Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](https://microsoft.com/openspecifications/promise) or the [Microsoft Community Promise](https://microsoft.com/openspecifications/community承诺). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting [iplg@microsoft.com](mailto:iplg@microsoft.com).
- **License Programs.** To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](https://microsoft.com/openspecifications/patent).
- **Trademarks.** The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit [www.microsoft.com/trademarks](https://www.microsoft.com/trademarks).
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

**Support.** For questions and support, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com).
## Revision Summary

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision History</th>
<th>Revision Class</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>6/27/2008</td>
<td>1.0</td>
<td>New</td>
<td>Initial Availability</td>
</tr>
<tr>
<td>8/15/2008</td>
<td>1.01</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>1/16/2009</td>
<td>1.02</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>7/13/2009</td>
<td>1.03</td>
<td>Major</td>
<td>Changes made for template compliance</td>
</tr>
<tr>
<td>8/28/2009</td>
<td>1.04</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>11/6/2009</td>
<td>1.05</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>2/19/2010</td>
<td>2.0</td>
<td>Minor</td>
<td>Updated the technical content</td>
</tr>
<tr>
<td>3/31/2010</td>
<td>2.01</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>4/30/2010</td>
<td>2.02</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>6/7/2010</td>
<td>2.03</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>6/29/2010</td>
<td>2.04</td>
<td>Editorial</td>
<td>Changed language and formatting in the technical content.</td>
</tr>
<tr>
<td>7/23/2010</td>
<td>2.04</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>9/27/2010</td>
<td>2.04</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>11/15/2010</td>
<td>2.05</td>
<td>Editorial</td>
<td>Changed language and formatting in the technical content.</td>
</tr>
<tr>
<td>12/17/2010</td>
<td>2.05</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>3/18/2011</td>
<td>2.05</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>6/10/2011</td>
<td>2.05</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>1/20/2012</td>
<td>2.6</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>4/11/2012</td>
<td>2.6</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>7/16/2012</td>
<td>2.6</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>10/8/2012</td>
<td>2.7</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>2/11/2013</td>
<td>2.7.1</td>
<td>Editorial</td>
<td>Changed language and formatting in the technical content.</td>
</tr>
<tr>
<td>7/30/2013</td>
<td>2.7.1</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>11/18/2013</td>
<td>2.7.1</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>2/10/2014</td>
<td>2.7.1</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>Date</td>
<td>Revision History</td>
<td>Revision Class</td>
<td>Comments</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td>----------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>4/30/2014</td>
<td>3.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>7/31/2014</td>
<td>3.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>10/30/2014</td>
<td>3.2</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>3/16/2015</td>
<td>4.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>9/4/2015</td>
<td>4.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>7/15/2016</td>
<td>4.1</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>9/14/2016</td>
<td>4.1</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>6/20/2017</td>
<td>4.2</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>9/19/2017</td>
<td>5.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>12/12/2017</td>
<td>5.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>4/27/2018</td>
<td>6.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>8/28/2018</td>
<td>7.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>12/11/2018</td>
<td>7.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>6/18/2019</td>
<td>8.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>11/19/2019</td>
<td>9.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>2/19/2020</td>
<td>9.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>4/22/2021</td>
<td>10.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>8/17/2021</td>
<td>11.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
</tbody>
</table>
# Table of Contents

1 Introduction ............................................................................................... 8
   1.1 Glossary ................................................................................................. 8
   1.2 References ........................................................................................... 11
      1.2.1 Normative References .................................................................... 11
      1.2.2 Informative References .................................................................. 11
   1.3 Overview ................................................................................................ 12
      1.3.1 Project Information ......................................................................... 12
      1.3.2 Project References .......................................................................... 12
      1.3.3 Project Items .................................................................................. 12
      1.3.4 Byte Ordering ................................................................................. 13
   1.4 Relationship to Protocols and Other Structures ...................................... 13
   1.5 Applicability Statement ......................................................................... 14
   1.6 Versioning and Localization .................................................................. 14
   1.7 Vendor-Extensible Fields ...................................................................... 14

2 Structures .................................................................................................. 15
   2.1 Conventions ......................................................................................... 15
      2.1.1 ABNF Rules ................................................................................ 15
         2.1.1.1 Common ABNF Rules .............................................................. 15
         2.1.1.2 ANYCHAR ............................................................................... 15
         2.1.1.3 EQ ........................................................................................ 15
         2.1.1.4 FLOAT .................................................................................. 15
         2.1.1.5 GUID .................................................................................... 16
         2.1.1.6 HEXINT32 ........................................................................... 16
         2.1.1.7 INT32 ................................................................................... 16
         2.1.1.8 LibidReference ..................................................................... 16
         2.1.1.9 ModuleIdentifier .................................................................. 17
         2.1.1.10 NWLN ............................................................................... 17
         2.1.1.11 PATH ............................................................................... 17
         2.1.1.12 ProjectReference ................................................................. 17
         2.1.1.13 QUOTEDCHAR .................................................................... 18
         2.1.1.14 VBABOOL .......................................................................... 18
         2.1.1.15 VbaIdentifier .................................................................... 18
      2.1.2 Pseudocode ................................................................................... 18
   2.2 File Structure ....................................................................................... 19
      2.2.1 Project Root Storage .................................................................... 19
      2.2.2 VBA Storage ................................................................................. 19
      2.2.3 _VBA_PROJECT Stream .................................................................. 20
      2.2.4 dir Stream ..................................................................................... 20
      2.2.5 Module Stream ............................................................................. 20
      2.2.6 SRP Streams ............................................................................... 20
      2.2.7 PROJECT Stream ......................................................................... 20
      2.2.8 PROJECTtm Stream ..................................................................... 20
      2.2.9 PROJECTtk Stream ..................................................................... 20
      2.2.10 Designer Storages ..................................................................... 20
      2.2.11 VBFrame Stream ....................................................................... 21
   2.3 Record Types ........................................................................................ 21
      2.3.1 PROJECT Stream: Project Information ......................................... 21
         2.3.1.1 ProjectProperties ................................................................. 21
         2.3.1.2 ProjectId ............................................................................. 21
         2.3.1.3 ProjectModule ...................................................................... 22
         2.3.1.4 ProjectDocModule ............................................................... 22
         2.3.1.5 ProjectStdModule ................................................................. 22
         2.3.1.6 ProjectClassModule ............................................................. 22
         2.3.1.7 ProjectDesignerModule ..................................................... 22
2.3.1.8 ProjectPackage ................................................................. 23
2.3.1.9 ProjectHelpFile .............................................................. 23
2.3.1.10 ProjectExeName32 .......................................................... 23
2.3.1.11 ProjectName ................................................................. 23
2.3.1.12 ProjectHelpId ............................................................... 23
2.3.1.13 ProjectDescription ........................................................ 24
2.3.1.14 ProjectVersionCompat32 .................................................. 24
2.3.1.15 ProjectProtectionState ..................................................... 24
2.3.1.16 ProjectPassword ............................................................ 25
2.3.1.17 ProjectVisibilityState ..................................................... 25
2.3.1.18 HostExtenders ............................................................... 26
2.3.1.19 ProjectWorkspace .......................................................... 26
2.3.1.20 ProjectWindowRecord ..................................................... 27

2.3.2 PROJECTtlk Stream: ActiveX Control Information ...................... 28
2.3.2.1 LICENSEINFO Record ....................................................... 28

2.3.3 PROJECTwm Stream: Module Name Information ............................ 29
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 ..................................... 31
  2.3.4.2.1.1 PROJECTSYSKIND Record ..................................... 32
  2.3.4.2.1.2 PROJECTCOMPATVERSION Record ............................... 33
  2.3.4.2.1.3 PROJECTLCID Record ........................................... 33
  2.3.4.2.1.4 PROJECTLCIDINVOKE Record ................................ 33
  2.3.4.2.1.5 PROJECTCODEPAGE Record .................................... 34
  2.3.4.2.1.6 PROJECTNAME Record ........................................... 34
  2.3.4.2.1.7 PROJECTDOSTRING Record ................................... 35
  2.3.4.2.1.8 PROJECTHELPFILEPATH Record ................................ 35
  2.3.4.2.1.9 PROJECTHELPCONTEXT Record ................................. 36
  2.3.4.2.1.10 PROJECTLIBFLAGS Record .................................... 36
  2.3.4.2.1.11 PROJECTVERSION Record ....................................... 37
  2.3.4.2.1.12 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 REFERENCERESPECIFIED Record ................................. 42
  2.3.4.2.2.5 REFERENCEREGISTERED Record .................................. 42
  2.3.4.2.2.6 REFERENCESPROJECT Record .................................... 43

  2.3.4.2.3 PROJECTMODULES Record .......................................... 44
  2.3.4.2.3.1 PROJECTCOOKIE Record ......................................... 44
  2.3.4.2.3.2 MODULE Record .................................................. 44
    2.3.4.2.3.2.1 MODULENAME Record ...................................... 46
    2.3.4.2.3.2.2 MODULENAMEUNICODE Record ................................ 46
    2.3.4.2.3.2.3 MODULESTREAMNAME Record ................................. 47
    2.3.4.2.3.2.4 MODULEDOSTRING Record ................................... 47
    2.3.4.2.3.2.5 MODULEOFFSET Record ....................................... 48
    2.3.4.2.3.2.6 MODULEHELPCONTEXT Record ................................ 48
    2.3.4.2.3.2.7 MODULECOOKIE Record ....................................... 49
    2.3.4.2.3.2.8 MODULETYPE Record .......................................... 49
    2.3.4.2.3.2.9 MODULEREADEONLY Record .................................. 49
    2.3.4.2.3.2.10 MODULEPRIVATE Record ..................................... 50

  2.3.4.3 Module Stream: Visual Basic Modules ................................ 50

2.3.5 VBFrame Stream: Designer Information .................................. 51
2.3.5.1 DesignerProperties ..................................................... 51
2.3.5.2 DesignerCaption .......................................................... 51
2.3.5.3 DesignerHeight ............................................................ 52
2.4 Algorithms ................................................................. 54
  2.4.1 Compression and Decompression .................................. 54
    2.4.1.1 Structures .................................................. 55
      2.4.1.1.1 CompressedContainer ................................ 55
      2.4.1.1.2 DecompressedBuffer .................................. 56
      2.4.1.1.3 DecompressedChunk ................................. 56
      2.4.1.1.4 CompressedChunk ................................. 56
      2.4.1.1.5 CompressedChunkHeader ............................ 57
      2.4.1.1.6 CompressedChunkData .............................. 57
      2.4.1.1.7 TokenSequence .................................... 58
      2.4.1.1.8 CopyToken ........................................ 58
    2.4.1.2 State Variables ........................................ 59
    2.4.1.3 Algorithms ............................................. 59
      2.4.1.3.1 Decompression Algorithm ........................... 59
      2.4.1.3.2 Decompressing a CompressedChunk .................. 60
      2.4.1.3.3 Decompressing a RawChunk ......................... 61
      2.4.1.3.4 Decompressing a TokenSequence .................... 61
      2.4.1.3.5 Decompressing a Token ............................ 61
      2.4.1.3.6 Compression algorithm ............................ 62
      2.4.1.3.7 Decompressing a DecompressedChunk ............... 62
      2.4.1.3.8 Decompressing a TokenSequence .................... 63
      2.4.1.3.9 Decompressing a Token ............................. 64
      2.4.1.3.10 Decompressing a RawChunk ....................... 65
      2.4.1.3.11 Byte Copy ......................................... 66
      2.4.1.3.12 Extract CompressedChunkSize ..................... 66
      2.4.1.3.13 Pack CompressedChunkSize ........................ 67
      2.4.1.3.14 Pack CompressedChunkSignature ................... 67
      2.4.1.3.15 Extract CompressedChunkFlag ..................... 67
      2.4.1.3.16 Pack CompressedChunkFlag ........................ 67
      2.4.1.3.17 Extract FlagBit .................................. 68
      2.4.1.3.18 Set FlagBit ....................................... 68
      2.4.1.3.19 CopyToken Algorithms ............................ 68
        2.4.1.3.19.1 CopyToken Help ................................ 69
        2.4.1.3.19.2 Unpack CopyToken ............................. 69
        2.4.1.3.19.3 Pack CopyToken ................................ 70
        2.4.1.3.19.4 Matching ...................................... 70
    2.4.2 Contents Hashes ............................................. 71
      2.4.2.1 Content Normalized Data ................................ 72
      2.4.2.2 Forms Normalized Data ............................... 73
      2.4.2.3 Content Hash .......................................... 75
      2.4.2.4 Agile Content Hash .................................... 75
    2.4.3 Data Encryption .............................................. 76
      2.4.3.1 Encrypted Data Structure ............................. 76
      2.4.3.2 Encryption ........................................... 76
      2.4.3.3 Decryption ............................................ 78
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. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

- **absolute path**: A string that identifies the location of a file and that begins with a drive identifier and root directory or network share and ends with the complete file name. Examples are C:\Documents\Work\example.txt and \netshare\Documents\Work\example.txt.

- **ActiveX control**: A reusable software control, such as a check box or button, that uses ActiveX technology and provides options to users or runs macros or scripts that automate a task. See also ActiveX object.

- **ActiveX control library**: A collection of controls that incorporate ActiveX technology and can be called by a Microsoft Visual Basic for Applications (VBA) project.

- **aggregatable server**: A COM server that can be contained by another COM server and can allow its interfaces to be used as if they were defined by the containing server.

- **automation server**: An application that exposes its functionality through COM interfaces to other applications, which are also called automation clients.

- **automation type library**: A COM server that provides descriptions of the automation-compatible types that are defined or referenced in an automation scope.

- **big-endian**: Multiple-byte values that are byte-ordered with the most significant byte stored in the memory location with the lowest address.

- **class identifier (CLSID)**: A GUID that identifies a software component; for instance, a DCOM object class or a COM class.

- **class module**: A module that contains the definition for a new object. Each instance of a class creates a new object, and procedures that are defined in the module become properties and methods of the object.

- **code page**: An ordered set of characters of a specific script in which a numerical index (code-point value) is associated with each character. Code pages are a means of providing support for character sets and keyboard layouts used in different countries. Devices such as the display and keyboard can be configured to use a specific code page and to switch from one code page (such as the United States) to another (such as Portugal) at the user's request.

- **compat version**: A 32-bit number that identifies the Office Model version used by a VBA project.

- **compilation constant**: A Microsoft Visual Basic identifier that is defined by using the #Const compiler directive or defined in the host application, and then used by other compiler directives to determine when or if certain blocks of Visual Basic code are compiled.

- **designer**: A visual design surface for adding and arranging controls on a user form and writing code for those controls.
**designer module**: A VBA module that extends the methods and properties of an ActiveX control that has been registered with the project.

**digest**: The fixed-length output string from a one-way hash function that takes a variable-length input string and is probabilistically unique for every different input string. Also, a cryptographic checksum of a data (octet) stream.

**document module**: A type of VBA project item that specifies a module for embedded macros and programmatic access operations that are associated with a document.

**embedded macro**: A macro that is saved with a document.

**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.

**floating-point number**: A number that is represented by a mantissa and an exponent according to a given base. The mantissa is typically a value between "0" and "1". To find the value of a floating-point number, the base is raised to the power of the exponent, and the mantissa is multiplied by the result.

**globally unique identifier (GUID)**: A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the GUID. See also universally unique identifier (UUID).

**hash**: A fixed-size result that is obtained by applying a one-way mathematical function, which is sometimes referred to as a hash algorithm, to an arbitrary amount of data. If the input data changes, the hash also changes. The hash can be used in many operations, including authentication and digital signing.

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

**Help topic identifier**: A unique identifier for an article that contains Help content.

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

**language code identifier (LCID)**: A 32-bit number that identifies the user interface human language dialect or variation that is supported by an application or a client computer.

**license key**: An array of bytes that enables access to a control according to the usage policies for that control.

**little-endian**: Multiple-byte values that are byte-ordered with the least significant byte stored in the memory location with the lowest address.

**module**: A collection of routines and data structures that performs a specific task or implements a specific abstract data type. Modules usually consist of two parts, a module header and a module body. A module header is a set of name/value attribute pairs that specify the linguistic characteristics of the module. A module body is the VBA source code, a set of declarations followed by procedures. VBA supports two types of modules, procedural modules and class modules.

**multibyte character set (MBCS)**: An alternative to Unicode for supporting character sets, like Japanese and Chinese, that cannot be represented in a single byte. Under MBCS, characters are encoded in either one or two bytes. In two-byte characters, the first byte, or "lead" byte, signals that both it and the following byte are to be interpreted as one character. The first byte comes
from a range of codes reserved for use as lead bytes. Which ranges of bytes can be lead bytes depends on the code page in use. For example, Japanese code page 932 uses the range 0x81 through 0x9F as lead bytes, but Korean code page 949 uses a different range.

**OLE compound file**: A form of structured storage, as described in [MS-CFB]. A compound file allows independent storages and streams to exist within a single file.

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

**procedural module**: A collection of subroutines and functions.

**project package**: An item in a VBA project that specifies a Designer class that can be extended in a designer module.

**reference**: A link in a project to another project, a .NET Framework assembly, or a compatible Component Object Model (COM) library. Adding a reference to a project enables use of the referenced item in a project, but does not copy it to the current project folder.

**relative path**: A path that is implied by the active working directory or is calculated based on a specified directory. If users enter a command that refers to a file and the full path is not entered, the active working directory is the relative path of the referenced file.

**right-to-left**: A reading and display order that is optimized for right-to-left languages.

**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".

**SHA-1**: An algorithm that generates a 160-bit hash value from an arbitrary amount of input data, as described in [RFC3174]. SHA-1 is used with the Digital Signature Algorithm (DSA) in the Digital Signature Standard (DSS), in addition to other algorithms and standards.

**storage**: An element of a compound file that is a unit of containment for one or more storages and streams, analogous to directories in a file system, as described in [MS-CFB].

**stream**: An element of a compound file, as described in [MS-CFB]. A stream contains a sequence of bytes that can be read from or written to by an application, and they can exist only in storages.

**twiddled type library**: A modified Automation type library in which all controls are marked as extensible. A twiddled type library is generated automatically by the Visual Basic Editor when a user adds one or more controls to a document.

**twip**: A unit of measurement that is used in typesetting and desktop publishing. It equals one-twentieth of a printer's point, or 1/1440 of an inch.

**Unicode**: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The Unicode standard [UNICODE5.0.0/2007] provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

**UTF-16**: A standard for encoding Unicode characters, defined in the Unicode standard, in which the most commonly used characters are defined as double-byte characters. Unless specified otherwise, this term refers to the UTF-16 encoding form specified in [UNICODES.0.0/2007] section 3.9.

**VBA environment**: An execution context that can host multiple VBA projects simultaneously and can be used to run those projects and source code, and track code dependencies.
**VBA host application**: An application that supports a VBA interpreter and can therefore run macros that are written in the VBA language.

**VBA identifier**: A VBA language token that is used to identify the name of an entity, such as a class, module, project, property, field, or variable.

**VBA project**: A collection of the modules, class modules, and user forms that are needed to create an application. Modules, class modules, and user forms can be imported into and exported from a project.

**Visual Basic for Applications (VBA)**: A macro-based programming language that derives from Visual Basic and can be used to customize and extend an application. Unlike Visual Basic, VBA code and macros can be run only from within a host application that supports VBA.

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

### 1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the [Errata](https://www.rfc-editor.org/info/rfc2119).

#### 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](mailto:dochelp@microsoft.com). We will assist you in finding the relevant information.


**Note**: Registration is required to download the document.


#### 1.2.2 Informative References


11 / 111
1.3 Overview

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, and project items.

1.3.1 Project Information

Records providing project information about the VBA project itself are contained within the following five streams:

- 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 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 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 to external ActiveX controls that are used by the VBA project.
- A REFERENCEREGISTERED (section 2.3.4.2.2.5) specifies a reference to external Automation type libraries that are used by the VBA project.
- A REFERENCEPROJECT (section 2.3.4.2.2.6) specifies a reference 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] section 4.2 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:

<table>
<thead>
<tr>
<th>Byte order</th>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big-endian</td>
<td>0x12</td>
<td>0x34</td>
<td>0x56</td>
<td>0x78</td>
</tr>
<tr>
<td>Little-endian</td>
<td>0x78</td>
<td>0x56</td>
<td>0x34</td>
<td>0x12</td>
</tr>
</tbody>
</table>

### 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.PerformanceCache (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:

\[
\text{FLOAT} = \begin{cases} 
  \text{[SIGN]} \ (1^{*}\text{DIGIT} \ "\" 1^{*}\text{DIGIT} \ \text{[EXP]} \ ) / \\
  \ (\ "\" \ 1^{*}\text{DIGIT} \ \text{[EXP]} \ ) / \\
  \ (1^{*}\text{DIGIT} \ ["\." \ [\text{EXP}] \ ]\ )
\end{cases}
\]

\[
\text{EXP} = \ "\text{e}\" \ \text{[SIGN]} \ 1^{*}\text{DIGIT}
\]

\[
\text{SIGN} = \ "\text{+}\" \ / \ "\text{-}\"
\]

2.1.1.5 GUID

Specifies a GUID.

ABNF syntax:

\[
\text{GUID} = \{\text{8HEXDIG} \ "\-\" \ 4\text{HEXDIG} \ "\-\" \ 4\text{HEXDIG} \ "\-\" \ 4\text{HEXDIG} \ "\-\" 12\text{HEXDIG}\}\}
\]

2.1.1.6 HEXINT32

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

ABNF syntax:

\[
\text{HEXINT32} = \ "\text{&H}\" \ 8\text{HEXDIG}
\]

2.1.1.7 INT32

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

ABNF syntax:

\[
\text{INT32} = \ [\ "\text{-}\" \] \ 1^{*}\text{DIGIT}
\]

2.1.1.8 LibidReference

Specifies the identifier of an Automation type library.

ABNF syntax:

\[
\text{LibidReference} = \text{"\"LibidReferenceKind LibidGuid} \\
\text{"\# LibidMajorVersion \"\" LibidMinorVersion} \\
\text{"\# LibidLcid} \\
\text{"\# LibidPath} \\
\text{"\# LibidRegName}
\]

\[
\text{LibidReferenceKind} = \ %x47 / \ %x48
\]

\[
\text{LibidGuid} = \ \text{GUID}
\]

\[
\text{LibidMajorVersion} = \ 1^{*}\text{4HEXDIG}
\]

\[
\text{LibidMinorVersion} = \ 1^{*}\text{4HEXDIG}
\]

\[
\text{LibidLcid} = \ 1^{*}\text{8HEXDIG}
\]
LibidPath          = *(%x01-22 / %x24-FF)
LibidRegName       = *255(%x01-FF)

<LibidReferenceKind>:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>%x47</td>
<td>&lt;LibidPath&gt; specifies a Windows file path.</td>
</tr>
<tr>
<td>%x48</td>
<td>&lt;LibidPath&gt; specifies a Macintosh path.</td>
</tr>
</tbody>
</table>

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

<LibidMajorVersion>: 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] section 3.3.5. 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>:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>%x41</td>
<td>The referenced VBA project is standalone and &lt;ProjectPath&gt; specifies a Windows file path.</td>
</tr>
<tr>
<td>%x42</td>
<td>The referenced VBA project is standalone and &lt;ProjectPath&gt; specifies a Macintosh path.</td>
</tr>
<tr>
<td>%x43</td>
<td>The referenced VBA project is embedded and &lt;ProjectPath&gt; specifies a Windows file path.</td>
</tr>
<tr>
<td>%x44</td>
<td>The referenced VBA project is embedded and &lt;ProjectPath&gt; specifies a Macintosh path.</td>
</tr>
</tbody>
</table>

<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-quotatio (") character.

2.1.1.14 VBABOOL
Specifies a Boolean value.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;0&quot;</td>
<td>FALSE</td>
</tr>
<tr>
<td>&quot;-1&quot;</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

ABNF syntax:

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

2.1.1.15 VbaIdentifier
Specifies a VBA Language identifier as specified by [MS-VBAL] section 3.3.5.

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.

![Diagram of VBA storage hierarchy]

**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 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 that specifies VBA project properties, project references, 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 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 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 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 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 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 as specified by [MS-OFORMS] section 2.
2.2.11 VBFrame Stream

A stream 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 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 2.3.4.2.1.5).

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] section 4.2.
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 2.3.4.2.1.8). 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*128 QUOTEDCHAR

   <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 2.3.4.2.1.6). SHOULD be an identifier as specified by [MS-VBAL] section 3.3.5. 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 2.3.4.2.1.9).

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 2.3.4.2.1.7).

2.3.1.14 ProjectVersionCompat32
Specifies the storage format version of the VBA project. MAY be missing<4>.

ABNF syntax:

```
ProjectVersionCompat32 = "VersionCompatible32=" DQUOTE "3932220000" 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:

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 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 2.3.4.2.1.5), and a Length parameter equal to the number of bytes in the password string including the terminating null character.

When the data specified by <EncryptedPassword> 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. \texttt{<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:

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

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

ExtenderIndex = HEXINT32
ExtenderGuid = GUID
LibName = "VBE" / *(%x21-3A / %x3C-FF)
CreationFlags = HEXINT32
```

\texttt{<HostExtenderRef>}: Specifies a reference to an aggregatable server’s Automation type library.

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

\texttt{<ExtenderGuid>}: Specifies the GUID of the Automation type library to extend.

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

\texttt{<CreationFlags>}: Specifies a host-provided flag as follows:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>MUST NOT create a new extended type library for the aggregatable server if one is already available to the VBA environment.</td>
</tr>
<tr>
<td>0x00000001</td>
<td>MUST create a new extended type library for the aggregatable server.</td>
</tr>
</tbody>
</table>

#### 2.3.1.19 ProjectWorkspace

Specifies a list of module editor window states.

ABNF syntax:

```latex
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
```

- `<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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Closed.</td>
</tr>
<tr>
<td>Z</td>
<td>Zoomed to fill the available viewing area.</td>
</tr>
<tr>
<td>I</td>
<td>Minimized to an icon.</td>
</tr>
</tbody>
</table>
2.3.2 PROJECTlk Stream: ActiveX Control Information

Specifies license information for ActiveX controls.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
| Version | Count |
| ... | LicenseInfoRecords (variable) |
| ... |

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.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
| ClassID (16 bytes) |
| ... |
| ... |
| SizeOfLicenseKey |
| LicenseKey (variable) |
| ... |
| LicenseRequired |

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. If 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, it can be 0x00000000 even though SizeOfLicenseKey is not zero.
2.3.3 PROJECTwm Stream: Module Name Information

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

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

NameMap (variable)

... ...

Terminator

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.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

ModuleName (variable)

... ...

ModuleNameUnicode (variable)

... ...

ModuleName (variable): A null-terminated string that specifies a module name. MUST contain MBCS characters encoded using the code page specified by PROJECTCODEPAGE (section 2.3.4.2.1.5). 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.

| 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 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| Reserved1 | Version |
| Reserved2 | Reserved3 | PerformanceCache (variable) |
| ... |

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, and modules. The entire stream MUST be compressed as specified in Compression (section 2.4.1).

| 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 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| InformationRecord (variable) |
| ... |
| ReferencesRecord (variable) |
| ... |
| ModulesRecord (variable) |
| ... |
| Terminator | Reserved |
| ... |

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.

| 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 |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

- **SysKindRecord**
  - ...

- **CompatVersionRecord**
  - ...

- **LcidRecord**
  - ...

- **LcidInvokeRecord**
  - ...

- **CodePageRecord**
  - ...

- **NameRecord (variable)**
  - ...

- **DocStringRecord (variable)**
  - ...

- **HelpFilePathRecord (variable)**
  - ...

- **HelpContextRecord**
  - ...
SysKindRecord (10 bytes): A PROJECTSYSKIND Record (section 2.3.4.2.1.1).
CompatVersionRecord (10 bytes): A PROJECTCOMPATVERSION Record (section 2.3.4.2.1.2).
LcidRecord (10 bytes): A PROJECTLCID Record (section 2.3.4.2.1.3).
LcidInvokeRecord (10 bytes): A PROJECTLCIDINVOKERecord (section 2.3.4.2.1.4).
CodePageRecord (8 bytes): A PROJECTCODEPAGE Record (section 2.3.4.2.1.5).
NameRecord (variable): A PROJECTNAME Record (section 2.3.4.2.1.6).
DocStringRecord (variable): A PROJECTDOCSTRING Record (section 2.3.4.2.1.7).
HelpFilePathRecord (variable): A PROJECTHELPFILEPATH Record (section 2.3.4.2.1.8).
HelpContextRecord (10 bytes): A PROJECTHELPCONTEXT Record (section 2.3.4.2.1.9).
LibFlagsRecord (10 bytes): A PROJECTLIBFLAGS Record (section 2.3.4.2.1.10).
VersionRecord (12 bytes): A PROJECTVERSION Record (section 2.3.4.2.1.11).
ConstantsRecord (variable): A PROJECTCONSTANTS Record (section 2.3.4.2.1.12). This field is optional.

2.3.4.2.1.1 PROJECTSYSKIND Record

Specifies the platform for which the VBA project is created.

<table>
<thead>
<tr>
<th>Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x0001.</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

[MS-OVBA] - v20210817
Office VBA File Format Structure
Copyright © 2021 Microsoft Corporation
Release: August 17, 2021
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>For 16-bit Windows Platforms.</td>
</tr>
<tr>
<td>0x00000001</td>
<td>For 32-bit Windows Platforms.</td>
</tr>
<tr>
<td>0x00000002</td>
<td>For Macintosh Platforms.</td>
</tr>
<tr>
<td>0x00000003</td>
<td>For 64-bit Windows Platforms.</td>
</tr>
</tbody>
</table>

2.3.4.2.1.2 PROJECTCOMPATVERSION Record

Specifies the VBA project’s compat version.

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x004A.

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

CompatVersion (4 bytes): An unsigned integer that specifies the compat version value for the VBA project.

2.3.4.2.1.3 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 0x000000409.

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

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
|    | Id |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ... |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ... |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**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.

### 2.3.4.2.1.5 PROJECTCODEPAGE Record

Specifies the **VBA project’s code page**.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
|    | Id |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ... |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ... |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**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.

### 2.3.4.2.1.6 PROJECTNAME Record

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

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
|    | Id |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ... |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ... |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**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 2.3.4.2.1.5). MUST NOT contain null characters.

### 2.3.4.2.1.7 PROJECTDOCSTRING Record
Specifies the description for the VBA project.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 |
| 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 2.3.4.2.1.5). 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.

### 2.3.4.2.1.8 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 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 |
| Id | SizeOfHelpFile1 |
| ... | HelpFile1 (variable) |
| ... |

**[MS-OVBA] - v20210817**
Office VBA File Format Structure
Copyright © 2021 Microsoft Corporation
Release: August 17, 2021
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 2.3.4.2.1.5). MUST NOT contain null characters.

Reserved (2 bytes): MUST be 0x003D. MUST be ignored.


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 2.3.4.2.1.5). MUST NOT contain null characters. MUST contain the same bytes as HelpFile1.

2.3.4.2.1.9 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 2.3.4.2.1.8).

2.3.4.2.1.10 PROJECTLIBFLAGS Record

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

2.3.4.2.1.11 PROJECTVERSION Record

Specifies the version of the VBA project.

| 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. |

2.3.4.2.1.12 PROJECTCONSTANTS Record

Specifies the compilation constants for the VBA project.

| Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x000C. |
| SizeOfConstants (4 bytes): An unsigned integer specifying the size of the compilation constants. |
| Constants (variable): A variable containing the constants. |
| SizeOfConstantsUnicode (4 bytes): An unsigned integer specifying the size of the compilation constants in Unicode. |
| ConstantsUnicode (variable): A variable containing the compilation constants in Unicode. |
**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 2.3.4.2.1.5). 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.<7>

**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 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.

```
0 1 2 3 4 5 6 7 8 9 1 0 1 2 3 4 5 6 7 8 9 2 0 1 2 3 4 5 6 7 8 9 3 0 1
```

**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 to an Automation type library or VBA project.

```
0 1 2 3 4 5 6 7 8 9 1 0 1 2 3 4 5 6 7 8 9 2 0 1 2 3 4 5 6 7 8 9 3 0 1
```

**NameRecord (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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x002F</td>
<td>ReferenceRecord is a REFERENCECONTROL (section 2.3.4.2.2.3).</td>
</tr>
<tr>
<td>0x0033</td>
<td>ReferenceRecord is a REFERENCEORIGINAL (section 2.3.4.2.2.4).</td>
</tr>
<tr>
<td>0x000D</td>
<td>ReferenceRecord is a REFERENCEREGISTERED (section 2.3.4.2.2.5).</td>
</tr>
<tr>
<td>0x000E</td>
<td>ReferenceRecord is a REFERENCEPROJECT (section 2.3.4.2.2.6).</td>
</tr>
</tbody>
</table>

2.3.4.2.2.2 REFERENCENAME Record

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

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Id</td>
<td></td>
<td>SizeOfName</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td>Name (variable)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td>SizeOfNameUnicode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td>NameUnicode (variable)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 2.3.4.2.1.5). 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 to a twiddled type library and its extended type library.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| OriginalRecord (variable) |
| ... |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1d |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| SizeTwiddled |
| ... |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Reserved1|
| ... |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Reserved2 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| NameRecordExtended (variable) |
| ... |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Reserved3 |
| ... |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Reserved4 |

[MS-OVBA] - v20210817
Office VBA File Format Structure
Copyright © 2021 Microsoft Corporation
Release: August 17, 2021
<table>
<thead>
<tr>
<th>Reserved5</th>
<th>OriginalTypeLib (16 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Cookie</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

**OriginalRecord (variable):** AREFERENCEORIGINAL Record (section 2.3.4.2.2.4) that specifies the Automation type library the twiddled type library was generated from. This field is optional.

**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 <8> 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 2.3.4.2.1.5). MUST NOT contain null characters.

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

**Reserved2 (2 bytes):** MUST be 0x0000. MUST be ignored.

**NameRecordExtended (variable):** AREFERENCENAME 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 2.3.4.2.1.5). 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 0x0000. 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.

<table>
<thead>
<tr>
<th>Id</th>
<th>SizeOfLibidOriginal</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>LibidOriginal (variable)</td>
</tr>
</tbody>
</table>

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 2.3.4.2.1.5). 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 to an Automation type library.

<table>
<thead>
<tr>
<th>Id</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>SizeOfLibid</td>
</tr>
<tr>
<td>...</td>
<td>Libid (variable)</td>
</tr>
</tbody>
</table>

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 2.3.4.2.1.5). 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 0x0000. MUST be ignored.

2.3.4.2.2.6  REFERENCEPROJECT Record

Specifies a reference to an external VBA project.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Size</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>SizeOfLibidAbsolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>LibidAbsolute (variable)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>SizeOfLibidRelative</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LibidRelative (variable)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MinorVersion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 2.3.4.2.1.5). 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 2.3.4.2.1.5). 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 2.3.4.2.1.11) 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 2.3.4.2.1.11) of the referenced VBA project.

2.3.4.2.3 PROJECTMODULES Record

Specifies data for the modules in the project.

| 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 |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

<table>
<thead>
<tr>
<th>Id</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Count</td>
</tr>
</tbody>
</table>

ProjectCookieRecord

<table>
<thead>
<tr>
<th>...</th>
</tr>
</thead>
</table>

Modules (variable)

| ... |

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 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

<table>
<thead>
<tr>
<th>Id</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Cookie</td>
</tr>
</tbody>
</table>

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 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 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).

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| Id | SizeOfModuleName |
| ... | ModuleName (variable) |

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 2.3.4.2.1.5). 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).

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| Id | SizeOfModuleNameUnicode |
| ... | ModuleNameUnicode (variable) |

...
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 1 0 1 2 3 4 5 6 7 8 9 2 0 1 2 3 4 5 6 7 8 9 3 0 1 |
|------------------------|-------------------|
| 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 2.3.4.2.1.5). 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 1 0 1 2 3 4 5 6 7 8 9 2 0 1 2 3 4 5 6 7 8 9 3 0 1 |
|------------------------|-------------------|
| Id                     | SizeOfDocString   |
Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x001C.


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 2.3.4.2.1.5). 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).

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 2.3.4.2.3.3).

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).
### 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 2.3.4.2.1.8).

#### 2.3.4.2.3.2.7 MODULECOOKIE Record

Specifies ignored data.

### 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.

### 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] section 4.2. MUST contain MBCS characters encoded using the code page specified in PROJECTCODEPAGE (section 2.3.4.2.1.5).
2.3.5 VBFrame Stream: Designer Information

The VBFrame stream 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 2.3.4.2.1.5).

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>:** 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>:** 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;0&quot;</td>
<td>&quot;Manual&quot; mode. <strong>DesignerTop</strong> (section 2.3.5.5) and <strong>DesignerLeft</strong> (section 2.3.5.4) coordinates of the designer are relative to the desktop window.</td>
</tr>
<tr>
<td>&quot;1&quot;</td>
<td>&quot;CenterOwner&quot; mode. Center the designer relative to its parent window.</td>
</tr>
<tr>
<td>&quot;2&quot;</td>
<td>&quot;Center&quot; mode. Center the designer relative to the desktop window.</td>
</tr>
<tr>
<td>&quot;3&quot;</td>
<td>&quot;WindowsDefault&quot; mode. Place the designer in the upper-left corner of screen.</td>
</tr>
</tbody>
</table>
2.3.5.12  DesignerTag
Specifies user-defined data associated with the designer.

ABNF syntax:

\[
\begin{align*}
\text{DesignerTag} & \quad = \ "Tag" \ \text{EQ} \ \text{DQUOTE} \ \text{DesignerTagText} \ \text{DQUOTE} \\
\text{DesignerTagText} & \quad = \ ^{\ast}130\text{QUOTEDCHAR}
\end{align*}
\]

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

ABNF syntax:

\[
\begin{align*}
\text{DesignerTypeInfoVer} & \quad = \ "TypeInfoVer" \ \text{EQ} \ \text{INT32}
\end{align*}
\]

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

ABNF syntax:

\[
\begin{align*}
\text{DesignerVisible} & \quad = \ "Visible" \ \text{EQ} \ \text{VBABOOL}
\end{align*}
\]

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

ABNF syntax:

\[
\begin{align*}
\text{DesignerWhatsThisButton} & \quad = \ "WhatsThisButton" \ \text{EQ} \ \text{VBABOOL}
\end{align*}
\]

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:

\[
\begin{align*}
\text{DesignerWhatsThisHelp} & \quad = \ "WhatsThisHelp" \ \text{EQ} \ \text{VBABOOL}
\end{align*}
\]

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. 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 tokens.

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 0b101101100100001.

### 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. On read, the end of stream 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.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
|   |   |   |   |   |   |   |   |   |   |   | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |   |

**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 Tokens. The number of Tokens in the final TokenSequence MUST be greater than or equal to 1. The number of Tokens in the final TokenSequence MUST less than or equal to eight. All other TokenSequences in the CompressedChunkData MUST contain eight Tokens.

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 Tokens. 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.
<table>
<thead>
<tr>
<th>variable</th>
<th>variable</th>
<th>1</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>2</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>Offset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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** 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** 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
    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
ENDFOR

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 2.4.1.3) to a CompressedChunk (section 2.4.1.4) 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 theLiteralToken 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 2.4.1.4) array exceeds 4096. The bytes from the DecompressedChunk (section 2.4.1.3) are copied, with no compression, into CompressedChunk.CompressedData (section 2.4.1.4). If fewer than 4096 bytes are copied then the remaining bytes in CompressedChunk.CompressedData.Data array are padded with the literal value 0x00. 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.

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
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:

```plaintext
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 2.4.1.4) from its **CompressedChunkHeader** (section 2.4.1.5). The pseudocode takes the following input parameter:

- **Header (2 bytes)**: An instance of a **CompressedChunkHeader** (section 2.4.1.5).

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 2.4.1.4) MUST be less than or equal to 4098. MUST be greater than or equal to three.

```plaintext
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 2.4.1.4). 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 2.4.1.5).

```
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 2.4.1.5) to 0b011.

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

Header (2 bytes): An instance of a CompressedChunkHeader (section 2.4.1.5).

```
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 2.4.1.5).

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 2.4.1.5).

```
SET temp1 TO Header BITWISE AND 0x7FFF
SET temp2 TO CompressedFlag LEFT SHIFT BY 15
```
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:

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

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.

```plaintext
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.

```plaintext
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 Hashes

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.

### 2.4.2.1 Content Normalized Data

The Contents Normalized Data algorithm produces an array of bytes as output:

**ContentNormalizedData(Variable):** A Buffer containing a variable number of bytes that contains the normalized representation of the **VBAStorage**.

**ContentNormalizedData** is generated by the following pseudocode:

```plaintext
FUNCTION ContentNormalizedData
    PARAMETERS Storage AS VBA Storage (section 2.3.4)
    RETURNS array of bytes

    DECLARE Buffer AS array of bytes
    SET Buffer TO resizable array of bytes

    APPEND Buffer WITH PROJECTNAME.ProjectName (section 2.3.4.2.1.6) of Storage

    APPEND Buffer WITH PROJECTCONSTANTS.Constants (section 2.3.4.2.1.12) of Storage

    FOR EACH REFERENCE (section 2.3.4.2.2.1) IN PROJECTREFERENCES.ReferenceArray (section 2.3.4.2.2) of Storage
        IF REFERENCE.ReferenceRecord.Id = 0x000D THEN
            APPEND Buffer with 0x7B
        ELSE IF REFERENCE.ReferenceRecord.Id = 0x000E THEN
            DECLARE TempBuffer AS array of bytes
            SET TempBuffer TO resizable array of bytes

            APPEND TempBuffer WITH REFERENCE.ReferenceRecord.SizeOfLibidAbsolute
            APPEND TempBuffer WITH REFERENCE.ReferenceRecord.LibidAbsolute
            APPEND TempBuffer WITH REFERENCE.ReferenceRecord.SizeOfLibidRelative
            APPEND TempBuffer WITH REFERENCE.ReferenceRecord.LibidRelative
            APPEND TempBuffer WITH REFERENCE.ReferenceRecord.MajorVersion
            APPEND TempBuffer WITH REFERENCE.ReferenceRecord.MinorVersion
            APPEND TempBuffer WITH 0x00

            DECLARE CopyIndex AS integer
            DECLARE CopyByte AS byte
            SET CopyIndex TO 0
            SET CopyByte TO TempBuffer[CopyIndex]

            WHILE NOT CopyByte = 0x00
```

---

[MS-OVBA] - v20210817
Office VBA File Format Structure
Copyright © 2021 Microsoft Corporation
Release: August 17, 2021
APPEND Buffer WITH CopyByte
SET CopyIndex TO CopyIndex + 1
SET CopyByte TO TempBuffer[CpopyIndex]

END WHILE
END IF
END FOR

FOR EACH ModuleStream (section 2.3.4.3) IN VBA Storage (section 2.3.4) of Storage

DEFINE CompressedContainer AS array of bytes
DEFINE Text AS array of bytes

SET CompressedContainer TO ModuleStream.CompressedSourceCode
SET Text TO result of Decompression(CompressedContainer) (section 2.4.1)

DECLARE Lines AS array of array of bytes
DECLARE TextBuffer AS array of bytes

SET Lines TO resizable array of array of bytes
SET TextBuffer TO resizable array of bytes

FOR EACH Char IN Text
  IF Char = 0x10 OR Char = 0x13 THEN
    APPEND Lines WITH TextBuffer
    SET TextBuffer TO resizable array of bytes
  ELSE
    APPEND TextBuffer WITH Char
  END IF
END FOR
END FOR

FOR EACH Line IN Lines
  IF NOT Line starts with "attribute" when ignoring case THEN
    APPEND Buffer WITH Line
  END IF
END FOR
END FOR
RETURN Buffer
END FUNCTION

2.4.2.2 Forms Normalized Data

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

FormsNormalizedData(Variable): A Bugger containing a variable number of bytes that contains the normalized representation of the Designer Storages portion of the VBAStorage.
**FormsNormalizedData** is generated by the following pseudocode:

```
FUNCTION NormalizeStorage
    PARAMETERS Storage AS storage
    RETURNS array of bytes

    DECLARE StorageBuffer AS array of bytes
    SET StorageBuffer TO resizable array of bytes

    FOR EACH StorageElement (stream or storage) IN Storage
        IF StorageElement is a stream THEN
            DECLARE TempBuffer AS array of bytes
            DECLARE TempBufferIndex AS integer
            SET TempBuffer TO array of bytes(1023) initialized to 0
            SET TempBufferIndex TO 0
            FOR EACH StreamByte IN StorageElement
                SET TempBuffer[TempBufferIndex] TO StreamByte
                SET TempBufferIndex TO TempBufferIndex + 1
                IF TempBufferIndex = 1023 OR StorageElement.EndOfStream
                    APPEND StorageBuffer WITH TempBuffer ' full 1023 bytes
                    SET TempBuffer TO array of bytes(1023) initialized to 0
                    SET TempBufferIndex TO 0
                END IF
            END FOR
        END IF
        ELSE IF StorageElement is a storage THEN
            APPEND StorageBuffer WITH NormalizeStorage(StorageElement)
        END IF
    END FOR

    RETURN StorageBuffer
END FUNCTION

FUNCTION FormsNormalizedData
    PARAMETERS DesignerStorage AS Designer Storage (section 2.2.10), ProjectStream AS PROJECT Stream (section 2.3.1)
    RETURNS array of bytes

    DECLARE Buffer AS array of bytes
    SET Buffer TO resizable array of bytes

    FOR EACH ProjectDesignerModule (section 2.3.1.7) IN ProjectStream
        FOR EACH StorageElement (stream or storage) IN DesignerStorage
            IF StorageElement is a stream THEN
                DECLARE TempBuffer AS array of bytes
                DECLARE TempBufferIndex AS integer
                SET TempBuffer TO array of bytes(1023) initialized to 0
                SET TempBufferIndex TO 0
                FOR EACH StreamByte IN StorageElement
                    SET TempBuffer[TempBufferIndex] TO StreamByte
                    SET TempBufferIndex TO TempBufferIndex + 1
                    IF TempBufferIndex = 1023 OR StorageElement.EndOfStream
                        APPEND StorageBuffer WITH TempBuffer ' full 1023 bytes
                        SET TempBuffer TO array of bytes(1023) initialized to 0
                        SET TempBufferIndex TO 0
                    END IF
                END FOR
            END IF
        END FOR
    END FOR
```

[MS-OVBA] - v20210817
Office VBA File Format Structure
Copyright © 2021 Microsoft Corporation
Release: August 17, 2021
SET TempBuffer[TempBufferIndex] TO StreamByte
SET TempBufferIndex TO TempBufferIndex + 1

IF TempBufferIndex = 1023 OR StorageElement.EndOfStream
    APPEND Buffer WITH TempBuffer ' full 1023 bytes
    SET TempBuffer TO array of bytes(1023) initialized to 0
    SET TempBufferIndex TO 0
END IF

END FOR
ELSE IF StorageElement is a storage THEN
    APPEND Buffer WITH NormalizeStorage(StorageElement)
END IF

END FOR
END FOR

RETURN Buffer

END FUNCTION

2.4.2.3 Content Hash

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

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

**CryptographicDigest** is generated by the following pseudocode:

```plaintext
SET CryptographicDigest TO the cryptographic digest of the ContentNormalizedData Buffer, as generated in the Content Normalized Data (section 2.4.2.1), as specified by the hashing algorithm.
```

2.4.2.4 Agile Content Hash

The Agile Content Hash algorithm produces an array of bytes as output:

**CryptographicDigest(Variable)**: The cryptographic digest of VBAStorage.

**CryptographicDigest** is generated by the following pseudocode:

```plaintext
Set ContentBuffer TO a resizable array of bytes

APPEND ContentBuffer WITH the ContentNormalizedData Buffer, as generated in Content Normalized Data (section 2.4.2.1).

APPEND ContentBuffer WITH the FormsNormalizedData Buffer, as generated in the Forms Normalized Data (section 2.4.2.2).

SET CryptographicDigest TO the cryptographic digest of ContentBuffer as specified by the hashing algorithm.
```
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.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |
| Seed | VersionEnc | ProjKeyEnc | IgnoredEnc (variable) |
| ... |
| DataLengthEnc |
| DataEnc (variable) |
| ... |

**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:

\[
\text{VersionEnc} = \text{Seed} \oplus \text{Version}
\]

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:

\[
\text{ProjKeyEnc} = \text{Seed} \oplus \text{ProjKey}
\]

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 AND 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.

\[ \text{Version} = \text{Seed XOR VersionEnc} \]

Version MUST be 2.

ProjKey is calculated using the following formula.

\[ \text{ProjKey} = \text{Seed XOR ProjKeyEnc} \]

To initialize states for the rest of the encoding:

1. SET UnencryptedByte1 TO ProjKey.
2. SET EncryptedByte1 TO ProjKeyEnc.
3. SET EncryptedByte2 TO VersionEnc.

The length of \text{IgnoredEnc} is computed as follows.

\[ \text{IgnoredLength} = (\text{Seed BAND 6}) / 2 \]

Decrypting of \text{IgnoredEnc} MUST be as follows.

FOR each ByteEnc IN IgnoredEnc:

1. SET Byte TO (ByteEnc XOR (EncryptedByte2 + UnencryptedByte1)).
2. SET EncryptedByte2 TO EncryptedByte1.
3. SET EncryptedByte1 TO ByteEnc.
4. SET UnencryptedByte1 TO Byte.

END FOR

\text{Length} is computed by the following pseudocode.

1. SET ByteIndex TO zero.
2. FOR each ByteEnc IN DataLengthEnc:
   1. SET Byte TO (ByteEnc XOR (EncryptedByte2 + UnencryptedByte1)).
   2. SET TempValue TO 256 raised to the power of ByteIndex.
   3. MULTIPLY TempValue by Byte.
   4. ADD TempValue TO Length.
   5. SET EncryptedByte2 TO EncryptedByte1.
   6. SET EncryptedByte1 TO ByteEnc.
   7. 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 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |
| 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:

```plaintext
FOR each Byte IN InputBytes:
    IF Byte EQUALS 0x00 THEN:
```

[MS-OVBA] - v20210817
Office VBA File Format Structure
Copyright © 2021 Microsoft Corporation
Release: August 17, 2021
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:

```plaintext
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
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 2.3.4.2.1.5).
- **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:

```plaintext
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 2.3.4.2.1.5).
- **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:

```plaintext
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.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Size</th>
<th>Structure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>0007</td>
<td>_VBA_PROJECT Stream: Version Dependent Project Information - _VBA_PROJECT</td>
<td></td>
</tr>
<tr>
<td>00000000</td>
<td>0002</td>
<td>unsigned integer - Reserved1</td>
<td>0x61CC</td>
</tr>
<tr>
<td>00000002</td>
<td>0002</td>
<td>unsigned integer - Version</td>
<td>0xFFFF</td>
</tr>
<tr>
<td>00000004</td>
<td>0001</td>
<td>BYTE - Reserved2</td>
<td>0x00</td>
</tr>
<tr>
<td>00000005</td>
<td>0002</td>
<td>unsigned integer - Reserved3</td>
<td>0x0001</td>
</tr>
<tr>
<td>00000007</td>
<td>0000</td>
<td>Blob - PerformanceCache</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: _VBA_PROJECT stream

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 for a VBA project. The dir (section 2.3.4.2) stream examples describe project information, project references and modules. The dir (section 2.3.4.2) stream 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.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Size</th>
<th>Structure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>0122</td>
<td>PROJECTINFORMATION Record - Information Record</td>
<td></td>
</tr>
<tr>
<td>00000000</td>
<td>000A</td>
<td>PROJECTSYSKIND Record - SysKindRecord</td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Size</td>
<td>Structure</td>
<td>Value</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>00000000</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x0001</td>
</tr>
<tr>
<td>00000002</td>
<td>0004</td>
<td>unsigned integer - Size</td>
<td>0x0000000004</td>
</tr>
<tr>
<td>00000006</td>
<td>0004</td>
<td>unsigned integer - SysKind</td>
<td>0x00000001</td>
</tr>
<tr>
<td>0000000A</td>
<td>000A</td>
<td>PROJECTLCID Record - LcidRecord</td>
<td></td>
</tr>
<tr>
<td>0000000A</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x0002</td>
</tr>
<tr>
<td>0000000C</td>
<td>0004</td>
<td>unsigned integer - Size</td>
<td>0x0000000004</td>
</tr>
<tr>
<td>00000010</td>
<td>0004</td>
<td>unsigned integer - Lcid</td>
<td>0x000000409</td>
</tr>
<tr>
<td>00000014</td>
<td>000A</td>
<td>PROJECTLCIDINVoke Record - LcidInvokeRecord</td>
<td></td>
</tr>
<tr>
<td>00000014</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x0014</td>
</tr>
<tr>
<td>00000016</td>
<td>0004</td>
<td>unsigned integer - Size</td>
<td>0x0000000004</td>
</tr>
<tr>
<td>0000001A</td>
<td>0004</td>
<td>unsigned integer - LcidInvoke</td>
<td>0x000000409</td>
</tr>
<tr>
<td>0000001E</td>
<td>0008</td>
<td>PROJECTCODEPAGE Record - CodePageRecord</td>
<td></td>
</tr>
<tr>
<td>0000001E</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x0003</td>
</tr>
<tr>
<td>00000020</td>
<td>0004</td>
<td>unsigned integer - Size</td>
<td>0x000000002</td>
</tr>
<tr>
<td>00000024</td>
<td>0002</td>
<td>unsigned integer - CodePage</td>
<td>0x04E4</td>
</tr>
<tr>
<td>00000026</td>
<td>0010</td>
<td>PROJECTNAME Record - NameRecord</td>
<td></td>
</tr>
<tr>
<td>00000026</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x0004</td>
</tr>
<tr>
<td>00000028</td>
<td>0004</td>
<td>unsigned integer - SizeOfProjectName</td>
<td>0x0000000A</td>
</tr>
<tr>
<td>0000002C</td>
<td>000A</td>
<td>array of bytes - ProjectName</td>
<td>VBAProject</td>
</tr>
<tr>
<td>00000036</td>
<td>0069</td>
<td>PROJECTDOCSSTRING Record - DocStringRecord</td>
<td></td>
</tr>
<tr>
<td>00000036</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x0005</td>
</tr>
<tr>
<td>0000003B</td>
<td>0004</td>
<td>unsigned integer - SizeOfDocString</td>
<td>0x0000001F</td>
</tr>
<tr>
<td>0000003C</td>
<td>001F</td>
<td>array of bytes - DocString</td>
<td>Example VBA Project Description</td>
</tr>
<tr>
<td>0000005B</td>
<td>0002</td>
<td>unsigned integer - Reserved</td>
<td>0x0040</td>
</tr>
<tr>
<td>0000005D</td>
<td>0004</td>
<td>unsigned integer - SizeOfDocStringUnicode</td>
<td>0x0000003E</td>
</tr>
<tr>
<td>00000061</td>
<td>003E</td>
<td>array of bytes - DocStringUnicode</td>
<td>Example VBA Project Description</td>
</tr>
<tr>
<td>0000009F</td>
<td>0042</td>
<td>PROJECTHELPFILEPATH Record - HelpFilePathRecord</td>
<td></td>
</tr>
<tr>
<td>0000009F</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x0006</td>
</tr>
<tr>
<td>000000A1</td>
<td>0004</td>
<td>unsigned integer - SizeOfHelpFile1</td>
<td>0x0000001B</td>
</tr>
<tr>
<td>000000A5</td>
<td>001B</td>
<td>array of bytes - HelpFile1</td>
<td>c:\example path\example.hlp</td>
</tr>
<tr>
<td>000000C0</td>
<td>0002</td>
<td>unsigned integer - Reserved</td>
<td>0x003D</td>
</tr>
</tbody>
</table>
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 Sheet3.1.4 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.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Size</th>
<th>Structure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>000001</td>
<td>03B</td>
<td>VBA_Canonical_ReferenceArray - ReferenceArray</td>
<td></td>
</tr>
<tr>
<td>000001</td>
<td>008</td>
<td>REFERENCE Record - Reference[0]</td>
<td></td>
</tr>
<tr>
<td>000001</td>
<td>001</td>
<td>REFERENCENAME Record - NameRecord</td>
<td></td>
</tr>
<tr>
<td>000001</td>
<td>002</td>
<td>unsigned integer - Id</td>
<td>0x0016</td>
</tr>
<tr>
<td>000001</td>
<td>004</td>
<td>unsigned integer - SizeOfName</td>
<td>0x00000006</td>
</tr>
<tr>
<td>000001</td>
<td>006</td>
<td>array of bytes - Name</td>
<td>stdole</td>
</tr>
<tr>
<td>000001</td>
<td>002</td>
<td>unsigned integer - Reserved</td>
<td>0x003E</td>
</tr>
<tr>
<td>000001</td>
<td>004</td>
<td>unsigned integer - SizeOfNameUnicode</td>
<td>0x0000000C</td>
</tr>
<tr>
<td>000001</td>
<td>000</td>
<td>array of bytes</td>
<td>stdole</td>
</tr>
<tr>
<td>Offset</td>
<td>Size</td>
<td>Structure</td>
<td>Value</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>34</td>
<td>C</td>
<td>NameUnicode</td>
<td></td>
</tr>
<tr>
<td>000001</td>
<td>40</td>
<td>REFERENCEREGISTERED Record - ReferenceRecord[0]</td>
<td></td>
</tr>
<tr>
<td>000001</td>
<td>40</td>
<td>unsigned integer - Id</td>
<td>0x000D</td>
</tr>
<tr>
<td>000001</td>
<td>42</td>
<td>unsigned integer - Size</td>
<td>0x00000068</td>
</tr>
<tr>
<td>000001</td>
<td>46</td>
<td>unsigned integer - SizeOfLibid</td>
<td>0x0000005E</td>
</tr>
<tr>
<td>000001</td>
<td>4A</td>
<td>array of bytes</td>
<td>*\G{00020430-0000-0000-C000-00000000004E6}#2.0#0#C:\Windows\system32\stdole2.tlb#OLE Automation</td>
</tr>
<tr>
<td>000001</td>
<td>A8</td>
<td>unsigned integer - Reserved1</td>
<td>0x00000000</td>
</tr>
<tr>
<td>000001</td>
<td>AC</td>
<td>unsigned integer - Reserved2</td>
<td>0x0000</td>
</tr>
<tr>
<td>000001</td>
<td>AE</td>
<td>REFERENCE Record - Reference[1]</td>
<td></td>
</tr>
<tr>
<td>000001</td>
<td>AE</td>
<td>unsigned integer - Id</td>
<td>0x0016</td>
</tr>
<tr>
<td>000001</td>
<td>B0</td>
<td>unsigned integer - SizeOfName</td>
<td>0x00000006</td>
</tr>
<tr>
<td>000001</td>
<td>B4</td>
<td>array of bytes</td>
<td>Office</td>
</tr>
<tr>
<td>000001</td>
<td>BA</td>
<td>unsigned integer - Reserved</td>
<td>0x003E</td>
</tr>
<tr>
<td>000001</td>
<td>BC</td>
<td>unsigned integer - SizeOfNameUnicode</td>
<td>0x0000000C</td>
</tr>
<tr>
<td>000001</td>
<td>C0</td>
<td>array of bytes</td>
<td>Office</td>
</tr>
<tr>
<td>000001</td>
<td>CC</td>
<td>REFERENCEREGISTERED Record - ReferenceRecord[1]</td>
<td></td>
</tr>
<tr>
<td>000001</td>
<td>CC</td>
<td>unsigned integer - Id</td>
<td>0x000D</td>
</tr>
<tr>
<td>000001</td>
<td>CE</td>
<td>unsigned integer - Size</td>
<td>0x0000009E</td>
</tr>
<tr>
<td>000001</td>
<td>D2</td>
<td>unsigned integer - SizeOfLibid</td>
<td>0x00000094</td>
</tr>
<tr>
<td>000001</td>
<td>D6</td>
<td>array of bytes</td>
<td>*\G{2DF8D04C-5BFA-101B-BDE5-00AA044DE52}#2.0#0#C:\Program Files\Common Files\Microsoft Shared\OFFICE12\MSO.DLL#Microsoft Office 12.0 Object Library</td>
</tr>
<tr>
<td>000002</td>
<td>6A</td>
<td>unsigned integer - Reserved1</td>
<td>0x00000000</td>
</tr>
<tr>
<td>000002</td>
<td>6E</td>
<td>unsigned integer - Reserved2</td>
<td>0x0000</td>
</tr>
<tr>
<td>000002</td>
<td>70</td>
<td>REFERENCE Record - Reference[2]</td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Size</td>
<td>Structure</td>
<td>Value</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>-----------------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>000002</td>
<td>70</td>
<td>REFERENCENAME Record - NameRecord</td>
<td></td>
</tr>
<tr>
<td>000002</td>
<td>70</td>
<td>unsigned integer - Id</td>
<td>0x0016</td>
</tr>
<tr>
<td>000002</td>
<td>72</td>
<td>unsigned integer - SizeOfName</td>
<td>0x0000000B</td>
</tr>
<tr>
<td>000002</td>
<td>76</td>
<td>array of bytes - Name</td>
<td>VBAProject1</td>
</tr>
<tr>
<td>000002</td>
<td>81</td>
<td>unsigned integer - Reserved</td>
<td>0x003E</td>
</tr>
<tr>
<td>000002</td>
<td>83</td>
<td>unsigned integer - SizeOfNameUnicode</td>
<td>0x00000016</td>
</tr>
<tr>
<td>000002</td>
<td>87</td>
<td>array of bytes - NameUnicode</td>
<td>VBAProject1</td>
</tr>
<tr>
<td>000002</td>
<td>9D</td>
<td>REFERENCPROJECT Record - ReferenceRecord[2]</td>
<td></td>
</tr>
<tr>
<td>000002</td>
<td>9D</td>
<td>unsigned integer - Id</td>
<td>0x000E</td>
</tr>
<tr>
<td>000002</td>
<td>9F</td>
<td>unsigned integer - Size</td>
<td>0x0000005E</td>
</tr>
<tr>
<td>000002</td>
<td>A3</td>
<td>unsigned integer - SizeOfLibidAbsolute</td>
<td>0x00000030</td>
</tr>
<tr>
<td>000002</td>
<td>A7</td>
<td>array of bytes - LibidAbsolute</td>
<td>*\CC:\Example Path\Example-ReferencedProject.xls</td>
</tr>
<tr>
<td>000002</td>
<td>D7</td>
<td>unsigned integer - SizeOfLibidRelative</td>
<td>0x00000020</td>
</tr>
<tr>
<td>000002</td>
<td>DB</td>
<td>array of bytes - LibidRelative</td>
<td>*\CExample-ReferencedProject.xls</td>
</tr>
<tr>
<td>000002</td>
<td>FB</td>
<td>unsigned integer - MajorVersion</td>
<td>0x49A95F46</td>
</tr>
<tr>
<td>000002</td>
<td>FF</td>
<td>unsigned integer - MinorVersion</td>
<td>0x000D</td>
</tr>
<tr>
<td>000003</td>
<td>01</td>
<td>REFERENCE Record - Reference[3]</td>
<td></td>
</tr>
<tr>
<td>000003</td>
<td>01</td>
<td>REFERENCENAME Record - NameRecord</td>
<td></td>
</tr>
<tr>
<td>000003</td>
<td>01</td>
<td>unsigned integer - Id</td>
<td>0x0016</td>
</tr>
<tr>
<td>000003</td>
<td>03</td>
<td>unsigned integer - SizeOfName</td>
<td>0x00000007</td>
</tr>
<tr>
<td>000003</td>
<td>07</td>
<td>array of bytes - Name</td>
<td>MSForms</td>
</tr>
<tr>
<td>000003</td>
<td>0E</td>
<td>unsigned integer - Reserved</td>
<td>0x003E</td>
</tr>
<tr>
<td>000003</td>
<td>10</td>
<td>unsigned integer - SizeOfNameUnicode</td>
<td>0x0000000E</td>
</tr>
<tr>
<td>Offset</td>
<td>Size</td>
<td>Structure</td>
<td>Value</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>000003 14</td>
<td>000 E</td>
<td>array of bytes - NameUnicode</td>
<td>MSForms</td>
</tr>
<tr>
<td>000003 22</td>
<td>018 0</td>
<td>REFERENCECONTROL Record - ReferenceRecord[3]</td>
<td></td>
</tr>
<tr>
<td>000003 22</td>
<td>007 5</td>
<td>REFERENCEORIGINAL Record - OriginalRecord</td>
<td></td>
</tr>
<tr>
<td>000003 22</td>
<td>000 2</td>
<td>unsigned integer - Id</td>
<td>0x0033</td>
</tr>
<tr>
<td>000003 24</td>
<td>000 4</td>
<td>unsigned integer - SizeOfLibidOriginal</td>
<td>0x0000006F</td>
</tr>
<tr>
<td>000003 28</td>
<td>000 4</td>
<td>array of bytes - LibidOriginal</td>
<td>*\G{(0D452EE1-E08F-101A-852E-02608C4D0BB4)#2.0#0#C:\Windows\system32\FM20.DLL#Microsoft Forms 2.0 Object Library</td>
</tr>
<tr>
<td>000003 97</td>
<td>000 2</td>
<td>unsigned integer - Id</td>
<td>0x002F</td>
</tr>
<tr>
<td>000003 99</td>
<td>000 4</td>
<td>unsigned integer - SizeTwiddled</td>
<td>0x0000003B</td>
</tr>
<tr>
<td>000003 9D</td>
<td>000 4</td>
<td>unsigned integer - SizeOfLibidTwiddled</td>
<td>0x00000031</td>
</tr>
<tr>
<td>000003 A1</td>
<td>000 4</td>
<td>array of bytes - LibidTwiddled</td>
<td>*\G{(00000000-0000-0000-0000-000000000000)#0.0#0##</td>
</tr>
<tr>
<td>000003 D2</td>
<td>000 4</td>
<td>unsigned integer - Reserved1</td>
<td>0x00000000</td>
</tr>
<tr>
<td>000003 D6</td>
<td>000 2</td>
<td>unsigned integer - Reserved2</td>
<td>0x0000</td>
</tr>
<tr>
<td>000003 D8</td>
<td>000 1</td>
<td>REFERENCENAME Record - NameRecordExtended</td>
<td></td>
</tr>
<tr>
<td>000003 D8</td>
<td>000 2</td>
<td>unsigned integer - Id</td>
<td>0x0016</td>
</tr>
<tr>
<td>000003 DA</td>
<td>000 4</td>
<td>unsigned integer - SizeOfName</td>
<td>0x00000007</td>
</tr>
<tr>
<td>000003 DE</td>
<td>000 7</td>
<td>array of bytes - Name</td>
<td>MSForms</td>
</tr>
<tr>
<td>000003 E5</td>
<td>000 2</td>
<td>unsigned integer - Reserved</td>
<td>0x003E</td>
</tr>
<tr>
<td>000003 E7</td>
<td>000 4</td>
<td>unsigned integer - SizeOfNameUnicode</td>
<td>0x0000000E</td>
</tr>
<tr>
<td>000003 EB</td>
<td>000 4</td>
<td>array of bytes - NameUnicode</td>
<td>MSForms</td>
</tr>
<tr>
<td>000003 F9</td>
<td>000 2</td>
<td>unsigned integer - Reserved3</td>
<td>0x0030</td>
</tr>
<tr>
<td>000003 FB</td>
<td>000 4</td>
<td>unsigned integer - SizeExtended</td>
<td>0x000000A3</td>
</tr>
</tbody>
</table>
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 REFERENCINGPROJECT (section 2.3.4.2.2.6) type. Reference[3] illustrates an example of a REFERENCEREGISTERED (section 2.3.4.2.2.5) 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 REFERENCINGPROJECT (section 2.3.4.2.2.6) record. Reference[3] is a REFERENCEREGISTERED (section 2.3.4.2.2.5) 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 REFERENCINGPROJECT (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: "\Example-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 2.3.4.2.1.11).

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 2.3.4.2.1.11).

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 LibidReferenceKind, "\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 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.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Size</th>
<th>Structure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>000004A2</td>
<td>01EA</td>
<td>PROJECTMODULES Record - ModulesRecord</td>
<td></td>
</tr>
</tbody>
</table>
Figure 5: Project modules stream

**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.
<table>
<thead>
<tr>
<th>Offset</th>
<th>Size</th>
<th>Structure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000512</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x001C</td>
</tr>
<tr>
<td>00000514</td>
<td>0004</td>
<td>unsigned integer - SizeOfDocString</td>
<td>0x00000000</td>
</tr>
<tr>
<td>00000518</td>
<td>0000</td>
<td>array of bytes - DocString</td>
<td></td>
</tr>
<tr>
<td>00000518</td>
<td>0002</td>
<td>unsigned integer - Reserved</td>
<td>0x0048</td>
</tr>
<tr>
<td>0000051A</td>
<td>0004</td>
<td>unsigned integer - SizeOfDocStringUnicode</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0000051E</td>
<td>0000</td>
<td>array of bytes - DocStringUnicode</td>
<td></td>
</tr>
<tr>
<td>0000051E</td>
<td>000A</td>
<td>MODULEOFFSET Record - OffsetRecord</td>
<td></td>
</tr>
<tr>
<td>0000051E</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x0031</td>
</tr>
<tr>
<td>00000520</td>
<td>0004</td>
<td>unsigned integer - Size</td>
<td>0x0000000004</td>
</tr>
<tr>
<td>00000524</td>
<td>0004</td>
<td>unsigned integer - TextOffset</td>
<td>0x00000000</td>
</tr>
<tr>
<td>00000528</td>
<td>000A</td>
<td>MODULEHELPCONTEXT Record - HelpContextRecord</td>
<td></td>
</tr>
<tr>
<td>00000528</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x001E</td>
</tr>
<tr>
<td>0000052A</td>
<td>0004</td>
<td>unsigned integer - Size</td>
<td>0x0000000004</td>
</tr>
<tr>
<td>0000052E</td>
<td>0004</td>
<td>unsigned integer - HelpContext</td>
<td>0x00000000</td>
</tr>
<tr>
<td>00000532</td>
<td>0008</td>
<td>MODULECOOKIE Record - CookieRecord</td>
<td></td>
</tr>
<tr>
<td>00000532</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x002C</td>
</tr>
<tr>
<td>00000534</td>
<td>0004</td>
<td>unsigned integer - Size</td>
<td>0x0000000002</td>
</tr>
<tr>
<td>00000538</td>
<td>0002</td>
<td>unsigned integer - Cookie</td>
<td>0xFFFF</td>
</tr>
<tr>
<td>0000053A</td>
<td>0006</td>
<td>MODULETYPE Record - TypeRecord</td>
<td></td>
</tr>
<tr>
<td>0000053A</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x0022</td>
</tr>
<tr>
<td>0000053C</td>
<td>0004</td>
<td>unsigned integer - Reserved</td>
<td>0x00000000</td>
</tr>
<tr>
<td>00000540</td>
<td>0002</td>
<td>unsigned integer - Terminator</td>
<td>0x002B</td>
</tr>
<tr>
<td>00000542</td>
<td>0004</td>
<td>unsigned integer - Reserved</td>
<td>0x00000000</td>
</tr>
</tbody>
</table>

**Figure 6: "ThisWorkbook" module record**

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 2.3.4.2.1.5).

**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 ModuleRecord (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.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Size</th>
<th>Structure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000546</td>
<td>00BE</td>
<td>MODULE Record - ModuleRecord</td>
<td></td>
</tr>
<tr>
<td>00000546</td>
<td>000C</td>
<td>MODULENAME Record - NameRecord</td>
<td></td>
</tr>
<tr>
<td>00000546</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x0019</td>
</tr>
<tr>
<td>00000546</td>
<td>0004</td>
<td>unsigned integer - SizeOfModuleName</td>
<td>0x00000006</td>
</tr>
<tr>
<td>00000546</td>
<td>0006</td>
<td>array of bytes - ModuleName</td>
<td></td>
</tr>
<tr>
<td>00000552</td>
<td>0012</td>
<td>MODULENAMEUNICODE Record - NameUnicodeRecord</td>
<td></td>
</tr>
<tr>
<td>00000552</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x0047</td>
</tr>
<tr>
<td>00000554</td>
<td>0004</td>
<td>unsigned integer - SizeOfModuleNameUnicode</td>
<td>0x0000000C</td>
</tr>
<tr>
<td>00000558</td>
<td>000C</td>
<td>array of bytes - ModuleNameUnicode</td>
<td></td>
</tr>
<tr>
<td>00000564</td>
<td>001E</td>
<td>MODULESTREAMNAME Record - StreamNameRecord</td>
<td></td>
</tr>
<tr>
<td>00000564</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x001A</td>
</tr>
<tr>
<td>00000566</td>
<td>0004</td>
<td>unsigned integer - SizeOfStreamName</td>
<td>0x00000006</td>
</tr>
<tr>
<td>0000056A</td>
<td>0006</td>
<td>array of bytes - StreamName</td>
<td></td>
</tr>
<tr>
<td>00000570</td>
<td>0002</td>
<td>unsigned integer - Reserved</td>
<td>0x0032</td>
</tr>
<tr>
<td>00000572</td>
<td>0004</td>
<td>unsigned integer - SizeOfStreamNameUnicode</td>
<td>0x0000000C</td>
</tr>
<tr>
<td>00000576</td>
<td>000C</td>
<td>array of bytes - StreamNameUnicode</td>
<td></td>
</tr>
<tr>
<td>00000582</td>
<td>005A</td>
<td>MODULEDOCSTRING Record - DocStringRecord</td>
<td></td>
</tr>
<tr>
<td>00000582</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x001C</td>
</tr>
</tbody>
</table>
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 Sheet3.1.2.3.2.2 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 2.3.4.2.1.8).

Figure 7: Sheet1 module record

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 Sheet3.1.2.3.2.2 Decompressed Module Stream Example (section 3.1.4).
3.1.2.3.2.3 UserForm1 Designer Module Record Example

The following example illustrates a designer module record.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Size</th>
<th>Structure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000604</td>
<td>0088</td>
<td>MODULE Record - ModuleRecord</td>
<td></td>
</tr>
<tr>
<td>00000604</td>
<td>000F</td>
<td>MODULENAME Record - NameRecord</td>
<td></td>
</tr>
<tr>
<td>00000604</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x0019</td>
</tr>
<tr>
<td>00000606</td>
<td>0004</td>
<td>unsigned integer - SizeOfModuleName</td>
<td>0x00000009</td>
</tr>
<tr>
<td>0000060A</td>
<td>0009</td>
<td>array of bytes - ModuleName</td>
<td>UserForm1</td>
</tr>
<tr>
<td>00000613</td>
<td>0018</td>
<td>MODULENAMEUNICODE Record - NameUnicodeRecord</td>
<td></td>
</tr>
<tr>
<td>00000613</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x0047</td>
</tr>
<tr>
<td>00000615</td>
<td>0004</td>
<td>unsigned integer - SizeOfModuleNameUnicode</td>
<td>0x00000012</td>
</tr>
<tr>
<td>00000619</td>
<td>0012</td>
<td>array of bytes - ModuleNameUnicode</td>
<td>UserForm1</td>
</tr>
<tr>
<td>0000062B</td>
<td>0027</td>
<td>MODULESTREAMNAME Record - StreamNameRecord</td>
<td></td>
</tr>
<tr>
<td>0000062B</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x001A</td>
</tr>
<tr>
<td>0000062D</td>
<td>0004</td>
<td>unsigned integer - SizeOfStreamName</td>
<td>0x00000009</td>
</tr>
<tr>
<td>00000631</td>
<td>0009</td>
<td>array of bytes - StreamName</td>
<td>UserForm1</td>
</tr>
<tr>
<td>0000063A</td>
<td>0002</td>
<td>unsigned integer - Reserved</td>
<td>0x0032</td>
</tr>
<tr>
<td>0000063C</td>
<td>0004</td>
<td>unsigned integer - SizeOfStreamNameUnicode</td>
<td>0x00000012</td>
</tr>
<tr>
<td>00000640</td>
<td>0012</td>
<td>array of bytes - StreamNameUnicode</td>
<td>UserForm1</td>
</tr>
<tr>
<td>00000652</td>
<td>000C</td>
<td>MODULEDOCSTRING Record - DocStringRecord</td>
<td></td>
</tr>
<tr>
<td>00000652</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x001C</td>
</tr>
<tr>
<td>00000654</td>
<td>0004</td>
<td>unsigned integer - SizeOfDocString</td>
<td>0x00000000</td>
</tr>
<tr>
<td>00000658</td>
<td>0000</td>
<td>array of bytes - DocString</td>
<td></td>
</tr>
<tr>
<td>00000658</td>
<td>0002</td>
<td>unsigned integer - Reserved</td>
<td>0x0048</td>
</tr>
<tr>
<td>0000065A</td>
<td>0004</td>
<td>unsigned integer - SizeOfDocStringUnicode</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0000065E</td>
<td>0000</td>
<td>array of bytes - DocStringUnicode</td>
<td></td>
</tr>
<tr>
<td>0000065E</td>
<td>000A</td>
<td>MODULEOFFSET Record - OffsetRecord</td>
<td></td>
</tr>
<tr>
<td>0000065E</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x0031</td>
</tr>
<tr>
<td>00000660</td>
<td>0004</td>
<td>unsigned integer - Size</td>
<td>0x00000004</td>
</tr>
<tr>
<td>00000664</td>
<td>0004</td>
<td>unsigned integer - TextOffset</td>
<td>0x00000000</td>
</tr>
<tr>
<td>00000668</td>
<td>000A</td>
<td>MODULEHELPCONTEXT Record - HelpContextRecord</td>
<td></td>
</tr>
<tr>
<td>00000668</td>
<td>0002</td>
<td>unsigned integer - Id</td>
<td>0x001E</td>
</tr>
</tbody>
</table>
### Figure 8: UserForm1 module record

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 UserForm3.1.2.3.2.3 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 might 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.

### Figure 9: Decompressed module data

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.

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

---

[MS-OVBA] - v20210817
Office VBA File Format Structure
Copyright © 2021 Microsoft Corporation
Release: August 17, 2021
### 3.1.4 Sheet1 Decompressed Module Stream Example

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

<table>
<thead>
<tr>
<th>Offset</th>
<th>Size</th>
<th>Structure</th>
<th>Value</th>
</tr>
</thead>
</table>
| 00000000 | 01D5 | array of bytes - TextDecompressedData | 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
Sub helloworld()
    MsgBox "Hello, World"
End Sub

**Figure 10: Decompressed module data**

**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.

```vba
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 CompilationExample()
    #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.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Size</th>
<th>Structure</th>
<th>Value</th>
</tr>
</thead>
</table>
| 00000000 | 0156 | array of bytes - TextDecompressedData | Attribute VB_Name = "UserForm1"
Attribute VB_Base = "0{842E9C5E-88B5-439A-912E-4C2D9AA0EC27}{2DC3C962-DA1C-47BA-AB63-...|

[MS-OVBA] - v20210817
Office VBA File Format Structure
Copyright © 2021 Microsoft Corporation
Release: August 17, 2021
## Text Decompressed Data

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

```plaintext
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.

```plaintext
Offset  Size  Structure  Value
000000  27F   array of byte text
```

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:

```plaintext
ID="917DED54-440B-4FD1-A5C1-74ACF261E600"
Document=ThisWorkbook/8H00000000
Document=Sheet1/8H00000000
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="0705D8E38EDDBF1DBF1DBF1DBF1"
DPB="0E0CD1EDCFF4E7F5E7F5E7"
GC="1517CAF1D6F9D7F9D706"
```

The preceding table illustrates an array of bytes that contains the example VBAPROJECTText (section 2.3.1).
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 2.3.4.2.1.8) 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 2.3.4.2.1.6).

ProjectHelpId (section 2.3.1.12): "HelpContextID="1"" is equivalent to the value specified in PROJECTHELPCONTEXT (section 2.3.4.2.1.9).

ProjectDescription (section 2.3.1.13): "Description="Example VBA Project Description"" is equivalent to the DocStringRecord (section 2.3.4.2.1.7) field in the PROJECTINFORMATION record (section 2.3.4.2.1).

ProjectVersionCompat (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="0E0CD1ECDF4E7F5E7F5E7"" 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** cannot 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

**WindowTitle**: "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.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Size</th>
<th>Structure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>0123</td>
<td>array of bytes - text</td>
<td>VERSION 5.00</td>
</tr>
</tbody>
</table>

```
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 library** 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 2.3.5.2): "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.

```
abcdefgijklmnopqrstuvwxyz.
```

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 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.

```
#aaabcdefaaaaghijaaaaaklaaamnopqaaaaaaaaaarstuvwxyzaaa
```

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 69 6A 6B 6C 6D 6E 6F 70 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
```
The following hex array represents the decompressed byte array of the example string as decompressed by the decompression algorithm:

```
7A 00 3C
```

**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.

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

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 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

The Visual Basic for Applications (VBA) cryptographic digest for a VBA project – see Contents Hash (section 2.4.2) – 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 updates to those products.

- Microsoft Office 97
- Microsoft Office 2000
- Microsoft Office XP
- Microsoft Office 2003
- the 2007 Microsoft Office system
- Microsoft Office 2010 suites
- Microsoft Office 2013
- Microsoft Office 2016
- Microsoft Office 2019
- Microsoft Office 2021

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates 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.


<2> Section 2.1.1.9: Microsoft Visual Basic for Applications 5.0 (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 2.3.4.2.1.12: Microsoft Visual Basic for Applications (VBA) will write user-specified values between −32768 and 32767. However, VBA will only read values between −9999 and 32767.

<8> Section 2.3.4.2.2.3: VBA 5.0 uses LibidTwiddled to specify a twiddled type library
6 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

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

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

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

The revision class None means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

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

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Revision class</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 Appendix A: Product Behavior</td>
<td>Updated list of supported products.</td>
<td>major</td>
</tr>
</tbody>
</table>
# Index

---

| VBA_PROJECT example | 84 |
| VBA_PROJECT stream | 20 |

## A

| ABNF rules | 15 |
| Algorithms |

| compression | 54 |
| contents hash | 71 |
| data encryption | 76 |
| decompression | 54 |
| encryption | 76 |
| password hash | 80 |

## B

| Byte ordering - overview | 13 |

## C

| Change tracking | 109 |
| Compression algorithms | 54 |
| Contents hash algorithm | 71 |
| Conventions | 15 |

## D

| Data encryption algorithm | 76 |
| Decompressed module stream example – Sheet1 | 100 |
| Decompressed module stream example - ThisWorkbook | 99 |
| Decompressed module stream example – UserForm1 | 100 |
| Decompression algorithms | 54 |
| Designer storage | 20 |

## E

| Encryption algorithm | 76 |
| Examples | 84 |

| VBA_PROJECT | 84 |
| decompressed module stream – Sheet1 | 100 |
| decompressed module stream - ThisWorkbook | 99 |
| decompressed module stream – UserForm1 | 100 |
| dir stream | 84 |
| maximum compression | 106 |
| no compression | 105 |
| normal compression | 105 |
| PROJECT stream | 101 |
| Sheet1 decompressed module stream | 100 |
| ThisWorkbook decompressed module stream | 99 |
| UserForm1 decompressed module stream | 100 |
| VBFrame stream | 104 |

## F

| Fields - vendor-extensible | 14 |
| File structure | 19 |

## G

| Glossary | 8 |

## I

| Informative references | 11 |
| Introduction | 8 |

## L

| Localization | 14 |

## M

| Maximum compression example | 106 |
| Module stream | 20 |

## N

| No compression example | 105 |
| Normal compression example | 105 |
| Normative references | 11 |

## O

| Overview (synopsis) | 12 |

## P

| VBA storage | 19 |
| VBA storage project information | 29 |
| VBFrame stream | 21 |
| VBFrame stream designer information | 51 |
| dir stream | 20 |
| dir stream example | 84 |

---

[MS-OVBA] - v20210817
Office VBA File Format Structure
Copyright © 2021 Microsoft Corporation
Release: August 17, 2021