ] NWLN ]
                     [ *WSP DesignerLeft *WSP [ Comment ] NWLN ]
                     [ *WSP DesignerTop *WSP [ Comment ] NWLN ]
                     [ *WSP DesignerWidth *WSP [ Comment ] NWLN ]
                     [ *WSP DesignerEnabled *WSP [ Comment ] NWLN ]
                     [ *WSP DesignerHelpContextId *WSP [ Comment ] NWLN ]
                     [ *WSP DesignerRTL *WSP [ Comment ] NWLN ]
                     [ *WSP DesignerShowModal *WSP [ Comment ] NWLN ]
                     [ *WSP DesignerStartupPosition *WSP [ Comment ] NWLN ]
```

```
                        [ *WSP DesignerTag *WSP [ Comment ] NWLN ]
                        [ *WSP DesignerTypeInfoVer *WSP [ Comment ] NWLN ]
                        [ *WSP DesignerVisible *WSP [ Comment ] NWLN ]
                        [ *WSP DesignerWhatsThisButton *WSP [ Comment ] NWLN ]
                        [ *WSP DesignerWhatsThisHelp *WSP [ Comment ] NWLN ]

Comment             =  "'" *ANYCHAR
```

<Comment>: Specifies a user-readable comment.

2.3.5.2 DesignerCaption

Specifies the title text of the designer.

ABNF syntax:

```
DesignerCaption         = "Caption" EQ DQUOTE DesignerCaptionText DQUOTE

DesignerCaptionText     = *130QUOTEDCHAR
```

2.3.5.3 DesignerHeight

Specifies the height of the designer in **twips**.

ABNF syntax:

```
DesignerHeight          = "ClientHeight" EQ FLOAT
```

2.3.5.4 DesignerLeft

Specifies the left edge of the designer in twips relative to the window specified by DesignerStartupPosition (section 2.3.5.11).

ABNF syntax:

```
DesignerLeft            = "ClientLeft" EQ FLOAT
```

2.3.5.5 DesignerTop

Specifies the position of the top edge of the designer in twips relative to the window specified by DesignerStartupPosition (section 2.3.5.11).

ABNF syntax:

```
DesignerTop             = "ClientTop" EQ FLOAT
```

2.3.5.6 DesignerWidth

Specifies the width of the designer in twips.

ABNF Syntax:

```
DesignerWidth           = "ClientWidth" EQ FLOAT
```

2.3.5.7 DesignerEnabled

Specifies whether the designer is enabled. The default is TRUE.

ABNF syntax:

```
DesignerEnabled         = "Enabled" EQ VBABOOL
```

2.3.5.8 DesignerHelpContextId

Specifies the Help topic identifier associated with this designer in the Help file as specified by ProjectHelpFile (section 2.3.1.9).

ABNF syntax:

```
DesignerHelpContextId   = "HelpContextID" EQ INT32
```

2.3.5.9 DesignerRTL

Specifies that the designer be shown with right and left coordinates reversed for **right-to-left** language use.

ABNF syntax:

```
DesignerRTL             = "RightToLeft" EQ VBABOOL
```

2.3.5.10 DesignerShowModal

Specifies whether the designer is a modal window. The default is TRUE.

ABNF syntax:

```
DesignerShowModal       = "ShowModal" EQ VBABOOL
```

2.3.5.11 DesignerStartupPosition

Specifies the startup position of the designer as follows.

ABNF syntax:

```
DesignerStartupPosition = "StartUpPosition" EQ RelativeParent

RelativeParent          = "0" / "1" / "2" / "3"
```

<RelativeParent>: Specifies the window used to determine the relative starting coordinates of the control window.

MUST be one of the following values:

Value	Meaning
"0"	"Manual" mode. DesignerTop (section 2.3.5.5) and DesignerLeft (section 2.3.5.4) coordinates of the designer are relative to the desktop window.
"1"	"CenterOwner" mode. Center the designer relative to its parent window.
"2"	"Center" mode. Center the designer relative to the desktop window.
"3"	"WindowsDefault" mode. Place the designer in the upper-left corner of screen.

2.3.5.12 DesignerTag

Specifies user-defined data associated with the designer.

ABNF syntax:

```
DesignerTag             = "Tag" EQ DQUOTE DesignerTagText DQUOTE

DesignerTagText         = *130QUOTEDCHAR
```

2.3.5.13 DesignerTypeInfoVer

Specifies the number of times the designer has been changed and saved. The default is 0.

ABNF syntax:

```
DesignerTypeInfoVer     = "TypeInfoVer" EQ INT32
```

2.3.5.14 DesignerVisible

Specifies whether the designer is visible. The default is TRUE.

ABNF syntax:

```
DesignerVisible         = "Visible" EQ VBABOOL
```

2.3.5.15 DesignerWhatsThisButton

Specifies whether a help button is shown for the designer. The default is FALSE.

ABNF syntax:

```
DesignerWhatsThisButton = "WhatsThisButton" EQ VBABOOL
```

2.3.5.16 DesignerWhatsThisHelp

Specifies whether a help topic is associated with this designer. The Help topic identifier is specified by DesignerHelpContextId (section 2.3.5.8).

ABNF syntax:

```
DesignerWhatsThisHelp    = "WhatsThisHelp" EQ VBABOOL
```

2.4 Algorithms

2.4.1 Compression and Decompression

To preserve space, VBA uses data compression on a contiguous sequence of records on various streams (1). The data compression technique is **run length encoding**.

The compression algorithm repeatedly reads 4096 bytes from the decompressed buffer into an array. Each group of 4096 bytes is called a chunk. The compression algorithm writes each 4096 byte chunk in an encoded and compressed format. Each output chunk is preceded by a two byte header which denotes the number of bytes in the chunk and the format of the chunk.

The compression algorithm searches for series of bytes that are repeated within the chunk. When series with multiple occurrences are found, the bytes in the first occurrence are encoded as literal tokens and the remaining occurrences are encoded as copy tokens which reference the first occurrence. The encoding for a repeated series of bytes is two bytes in length, thus matches of three bytes or more are required for encoding to be beneficial. Tokens are organized into groups of eight called a Token Sequence, which includes a flag byte. The flag byte is written in advance of the eight tokens. Each bit in the flag byte is used to identify the type of one of the token.

If the compression algorithm fails in producing enough copy tokens to compensate for the space overhead of the copy tokens and the flag bytes, the 4096 byte input chunk is written to the output chunk without any encoding.

The decompression algorithm reads one compressed chunk at a time. Each compressed chunk is decoded into 4096 bytes of uncompressed data which is written to output. For each chunk, the size and format style are extracted from the chunk header. The chunk is then read and decoded according to the format specified in the header.

When the chunk header format specifies that the chunk contains no copy tokens, the 4096 remaining bytes are copied to output. When the chunk header format specifies that copy tokens exist in the chunk, the Token Sequences are decoded. Literal tokens are copied to output. Copy tokens are decoded to find the first occurrence of the byte sequence the copy token represents which is then copied to output.

The pseudocode and record specifications for Compression and Decompression use the following conventions.

- LEFT SHIFT: Bits in the operand are moved from the least significant to the most significant positions. High order bits are truncated. Low order bits become zero.
- RIGHT SHIFT: Bits in the operand are moved from the most significant position to the least significant positions. Low order bits are truncated. High order bits become zero.
- A literal bit sequence is denoted with the initial characters 0b. For example, the literal constant 0xB721 would appear as the binary literal 0b1011011100100001.

2.4.1.1 Structures

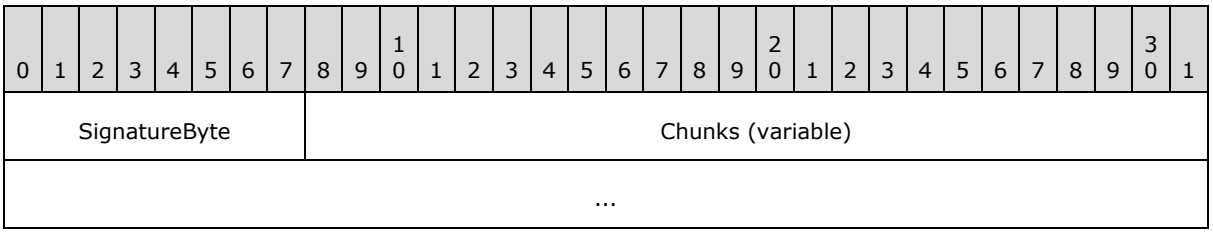
2.4.1.1.1 CompressedContainer

A **CompressedContainer** is an array of bytes holding the compressed data. The **Decompression** algorithm (section 2.4.1.3.1) processes a **CompressedContainer** to populate a

DecompressedBuffer. The **Compression** algorithm (section 2.4.1.3.6) processes a **DecompressedBuffer** to produce a **CompressedContainer**.

A **CompressedContainer** MUST be the last array of bytes in a stream (1). On read, the end of stream (1) indicator determines when the entire **CompressedContainer** has been read.

The **CompressedContainer** is a **SignatureByte** followed by array of **CompressedChunk** (section 2.4.1.1.4) structures.

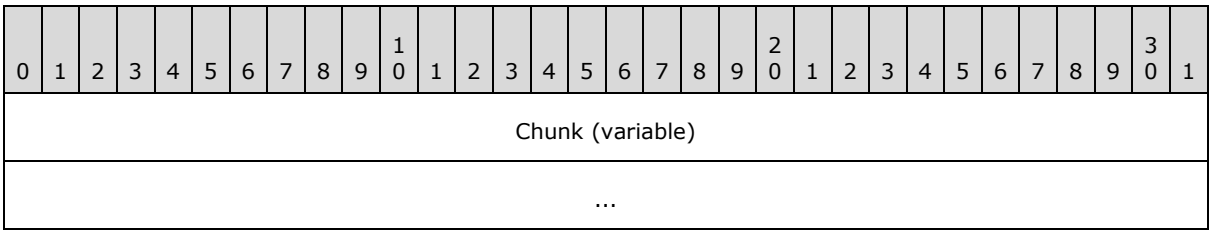


SignatureByte (1 byte): Specifies the beginning of the **CompressedContainer.** MUST be 0x01. The **Decompression** algorithm (section 2.4.1.3.1) reads **SignatureByte**. The **Compression** algorithm (section 2.4.1.3.6) writes **SignatureByte**.

Chunks (variable): An array of **CompressedChunk** (section 2.4.1.1.4) records. Specifies the compressed data. Read by the **Decompression** algorithm. Written by the **Compression** algorithm.

2.4.1.1.2 DecompressedBuffer

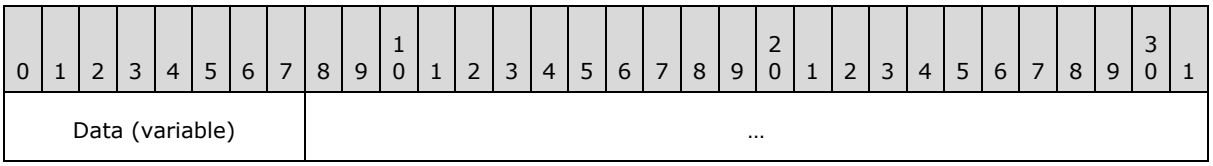
The **DecompressedBuffer** is a resizable array of bytes that contains the same data as the **CompressedContainer** (section 2.4.1.1.1), but the data is in an uncompressed format.



Chunk (variable): An array of **DecompressedChunk** (section 2.4.1.1.3) structures. The number of bytes in the last **DecompressedChunk** in a **DecompressedBuffer** (section 2.4.1.1.2) MUST be greater than zero. The number of bytes in the last **DecompressedChunk** in a **DecompressedBuffer** MUST be less than or equal to 4096. The number of bytes in all other **DecompressedChunks** MUST be 4096. Read by the **Compression** algorithm (section 2.4.1.3.6). Written by the **Decompression** algorithm (section 2.4.1.3.1).

2.4.1.1.3 DecompressedChunk

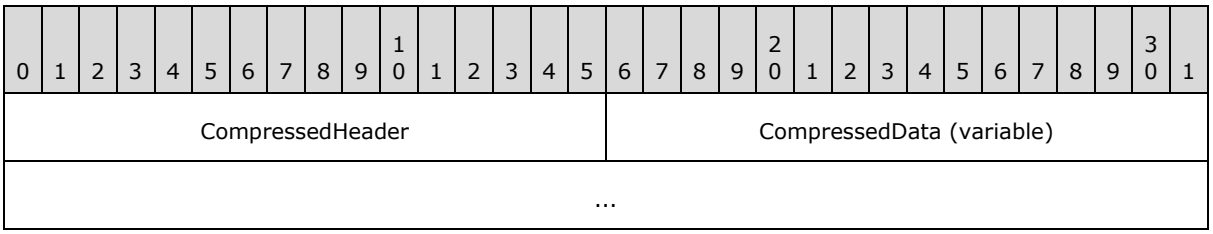
A **DecompressedChunk** is a resizable array of bytes in the **DecompressedBuffer** (section 2.4.1.1.2). The byte array is the data from a **CompressedChunk** (section 2.4.1.1.4) in uncompressed format.



Data (variable): An array of bytes. Each byte specifies a copy of one byte of the **DecompressedBuffer** (section 2.4.1.1.2).

2.4.1.1.4 CompressedChunk

A **CompressedChunk** is a record that encodes all data from a **DecompressedChunk** (section 2.4.1.1.3) in compressed format. A **CompressedChunk** has two parts: a **CompressedChunkHeader** (section 2.4.1.1.5) followed by a **CompressedChunkData** (section 2.4.1.1.6). The number of bytes in a **CompressedChunk** MUST be greater than or equal to 3. The number of bytes in a **CompressedChunk** MUST be less than or equal to 4098.

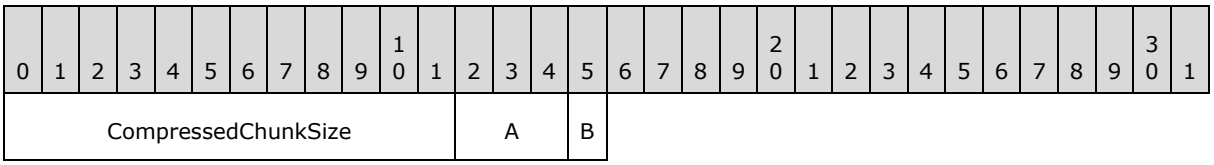


CompressedHeader (2 bytes): A **CompressedChunkHeader**. Read by the **Decompressing a CompressedChunk** algorithm (section 2.4.1.3.2). Written by the **Compressing a DecompressedChunk** algorithm (section 2.4.1.3.7).

CompressedData (variable): A **CompressedChunkData**. The size of **CompressedData** MUST be greater than zero. The size of **CompressedData** MUST be less than or equal to 4096. Read by the **Decompressing a CompressedChunk** algorithm. Written by the **Compressing a DecompressedChunk**.

2.4.1.1.5 CompressedChunkHeader

A **CompressedChunkHeader** is the first record in a **CompressedChunk** (section 2.4.1.1.4). A **CompressedChunkHeader** specifies the size of the entire **CompressedChunk** and the data encoding format in **CompressedChunk**.**CompressedData**. **CompressedChunkHeader** information is used by the **Decompressing a CompressedChunk** (section 2.4.1.3.2) and **Compressing a DecompressedChunk** (section 2.4.1.3.7) algorithms.



CompressedChunkSize (12 bits): An unsigned integer that specifies the number of bytes in the **CompressedChunk** minus 3. MUST be greater than or equal to zero. If **CompressedChunkFlag** is equal to 0b1, this element MUST be less than or equal to 4095. If **CompressedChunkFlag** is equal to 0b0, this element MUST be 4095. Read by the **Extract CompressedChunkSize** (section 2.4.1.3.12) algorithm. Written by the **Pack CompressedChunkSize** (section 2.4.1.3.13) algorithm.

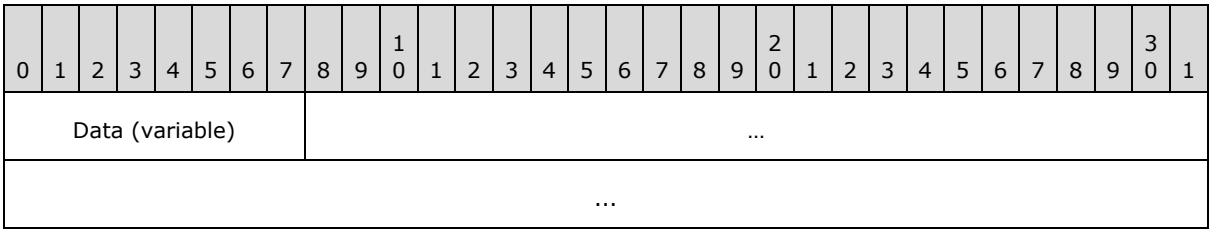
A – CompressedChunkSignature (3 bits): MUST be 0b011. Written by the **Pack CompressedChunkSignature** (section 2.4.1.3.14) algorithm.

B – CompressedChunkFlag (1 bit): A bit specifying how **CompressedChunk.CompressedData** is compressed. If this is 0b1, **CompressedChunk.CompressedData** is in compressed format. If this is 0b0, **CompressedChunk.CompressedData** contains uncompressed data. Read by the **Extract CompressedChunkFlag** (section 2.4.1.3.15) algorithm. Written by the **Pack CompressedChunkFlag** (section 2.4.1.3.16) algorithm.

2.4.1.1.6 CompressedChunkData

If **CompressedChunkHeader.CompressedChunkFlag** (section 2.4.1.1.5) is 0b0, **CompressedChunkData** contains an array of **CompressedChunkHeader.CompressedChunkSize** elements plus 3 bytes of uncompressed data.

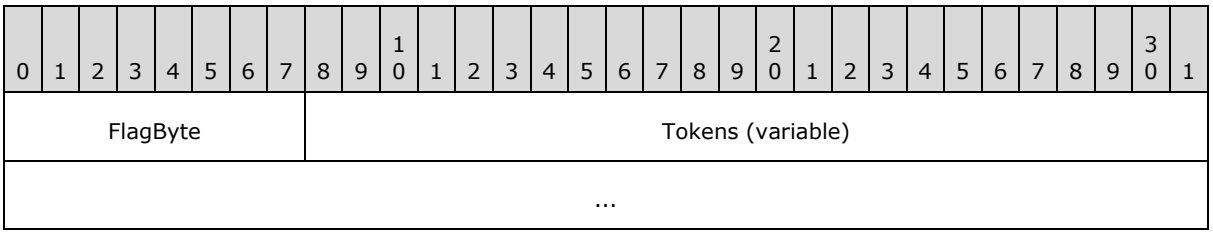
If **CompressedChunkHeader CompressedChunkFlag** is 0b1, **CompressedChunkData** contains an array of **TokenSequence** (section 2.4.1.1.7) elements.



Data (variable): An array of bytes. Specifies an encoding of bytes from the **DecompressedBuffer** (section 2.4.1.1.2). The size of **Data** in bytes MUST be **CompressedChunk.CompressedChunkHeader.CompressedChunkSize** (section 2.4.1.1.4) plus 3. Bytes from the **DecompressedChunk** (section 2.4.1.1.3) are encoded and written to **Data** by the **Compressing a DecompressedChunk** (section 2.4.1.3.7) algorithm. **Data** is read from the **CompressedChunk** to be decoded and written to the **DecompressedChunk** by the **Decompressing a CompressedChunk** (section 2.4.1.3.2) algorithm.

2.4.1.1.7 TokenSequence

A **TokenSequence** is a **FlagByte** followed by an array of **Token**s. The number of **Tokens** in the final **TokenSequence** MUST be greater than or equal to 1. The number of **Token**s in the final **TokenSequence** MUST less than or equal to eight. All other **TokenSequence**s in the **CompressedChunkData** MUST contain eight **Token**s.



FlagByte(1 byte): An array of bits. Each bit specifies the type of a **Token** in the **TokenSequence**. A value of 0b0 specifies a **LiteralToken**. A value of 0b1 specifies a **CopyToken** (section 2.4.1.1.8). The least significant bit in the **FlagByte** denotes the first

Token in the **TokenSequence**. The most significant bit in the **FlagByte** denotes the last **Token** in the **TokenSequence**. The correspondence between a **FlagByte** element and a **Token** element is maintained by the **Decompressing a TokenSequence** (section 2.4.1.3.4) and the **Compressing a TokenSequence** (section 2.4.1.3.8) algorithms.

Tokens (variable): An array of **Token**s. Each **Token** can either be a **LiteralToken** or a **CopyToken** as specified by the corresponding bit in **FlagByte**. A **LiteralToken** is a copy of one byte, in uncompressed format, from the **DecompressedBuffer** (section 2.4.1.1.2). A **CopyToken** is a 2-byte encoding of 3 or more bytes from the **DecompressedBuffer**. Read by the **Decompressing a TokenSequence** algorithm. Written by the **Compressing a TokenSequence** algorithm.

2.4.1.1.8 CopyToken

CopyToken is a two-byte record interpreted as an unsigned 16-bit integer in little-endian order. A **CopyToken** is a compressed encoding of an array of bytes from a **DecompressedChunk** (section 2.4.1.1.3). The byte array encoded by a **CopyToken** is a byte-for-byte copy of a byte array elsewhere in the same **DecompressedChunk**, called a **CopySequence** (section 2.4.1.3.19).

The starting location, in a **DecompressedChunk**, is determined by the **Compressing a Token** (section 2.4.1.3.9) and the **Decompressing a Token** (section 2.4.1.3.5) algorithms. Packed into the **CopyToken** is the **Offset**, the distance, in byte count, to the beginning of the **CopySequence**. Also packed into the **CopyToken** is the **Length**, the number of bytes encoded in the **CopyToken**. **Length** also specifies the count of bytes in the **CopySequence**. The values encoded in **Offset** and **Length** are computed by the **Matching** (section 2.4.1.3.19.4) algorithm.

variable	variable	16	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Length	Offset																

Length (variable): A variable bit unsigned integer that specifies the number of bytes contained in a **CopySequence** minus three. MUST be greater than or equal to zero. MUST be less than 4093. The number of bits used to encode **Length** MUST be greater than or equal to four. The number of bits used to encode **Length** MUST be less than or equal to 12. The number of bits used to encode **Length** is computed and used in the **Unpack CopyToken** (section 2.4.1.3.19.2) and the **Pack CopyToken** (section 2.4.1.3.19.3) algorithms.

Offset (variable): A variable bit unsigned integer that specifies the distance, in byte count, from the beginning of a duplicate set of bytes in the DecompressedBuffer to the beginning of a CopySequence. The value stored in **Offset** is the distance minus three. MUST be greater than zero. MUST be less than 4096. The number of bits used to encode **Offset** MUST be greater than or equal to four. The number of bits used to encode **Offset** MUST be less than or equal to 12. The number of bits used to encode **Offset** is computed and used in the **Unpack CopyToken** and the **Pack CopyToken** algorithms.

2.4.1.2 State Variables

The following state is maintained for the **CompressedContainer** (section 2.4.1.1.1):

CompressedRecordEnd: The location of the byte after the last byte in the **CompressedContainer** (section 2.4.1.1.1).

CompressedCurrent: The location of the next byte in the **CompressedContainer** (section 2.4.1.1.1) to be read by decompression or to be written by compression.

The following state is maintained for the current **CompressedChunk** (section 2.4.1.1.4):

CompressedChunkStart: The location of the first byte of the **CompressedChunk** (section 2.4.1.1.4) within the **CompressedContainer** (section 2.4.1.1.1).

The following state is maintained for a **DecompressedBuffer** (section 2.4.1.1.2):

DecompressedCurrent: The location of the next byte in the **DecompressedBuffer** (section 2.4.1.1.2) to be written by decompression or to be read by compression.

DecompressedBufferEnd: The location of the byte after the last byte in the **DecompressedBuffer** (section 2.4.1.1.2).

The following state is maintained for the current **DecompressedChunk** (section 2.4.1.1.3):

DecompressedChunkStart: The location of the first byte of the **DecompressedChunk** (section 2.4.1.1.3) within the **DecompressedBuffer** (section 2.4.1.1.2).

2.4.1.3 Algorithms

2.4.1.3.1 Decompression Algorithm

The Decompression algorithm pseudocode decodes the data in a **CompressedContainer** (section 2.4.1.1.1) and writes the uncompressed bytes to a **DecompressedBuffer** (section 2.4.1.1.2). The pseudocode first validates **CompressedContainer SignatureByte** (section 2.4.1.1.1). If validation fails, then the **CompressedContainer** (section 2.4.1.1.1) is corrupt and cannot be decoded. The pseudocode then iterates over the **CompressedChunks** (section 2.4.1.1.4). On each iteration, the current **CompressedChunk** is decoded.

The pseudocode to decompress the **CompressedContainer** (section 2.4.1.1.1) into the **DecompressedBuffer** (section 2.4.1.1.2) uses the state variables described in State Variables (section 2.4.1.2): **CompressedCurrent**, **CompressedRecordEnd**, and **DecompressedCurrent**. These state variables MUST be initialized by the caller. **CompressedChunkStart** is also used.

- IF the byte located at CompressedCurrent EQUALS 0x01 THEN
  - INCREMENT CompressedCurrent
  - WHILE CompressedCurrent is LESS THAN CompressedRecordEnd
    - SET CompressedChunkStart TO CompressedCurrent
    - CALL Decompressing a CompressedChunk
  - END WHILE
- ELSE
  - RAISE ERROR
- ENDIF

2.4.1.3.2 Decompressing a CompressedChunk

The **Decompressing a CompressedChunk** pseudocode decodes the data in a **CompressedChunk** (section 2.4.1.1.4) and writes the uncompressed bytes to the **DecompressedBuffer** (section 2.4.1.1.2).

The **Decompressing a CompressedChunk** pseudocode inspects **CompressedChunk.CompressedChunkHeader CompressedChunkFlag** (section 2.4.1.1.5) to determine the encoding format of **CompressedChunk CompressedData** (section 2.4.1.1.4), and then decodes the **CompressedChunkData** (section 2.4.1.1.6) using the format.

The pseudocode for **Decompressing a CompressedChunk** uses the state variables described in State Variables (section 2.4.1.2): **DecompressedChunkStart**, **DecompressedCurrent**, **CompressedRecordEnd**, **CompressedCurrent**, and **CompressedChunkStart**.

- SET Header TO the **CompressedChunkHeader** (section 2.4.1.1.5) located at **CompressedChunkStart**
- CALL **Extract CompressedChunkSize** (section 2.4.1.3.12) with Header returning **Size**
- CALL **Extract CompressedChunkFlag** (section 2.4.1.3.15) with Header returning **CompressedFlag**
- SET **DecompressedChunkStart** TO **DecompressedCurrent**
- SET **CompressedEnd** TO the minimum of **CompressedRecordEnd** and (**CompressedChunkStart** PLUS **Size**)
- SET **CompressedCurrent** TO **CompressedChunkStart** PLUS 2
- IF **CompressedFlag** EQUALS 1 THEN
  - WHILE **CompressedCurrent** is LESS THAN **CompressedEnd**
    - CALL **Decompressing a TokenSequence** (section 2.4.1.3.4) with **CompressedEnd**
  - END WHILE
- ELSE
  - CALL **Decompressing a RawChunk** (section 2.4.1.3.3)
- ENDIF

2.4.1.3.3 Decompressing a RawChunk

The **Decompressing a RawChunk** pseudocode is called when the **CompressedChunkFlag** of the current **CompressedChunk** (section 2.4.1.1.4) is 0b0. **CompressedChunk.CompressedChunkData** (section 2.4.1.1.6) MUST contain 4096 bytes of uncompressed data. The **Decompressing a RawChunk** pseudocode copies the uncompressed data to the **DecompressedBuffer** (section 2.4.1.1.2).

The pseudocode for **Decompressing a CompressedChunk** uses the state variables described in State Variables (section 2.4.1.2): **DecompressedCurrent**, **CompressedCurrent**.

- APPEND 4096 bytes from CompressedCurrent TO DecompressedCurrent
- INCREMENT DecompressedCurrent BY 4096
- INCREMENT CompressedCurrent BY 4096

2.4.1.3.4 Decompressing a TokenSequence

The pseudocode for **Decompressing a TokenSequence** decodes the compressed data in a single **TokenSequence** (section 2.4.1.1.7) out of a **CompressedChunk** (section 2.4.1.1.4). The uncompressed data is written to the **DecompressedBuffer** (section 2.4.1.1.2).

The pseudocode for **Decompressing a TokenSequence** take the following input parameter.

CompressedEnd: Specifies the location of the byte after the last byte in the current **CompressedChunk** (section 2.4.1.1.4).

The pseudocode for **Decompressing a TokenSequence** uses the state variable described in State Variables (section 2.4.1.2): **CompressedCurrent**.

- SET Byte TO the **FlagByte** (section 2.4.1.1.7) located at CompressedCurrent
- INCREMENT CompressedCurrent
- IF CompressedCurrent is LESS THAN CompressedEnd THEN
  - FOR index FROM 0 TO 7 INCLUSIVE
    - IF CompressedCurrent is LESS THAN CompressedEnd THEN
      - CALL **Decompressing a Token** (section 2.4.1.3.5) with index and Byte
    - ENDIF
  - ENDFOR
- ENDIF

2.4.1.3.5 Decompressing a Token

The **Decompressing a Token** pseudocode decodes a single token producing uncompressed data. The uncompressed data is written to the **DecompressedBuffer** (section 2.4.1.1.2).

The **Decompressing a Token** pseudocode takes the following input parameters:

Index: An unsigned integer that specifies the element of a **TokenSequence** (section 2.4.1.1.7) to decompress. MUST be greater than or equal to zero. MUST be less than or equal to 7.

Byte (1 byte): The FlagByte of the current **TokenSequence** (section 2.4.1.1.7).

The pseudocode for decompressing a token uses the state variables described in State Variables (section 2.4.1.2): **CompressedCurrent**, **DecompressedCurrent**.

- CALL **Extract FlagBit** (section 2.4.1.3.17) with index and Byte returning Flag
- IF Flag EQUALS 0 THEN
  - COPY the byte at CompressedCurrent TO DecompressedCurrent
  - INCREMENT DecompressedCurrent
  - INCREMENT CompressedCurrent
- ELSE

- SET Token TO the **CopyToken** (section 2.4.1.1.8) at CompressedCurrent
- CALL **Unpack CopyToken** (section 2.4.1.3.19.2) with Token returning Offset and Length
- SET CopySource TO DecompressedCurrent MINUS Offset
- CALL **Byte Copy** (section 2.4.1.3.11) with CopySource, DecompressedCurrent, and Length
- INCREMENT DecompressedCurrent BY Length
- INCREMENT CompressedCurrent BY 2

- ENDIF

2.4.1.3.6 Compression algorithm

The pseudocode for the **Compression algorithm** uses the state variables described in State Variables (section 2.4.1.2): **DecompressedCurrent**, **DecompressedBufferEnd**, and **CompressedCurrent**. These state variables MUST be initialized by the caller. **CompressedChunkStart** and **DecompressedChunkStart** are also used.

- SET SignatureByte TO 0x01
- INCREMENT CompressedCurrent
- WHILE DecompressedCurrent is LESS THAN DecompressedBufferEnd
  - SET CompressedChunkStart TO CompressedCurrent
  - SET DecompressedChunkStart TO DecompressedCurrent
  - CALL **Compressing a DecompressedChunk** (section 2.4.1.3.7)
- END WHILE

2.4.1.3.7 Compressing a DecompressedChunk

The pseudocode to compress a **DecompressedChunk** (section 2.4.1.1.3) to a **CompressedChunk** (section 2.4.1.1.4) uses the state variables described in State Variables (section 2.4.1.2): **CompressedChunkStart, CompressedCurrent, DecompressedChunkStart, DecompressedBufferEnd**, and **DecompressedCurrent**.

- SET CompressedEnd TO CompressedChunkStart PLUS 4098
- SET CompressedCurrent TO the CompressedChunkStart PLUS 2
- SET DecompressedEnd TO the minimum of (DecompressedChunkStart PLUS 4096) and DecompressedBufferEnd
- WHILE (DecompressedCurrent is LESS THAN DecompressedEnd) AND (CompressedCurrent is LESS THAN CompressedEnd)
  - CALL Compressing a TokenSequence (section 2.4.1.3.8) with CompressedEnd and DecompressedEnd
- END WHILE
- IF DecompressedCurrent is LESS THAN DecompressedEnd THEN

- CALL Compressing a RawChunk (section 2.4.1.3.10) with DecompressedEnd MINUS 1
- SET CompressedFlag TO 0

- ELSE
- SET CompressedFlag TO 1

- ENDIF
- SET Size TO CompressedCurrent MINUS CompressedChunkStart
- SET Header TO 0x0000
- CALL Pack CompressedChunkSize (section 2.4.1.3.13) with Size and Header
- CALL Pack CompressedChunkFlag (section 2.4.1.3.16) with CompressedFlag and Header
- CALL Pack CompressedChunkSignature (section 2.4.1.3.14) with Header
- SET the CompressedChunkHeader (section 2.4.1.1.5) located at CompressedChunkStart TO Header

2.4.1.3.8 Compressing a TokenSequence

The **Compressing a TokenSequence** pseudocode encodes a sub array of the **DecompressedChunk** (section 2.4.1.1.3) into a **TokenSequence** (section 2.4.1.1.7). The TokenSequence is written to the **CompressedChunk** (section 2.4.1.1.4). The location of the **FlagByte** of the **TokenSequence** (section 2.4.1.1.7) is reserved and then the Compressing a Token algorithm (section 2.4.1.3.9) is called to manufacture the individual **Tokens**. After the encoding of each **Token** has been computed, the **FlagByte** is updated.

The **Compressing a TokenSequence** pseudocode takes the following input parameters.

CompressedEnd: The location of the next byte after the end of the current **CompressedChunk** (section 2.4.1.1.4).

DecompressedEnd: The location of the next byte after the end of the current **DecompressedChunk** (section 2.4.1.1.3).

The pseudocode for **Compressing a TokenSequence** uses the state variables described in State Variables (section 2.4.1.2): **CompressedCurrent** and **DecompressedCurrent**.

- SET FlagByteIndex TO CompressedCurrent
- SET TokenFlags TO 0b00000000
- INCREMENT CompressedCurrent
- FOR index FROM 0 TO 7 INCLUSIVE
  - IF (DecompressedCurrent is LESS THAN DecompressedEnd)
    - AND (CompressedCurrent is LESS THAN CompressedEnd) THEN
      - CALL Compressing a Token with CompressedEnd, DecompressedEnd,
        - index, and TokenFlags,

        - returning TokenFlags
    - ENDIF
- ENDFOR
- SET the byte at location FlagByteIndex TO TokenFlags

2.4.1.3.9 Compressing a Token

The **Compressing a Token** pseudocode uses the Matching algorithm (section 2.4.1.3.19.4) to determine the type of **Token** that can be placed at **CompressedCurrent**, manufactures the **Token**, and places the **Token** in the **CompressedChunk** (section 2.4.1.1.4) at **CompressedCurrent**. If placing the **Token** at **CompressedCurrent** would exceed the boundaries of the current **CompressedChunk** (section 2.4.1.1.4), the **Token** is not inserted and **CompressedCurrent** is set to a value that will signal calling algorithms that the **CompressedChunk** (section 2.4.1.1.4) is full.

The **Compressing a Token** pseudocode takes the following input parameter.

CompressedEnd: The location of the next byte after the end of the current **CompressedChunk** (section 2.4.1.1.4).

DecompressedEnd: The location of the first byte after the end of the **DecompressedChunk** (section 2.4.1.1.3).

Index: An integer that specifies the ordinal of the **Token** within a **TokenSequence** (section 2.4.1.1.7) being encoded. MUST be greater than or equal to 0. MUST be less than or equal to 7.

The **Compressing a Token** pseudocode takes the following input/output parameter.

Flags(1 byte): The **FlagByte** of the current **TokenSequence** (section 2.4.1.1.7).

The pseudocode for **Compressing a Token** uses the state variables described in State Variables (section 2.4.1.2): **CompressedCurrent** and **DecompressedCurrent**.

- SET Offset TO zero
- CALL Matching (section 2.4.1.3.19.4) with DecompressedEnd returning Offset and Length
- IF Offset is not zero THEN
    - IF (CompressedCurrent PLUS 1) is LESS THAN CompressedEnd THEN
        - CALL Pack CopyToken (section 2.4.1.3.19.3) with Offset and Length returning Token
        - APPEND the bytes of the CopyToken (section 2.4.1.1.8) Token TO CompressedCurrent in little-endian order
        - CALL Set FlagBit (section 2.4.1.3.18) with index, 1, and Flags
        - INCREMENT CompressedCurrent BY 2
        - INCREMENT DecompressedCurrent BY Length
    - ELSE
        - SET CompressedCurrent TO CompressedEnd

- ENDIF
- ELSE
  - IF CompressedCurrent is LESS THAN CompressedEnd THEN
    - APPEND the byte of the LiteralToken at DecompressedCurrent
      - TO CompressedCurrent
    - INCREMENT CompressedCurrent
    - INCREMENT DecompressedCurrent
  - ELSE
    - SET CompressedCurrent TO CompressedEnd
  - ENDIF
- ENDIF

2.4.1.3.10 Compressing a RawChunk

The **Compressing a RawChunk** pseudocode is called when the number of bytes in a **CompressedChunk.CompressedData** (section 2.4.1.1.4) array exceeds 4096. The bytes from the **DecompressedChunk** (section 2.4.1.1.3) are copied, with no compression, into **CompressedChunk.CompressedData** (section 2.4.1.1.4). If fewer than 4096 bytes are copied then the remaining bytes in **CompressedChunk.CompressedData.Data** array are padded with the literal value 0x00.<11>

Compressing a RawChunk takes the following input parameter.

LastByte: Specifies the location of the last byte of the **DecompressedChunk**.

The pseudocode for **Compressing a RawChunk** uses the state variables described in State Variables (section 2.4.1.2): **CompressedCurrent, CompressedChunkStart, DecompressedChunkStart,** and **DecompressedCurrent**.

- SET CompressedCurrent TO CompressedChunkStart PLUS 2
- SET DecompressedCurrent TO DecompressedChunkStart
- SET PadCount TO 4096
- FOR each byte, B, FROM DecompressedChunkStart TO LastByte INCLUSIVE
  - COPY B TO CompressedCurrent
  - INCREMENT CompressedCurrent
  - INCREMENT DecompressedCurrent
  - DECREMENT PadCount
- ENDFOR
- FOR counter FROM 1 TO PadCount INCLUSIVE

- COPY 0x00 TO CompressedCurrent
- INCREMENT CompressedCurrent

- ENDFOR

2.4.1.3.11 Byte Copy

The **Byte Copy** pseudocode will copy a source sequence of bytes to a destination sequence of bytes. The source and destination sequences are allowed to overlap; thus it is possible for the **Byte Copy** operation to modify bytes in the source sequence.

Byte copy takes the following input parameters:

CopySource: Specifies the location, in the **DecompressedBuffer**, of the first byte of the source sequence.

DestinationSource: Specifies the location, in the **DecompressedBuffer**, of the first byte of the destination sequence.

ByteCount: Specifies the number of bytes to copy. MUST be greater than 0.

The pseudocode follows:

- SET SrcCurrent TO CopySource
- SET DstCurrent TO DestinationSource
- FOR counter FROM 1 TO ByteCount INCLUSIVE
  - COPY the byte at SrcCurrent TO DstCurrent
  - INCREMENT SrcCurrent
  - INCREMENT DstCurrent
- ENDFOR

2.4.1.3.12 Extract CompressedChunkSize

The Extract CompressedChunkSize pseudocode is used to unpack the size of a **CompressedChunk** (section 2.4.1.1.4) from its **CompressedChunkHeader** (section 2.4.1.1.5). The pseudocode takes the following input parameter:

Header (2 bytes): An instance of a **CompressedChunkHeader** (section 2.4.1.1.5).

The Extract CompressedChunkSize pseudocode takes the following output parameter:

Size (2 bytes): An unsigned 16-bit integer. The number of bytes in the **CompressedChunk** (section 2.4.1.1.4) MUST be less than or equal to 4098. MUST be greater than or equal to three.

- SET temp TO Header BITWISE AND 0x0FFF
- SET Size TO temp PLUS 3

2.4.1.3.13 Pack CompressedChunkSize

Pack CompressedChunkSize pseudocode takes the following input parameters:

Size: An unsigned 16-bit integer. The number of bytes in the **CompressedChunk** (section 2.4.1.1.4). MUST be less than or equal to 4098. MUST be greater than or equal to three.

Pack CompressedChunkSize pseudocode take the following input/output parameter:

Header: An instance of a **CompressedChunkHeader** (section 2.4.1.1.5).

- SET temp1 TO Header BITWISE AND 0xF000
- SET temp2 TO Size MINUS 3
- SET Header TO temp1 BITWISE OR temp2

2.4.1.3.14 Pack CompressedChunkSignature

Pack CompressedChunkSignature sets the **CompressedChunkSignature** of a **CompressedChunkHeader** (section 2.4.1.1.5) to 0b011.

The **Pack CompressedChunkSignature** pseudocode takes the following input/output parameter:

Header (2 bytes): An instance of a **CompressedChunkHeader** (section 2.4.1.1.5).

- SET temp TO Header BITWISE AND 0x8FFF
- SET Header TO temp BITWISE OR 0x3000

2.4.1.3.15 Extract CompressedChunkFlag

The **Extract CompressedChunkFlag** pseudocode takes the following input parameter:

Header (2 bytes): An instance of a **CompressedChunkHeader** (section 2.4.1.1.5).

The **Extract CompressedChunkFlag** pseudocode takes the following output parameter:

CompressedFlag: An unsigned integer. The value returned MUST be zero or one.

- SET temp TO Header BITWISE AND 0x8000
- SET CompressedFlag TO temp RIGHT SHIFT BY 15

2.4.1.3.16 Pack CompressedChunkFlag

The **Pack CompressedChunkFlag** pseudocode takes the following input parameter:

CompressedFlag: An unsigned integer. MUST be zero or one.

The **Pack CompressedChunkFlag** pseudocode takes the following input/output parameter:

Header (2 bytes): An instance of a **CompressedChunkHeader** (section 2.4.1.1.5).

- SET temp1 TO Header BITWISE AND 0x7FFF
- SET temp2 TO CompressedFlag LEFT SHIFT BY 15
- SET Header TO temp1 BITWISE OR temp2

2.4.1.3.17 Extract FlagBit

The **Extract FlagBit** pseudocode takes the following input parameters:

Index: An unsigned integer specifying which FlagBit to extract. MUST be greater than or equal to zero and less than eight.

Byte (1 byte): An instance of a **FlagByte**.

The **Extract FlagBit** pseudocode returns the following output parameters:

Flag: An integer. The value of the bit in **Byte** at location **Index**. The value returned MUST be zero or one.

- SET Flag TO (Byte RIGHT SHIFT BY Index) BITWISE AND 1

2.4.1.3.18 Set FlagBit

The **Set FlagBit** pseudocode sets a specified bit in a FlagByte to 0b0 or 0b1.

The **Set FlagBit** pseudocode takes the following input parameters:

Index: An unsigned integer specifying which FlagBit to set. MUST be greater than or equal to zero. MUST be less than eight.

Flag: An integer. Specifies the bit value to set at location **Index** in **Byte**. MUST be zero or one.

The **Set FlagBit** pseudocode takes the following input/output parameters:

Byte (1 byte): An instance of a **FlagByte**.

- SET temp1 TO Flag LEFT SHIFT BY Index
- SET temp2 TO Byte BITWISE AND (BITWISE NOT temp1)
- SET Byte TO temp2 BITWISE OR temp1

2.4.1.3.19 CopyToken Algorithms

Packed into a **CopyToken** (section 2.4.1.1.8) are an **Offset** value and a **Length** value. The **Offset**, **Length** pair specify the start and length of a sequence of bytes, called a **CopySequence**, in the **DecompressedChunk**. A **CopySequence** is an array of bytes in the **DecompressedChunk** (section 2.4.1.1.3) that are duplicated starting at **DecompressedCurrent**. The **Matching algorithm** (section 2.4.1.3.19.4) will search for a **CopySequence**.

The start of a **CopySequence** MUST be before **DecompressedCurrent**. The start of the **CopySequence** MUST be at or after **DecompressedChunkStart**. The number of bytes in a **CopySequence** MUST be greater than or equal to three. The number of bytes in a **CopySequence** MUST be less than 4096.

Offset specifies the start of the **CopySequence**. **Offset** is the difference between **DecompressedCurrent** and the start of the **CopySequence** minus one. **Length** is the number of bytes minus three in the **CopySequence**.

The number of bits used to pack **Offset** and **Length** is a function of the relationship between **DecompressedCurrent** and **DecompressedChunkStart** as specified as:

DecompressedCurrent minus DecompressedChunkStart	Number of bits used to pack Length	Largest possible value for Length	Number of bits used to pack Offset
1 to 16	12	4098	4
17 to 32	11	2050	5
33 to 64	10	1026	6
65 to 128	9	514	7
129 to 256	8	258	8
257 to 512	7	130	9
513 to 1024	6	66	10
1025 to 2048	5	34	11
2049 to 4096	4	18	12

The **CopyToken Help algorithm** (section 2.4.1.3.19.1) returns values that are used by the **Unpack CopyToken** (section 2.4.1.3.19.2) and **Pack CopyToken** (section 2.4.1.3.19.3) algorithms to manipulate the **Offset** and **Length** fields of a **CopyToken**.

2.4.1.3.19.1 CopyToken Help

CopyToken Help derived bit masks are used by the **Unpack CopyToken** (section 2.4.1.3.19.2) and the **Pack CopyToken** (section 2.4.1.3.19.3) algorithms. **CopyToken Help** also derives the maximum length for a **CopySequence** (section 2.4.1.3.19) which is used by the **Matching algorithm** (section 2.4.1.3.19.4)**.**

The pseudocode uses the state variables described in State Variables (section 2.4.1.2): **DecompressedCurrent** and **DecompressedChunkStart**.

The pseudocode for **CopyToken Help** returns the following output parameters:

LengthMask (2 bytes): An unsigned 16-bit integer. A bitmask used to access **CopyToken**.**Length**.

OffsetMask (2 bytes): An unsigned 16-bit integer. A bitmask used to access **CopyToken**.**Offset**.

BitCount (2 bytes): An unsigned 16-bit integer. The number of bits set to 0b1 in **OffsetMask**.

MaximumLength (2 bytes): An unsigned 16-bit integer. The largest possible integral value that can fit into **CopyToken**.**Length**.

- SET difference TO DecompressedCurrent MINUS DecompressedChunkStart
- SET BitCount TO the smallest integer that is GREATER THAN OR EQUAL TO LOGARITHM base 2 of difference
- SET BitCount TO the maximum of BitCount and 4
- SET LengthMask TO 0xFFFF RIGHT SHIFT BY BitCount
- SET OffsetMask TO BITWISE NOT LengthMask

- SET MaximumLength TO (0xFFFF RIGHT SHIFT BY BitCount) PLUS 3

2.4.1.3.19.2 Unpack CopyToken

The **Unpack CopyToken** pseudocode will compute the specifications of a **CopySequence** (section 2.4.1.3.19) that are encoded in a CopyToken.

The pseudocode for **Unpack CopyToken** takes the following input parameters:

Token (2 bytes): A **CopyToken** (section 2.4.1.1.8).

The pseudocode takes the following output parameters:

Offset (2 bytes): An unsigned 16-bit integer that specifies the beginning of a **CopySequence** (section 2.4.1.3.19).

Length (2 bytes): An unsigned 16-bit integer that specifies the length of a **CopySequence** (section 2.4.1.3.19) as follows:

1. CALL **CopyToken Help** (section 2.4.1.3.19.1) returning **LengthMask**, **OffsetMask**, and **BitCount**.
2. SET **Length** TO (Token **BITWISE** AND **LengthMask**) PLUS 3.
3. SET **temp1** TO Token **BITWISE** AND **OffsetMask**.
4. SET **temp2** TO 16 MINUS **BitCount**.
5. SET **Offset** TO (**temp1** RIGHT SHIFT BY **temp2**) PLUS 1.

2.4.1.3.19.3 Pack CopyToken

The **Pack CopyToken** pseudocode will take the Offset and Length values that specify a **CopySequence** (section 2.4.1.3.19) and pack them into a **CopyToken** (section 2.4.1.1.8).

The **Pack CopyToken** pseudocode takes the following input parameters:

Offset (2 bytes): An unsigned 16-bit integer that specifies the beginning of a **CopySequence** (section 2.4.1.3.19).

Length (2 bytes): An unsigned 16-bit integer that specifies the length of a **CopySequence** (section 2.4.1.3.19).

The **Pack CopyToken** pseudocode takes the following output parameters:

Token (2 bytes): A **CopyToken** (section 2.4.1.1.8).

- CALL **CopyToken Help** (section 2.4.1.3.19.1) returning LengthMask, OffsetMask, and BitCount
- SET temp1 TO Offset MINUS 1
- SET temp2 TO 16 MINUS BitCount
- SET temp3 TO Length MINUS 3
- SET Token TO (temp1 LEFT SHIFT BY temp2) BITWISE OR temp3

2.4.1.3.19.4 Matching

The **Matching** pseudocode is used to search for a **CopySequence** (section 2.4.1.3.19) in a **DecompressedChunk** (section 2.4.1.1.3), based on an array of bytes in the same DecompressedChunk. The pseudocode uses the state variables described in State Variables (section 2.4.1.2): **DecompressedCurrent**, and **DecompressedChunkStart**.

The Matching pseudocode takes the following input parameters:

DecompressedEnd: Specifies the location of the byte after the last byte in the current **DecompressedChunk**.

The **Matching** pseudocode returns the following output parameters:

Offset: If a match is found, then the number of bytes between the start of the **CopySequence** (section 2.4.1.3.19) and **DecompressedCurrent**. If a match is not found, then zero.

Length: If a match is found, then the number of bytes in the **CopySequence** (section 2.4.1.3.19). If a match is not found, then zero.

- SET Candidate TO DecompressedCurrent MINUS 1
- SET BestLength TO 0
- WHILE Candidate is GREATER THAN OR EQUAL TO DecompressedChunkStart
  - SET C TO Candidate
  - SET D TO DecompressedCurrent
  - SET Len TO 0
  - WHILE (D is LESS THAN DecompressedEnd)
  - and (the byte at D EQUALS the byte at C)
    - INCREMENT Len
    - INCREMENT C
    - INCREMENT D
  - END WHILE
  - IF Len is GREATER THAN BestLength THEN
    - SET BestLength TO Len
    - SET BestCandidate TO Candidate
  - ENDIF
  - DECREMENT Candidate
- END WHILE
- IF BestLength is GREATER THAN OR EQUAL TO 3 THEN
  - CALL **CopyToken Help** (section 2.4.1.3.19.1) returning MaximumLength

- SET Length TO the MINIMUM of BestLength and MaximumLength
- SET Offset TO DecompressedCurrent MINUS BestCandidate

- ELSE
  - SET Length TO 0
  - SET Offset TO 0
- ENDIF

2.4.2 Contents Hash

The Contents Hash is a cryptographic **digest** of a subset of the information stored in the VBA Storage (section 2.3.4).

Conventions:

- APPEND specifies appending the bytes of a field to the end of a resizable array of bytes.
- APPEND specifies appending the MBCS bytes of a string without null termination to the end of a resizable array of bytes.
- FOR each specifies iteration over a collection of records in their stored order.

This Contents Hash algorithm requires one parameter as input:

VBAStorage(Variable): The VBA Storage (section 2.3.4) to calculate a hash for.

The Contents Hash algorithm produces an array of bytes as output:

CryptographicDigest(16 bytes): The cryptographic digest of **VBAStorage**.

CryptographicDigest is generated by the following pseudocode:

- SET Buffer TO a resizable array of bytes
- APPEND Buffer WITH PROJECTNAME**.ProjectName** (section 2.3.4.2.1.5)
- APPEND Buffer WITH PROJECTCONSTANTS**.ProjectConstants** (section 2.3.4.2.1.11)
- FOR each REFERENCE (section 2.3.4.2.2.1) IN PROJECTREFERENCES**.ReferenceArray** (section 2.3.4.2.2)
- IF Ref.ReferenceRecord.Id = 0x000D THEN
  - APPEND Buffer WITH 0x7B
- END IF
- IF Ref.ReferenceRecord.Id = 0x000E THEN
  - SET TempBuffer TO a resizable array of bytes
  - APPEND TempBuffer WITH Ref.ReferenceRecord.SizeOfAbsoluteLibId
  - APPEND TempBuffer WITH Ref.ReferenceRecord.AbsoluteLibId

- APPEND TempBuffer WITH Ref.ReferenceRecord.SizeOfRelativeLibId
- APPEND TempBuffer WITH Ref.ReferenceRecord.RelativeLibId
- APPEND TempBuffer WITH Ref.ReferenceRecord.MajorVersion
- APPEND TempBuffer WITH Ref.ReferenceRecord.MinorVersion
- APPEND TempBuffer WITH 0x00
- SET CopyIndex TO 0
- SET CopyByte TO TempBuffer[CopyIndex]
- WHILE NOT CopyByte EQUALS 0x00
  - APPEND Buffer WITH CopyByte
  - INCREMENT CopyIndex
  - SET CopyByte TO TempBuffer[CopyIndex]
- END WHILE

- END IF
- END FOR
- FOR each ModuleStream ModStream (section 2.3.4.3) IN VBA Storage (section 2.3.4)
- SET CompressedContainer TO ModStream.CompressedSourceCode
- CALL Decompression (section 2.4.1) with CompressedContainer RETURNING DecompressedBuffer
- SET Text TO the string representation of the bytes in DecompressedBuffer
- SET Lines TO a resizable array of strings
- SET TextBuffer TO ""
- FOR each character Char IN Text
  - IF Char is Carriage Return or Line Feed THEN
    - ADD TextBuffer TO the end of Lines
    - SET TextBuffer TO ""
  - ELSE
    - ADD Char TO the end of TextBuffer
  - END IF
- END FOR
- FOR each Line IN Lines
  - IF Line does not start with "Attribute " ignoring case THEN

- APPEND Buffer WITH Line

- END IF

- END FOR

- END FOR

- SET CryptographicDigest TO the cryptographic digest of Buffer as specified by the MD5 algorithm in [RFC1321].

2.4.3 Data Encryption

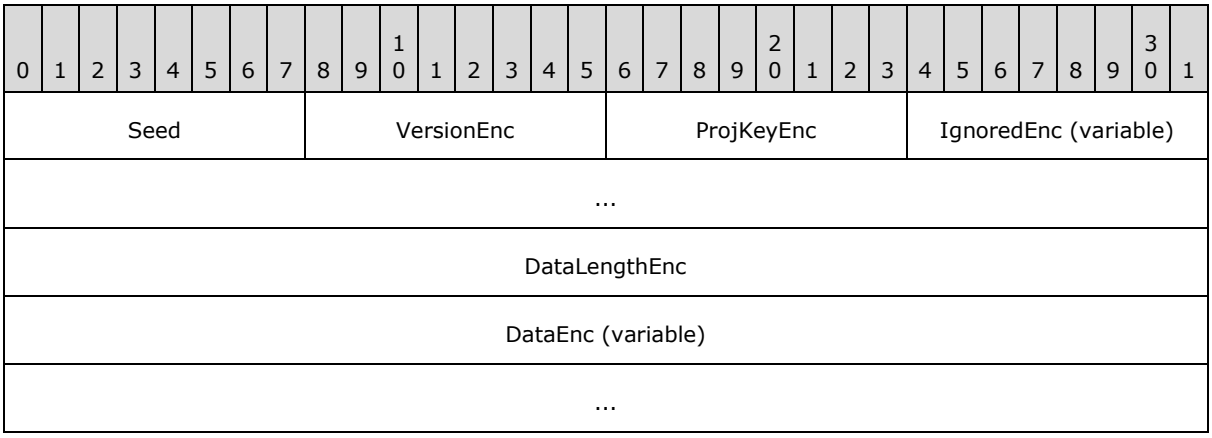
VBA uses a reversible encryption algorithm for selected data.

Conventions:

- XOR specifies a bit-wise exclusive OR operation.
- BAND specifies a bit-wise AND operation.
- All operations resulting in integer overflow MUST only store low-order bits, resulting in high-order bit truncation.

2.4.3.1 Encrypted Data Structure

Specifies encrypted data. This structure has the following format.



Seed (1 byte): Specifies the encryption seed.

VersionEnc (1 byte): Encrypted as specified in section 2.4.3.2. Specifies the encryption version.

ProjKeyEnc (1 byte): Encrypted as specified in section 2.4.3.2. Specifies the project-specific encryption key.

IgnoredEnc (variable): Encrypted as specified in section 2.4.3.2. An array of arbitrary bytes for obfuscation.

DataLengthEnc (4 bytes): Encrypted as specified in section 2.4.3.2. Specifies the length in bytes of **DataEnc**.

DataEnc (variable): Encrypted as specified in section 2.4.3.2. Specifies the data encrypted by the algorithm.

2.4.3.2 Encryption

This encryption accepts two parameters as input:

Data (Variable): An array of bytes to be obfuscated.

Length (4 bytes): An unsigned integer that specifies the length of **Data**.

The algorithm will yield an array of bytes as defined in Encrypted Data Structure (section 2.4.3.1).

To encrypt **Data**, an implementation MUST maintain the following states:

UnencryptedByte1 (1 byte): Specifies the last unencrypted byte read or written.

EncryptedByte1 (1 byte): Specifies the last encrypted byte read or written.

EncryptedByte2 (1 byte): Specifies the next-to-last encrypted byte read or written.

Version (1 byte): Specifies the encryption version.

ProjKey (1 byte): Specifies a project-specific encryption key.

IgnoredLength (1 byte): Specifies the length in bytes of **IgnoredEnc**.

Each field MUST be encrypted in the following order:

VersionEnc is calculated using the following formula:

```
VersionEnc = Seed XOR Version
```

Version MUST be 2.

ProjKey is the checksum of the project identifier as computed by the following pseudocode:

- SET **ProjKey** TO 0.
- FOR each **CharacterByte** IN the string ProjectId.ProjectCLSID (section 2.3.1.2).
  - ADD **CharacterByte** TO **ProjKey**.
- END FOR

ProjKeyEnc is calculated using the following formula:

```
ProjKeyEnc = Seed XOR ProjKey
```

Initialize states for the rest of the encoding:

- SET **UnencryptedByte1** TO **ProjKey**.
- SET **EncryptedByte1** TO **ProjKeyEnc**.
- SET **EncryptedByte2** TO **VersionEnc** .

IgnoredEnc is computed by the following pseudocode:

- SET **IgnoredLength** TO `(Seed BAND 6) / 2`.
- FOR Counter FROM 1 TO **IgnoredLength** INCLUSIVE:
  - SET **TempValue** TO any value.
  - SET **ByteEnc** TO (**TempValue** XOR `(EncryptedByte2 + UnencryptedByte1))`.
  - APPEND **IgnoredEnc** WITH **ByteEnc**.
  - SET **EncryptedByte2** TO **EncryptedByte1**.
  - SET **EncryptedByte1** TO **ByteEnc**.
  - SET **UnencryptedByte1** TO **TempValue**.
- END FOR

DataLengthEnc is computed by the following pseudocode:

- FOR each **Byte** IN **Length** in little endian order:
  - SET **ByteEnc** TO `(Byte XOR (EncryptedByte2 + UnencryptedByte1))`.
  - APPEND **DataLengthEnc** WITH **ByteEnc**.
  - SET **EncryptedByte2** TO **EncryptedByte1**.
  - SET **EncryptedByte1** TO **ByteEnc**.
  - SET **UnencryptedByte1** TO **Byte**.
- END FOR

DataEnc is computed by the following pseudocode:

- FOR each **DataByte** IN **Data**:
  - SET **ByteEnc** TO `(DataByte XOR (EncryptedByte2 + UnencryptedByte1))`.
  - APPEND **DataEnc** WITH **ByteEnc**.
  - SET **EncryptedByte2** TO **EncryptedByte1**.
  - SET **EncryptedByte1** TO **ByteEnc**.
  - SET **UnencryptedByte1** TO **DataByte**.
- END FOR

2.4.3.3 Decryption

This decryption algorithm accepts an Encrypted Data Structure (section 2.4.3.1) as input and will yield:

Length (4 bytes): An unsigned integer that specifies the length of **Data**.

Data (variable): An array of unencrypted bytes.

To decrypt **Data** from an Encrypted Data Structure (section 2.4.3.1), an implementation MUST maintain the following states:

UnencryptedByte1 (1 byte): Specifies the last unencrypted byte read or written.

EncryptedByte1 (1 byte): Specifies the last encrypted byte read or written.

EncryptedByte2 (1 byte): Specifies the next-to-last encrypted byte read or written.

Version (1 byte): Specifies the encryption version.

ProjKey (1 byte): Specifies a project-specific encryption key.

IgnoredLength (1 byte): Specifies the length in bytes of **IgnoredEnc**.

MUST decrypt in order as follows.

Version is calculated using the following formula.

```
Version = Seed XOR VersionEnc
```

Version MUST be 2.

ProjKey is calculated using the following formula.

```
ProjKey = Seed XOR ProjKeyEnc
```

To initialize states for the rest of the encoding:

- SET **UnencryptedByte1** TO **ProjKey**.
- SET **EncryptedByte1** TO **ProjKeyEnc**.
- SET **EncryptedByte2** TO **VersionEnc**.

The length of **IgnoredEnc** is computed as follows.

```
IgnoredLength = (Seed BAND 6) / 2
```

Decrypting of **IgnoredEnc** MUST be as follows.

- FOR each **ByteEnc** IN **IgnoredEnc**:
  - SET **Byte** TO `(ByteEnc XOR (EncryptedByte2 + UnencryptedByte1))`.
  - SET **EncryptedByte2** TO **EncryptedByte1**.
  - SET **EncryptedByte1** TO **ByteEnc**.
  - SET **UnencryptedByte1 TO** Byte.
- END FOR

Length is computed by the following pseudocode.

- SET **ByteIndex** TO zero.
- FOR each **ByteEnc** IN **DataLengthEnc**:
  - SET **Byte** TO `(ByteEnc XOR (EncryptedByte2 + UnencryptedByte1))`.
  - SET **TempValue** TO 256 raised to the power of **ByteIndex**.
  - MULTIPLY **TempValue** by **Byte**.
  - ADD **TempValue** TO **Length**.
  - SET **EncryptedByte2** TO **EncryptedByte1**.
  - SET **EncryptedByte1** TO **ByteEnc**.
  - SET **UnencryptedByte1** TO **Byte**.
  - INCREMENT **ByteIndex**
- END FOR

Length is equal to the length of **DataEnc**.

Data is computed using the following pseudocode.

- FOR each **ByteEnc** IN **DataEnc**:
  - SET **Byte** TO `(ByteEnc XOR (EncryptedByte2 + UnencryptedByte1))`.
  - APPEND **Data** WITH **Byte**.
  - SET **EncryptedByte2** TO **EncryptedByte1**.
  - SET **EncryptedByte1** TO **ByteEnc**.
  - SET **UnencryptedByte1** TO **Byte**.
- END FOR

2.4.4 Password Hash

VBA employs a custom format for storing a password hash, obfuscating the password with random data. That random data is stored with the VBA project so the hash can be verified without the need to store the original password string.

2.4.4.1 Password Hash Data Structure

The password data structure specifies a password hash and additional random byte data to obfuscate the hash.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reserved								GrbitKey								GrbithashNull															

...	KeyNoNulls	
...	PasswordHashNoNulls	Terminator

Reserved (1 byte): MUST be 0xFF. MUST be ignored.

GrbitKey (4 bits): Each bit specifies a corresponding null byte of **Key** as specified by Encode Nulls (section 2.4.4.2).

GrbitHashNull (20 bits): Each bit specifies a corresponding null byte of **PasswordHash** as specified by Encode Nulls (section 2.4.4.2).

KeyNoNulls (4 bytes): Specifies the **Key** for the Password Hash Algorithm (section 2.4.4.4) with null bytes removed as specified by Encode Nulls (section 2.4.4.2). **Key** is any value.

Key is encoded into **KeyNoNulls** as specified by the following pseudocode:

- CALL Encode Nulls (section 2.4.4.2) with Key RETURNING GrbitKey and KeyNoNulls

Decoding is specified by the following pseudocode:

- CALL Decode Nulls (section 2.4.4.3) with KeyNoNulls and GrbitKey RETURNING Key

PasswordHashNoNulls (20 bytes): Specifies the **PasswordHash** result of the Password Hash Algorithm (section 2.4.4.4) with null bytes removed as specified by Encode Nulls (section 2.4.4.2).

PasswordHash is the 160-bit cryptographic digest of a password combined with **Key** as specified by Password Hash Algorithm (section 2.4.4.4).

Encoding is specified by the following pseudocode:

- CALL Encode Nulls (section 2.4.4.2) with PasswordHash RETURNING GrbitHashNull and PasswordHashNoNulls

Decoding is specified by the following pseudocode:

- CALL Decode Nulls (section 2.4.4.3) with PasswordHashNoNulls and GrbitHashNull RETURNING PasswordHash

Terminator (1 byte): MUST be 0x00.

2.4.4.2 Encode Nulls

The Password Hash stores **Key** and **PasswordHash** with null bytes removed. The fields are encoded by replacing 0x00 bytes with 0x01 and setting a bit on the bit-fields **GrbitKey** and **GrbitHashNull**, respectively.

This algorithm accepts the following as parameters:

InputBytes (variable): An input array of bytes to be encoded.

GrbitNull (variable): An output array of bits specifying null bytes in **InputBytes**.

EncodedBytes (variable): An output array of encoded bytes.

Encoding is computed by the following pseudocode:

- FOR each **Byte** IN **InputBytes**:
  - IF **Byte** EQUALS 0x00 THEN:
    - APPEND **EncodedBytes** WITH 0x01.
    - APPEND **GrbitNull** WITH one bit set to FALSE.
  - ELSE:
    - APPEND **EncodedBytes** WITH **Byte**.
    - APPEND **GrbitNull** WITH one bit set to TRUE.
  - END IF
- END FOR

2.4.4.3 Decode Nulls

The Password Hash stores **Key** and **PasswordHash** with null bytes removed as specified by Encode Nulls (section 2.4.4.2). The fields are decoded by reading bit-fields **GrbitKey** and **GrbitHashNull**, and replacing corresponding bytes in **Key** and **PasswordHash** with 0x00.

This algorithm accepts the following as parameters:

EncodedBytes (variable): An input array of bytes to be encoded.

GrbitNull (variable): An input array of bits specifying null bytes in **DecodedBytes**.

DecodedBytes (variable): An output array of encoded bytes.

Decoding is computed by the following pseudocode:

- SET **Index** TO 0.
- FOR each **Bit** IN **GrbitNull**:
  - IF **Bit** EQUALS FALSE THEN:
    - APPEND **DecodedBytes** WITH 0x00.
  - ELSE:
    - APPEND **DecodedBytes** WITH `EncodedBytes[Index]`.
  - END IF
  - INCREMENT **Index**.
- END FOR

2.4.4.4 Password Hash Algorithm

This Password Hash Algorithm accepts the following as parameters:

Password (Variable): An array of bytes to be obfuscated. MUST contain MBCS characters encoded using the code page specified by PROJECTCODEPAGE (section 2.3.4.2.1.4).

Key (4 Bytes): An array of 4 bytes of any value.

When comparing a new password to an old cryptographic digest, MUST be the same value as stored in the old password's Password Hash Data Structure**.Key** (section 2.4.4).

The algorithm will yield **PasswordHash**, an array of 20 bytes.

The pseudocode for computing the hash is as follows:

- LET **BytesToHash** be a variable array of bytes.
- APPEND **BytesToHash** WITH **Password**.
- APPEND **BytesToHash** WITH **Key**.
- SET **PasswordHash** TO the **SHA-1** cryptographic digest of **BytesToHash**, as specified by [RFC3174].

2.4.4.5 Password Hash Validation

To verify a password against the stored hash, a new password hash MUST be generated using the same **Key** as the existing password. The new hash can then be compared to the hash in the VBA project.

Validation accepts the following as parameters:

NewPassword (Variable): An array of bytes specifying a password to validate. MUST contain MBCS characters encoded using the code page specified by PROJECTCODEPAGE (section 2.3.4.2.1.4).

Key (4 bytes): An array of 4 bytes. MUST be the value stored in Password Hash Data Structure**.Key** (section 2.4.4).

OldPasswordHash (20 bytes): A cryptographic digest. MUST be the value stored in Password Hash Data Structure**.PasswordHash** (section 2.4.4).

The algorithm yields a Boolean value, **Valid**, specifying that **NewPassword** is valid.

Validation is computed by the following pseudocode:

- LET **NewHash** be an array of 20 bytes.
- CALL Password Hash Algorithm (section 2.4.4.4) with **NewPassword** and Key RETURNING **NewHash**.
- IF **NewHash** EQUALS **OldPasswordHash** THEN:
  - SET **Valid** TO TRUE.
- ELSE:
  - SET **Valid** TO FALSE.
- END IF

3 Structure Examples

The following sections provide structure examples of features of this file format. Note that these examples are illustrative of this file format specification, and do not cover all possible structure usage scenarios.

The examples in section 3.1 illustrate the structures of a single VBA project storage as it could be used in a workbook as described in [MS-XLS] section 2.1.7.18.

The examples in section 3.2 illustrate byte arrays that are compressed and decompressed by using the compression and decompression algorithms in section 2.4.1. Note that these examples are illustrative of this file format specification, and do not cover all possible usage scenarios.

3.1 VBA Storage Information Example

3.1.1 _VBA_PROJECT Example

The following table illustrates a _VBA_PROJECT (section 2.3.4.1) example that describes the version-dependent information for the VBA project.

_VBA_PROJECT stream			
Offset	**Size**	**Structure**	**Value**
00000000	0007	_VBA_PROJECT Stream: Version Dependent Project Information **- _VBA_PROJECT**	
00000000	0002	unsigned integer **- Reserved1**	0x61CC
00000002	0002	unsigned integer **- Version**	0xFFFF
00000004	0001	BYTE **- Reserved2**	0x00
00000005	0002	unsigned integer **- Reserved3**	0x0001
00000007	0000	Blob **- PerformanceCache**	

Version: 0xFFFF specifies the version of VBA used to create the VBA project. Write this field as 0xFFFF.

PerformanceCache: This record is empty on write.

3.1.2 dir Stream Example

The following examples illustrate a dir (section 2.3.4.2) stream (1) for a VBA project. The dir (section 2.3.4.2) stream (1) examples describe project information, project references (1) and modules. The dir (section 2.3.4.2) stream (1) ends with an unsigned integer, **Terminator**, and a **Reserved** field.

3.1.2.1 Project Information Example

The following example illustrates a PROJECTINFORMATION (section 2.3.4.2.1) record for a VBA project.

ProjectInformation record			
Offset	**Size**	**Structure**	**Value**
00000000	0122	PROJECTINFORMATION Record **- Information Record**	
00000000	000A	PROJECTSYSKIND Record **- SysKindRecord**	
00000000	0002	unsigned integer **- Id**	0x0001
00000002	0004	unsigned integer **- Size**	0x00000004
00000006	0004	unsigned integer **- SysKind**	0x00000001
0000000A	000A	PROJECTLCID Record **- LcidRecord**	
0000000A	0002	unsigned integer **- Id**	0x0002
0000000C	0004	unsigned integer **- Size**	0x00000004
00000010	0004	unsigned integer **- Lcid**	0x00000409
00000014	000A	PROJECTLCIDINVOKE Record **- LcidInvokeRecord**	
00000014	0002	unsigned integer **- Id**	0x0014
00000016	0004	unsigned integer **- Size**	0x00000004
0000001A	0004	unsigned integer **- LcidInvoke**	0x00000409
0000001E	0008	PROJECTCODEPAGE Record **- CodePageRecord**	
0000001E	0002	unsigned integer **- Id**	0x0003
00000020	0004	unsigned integer **- Size**	0x00000002
00000024	0002	unsigned integer **- CodePage**	0x04E4
00000026	0010	PROJECTNAME Record **- NameRecord**	
00000026	0002	unsigned integer **- Id**	0x0004
00000028	0004	unsigned integer **- SizeOfProjectName**	0x0000000A
0000002C	000A	array of bytes **- ProjectName**	VBAProject
00000036	0069	PROJECTDOCSTRING Record **- DocStringRecord**	
00000036	0002	unsigned integer **- Id**	0x0005
00000038	0004	unsigned integer **- SizeOfDocString**	0x0000001F
0000003C	001F	array of bytes **- DocString**	Example VBA Project Description
0000005B	0002	unsigned integer **- Reserved**	0x0040
0000005D	0004	unsigned integer **- SizeOfDocStringUnicode**	0x0000003E

ProjectInformation record			
00000061	003E	array of bytes **- DocStringUnicode**	Example VBA Project Description
0000009F	0042	PROJECTHELPFILEPATH Record **- HelpFilePathRecord**	
0000009F	0002	unsigned integer **- Id**	0x0006
000000A1	0004	unsigned integer **- SizeOfHelpFile1**	0x0000001B
000000A5	001B	array of bytes **- HelpFile1**	c:\example path\example.hlp
000000C0	0002	unsigned integer **- Reserved**	0x003D
000000C2	0004	unsigned integer **- SizeOfHelpFile2**	0x0000001B
000000C6	001B	array of bytes **- HelpFile2**	c:\example path\example.hlp
000000E1	000A	PROJECTHELPCONTEXT Record **- HelpContextRecord**	
000000E1	0002	unsigned integer **- Id**	0x0007
000000E3	0004	unsigned integer **- Reserved**	0x00000004
000000E7	0004	unsigned integer **- HelpContext**	0x00000001
000000EB	000A	PROJECTLIBFLAGS Record **- LibFlagsRecord**	
000000EB	0002	unsigned integer **- Id**	0x0008
000000ED	0004	unsigned integer **- Size**	0x00000004
000000F1	0004	unsigned integer **- ProjectLibFlags**	0x00000000
000000F5	000C	PROJECTVERSION Record **- VersionRecord**	
000000F5	0002	unsigned integer **- Id**	0x0009
000000F7	0004	unsigned integer **- Reserved**	0x00000004
000000FB	0004	unsigned integer **- VersionMajor**	0x49B5196B
000000FF	0002	unsigned integer **- VersionMinor**	0x0006
00000101	0021	PROJECTCONSTANTS Record **- ConstantsRecord**	
00000101	0002	unsigned integer **- Id**	0x000C
00000103	0004	unsigned integer **- SizeOfConstants**	0x00000007
00000107	0007	array of bytes **- Constants**	abc = 1
0000010E	0002	unsigned integer **- Reserved**	0x003C
00000110	0004	unsigned integer **- SizeOfConstantsUnicode**	0x0000000E
00000114	000E	array of bytes **- ConstantsUnicode**	abc = 1

The preceding table illustrates a PROJECTINFORMATION (section 2.3.4.2.1) record. The PROJECTINFORMATION (section 2.3.4.2.1) record stores the VBA project's properties.

SysKindRecord.SysKind: 0x00000001 specifies this project is intended for the 32-bit Windows Platform.

CodePageRecord.CodePage: 0x04E4 specifies 1252 - Western Latin code page. For more information, see specified in [MC-CPB].

NameRecord.ProjectName: "VBAProject" specifies the name of the VBA project in MBCS characters.

DocStringRecord: Specifies the description, "Example VBA Project Description" for the VBA project. The description of the project can be used to provide additional information about the project beyond the **ProjectName**.

DocStringRecord.DocString: "Example VBA Project Description" specifies the description for the project in MBCS characters.

DocStringRecord.DocStringUnicode: "Example VBA Project Description" specifies the description for the project in Unicode characters. This value is equivalent to the **DocString** field value.

HelpFilePathRecord.HelpFile1: "c:\example path\example.hlp" specifies the path to a Help file for this VBA project in MBCS characters.

HelpFilePathRecord.HelpFile2: "c:\example path\example.hlp" specifies the path to a Help file for this VBA project in MBCS characters.

HelpContextRecord.HelpContext: 0x00000001 specifies the Help topic identifier for the VBA project, which is the help topic the user will see when the **HelpFilePathRecord.HelpFile1** is requested. For example, if the user requested help for this context, the **HelpFilePathRecord.HelpFile1,** "c:\example path\example.hlp", would be opened and the user would see the help topic corresponding to the **HelpContext**, 1, in this example.

VersionRecord.VersionMajor: 0x49B5196B specifies the major version of the VBA project.

VersionRecord.VersionMinor: 0x0006 specifies the minor version of the VBA project.

ConstantsRecord: Specifies compilation constants for the VBA project. **Constants** are used to conditionally compile code within the VBA project. The Constant, "abc = 1" from this example is illustrated in the Sheet1 Decompressed Module Stream Example (section 3.1.4).

ConstantsRecord.Constants: "abc = 1" specifies the compilation constants for the VBA project in MBCS characters.

ConstantsRecord.ConstantsUnicode: "abc = 1" specifies the compilation constants for the VBA project in Unicode characters. This value is equivalent to the **Constants** field value.

3.1.2.2 Project Reference Information Example

The following example illustrates the PROJECTREFERENCES (section 2.3.4.2.2) record for the VBA project. This project includes four references (1).

Project reference record			
Offset	**Size**	**Structure**	**Value**
00000122	0380	VBA_Canonical_ReferenceArray **- ReferenceArray**	
00000122	008C	REFERENCE Record **- Reference[0]**	
00000122	001E	REFERENCENAME Record **- NameRecord**	
00000122	0002	unsigned integer **- Id**	0x0016
00000124	0004	unsigned integer **- SizeOfName**	0x00000006
00000128	0006	array of bytes **- Name**	stdole
0000012E	0002	unsigned integer **- Reserved**	0x003E
00000130	0004	unsigned integer **- SizeOfNameUnicode**	0x0000000C
00000134	000C	array of bytes **- NameUnicode**	stdole
00000140	006E	REFERENCEREGISTERED Record **- ReferenceRecord[0]**	
00000140	0002	unsigned integer **- Id**	0x000D
00000142	0004	unsigned integer **- Size**	0x00000068
00000146	0004	unsigned integer **- SizeOfLibid**	0x0000005E
0000014A	005E	array of bytes **- Libid**	*\G{00020430-0000-0000-C000-000000000046}#2.0#0#C:\Windows\system32\stdole2.tlb#OLE Automation
000001A8	0004	unsigned integer **- Reserved1**	0x00000000
000001AC	0002	unsigned integer **- Reserved2**	0x0000
000001AE	00C2	REFERENCE Record **- Reference[1]**	

Project reference record			
00000 1AE	001 E	REFERENCENAM E Record **- NameRecord**	
00000 1AE	000 2	unsigned integer **- Id**	0x0016
00000 1B0	000 4	unsigned integer **- SizeOfName**	0x00000006
00000 1B4	000 6	array of bytes **- Name**	Office
00000 1BA	000 2	unsigned integer **- Reserved**	0x003E
00000 1BC	000 4	unsigned integer **- SizeOfNameUnicode**	0x0000000C
00000 1C0	000 C	array of bytes **- NameUnicode**	Office
00000 1CC	00 A4	REFERENCEREGI STERED Record **- ReferenceRecord[1]**	
00000 1CC	000 2	unsigned integer **- Id**	0x000D
00000 1CE	000 4	unsigned integer **- Size**	0x0000009E
00000 1D2	000 4	unsigned integer **- SizeOfLibid**	0x00000094
00000 1D6	009 4	array of bytes **- Libid**	*\G{2DF8D04C-5BFA-101B-BDE5-00AA0044DE52}#2.0#0#C:\Program Files\Common Files\Microsoft Shared\OFFICE12\MSO.DLL#Microsoft Office 12.0 Object Library
00000 26A	000 4	unsigned integer **- Reserved1**	0x00000000
00000 26E	000 2	unsigned integer **- Reserved2**	0x0000
00000 270	009 1	REFERENCE Record **- Reference[2]**	
00000 270	002 D	REFERENCENAM E Record **- NameRecord**	
00000 270	000 2	unsigned integer **- Id**	0x0016
00000	000	unsigned	0x0000000B

Project reference record			
272	4	integer **- SizeOfName**	
00000276	000B	array of bytes **- Name**	VBAProject1
00000281	0002	unsigned integer **- Reserved**	0x003E
00000283	0004	unsigned integer **- SizeOfNameUnicode**	0x00000016
00000287	0016	array of bytes **- NameUnicode**	VBAProject1
0000029D	0064	REFERENCEPROJECT Record **- ReferenceRecord[2]**	
0000029D	0002	unsigned integer **- Id**	0x000E
0000029F	0004	unsigned integer **- Size**	0x0000005E
000002A3	0004	unsigned integer **- SizeOfLibidAbsolute**	0x00000030
000002A7	0030	array of bytes **- LibidAbsolute**	*\CC:\Example Path\Example-ReferencedProject.xls
000002D7	0004	unsigned integer **- SizeOfLibidRelative**	0x00000020
000002DB	0020	array of bytes **- LibidRelative**	*\CExample-ReferencedProject.xls
000002FB	0004	unsigned integer **- MajorVersion**	0x49A95F46
000002FF	0002	unsigned integer **- MinorVersion**	0x000D
00000301	01A1	REFERENCE Record **- Reference[3]**	
00000301	0021	REFERENCENAME Record **- NameRecord**	
00000301	0002	unsigned integer **- Id**	0x0016
00000	000	unsigned integer **-**	0x00000007

Project reference record			
303	4	**SizeOfName**	
00000307	0007	array of bytes **- Name**	MSForms
00000030E	0002	unsigned integer **- Reserved**	0x003E
00000310	0004	unsigned integer **- SizeOfNameUnicode**	0x0000000E
00000314	000E	array of bytes **- NameUnicode**	MSForms
00000322	0180	REFERENCECONTROL Record **- ReferenceRecord[3]**	
00000322	0075	REFERENCEORIGINAL Record **- OriginalRecord**	
00000322	0002	unsigned integer **- Id**	0x0033
00000324	0004	unsigned integer **- SizeOfLibidOriginal**	0x0000006F
00000328	006F	array of bytes **- LibidOriginal**	*\G{0D452EE1-E08F-101A-852E-02608C4D0BB4}#2.0#0#C:\Windows\system32\FM20.DLL#Microsoft Forms 2.0 Object Library
00000397	0002	unsigned integer **- Id**	0x002F
00000399	0004	unsigned integer **- SizeTwiddled**	0x0000003B
0000039D	0004	unsigned integer **- SizeOfLibidTwiddled**	0x00000031
000003A1	0031	array of bytes **- LibidTwiddled**	*\G{00000000-0000-0000-0000-000000000000}#0.0#0##
000003D2	0004	unsigned integer **- Reserved1**	0x00000000
000003D6	0002	unsigned integer **- Reserved2**	0x0000
000003D8	0021	REFERENCENAME Record **- NameRecordExtended**	

Project reference record			
00000 3D8	000 2	unsigned integer **- Id**	0x0016
00000 3DA	000 4	unsigned integer **- SizeOfName**	0x00000007
00000 3DE	000 7	array of bytes **- Name**	MSForms
00000 3E5	000 2	unsigned integer **- Reserved**	0x003E
00000 3E7	000 4	unsigned integer **- SizeOfNameUnicode**	0x0000000E
00000 3EB	000 E	array of bytes **- NameUnicode**	MSForms
00000 3F9	000 2	unsigned integer **- Reserved3**	0x0030
00000 3FB	000 4	unsigned integer **- SizeExtended**	0x000000A3
00000 3FF	000 4	unsigned integer **- SizeOfLibidExtended**	0x00000085
00000 403	008 5	array of bytes **- LibidExtended**	*\G{896C2D83-5466-46ED-8FAE-4C3E4F85E710}#2.0#0#C:\Users\jsmith\AppData\Local\Temp\VBE\MSForms.exd#Microsoft Forms 2.0 Object Library
00000 488	000 4	unsigned integer **- Reserved4**	0x00000000
00000 48C	000 2	unsigned integer **- Reserved5**	0x0000
00000 48E	001 0	GUID **- OriginalTypeLib**	E1 2E 45 0D 8F E0 1A 10 85 2E 02 60 8C 4D 0B B4
00000 49E	000 4	unsigned integer **- Cookie**	0x00000001

The example described in preceding table illustrates a set of four external references for the example VBA project. Description for **Reference[1]** is omitted as it duplicates the example of a REFERENCEREGISTERED (section 2.3.4.2.2.5) type, illustrated in **Reference[0]**. **Reference[2]** illustrates an example of a REFERENCEPROJECT (section 2.3.4.2.2.6) type. **Reference[3]** illustrates an example of a REFERENCECONTROL (section 2.3.4.2.2.3) type.

ReferenceArray: Specifies an array of four REFERENCE (section 2.3.4.2.2.1) records. In this array, **Reference[0]** and **Reference[1]** are REFERENCEREGISTERED (section 2.3.4.2.2.5)

type records. **Reference[2]** is a REFERENCEPROJECT (section 2.3.4.2.2.6) record. **Reference[3]** is a REFERENCECONTROL (section 2.3.4.2.2.3) record.

Reference[0]: Specifies a record of type REFERENCEREGISTERED (section 2.3.4.2.2.5).

Reference[0].NameRecord.Name: "stdole" specifies a reference to the stdole2.tlb Automation type library in MBCS characters.

Reference[0].NameRecord.NameUnicode: "stdole" specifies a reference to the stdole2.tlb Automation type library in Unicode characters. This value is equivalent to the **Name** field value.

Reference[0].ReferenceRecord[0].Libid: "*\G{00020430-0000-0000-C000-000000000046}#2.0#0#C:\Windows\system32\stdole2.tlb#OLE Automation" specifies a LibidReference (section 2.1.1.8) and conforms to the ABNF Syntax for Libid references.

The **LibidReferenceKind,** "*\G", specifies the **LibidPath** is a Windows Path.

The **LibidGuid, "**{00020430-0000-0000-C000-000000000046}", specifies the **CLSID** of the "OLE Automation" Automation type library.

The **LibidMajorVersion** is 2.

The **LibidMinorVersion** is 0.

The **LibidLCID** is 0.

The **LibidPath** is "C:\Windows\system32\stdole2.tlb".

The **LibidRegName** is "OLE Automation".

Reference[2]: Specifies a reference of type REFERENCEPROJECT (section 2.3.4.2.2.6). This reference illustrates the information required to reference another VBA project that exists in another Excel workbook file. The **ProjectName** of the referenced workbook cannot match the **ProjectName** of the referencing workbook.

Reference[2].NameRecord.Name: "VBAProject1" specifies the **ProjectName** of the referenced VBA project in MBCS characters.

Reference[2].NameRecord.NameUnicode: "VBAProject1" specifies the **ProjectName** of the referenced VBA project in Unicode characters. This value is equivalent to the **Name** field value.

Reference[2].ReferenceRecord[2].LibidAbsolute: "*\CC:\Example Path\Example-ReferencedProject.xls" specifies the absolute path to the file containing the referenced VBA project.

The **Projectkind,** "*\C" specifies a Windows file path.

The **ProjectPath** is "C:\Example Path\ReferencedProject.xls".

Reference[2].ReferenceRecord[2].LibidRelative: "*\CExample-ReferencedProject.xls" specifies the relative path to the file containing the referenced VBA project. In this example, both files exist in the same directory ("C:\Example Path").

The **Projectkind,** "*\C" specifies a Windows file path.

The **ProjectPath** is "Example-ReferencedProject.xls", as it is relative, there is no additional file path. If the referenced file existed in the subdirectory "Test" the Project Path would be "\Test\Example-ReferencedProject.xls".

Reference[2].ReferenceRecord[2].MajorVersion: "0x49A95F46" specifies the **MajorVersion** of the referenced VBA project. The **MajorVersion** is equivalent to the **VersionMajor** of the referenced VBA project's PROJECTVERSION record (section 2.3.4.2.1.10).

Reference[2].ReferenceRecord[2].MinorVersion: "0x000D" specifies the **MinorVersion** of the referenced VBA project. The **MinorVersion** is equivalent to the **VersionMinor** of the referenced VBA project's PROJECTVERSION record (section 2.3.4.2.1.10).

Reference[3]: Specifies a reference of type REFERENCECONTROL (section 2.3.4.2.2.3) to an **ActiveX control library**.

Reference[3].NameRecord.Name: "MSForms" specifies the name of an Office Form ActiveX control in MBCS characters as described in [MS-OFORMS].

Reference[3].NameRecord.NameUnicode: "MSForms" specifies the name of an Office Form ActiveX control in Unicode characters as described in [MS-OFORMS]. This value is equivalent to the **Name** field value.

Reference[3].ReferenceRecord[3].OriginalRecord.LibidOriginal: "*\G{0D452EE1-E08F-101A-852E-02608C4D0BB4}#2.0#0#C:\Windows\system32\FM20.DLL#Microsoft Forms 2.0 Object Library" specifies the Office Form ActiveX control library identifier.

The **LibidReferenceKind,** "*\G", specifies the **LibidPath** is a Windows Path.

The **LibidGuid**, "{0D452EE1-E08F-101A-852E-02608C4D0BB4}", specifies the **ClassId** of the Office Form ActiveX control as described in [MS-OFORMS].

The **LibidMajorVersion** is 2.

The **LibidMinorVersion** is 0.

The **LibidLCID** is 0.

The **LibidPath** is "C:\Windows\system32\FM20.DLL".

The **LibidRegName** is "Microsoft Forms 2.0 Object Library".

Reference[3].ReferenceRecord[3].LibidTwiddled: *\G{00000000-0000-0000-0000-000000000000}#0.0#0## specifies the **ReferenceRecord** does not reference a twiddled type library.

The **LibidRefernceKind**, "*\G" specifies a Windows file path.

The **LibidGuid** is {00000000-0000-0000-0000-000000000000}.

The **LibidMajorVersion** is 0.

The **LibidMinorVersion** is 0.

The **LibidLCID** is 0.

The **LibidPath** and **LibidRegName** are empty, signifying the **ReferenceRecord** is not a twiddled type library.

Reference[3].ReferenceRecord[3].NameRecordExtended.Name: "MSForms" specifies the name of the extended type library in MBCS characters

Reference[3].ReferenceRecord[3].NameRecordExtended.NameUnicode: "MSForms" specifies the name of the extended type library in Unicode characters. This value is equivalent to the **Name** field value.

Reference[3].ReferenceRecord[3].LibidExtended: *\G{896C2D83-5466-46ED-8FAE-4C3E4F85E710}#2.0#0#C:\Users\jsmith\AppData\Local\Temp\VBE\MSForms.exd#Microsoft Forms 2.0 Object Library specifies the Office Form ActiveX control library as the extended control library as described in [MS-OFORMS].

The **LibidReferenceKind,** "*\G" specifies the **LibidPath** is a Windows path.

The **LibidGuid,** {896C2D83-5466-46ED-8FAE-4C3E4F85E710} specifies the **ClassID** of the Office Form extended control library described in [MS-OFORMS].

The **LibidMajorVersion** is 2.

The **LibidMinorVersion** is 0.

The **LibidLCID** is 0.

The **LibidPath** is "C:\Users\jsmith\AppData\Local\Temp\VBE\MSForms.exd".

The **LibidRegName** is "Microsoft Forms 2.0 Object Library".

Reference[3].ReferenceRecord[3].OriginalTypeLib: E1 2E 45 0D 8F E0 1A 10 85 2E 02 60 8C 4D 0B B4 specifies the CLSID of the Automation type library the extended type library was generated from. This value is equivalent to the **LibidGuid** value of this record's **LibidOriginal** field, "{0D452EE1-E08F-101A-852E-02608C4D0BB4}".

3.1.2.3 Module Information Example

3.1.2.3.1 PROJECT MODULES Example

The following illustrates a PROJECTMODULES (section 2.3.4.2.3) example that includes three modules for the VBA project.

Project modules stream			
Offset	**Size**	**Structure**	**Value**
000004A2	01EA	PROJECTMODULES Record **- ModulesRecord**	
000004A2	0002	unsigned integer **- Id**	0x000F
000004A4	0004	unsigned integer **- Size**	0x00000002
000004A8	0002	unsigned integer **- Count**	0x0003
000004AA	0008	PROJECTCOOKIE Record **- ProjectCookieRecord**	
000004AA	0002	unsigned integer **- Id**	0x0013
000004AC	0004	unsigned integer **- Size**	0x00000002
000004B0	0002	unsigned integer **- Cookie**	0xFFFF

Count: 0x0003 specifies 3 modules for the project.

ProjectCookieRecord.Cookie: 0xFFFF specifies ignored data. Write this field as 0xFFFF.

3.1.2.3.2 Module Record Examples

3.1.2.3.2.1 ThisWorkbook Document Module Record Example

This module record example describes a typical document module record.

"ThisWorkbook" module record			
Offset	**Size**	**Structure**	**Value**
000004B2	0094	MODULE Record **- ModuleRecord**	
000004B2	0012	MODULENAME Record **- NameRecord**	
000004B2	0002	unsigned integer **- Id**	0x0019
000004B4	0004	unsigned integer **- SizeOfModuleName**	0x0000000C
000004B8	000C	array of bytes **- ModuleName**	ThisWorkbook
000004C4	001E	MODULENAMEUNICODE Record **- NameUnicodeRecord**	
000004C4	0002	unsigned integer **- Id**	0x0047
000004C6	0004	unsigned integer **- SizeOfModuleNameUnicode**	0x00000018
000004CA	0018	array of bytes **- ModuleNameUnicode**	ThisWorkbook
000004E2	0030	MODULESTREAMNAME Record **- StreamNameRecord**	
000004E2	0002	unsigned integer **- Id**	0x001A
000004E4	0004	unsigned integer **- SizeOfStreamName**	0x0000000C
000004E8	000C	array of bytes **- StreamName**	ThisWorkbook
000004F4	0002	unsigned integer **- Reserved**	0x0032
000004F6	0004	unsigned integer **- SizeOfStreamNameUnicode**	0x00000018
000004FA	0018	array of bytes **- StreamNameUnicode**	ThisWorkbook
00000512	000C	MODULEDOCSTRING Record **- DocStringRecord**	
00000512	0002	unsigned integer **- Id**	0x001C
00000514	0004	unsigned integer **- SizeOfDocString**	0x00000000
00000518	0000	array of bytes **- DocString**	
00000518	0002	unsigned integer **- Reserved**	0x0048
0000051A	0004	unsigned integer **- SizeOfDocStringUnicode**	0x00000000
0000051E	0000	array of bytes **- DocStringUnicode**	

"ThisWorkbook" module record			
0000051E	000A	MODULEOFFSET Record **- OffsetRecord**	
0000051E	0002	unsigned integer **- Id**	0x0031
00000520	0004	unsigned integer **- Size**	0x00000004
00000524	0004	unsigned integer **- TextOffset**	0x00000000
00000528	000A	MODULEHELPCONTEXT Record **- HelpContextRecord**	
00000528	0002	unsigned integer **- Id**	0x001E
0000052A	0004	unsigned integer **- Size**	0x00000004
0000052E	0004	unsigned integer **- HelpContext**	0x00000000
00000532	0008	MODULECOOKIE Record **- CookieRecord**	
00000532	0002	unsigned integer **- Id**	0x002C
00000534	0004	unsigned integer **- Size**	0x00000002
00000538	0002	unsigned integer **- Cookie**	0xFFFF
0000053A	0006	MODULETYPE Record **- TypeRecord**	
0000053A	0002	unsigned integer **- Id**	0x0022
0000053C	0004	unsigned integer **- Reserved**	0x00000000
00000540	0002	unsigned integer **- Terminator**	0x002B
00000542	0004	unsigned integer **- Reserved**	0x00000000

The preceding table illustrates the module record for the record named "ThisWorkbook". The **DocStringRecord** and **HelpContextRecord** descriptions for this module example are omitted as they are empty for this example and illustrated in the next example. The decompressed code can be found in the corresponding ThisWorkbook Decompressed Module Stream Example (section 3.1.3).

NameRecord.ModuleName: "ThisWorkbook" specifies the name of the module in MBCS characters as specified by the PROJECTCODEPAGE (section 2.3.4.2.1.4).

NameUnicodeRecord.ModuleNameUnicode: "ThisWorkbook" specifies the name of the module in Unicode characters. This value is equivalent to the **NameRecord.ModuleName** field value.

StreamNameRecord.StreamName: "ThisWorkbook" specifies the stream name in MBCS characters of the ModuleStream (section 2.3.4.3) in the VBA Storage (section 2.3.4) corresponding to the containing MODULE Record (section 2.3.4.2.3.2).

StreamNameRecord.StreamNameUnicode: "ThisWorkbook" specifies the stream name in Unicode characters of the ModuleStream (section 2.3.4.3) in the VBA Storage (section 2.3.4) corresponding to the containing MODULE Record (section 2.3.4.2.3.2). This value is equivalent to the **StreamName** field value.

OffsetRecord: Specifies the location of the source code in the module stream that corresponds to this module record. The corresponding module stream can be found in ThisWorkbook Decompressed Module Stream Example (section 3.1.3).

OffsetRecord.TextOffset: 0x00000000 specifies the code in the corresponding Module stream as described by the **ModuleName** record begins at 0x00000000.

CookieRecord.Cookie: 0xFFFF specifies ignored data. Write this field as 0xFFFF.

TypeRecord.Id: 0x0022 specifies this module is a document module, class module, or designer module.

3.1.2.3.2.2 Sheet1 Document Module Record Example

This example illustrates a document module record, it differs from the previous module record example in record values. This example is included to illustrate the MODULEDOCSTRING (section 2.3.4.2.3.2.4) and MODULEHELPCONTEXT (section 2.3.4.2.3.2.6) records.

Sheet1 module record			
Offset	**Size**	**Structure**	**Value**
00000546	00BE	MODULE Record **- ModuleRecord**	
00000546	000C	MODULENAME Record **- NameRecord**	
00000546	0002	unsigned integer **- Id**	0x0019
00000548	0004	unsigned integer **- SizeOfModuleName**	0x00000006
0000054C	0006	array of bytes **- ModuleName**	Sheet1
00000552	0012	MODULENAMEUNICODE Record **- NameUnicodeRecord**	
00000552	0002	unsigned integer **- Id**	0x0047
00000554	0004	unsigned integer **- SizeOfModuleNameUnicode**	0x0000000C
00000558	000C	array of bytes **- ModuleNameUnicode**	Sheet1
00000564	001E	MODULESTREAMNAME Record **- StreamNameRecord**	
00000564	0002	unsigned integer **- Id**	0x001A
00000566	0004	unsigned integer **- SizeOfStreamName**	0x00000006
0000056A	0006	array of bytes **- StreamName**	Sheet1
00000570	0002	unsigned integer **- Reserved**	0x0032
00000572	0004	unsigned integer **- SizeOfStreamNameUnicode**	0x0000000C
00000576	000C	array of bytes **- StreamNameUnicode**	Sheet1
00000582	005A	MODULEDOCSTRING Record **- DocStringRecord**	
00000582	0002	unsigned integer **- Id**	0x001C

Sheet1 module record			
00000584	0004	unsigned integer **- SizeOfDocString**	0x0000001A
00000588	001A	array of bytes **- DocString**	Example Module Description
000005A2	0002	unsigned integer **- Reserved**	0x0048
000005A4	0004	unsigned integer **- SizeOfDocStringUnicode**	0x00000034
000005A8	0034	array of bytes **- DocStringUnicode**	Example Module Description
000005DC	000A	MODULEOFFSET Record **- OffsetRecord**	
000005DC	0002	unsigned integer **- Id**	0x0031
000005DE	0004	unsigned integer **- Size**	0x00000004
000005E2	0004	unsigned integer **- TextOffset**	0x00000000
000005E6	000A	MODULEHELPCONTEXT Record **- HelpContextRecord**	
000005E6	0002	unsigned integer **- Id**	0x001E
000005E8	0004	unsigned integer **- Size**	0x00000004
000005EC	0004	unsigned integer **- HelpContext**	0x00000002
000005F0	0008	MODULECOOKIE Record **- CookieRecord**	
000005F0	0002	unsigned integer **- Id**	0x002C
000005F2	0004	unsigned integer **- Size**	0x00000002
000005F6	0002	unsigned integer **- Cookie**	0xFFFF
000005F8	0006	MODULETYPE Record **- TypeRecord**	
000005F8	0002	unsigned integer **- Id**	0x0022
000005FA	0004	unsigned integer **- Reserved**	0x00000000
000005FE	0002	unsigned integer **- Terminator**	0x002B
00000600	0004	unsigned integer **- Reserved**	0x00000000

The preceding table illustrates a module record for a document module with a description and link to a Windows Help file (.hlp). The description is described in the **DocStringRecord**. The link to the Help file is illustrated in the **HelpFilePathRecord** of the PROJECTINFORMATION (section 2.3.4.2.1). The link to the Help topic for this example is illustrated in the **HelpContextRecord**. The decompressed code can be found in the corresponding Sheet1 Decompressed Module Stream Example (section 3.1.4).

DocStringRecord.DocString: "Example Module Description" specifies the description of the module in MBCS characters.

DocStringRecord.DocStringUnicode: "Example Module Description" specifies the description of the module in Unicode characters. This value is equivalent to the **DocString** field value.

HelpContextRecord.HelpContext: 0x00000002 specifies the Help topic identifier in the Help file specified by PROJECTHELPFILEPATH Record (section 2.3.4.2.1.7).

3.1.2.3.2.3 UserForm1 Designer Module Record Example

The following example illustrates a designer module record.

UserForm1 module record			
Offset	**Size**	**Structure**	**Value**
00000604	0088	MODULE Record **- ModuleRecord**	
00000604	000F	MODULENAME Record **- NameRecord**	
00000604	0002	unsigned integer **- Id**	0x0019
00000606	0004	unsigned integer **- SizeOfModuleName**	0x00000009
0000060A	0009	array of bytes **- ModuleName**	UserForm1
00000613	0018	MODULENAMEUNICODE Record **- NameUnicodeRecord**	
00000613	0002	unsigned integer **- Id**	0x0047
00000615	0004	unsigned integer **- SizeOfModuleNameUnicode**	0x00000012
00000619	0012	array of bytes **- ModuleNameUnicode**	UserForm1
0000062B	0027	MODULESTREAMNAME Record **- StreamNameRecord**	
0000062B	0002	unsigned integer **- Id**	0x001A
0000062D	0004	unsigned integer **- SizeOfStreamName**	0x00000009
00000631	0009	array of bytes **- StreamName**	UserForm1
0000063A	0002	unsigned integer **- Reserved**	0x0032
0000063C	0004	unsigned integer **- SizeOfStreamNameUnicode**	0x00000012
00000640	0012	array of bytes **- StreamNameUnicode**	UserForm1
00000652	000C	MODULEDOCSTRING Record **- DocStringRecord**	
00000652	0002	unsigned integer **- Id**	0x001C
00000654	0004	unsigned integer **- SizeOfDocString**	0x00000000
00000658	0000	array of bytes **- DocString**	
00000658	0002	unsigned integer **- Reserved**	0x0048
0000065A	0004	unsigned integer **- SizeOfDocStringUnicode**	0x00000000
0000065E	0000	array of bytes **- DocStringUnicode**	

UserForm1 module record			
0000065E	000A	MODULEOFFSET Record **- OffsetRecord**	
0000065E	0002	unsigned integer **- Id**	0x0031
00000660	0004	unsigned integer **- Size**	0x00000004
00000664	0004	unsigned integer **- TextOffset**	0x00000000
00000668	000A	MODULEHELPCONTEXT Record **- HelpContextRecord**	
00000668	0002	unsigned integer **- Id**	0x001E
0000066A	0004	unsigned integer **- Size**	0x00000004
0000066E	0004	unsigned integer **- HelpContext**	0x00000000
00000672	0008	MODULECOOKIE Record **- CookieRecord**	
00000672	0002	unsigned integer **- Id**	0x002C
00000674	0004	unsigned integer **- Size**	0x00000002
00000678	0002	unsigned integer **- Cookie**	0xFFFF
0000067A	0006	MODULETYPE Record **- TypeRecord**	
0000067A	0002	unsigned integer **- Id**	0x0022
0000067C	0004	unsigned integer **- Reserved**	0x00000000
00000680	0006	MODULEPRIVATE Record **- PrivateRecord**	
00000680	0002	unsigned integer **- Id**	0x0028
00000682	0004	unsigned integer **- Reserved**	0x00000000
00000686	0002	unsigned integer **- Terminator**	0x002B
00000688	0004	unsigned integer **- Reserved**	0x00000000

The preceding table illustrates a Module record for a designer module. The fields for this example are omitted, with the exception of **PrivateRecord**. The decompressed data can be found in the corresponding UserForm1 Decompressed Module Stream Example (section 3.1.5).

PrivateRecord: The presence of this record with a value of 0x0028 for the identifier specifies that the module is only usable from within this VBA project. Referencing VBA projects may not call this module.

3.1.3 ThisWorkbook Decompressed Module Stream Example

The following example illustrates the decompressed module stream for the "ThisWorkbook" module record.

Decompressed module data			
Offset	**Size**	**Structure**	**Value**
00000000	0163	array of bytes **- TextDecompressedData**	Attribute VB_Name = "ThisWorkbook"\r\nAttribute VB_Base = "0{00020819-0000-0000-C000-000000000046}"\r\nAttribute VB_GlobalNameSpace = False\r\nAttribute VB_Creatable = False\r\nAttribute VB_PredeclaredId = True\r\nAttribute VB_Exposed = True\r\nAttribute VB_TemplateDerived = False\r\nAttribute VB_Customizable = True\r\nSub helloworld()\r\nMsgBox "Hello, World"\r\nEnd Sub\r\n

The preceding table illustrates the decompressed module data for the "ThisWorkbook" module.

TextDecompressedData: Specifies the attributes of the "ThisWorkbook" module and the code for the module, as described in [MS-VBAL]. The following text is formatted for readability.

```
Attribute VB_Name = "ThisWorkbook"
Attribute VB_Base = "0{00020819-0000-0000-C000-000000000046}"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = True
Attribute VB_TemplateDerived = False
Attribute VB_Customizable = True
Sub helloworld()
    MsgBox "Hello, World"
End Sub
```

3.1.4 Sheet1 Decompressed Module Stream Example

The following example illustrates the decompressed module stream example for the "Sheet1" module record.

Decompressed module data			
Offset	**Size**	**Structure**	**Value**
00000000	01D5	array of bytes **- TextDecompressedData**	Attribute VB_Name = "Sheet1"\r\nAttribute VB_Base = "0{00020820-0000-0000-C000-000000000046}"\r\nAttribute VB_GlobalNameSpace = False\r\nAttribute VB_Creatable = False\r\nAttribute VB_PredeclaredId = True\r\nAttribute VB_Exposed = True\r\nAttribute VB_TemplateDerived = False\r\nAttribute VB_Customizable = True\r\nAttribute VB_HelpID = 2\r\nAttribute VB_Description = "Example Module Description"\r\nSub CompiliationExample()\r\n\r\n#If abc Then\r\n MsgBox "abc=1"\r\n#End If\r\n \r\nEnd Sub\r\n\r\n

TextDecompressedData: Specifies the attributes of the "Sheet1" module and the code for the module as described in [MS-VBAL]. The following text is formatted for readability.

```
Attribute VB_Name = "Sheet1"
Attribute VB_Base = "0{00020820-0000-0000-C000-000000000046}"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = True
Attribute VB_TemplateDerived = False
Attribute VB_Customizable = True
Attribute VB_HelpID = 2
Attribute VB_Description = "Example Module Description"
Sub CompiliationExample()

#If abc Then
    MsgBox "abc=1"
#End If

End Sub
```

3.1.5 UserForm1 Decompressed Module Stream Example

The following example illustrates the decompressed module stream example for the "UserForm1" module record.

Decompressed module data			
Offset	**Size**	**Structure**	**Value**
00000000	0156	array of bytes **- TextDecompressedData**	Attribute VB_Name = "UserForm1"\r\nAttribute VB_Base = "0{842E9C5E-88B5-439A-912E-4C2D9AA0EC27}{2DC3C962-DA1C-47BA-AB63-E9D578FC2637}"\r\nAttribute VB_GlobalNameSpace = False\r\nAttribute VB_Creatable = False\r\nAttribute VB_PredeclaredId = True\r\nAttribute VB_Exposed = False\r\nAttribute VB_TemplateDerived = False\r\nAttribute VB_Customizable = False\r\n

TextDecompressedData: Specifies the attributes of the "UserForm1" module. The following text is formatted for readability.

```
Attribute VB_Name = "UserForm1"
Attribute VB_Base = "0{842E9C5E-88B5-439A-912E-4C2D9AA0EC27}{2DC3C962-DA1C-47BA-
AB63-E9D578FC2637}"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Attribute VB_TemplateDerived = False
Attribute VB_Customizable = False
```

3.1.6 PROJECT Stream Example

This example illustrates the properties of the VBA project in the Project Stream.

Project stream			
Offset	**Size**	**Structure**	**Value**
00000000	027F	array of bytes - **text**	ID="{917DED54-440B-4FD1-A5C1-74ACF261E600}"\r\nDocument=ThisWorkbook/&H00000000\r\nDocument=Sheet1/&H00000000\r\nPackage={AC9F2F90-E877-11CE-9F68-00AA00574A4F}\r\nBaseClass=UserForm1\r\nHelpFile="c:\example path\example.hlp"\r\nName="VBAProject"\r\nHelpContextID="1"\r\nDescription="Example VBA Project Description"\r\nVersionCompatible32="393222000"\r\nCMG="0705D8E3D8EDDBF1DBF1DBF1DBF1"\r\nDPB="0E0CD1ECDFF4E7F5E7F5E7"\r\nGC="1517CAF1D6F9D7F9D706"\r\n\r\n[Host Extender Info]\r\n&H00000001={3832D640-CF90-11CF-8E43-00A0C911005A};VBE;&H00000000\r\n\r\n[Workspace]\r\nThisWorkbook=23, 23, 911, 280, \r\nSheet1=69, 69, 724, 317, C\r\nUserForm1=0, 0, 0, 0, C, 46, 46, 701, 294, Z

The preceding table illustrates an array of bytes that contains the example VBAPROJECTText (section 2.3.1). The VBAPROJECTText (section 2.3.1) conforms to the ABNF syntax.

text: The example VBAPROJECTText (section 2.3.1) follows. The following text is formatted for readability:

```
ID="{917DED54-440B-4FD1-A5C1-74ACF261E600}"
Document=ThisWorkbook/&H00000000
Document=Sheet1/&H00000000
Package={AC9F2F90-E877-11CE-9F68-00AA00574A4F}
BaseClass=UserForm1
HelpFile="c:\example path\example.hlp"
Name="VBAProject"
HelpContextID="1"
Description="Example VBA Project Description"
VersionCompatible32="393222000"
CMG="0705D8E3D8EDDBF1DBF1DBF1DBF1"
DPB="0E0CD1ECDFF4E7F5E7F5E7"
GC="1517CAF1D6F9D7F9D706"

[Host Extender Info]
&H00000001={3832D640-CF90-11CF-8E43-00A0C911005A};VBE;&H00000000

[Workspace]
ThisWorkbook=23, 23, 911, 280,
Sheet1=69, 69, 724, 317, C
UserForm1=0, 0, 0, 0, C, 46, 46, 701, 294, Z
```

ProjectID (section 2.3.1.2): "ID="{917DED54-440B-4FD1-A5C1-74ACF261E600}"" specifies the CLSID of the VBA project's Automation type library.

ProjectDocModule (section 2.3.1.4): specifies the module names, "ThisWorkbook" and "Sheet1", of the document modules in the VBA project. "&H00000000" specifies the modules are document modules. This example contains no ProjectStdModule (section 2.3.1.5) or ProjectClassModule (section 2.3.1.6) properties as there are no procedural modules or class modules.

ProjectPackage (section 2.3.1.8): "Package={AC9F2F90-E877-11CE-9F68-00AA00574A4F}" specifies the CLSID for the designer module, "UserForm1", as specified in the ProjectDesignerModule (section 2.3.1.7) property.

ProjectHelpFile (section 2.3.1.9): "HelpFile="c:\example path\example.hlp"" is equivalent to the value specified in PROJECTHELPFILEPATH (section 2.3.4.2.1.7) field in the PROJECTINFORMATION record (section 2.3.4.2.1).

ProjectName (section 2.3.1.11): "Name="VBAProject"" is equivalent to the value specified in PROJECTNAME (section 2.3.4.2.1.5).

ProjectHelpId (section 2.3.1.12): "HelpContextID="1"" is equivalent to the value specified in PROJECTHELPCONTEXT (section 2.3.4.2.1.8).

ProjectDescription (section 2.3.1.13): "Description="Example VBA Project Description"" is equivalent to the DocStringRecord (section 2.3.4.2.1.6) field in the PROJECTINFORMATION record (section 2.3.4.2.1).

ProjectVersionCompat32 (section 2.3.1.14): "VersionCompatible32="393222000"" specifies the VBA version of the VBA project.

ProjectProtectionState (section 2.3.1.15): "CMG="0705D8E3D8EDDBF1DBF1DBF1DBF1"" specifies no sources are restricted access to the VBA project. The value is obfuscated by Data Encryption (section 2.4.3). The following is the decrypted value of the ProjectProtectionState (section 2.3.1.15) as specified by an Encrypted Data Structure (section 2.4.3.1). The text is formatted for readability:

```
Seed:           0x07
Version:        0x02
ProjKey:        0xDF
Ignored:        0x070707
DataLength:     0x00000004
Data:           0x00000000
```

ProjectPassword (section 2.3.1.16): "DPB="0E0CD1ECDFF4E7F5E7F5E7"" specifies the VBA project has no password. The value is obfuscated by Data Encryption (section 2.4.3). The following is the decrypted value of the ProjectProtectionState (section 2.3.1.15) as specified by an Encrypted Data Structure (section 2.4.3.1). The text is formatted for readability:

```
Seed:           0x0E
Version:        0x02
ProjKey:        0xDF
Ignored:        0x070707
DataLength:     0x00000001
Data:           0x00
```

ProjectVisibilityState (section 2.3.1.17): "GC="1517CAF1D6F9D7F9D706"" specifies the VBA project is visible. The value is obfuscated by Data Encryption (section 2.4.3). The following text is the decrypted value of ProjectVisibilityState (section 2.3.1.17) as specified by an Encrypted Data Structure (section 2.4.3.1). The text is formatted for readability:

```
Seed:           0x15
Version:        0x02
ProjKey:        0xDF
Ignored:        0x0707
DataLength:     0x00000001
Data:           0xFF
```

HostExtenderRef (section 2.3.1.18): "&H00000001={3832D640-CF90-11CF-8E43-00A0C911005A};VBE;&H00000000", specifies the list of host extenders. There is only one host extender for the VBA project.

ExtenderIndex: "&H00000001" specifies the host extender entry is "1".

ExtenderGuid: "{3832D640-CF90-11CF-8E43-00A0C911005A}" specifies the GUID of the Automation type library to extend.

LibName: "VBE" specifies a built in name for the VBA Automation type library.

CreationFlags: "&H00000000" specifies that a new extended type library for the aggregatable server must not be created if there is one available.

ProjectWorkspace record (section 2.3.1.19) specifies module window states for the three modules in the VBA project.

The first ProjectWorkspace record (section 2.3.1.19) specifies the module window state for the "ThisWorkbook" module. The **ModuleIdentifier** value, "ThisWorkbook" specifies the name of the module. The first **CodeWindow** value, "23, 23, 911, 280", specifies the coordinates of the window as follows:

WindowLeft 23

WindowTop 23

WindowRight 911

WindowBottom 280

There is no value for **WindowState** for this module.

The second ProjectWorkspace record (section 2.3.1.19) specifies the module window state for the "Sheet1" module. **ModuleIdentifier** "Sheet1" specifies the name of the module. The **CodeWindow** Value, "69, 69, 724, 317" specifies the coordinates of the window as follows:

WindowLeft = 69

WindowTop = 69

WindowRight = 724

WindowBottom = 317

The **WindowState**, "C" specifies the code window for this module is closed.

The third ProjectWorkspace record (section 2.3.1.19) specifies the module windows state for the "UserForm1" designer module. **ModuleIdentifier** "UserForm1" specifies the name of the module. The **CodeWindow** value, "0, 0, 0, 0", specifies no code window coordinates for this ProjectWorkspace record (section 2.3.1.19). The WindowState, "C", specifies the code window for this module is closed. The **DesignerWindow** value, "46, 46, 701, 294", specifies the coordinates of the window as follows:

WindowLeft = 46

WindowTop = 46

WindowRight = 701

WindowBottom = 294

WindowState: "Z" specifies the **DesignerWindow** is zoomed to fill the available viewing area.

3.1.7 VBFrame Stream Example

The following example illustrates the extended properties for a designer module.

VBFrame stream			
Offset	**Size**	**Structure**	**Value**
00000000	0123	array of bytes **- text**	VERSION 5.00\r\nBegin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} UserForm1 \r\n Caption = "UserForm1"\r\n ClientHeight = 3210\r\n ClientLeft = 45\r\n ClientTop = 345\r\n ClientWidth = 4710\r\n StartUpPosition = 1 'CenterOwner\r\n TypeInfoVer = 1\r\nEnd\r\n

The preceding table illustrates the VBFrame Stream (section 2.2.11). This stream contains an Office Form ActiveX control library as described in [MS-OFORMS].

text: This VBFrame Stream (section 2.2.11) describes the extended properties for the "UserForm1" designer module. The following text is formatted for readability:

```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} UserForm1
   Caption         =   "UserForm1"
   ClientHeight    =   3210
   ClientLeft      =   45
   ClientTop       =   345
   ClientWidth     =   4710
   StartUpPosition =   1  'CenterOwner
   TypeInfoVer     =   2
End
```

DesignerCLSID: "{C62A69F0-16DC-11CE-9E98-00AA00574A4F}" specifies the class identifier (CLSID) of the Office Form ActiveX control as described in [MS-OFORMS].

DesignerName: "UserForm1" specifies the name of the designer module.

DesignerCaption (section 2.3.5.2): "UserForm1" specifies the title text of the designer.

DesignerHeight (section 2.3.5.3): "ClientHeight = 3210" specifies the height of the designer is 3210 twips.

DesignerLeft (section 2.3.5.4): "ClientLeft = 45" specifies the left edge of the designer is 45 twips from the DesignerStartupPosition (section 2.3.5.11).

DesignerTop (section 2.3.5.5): "ClientTop = 345" specifies the top edge of the designer is 345 twips from the DesignerStartupPosition (section 2.3.5.11).

DesignerWidth (section 2.3.5.6): "ClientWidth = 4710" specifies the width of the designer is 4710 twips.

DesignerStartupPosition (section 2.3.5.11): "StartUpPosition = 1" specifies the **RelativeParent** value of "1". This specifies the designer is centered relative to its parent window. The text "'CenterOwner" is a comment as described in [MS-VBAL].

DesignerTypeInfoVer (section 2.3.5.13): "TypeInfoVer = 2" specifies the designer has been changed and saved 2 times.

3.2 Compression/Decompression Examples

3.2.1 No Compression Example

The following string illustrates an ASCII text string with a set of characters that cannot be compressed by the compression algorithm specified in section 2.4.1.

```
abcdefghijklmnopqrstuv.
```

This example is provided to demonstrate the results of compressing and decompressing the string using an interoperable implementation of the algorithm specified in section 2.4.1.

The following hex array represents the compressed byte array of the example string as compressed by the compression algorithm.

```
01 19 B0 00 61 62 63 64 65 66 67 68 00 69 6A 6B 6C
6D 6E 6F 70 00 71 72 73 74 75 76 2E
```

The following hex array represents the decompressed byte array of the example string as decompressed by the decompression algorithm.

```
61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71
72 73 74 75 76 2E
```

3.2.2 Normal Compression Example

The following string illustrates an ASCII text string with a typical set of characters that can be compressed by the compression algorithm.

```
#aaabcdefaaaaghijaaaaaklaaamnopqaaaaaaaaaaaarstuvwxyzaaa
```

This example is provided to demonstrate the results of compressing and decompressing the example string using an interoperable implementation of the algorithm specified in section 2.4.1.

The following hex array represents the compressed byte array of the example string as compressed by the compression algorithm:

```
01 2F B0 00 23 61 61 61 62 63 64 65 82 66 00 70
61 67 68 69 6A 01 38 08 61 6B 6C 00 30 6D 6E 6F
70 06 71 02 70 04 10 72 73 74 75 76 10 77 78 79
7A 00 3C
```

The following hex array represents the decompressed byte array of the example string as decompressed by the decompression algorithm:

```
23 61 61 61 62 63 64 65  66 61 61 61 61 67 68 69
6a 61 61 61 61 61 6B 6C  61 61 61 6D 6E 6F 70 71
61 61 61 61 61 61 61 61  61 61 61 61 72 73 74 75
76 77 78 79 7A 61 61 61
```

3.2.3 Maximum Compression Example

The following illustrates a set of repeating characters that represent a string that can be maximally compressed using the compression algorithm.

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

This example is provided to demonstrate the results of compressing and decompressing the example string using an interoperable implementation of the algorithm specified in section 2.4.1.

The following hex array represents the compressed byte array of the example string as compressed by the compression algorithm:

```
01 03 B0 02 61 45 00
```

The following hex array represents the decompressed byte array of the example string as decompressed by the decompression algorithm:

```
61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
61 61 61 61 61 61 61 61 61
```

4 Security Considerations

4.1 Project Integrity Verification

Visual Basic for Applications (VBA) uses the **MD5** algorithm to create a cryptographic digest for the VBA project – see Contents Hash (section 2.4.2). This cryptographic digest can be stored externally and used to verify the integrity of the VBA project.

4.2 Encryption Method

When data in a VBA project, such as a password, is encrypted, the information necessary to decrypt the data is stored with the encrypted data. The design of this encryption is to obfuscate sensitive information, not to secure it. For more information, see Data Encryption (section 2.4.3). Following is a list of encrypted items:

- Project password – see ProjectPassword (section 2.3.1.16)
- Project protection state – see ProjectProtectionState (section 2.3.1.15)
- Project visibility state – see ProjectVisibilityState (section 2.3.1.17)

5 Appendix A: Product Behavior

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

- Microsoft Office 97
- Microsoft Office 2000
- Microsoft Office XP
- Microsoft Office 2003
- the 2007 Microsoft Office system
- Microsoft Office 2010 suites
- Microsoft Office 2013

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 1.5: This persistence format provides interoperability with applications that create or read documents conforming to this structure, including Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, and PowerPoint 2010.

<2> Section 2.1.1.9: VBA 5.0 does not require the name to be an identifier.

<3> Section 2.3.1.11: VBA 5.0 uses the file name of the containing document.

<4> Section 2.3.1.14: VBA 5.0 does not write this record.

<5> Section 2.3.1.15: VBA 5.0 will save 0x00000000 regardless of protection state.

<6> Section 2.3.1.16: VBA 5.0 will save the encrypted plain text password.

<7> Section 2.3.2.1: MAY be 0x00000000 even though SizeOfLicenseKey is not zero. This happens when a document is originally created with an ActiveX control that requires license-aware object creation, and then resaved after the ActiveX control removes that requirement.

<8> Section 2.3.4.2.1.11: VBA will write user-specified values between −32768 and 32767. However, VBA will only read values between -9999 and 32767.

<9> Section 2.3.4.2.2.3: VBA 5.0 does not use **OriginalRecord.**

<10> Section 2.3.4.2.2.3: VBA 5.0 uses **LibidTwiddled** to specify a twiddled type library

<11> Section 2.4.1.3.10: The 0x00 byte padding is indistinguishable from bytes in the original **DecompressedChunk**. Thus, it is possible for an application of the **Compression algorithm**

followed by an application of the **Decompression algorithm** to result in a **DecompressedBuffer** that contains more bytes than the original.

6 Change Tracking

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

7 Index

_VBA_PROJECT example 84
_VBA_PROJECT stream 19

A

ABNF rules 14
Algorithms
  compression 56
  contents hash 74
  data encryption 76
  decompression 56
  encryption 76
  password hash 80
Applicability 12

B

Byte ordering - overview 11

C

Change tracking 113
Compression algorithms 56
Contents hash algorithm 74
Conventions 14

D

Data encryption algorithm 76
Decompressed module stream example – Sheet1 102
Decompressed module stream example - ThisWorkbook 101
Decompressed module stream example – UserForm1 103
Decompression algorithms 56
Designer storage 20
Details
  _VBA_PROJECT stream 19
  ABNF rules 14
  compression algorithms 56
  contents hash algorithm 74
  conventions 14
  data encryption algorithm 76
  decompression algorithms 56
  designer storage 20
  dir stream 19
  file structure 18
  module stream 19
  password hash algorithm 80
  project root storage 19
  PROJECT stream 19
  PROJECT stream project information 20
  PROJECTlk stream 20
  PROJECTlk stream ActiveX control information 27
  PROJECTwm stream 19
  PROJECTwm stream module name information 28
  pseudocode 18
  SRP stream 19
  VBA storage 19
  VBA storage project information 29
  VBFrame stream 20
  VBFrame stream designer information 52
dir stream 19
dir stream example 84

E

Encryption algorithm 76
Examples 84
  _VBA_PROJECT 84
  decompressed module stream – Sheet1 102
  decompressed module stream - ThisWorkbook 101
  decompressed module stream – UserForm1 103
  dir stream 84
  maximum compression 109
  no compression 108
  normal compression 108
  PROJECT stream 103
  Sheet1 decompressed module stream 102
  ThisWorkbook decompressed module stream 101
  UserForm1 decompressed module stream 103
  VBFrame stream 107

F

Fields - vendor-extensible 13
File structure 18

G

Glossary 8

I

Informative references 10
Introduction 8

L

Localization 12

M

Maximum compression example 109
Module stream 19

N

No compression example 108
Normal compression example 108
Normative references 9

O

Overview (synopsis) 10

P

Password hash algorithm 80
Product behavior 111
Project information - overview 10
Project items - overview 11
Project references - overview 10
Project root storage 19
PROJECT stream 19
PROJECT stream example 103
PROJECT stream project information 20
PROJECTlk stream 20
PROJECTlk stream ActiveX control information 27
PROJECTwm stream 19
PROJECTwm stream module name information 28
Pseudocode 18

R

References 9
  informative 10
  normative 9
Relationship to protocols and other structures 12

S

Security
  encryption method 110
  project integrity verification 110
Security – encryption method 110
Security - project integrity verification 110
Sheet1 decompressed module stream example 102
SRP stream 19
Structures
  _VBA_PROJECT stream 19
  ABNF rules 14
  compression algorithms 56
  contents hash algorithm 74
  conventions 14
  data encryption algorithm 76
  decompression algorithms 56
  designer storage 20
  dir stream 19
  module stream 19
  password hash algorithm 80
  project root storage 19
  PROJECT stream 19
  PROJECT stream project information 20
  PROJECTlk stream 20
  PROJECTlk stream ActiveX control information 27
  PROJECTwm stream 19
  PROJECTwm stream module name information 28
  pseudocode 18
  SRP stream 19
  VBA storage 19
  VBA storage project information 29
  VBFrame stream 20
  VBFrame stream designer information 52

T

ThisWorkbook decompressed module stream example 101
Tracking changes 113

U

UserForm1 decompressed module stream example 103

V

VBA storage 19
VBA storage project information 29
VBFrame stream 20
VBFrame stream designer information 52
VBFrame stream example 107
Vendor-extensible fields 13
Versioning 12