[MS-CIFO]:

Content Index Format Structure

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Support. For questions and support, please contact dochelp@microsoft.com.

Revision Summary

Date	Revision History	Revision Class	Comments
4/4/2008	0.1	New	Initial Availability
6/27/2008	1.0	Major	Revised and edited the technical content
12/12/2008	1.01	Editorial	Revised and edited the technical content
7/13/2009	1.02	Major	Revised and edited the technical content
8/28/2009	1.03	Editorial	Revised and edited the technical content
11/6/2009	1.04	Editorial	Revised and edited the technical content
2/19/2010	2.0	Editorial	Revised and edited the technical content
3/31/2010	2.01	Editorial	Revised and edited the technical content
4/30/2010	2.02	Editorial	Revised and edited the technical content
6/7/2010	2.03	Editorial	Revised and edited the technical content
6/29/2010	2.04	Editorial	Changed language and formatting in the technical content.
7/23/2010	2.05	Minor	Clarified the meaning of the technical content.
9/27/2010	2.05	None	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.05	None	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.05	None	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	2.05	None	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	2.6	Minor	Clarified the meaning of the technical content.
1/20/2012	2.7	Minor	Clarified the meaning of the technical content.
4/11/2012	2.7	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	2.7	None	No changes to the meaning, language, or formatting of the technical content.
9/12/2012	2.7	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	2.7	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2013	2.7	None	No changes to the meaning, language, or formatting of the technical content.
7/30/2013	2.7	None	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	2.7	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
2/10/2014	2.7	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	2.7	None	No changes to the meaning, language, or formatting of the technical content.
7/31/2014	2.7	None	No changes to the meaning, language, or formatting of the technical content.
10/30/2014	2.7	None	No changes to the meaning, language, or formatting of the technical content.
6/23/2016	2.7	None	No changes to the meaning, language, or formatting of the technical content.
9/14/2016	2.7	None	No changes to the meaning, language, or formatting of the technical content.
10/17/2016	2.7	None	No changes to the meaning, language, or formatting of the technical content.
6/20/2017	3.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction 7
1.1 Glossary 7
1.2 References 10
1.2.1 Normative References 10
1.2.2 Informative References 10
1.3 Structure Overview (Synopsis) 10
1.4 Relationship to Protocols and Other Structures 10
1.5 Applicability Statement 11
1.6 Versioning and Localization 11
1.7 Vendor-Extensible Fields 11

2 Structures 12
2.1 Common Constants 12
2.1.1 Property Identifier 12
2.1.2 MaxOccBuckets Table 12
2.2 Common Structures 17
2.2.1 BitStream File Format 17
2.2.1.1 BitStream Page Structure 17
2.2.1.2 BitStream DWORD 18
2.2.1.3 BitStreamPosition 18
2.2.2 BitStream Field Structures 19
2.2.2.1 BitCompress(K) 19
2.2.2.2 PidCompress 22
2.2.2.3 DocIDCountCompress 22
2.2.2.4 PrefixSuffixCompress 23
2.2.3 Index Keys 24
2.2.3.1 String Normalization 24
2.2.3.2 Content 25
2.2.3.3 BOF 25
2.2.3.4 EOF 25
2.2.3.5 Max 25
2.2.3.6 Basic Scope 25
2.2.3.7 Compound Scope 27
2.2.3.8 Anchor Scope 28
2.2.4 Recoverable Storage File Format 28
2.2.4.1 Header File Format 28
2.2.4.2 Data File Format 30
2.2.5 CheckSummed Recoverable Storage File Format 31
2.2.5.1 CheckSummedRecord Structure 31
2.2.6 Sparse Array File Format 31
2.2.6.1 SparseArrayBlock Structure 33
2.2.6.2 SparseArrayBlockData Structure 33
2.3 Content Index File Format 34
2.3.1 ContentIndexRecord 36
2.4 Scope Index File Format 45
2.4.1 ScopeIndexRecord 45
2.5 Index Directory File Format 47
2.5.1 File Layout 47
2.5.2 First Page Structure 48
2.5.3 Page Structure 49
2.5.4 Page Header Structure 50
2.5.5 File Header Structure 50
2.5.6 Record Buffer Structure 51
2.5.7 Record Structure 51
2.6 Content Index Extension File Format 54

2.6.1 KeyExtensionData Structure 54
2.6.1.1 ExtensionCompressionTablePage Structure 55
2.6.1.1.1 SymbolCategory Structure 56
2.6.1.1.2 CodingTableEntry Structure 57
2.6.1.2 ExtensionDataPage Structure 57
2.6.1.2.1 DirectoryEntry Structure 58
2.6.1.2.2 EncodedDOCIDDelta Structure 59
2.7 Document Set Files 59
2.7.1 List Document Set 60
2.7.2 Bitmap Document Set 62
2.7.3 Indexed Bitmap Document Set 64
2.8 Average Document Length File Format 66
2.8.1 CAVDLItem Structure 66
2.9 Merge Process 67
2.10 Merge Log File Format 67
2.10.1 User Header Format 68
2.10.2 File Content 69
2.10.3 CMergeSplitKey Structure 70
2.11 Query-Independent Rank Files 72
2.12 Detected Language Files 72
2.13 Index Table File Format 72
2.13.1 User Header 72
2.13.2 CIndexRecord 72
2.13.3 IndexType Enumeration 73
2.14 Click Distance File 75
2.15 Index Lexicon File 75
2.16 Diacritic Settings File 76
2.17 Full-Text Index Component 76
2.17.1 Naming Convention for the Full-Text Index Component Files 77
2.18 Full-Text Index Catalog 78
2.18.1 Main Catalog 80
2.18.2 Anchor Text Catalog 80
2.18.3 Active Anchor Text Catalog 81

3 Structure Examples 82
3.1 Full-text Index Catalog 82
3.1.1 Compound Scope Index Directory 83
3.1.2 Compound Scope Index 85
3.1.3 Basic Scope Index Directory 86
3.1.4 Basic Scope Index 88
3.1.5 Content Index File 90
3.1.6 Index Directory 92
3.1.6.1 Content Index Record 92
3.1.6.2 Content Index Record with Skips 95
3.1.7 Document Set Files 95
3.1.8 Average Document Length Files 101
3.1.9 Detected Language Files 104
3.1.10 Query-Independent Rank Files 106
3.1.11 Index Table File 109
3.1.12 Index Lexicon File 113
3.1.13 Diacritic Settings File 113
3.2 CIX File 113
3.2.1 Physical File on Disk 114
3.2.2 ExtensionCompressionTablePage 115
3.2.2.1 Page start, symbol category descriptors 115
3.2.2.2 Coding Table 115
3.2.2.3 End of Page 116
3.2.3 ExtensionDataPage 116

3.2.3.1 Page start, page directory 116
3.2.3.2 DOCID Bit Stream 117
3.2.3.3 OccCount Bit Stream 118
4 Security Considerations 120
5 Appendix A: Character Normalization Tables 121
6 Appendix B: Product Behavior 200
7 Change Tracking 203
8 Index 204

1 Introduction

This document specifies the Content Index Format Structure that contains the data needed to perform queries.

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:

anchor scope index key: An index key that contains an encoded document identifier. It is used in conjunction with a scope index record that stores links from the item that is referenced by the document identifier.

anchor text: The text that is included with a hyperlink to describe the target content of a hyperlink.

authority page: A webpage that a site collection administrator designated as more relevant than other webpages. This is typically the **URL** of the home page for the intranet of an organization. The higher the authority level assigned to a page, the higher the page appears in search results. Also referred to as authoritative page.

basic scope index: A scope index file that contains records with **basic scope index keys** or **anchor scope index keys**.

basic scope index key: An **index key** that references a scope index record and contains information about a property and its value.

beginning-of-file (BOF) key: An **index key** that is stored near the beginning of a content index file. It references a content index record that stores the maximum occurrence for a specified property.

BitStream: A sequence of bits that represents the compressed data for a full-text index catalog.

BitStream field: A section of bits that is part of a BitStream and is 32 or fewer bits.

BitStream field structure: A structure that contains one or more BitStream fields.

BitStream file: A content index file, a scope index file, or a content index extension (.cix) file that is used to store compressed data for a full-text index catalog. It stores the data as a series of BitStreams that are organized into BitStream pages.

BitStream page: A 4,096-byte segment of data in a BitStream file. It stores 32,704 bits, using an array of 4-byte blocks.

BitStreamPosition: A data structure that is used to specify the location of a BitStream field or field structure in a BitStream file.

CheckSummedRecord: A record that stores data fields and the corresponding checksum for each of those fields.

CIndexRecord: A record in an index table file.

compound scope index: A file that is in a search scope index and contains records that store **compound scope index keys** or **anchor scope index keys**.

compound scope index key: A key that is used to locate a scope index record. It is based on a compound scope identifier.

content index extension (.cix) file: A file that is part of a full-text index catalog. It is used to store compressed document identifiers and OccCount values for data that is stored in an associated content index file.

content index file: A file that is part of a full-text index catalog. It is used to store data from items as an inverted index and it enables searches for specific terms across items.

content index key: A key that references a record in a content index file. It consists of a property identifier and a normalized token.

content index record: A part of a content index file that is used to store all of the document identifiers for items that have a unique combination of a token and a property identifier.

DocID skip: A forward link that allows the reader of a **content index record** or a scope index record to skip a group of **document identifiers**.

DocIDDelta: A number that represents the incremental difference in value between a document identifier and the document identifier that immediately precedes it in a list that is sorted in ascending order.

document identifier: An integer that uniquely identifies a crawled item.

end-of-file (EOF) key: An **index key** that is stored near the end of a content index file. It references a content index record that stores the maximum occurrence for a specified property.

full-text index component: A set of files that contain all of the index keys that are extracted from a set of items.

index directory file: A file that is part of a full-text index catalog. It is used to store index keys from an associated content index file, which facilitates finding a specific content index record in the content index file.

index directory level: An array of **index directory pages** that contains **index keys** from an associated index and the positions of those keys in the index.

index directory page: A page that conforms to the index directory page structure that stores index directory records.

index identifier: An integer that uniquely identifies a full-text index component within a full-text index catalog.

index key: A key that references a record in a content index file or a scope index file. It consists of an index key string and a property identifier.

index key string: A sequence of bytes that specifies the value that is used to sort records in a content index file or a scope index file.

index server: A server that is assigned the task of crawling.

index table file: A directory that is used to store an inventory of files in a full-text index catalog.

inverted index: For each token that is encountered in a corpus of indexed items, a data structure that stores a list of postings that identify which documents matched and a list of occurrences that identify which position in each document.

item: A unit of content that can be indexed and searched by a search application.

log2: A function that returns an integer specifying the minimum number of bits that are required to represent the integer part of an input parameter.

master index component: A full-text index component that contains index keys that are extracted from a set of items. In a full-text index catalog, there is only one master index component. It is referenced by an itMaster CIndexRecord.

max key: An index key that references the last record in a content index file or a scope index file.

MaxOccBucket: An integer that is used to store the approximate number of tokens for a specific item and property.

metadata schema: A schema that is used to manage information about an item.

OccCount: An integer that is used to store the number of instances of a token for a specific item and property.

prefix length: An integer that represents the number of identical bytes at the beginning of the current and previous index key strings. See also **suffix length**.

property identifier: A unique integer or a 16-bit, numeric identifier that is used to identify a specific attribute or property.

query server: A server that has been assigned the task of fulfilling search queries.

rank: An integer that represents the relevance of a specific item for a search query. It can be a combination of static rank and dynamic rank. See also static rank and dynamic rank.

ranking: A process in which an integer that represents the relevance of a specific item for a search query is assigned to that item. It can be a combination of static rank and dynamic rank.

scope index key: A **basic scope index key** or a compound scope index key that references a scope index record.

search application: A unique group of search settings that is associated, one-to-one, with a shared service provider.

search query: A complete set of conditions that are used to generate search results, including query text, sort order, and ranking parameters.

search scope: A list of attributes that define a collection of items.

search scope compilation identifier: An integer that identifies the version of the list of search scopes that is associated with a scopes compilation event on a search server.

split key: A content index key that references a record in a target content index file. All of the records before the referenced record have been written to the file successfully.

suffix length: An integer that represents the number of bytes of the current index key string minus the number of identical bytes at the beginning of the current and previous index key strings. See also **prefix length**.

token: A word in an item or a search query that translates into a meaningful word or number in written text. A token is the smallest textual unit that can be matched in a search query. Examples include "cat", "AB14", or "42".

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).

Uniform Resource Locator (URL): A string of characters in a standardized format that identifies a document or resource on the World Wide Web. The format is as specified in [RFC1738].

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

1.2 References

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

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

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

[MS-QSSWS] Microsoft Corporation, "Search Query Shared Services Protocol".

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992, http://www.ietf.org/rfc/rfc1321.txt

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

1.2.2 Informative References

None.

1.3 Structure Overview (Synopsis)

This document specifies the data structures that make up the full-text index catalog.

The full-text index catalog, defined in section 2.18, is the top-level concept defined in this document. It consists of a set of files which contain the data necessary for resolving full-text queries. The full-text index catalog is constructed by the index server by processing the text extracted from multiple properties of the **items** that are crawled. The index server creates the full text index catalog as a part of crawling.

The full-text index catalog consists of one or more **full-text index components**, each of which stores the indexed content of a subset of the items and which are included in the full-text index catalog.

Each full-text index component, defined in section 2.17, is composed of several files that have specific formats. Besides the actual data, files in each full-text index component contain additional structures which allow the **search queries** to efficiently locate and retrieve the data required to satisfy these queries.

In addition to the full-text index components, the full-text index catalog contains files that store the inventory of the catalog and the statistics necessary for the **ranking** of items. The full-text index catalog is defined in section 2.18.

1.4 Relationship to Protocols and Other Structures

None.

1.5 Applicability Statement

These structures are only applicable to the inter-server communication between the **index server** and the **query server**.

1.6 Versioning and Localization

None.

1.7 Vendor-Extensible Fields

None.

2 Structures

2.1 Common Constants

2.1.1 Property Identifier

Property identifiers are unique numeric constants used to denote properties extracted from items that are stored in the full-text index catalog.

The properties listed in the following table are not directly extracted from the items, but instead are automatically generated as defined in the respective sections.

Value	Name	Meaning	Detailed information
95	**pidSiteScope**	All generated string values for folders in the **URL** of the item	Section 2.18.1
96	**pidClickDistance**	This property identifier is used in the representation of the click distance file	Section 2.14

Additionally, the property identifier values listed in the following table are used in the representation of the full-text index catalogs.

Value	Name	Meaning
0x7FFFFFFF	**pidMaximum**	This property identifier is used for composing a **max key** that is guaranteed to be bigger than any other valid **index key**.
0x7FFEFFFF	**pidEOFile**	This property identifier value is used for composing an **end-of-file (EOF) key** that is associated with the accumulated content of all the indexed properties of a document. The **content index record** that contains this key stores the sum of the lengths in tokens of all the indexed properties of each document included in the **content index file**.
0x7FFEFFF1	**pidCompoundScope**	This property identifier value is used for composing a Compound Scope Index Key (section 2.2.3.7)

2.1.2 MaxOccBuckets Table

The **MaxOccBuckets** table defines the relationship between **MaxOccBucket** values and the upper bound estimation of maximum occurrence for a given property in an item. The estimated value for

maximum occurrence MUST be greater than or equal to the actual value of maximum occurrence. If maximum occurrence is greater than 474,449, **MaxOccBucket** MUST be equal to 127.

MaxOccBucket	Max Occurrence
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	11
11	12
12	13
13	14
14	15
15	16
16	17
17	18
18	19
19	20
20	22
21	24
22	26
23	28
24	30
25	33
26	36
27	39
28	42
29	46

MaxOccBucket	Max Occurrence
30	50
31	55
32	60
33	66
34	72
35	79
36	86
37	94
38	103
39	113
40	124
41	136
42	149
43	163
44	179
45	196
46	215
47	236
48	259
49	284
50	312
51	343
52	377
53	414
54	455
55	500
56	550
57	605
58	665
59	731
60	804
61	884

MaxOccBucket	Max Occurrence
62	972
63	1069
64	1175
65	1292
66	1421
67	1563
68	1719
69	1890
70	2079
71	2286
72	2514
73	2765
74	3041
75	3345
76	3679
77	4046
78	4450
79	4895
80	5384
81	5922
82	6514
83	7165
84	7881
85	8669
86	9535
87	10488
88	11536
89	12689
90	13957
91	15352
92	16887
93	18575

MaxOccBucket	Max Occurrence
94	20432
95	22475
96	24722
97	27194
98	29913
99	32904
100	36194
101	39813
102	43794
103	48173
104	52990
105	58289
106	64117
107	70528
108	77580
109	85338
110	93871
111	103258
112	113583
113	124941
114	137435
115	151178
116	166295
117	182924
118	201216
119	221337
120	243470
121	267817
122	294598
123	324057
124	356462
125	392108

MaxOccBucket	Max Occurrence
126	431318
127	474449

2.2 Common Structures

2.2.1 BitStream File Format

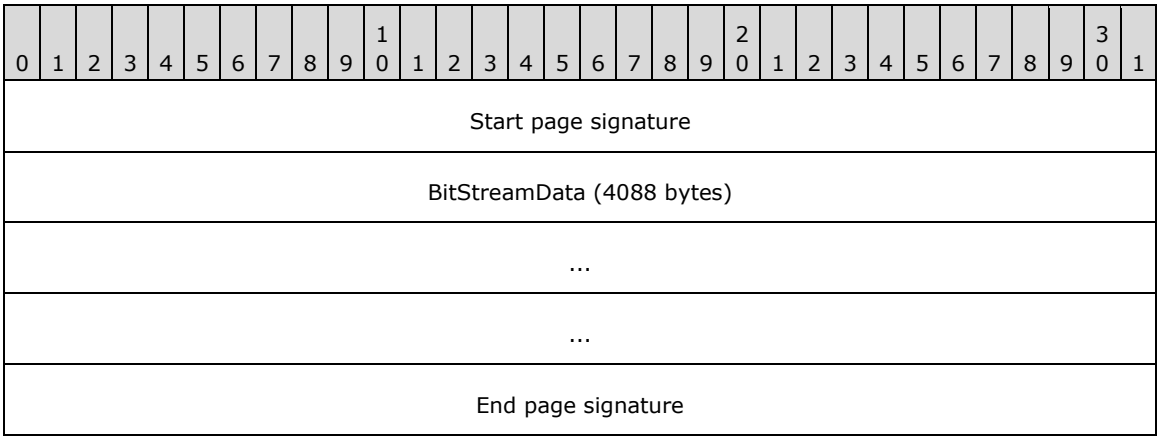
The **BitStream file** is a generic file format used for storing compressed data specific to the full-text index catalog. This data is a sequence of unsigned integer values represented by various-sized **BitStream fields** which are segments of a **BitStream**.

The top level structure of a BitStream file is an array of **BitStream pages** of 4,096 bytes each. Subsequently, the size of the BitStream files MUST be a multiple of 4,096 bytes. The structure of a page is defined in section 2.2.1.1.

Each BitStream page stores a segment of 32,704 BitStream bits, using an array of 4-byte blocks. The order in which bits of the BitStream are mapped to each 4-byte block is defined in section 2.2.1.2.

2.2.1.1 BitStream Page Structure

Each **BitStream file** is composed of one or more 4,096-byte **BitStream pages**. The structure of each BitStream page is defined as shown in the following table.



Start page signature (4 bytes): DWORD (see [MS-DTYP]) used to ensure the validity of the page. The value MUST be nonzero and it MUST be equal to the value of the **End page signature** field.

BitStreamData (4088 bytes): Array of DWORD elements. The array stores consecutive 32-bit segments of the **BitStream**. The mapping of 32-bit segments of the BitStream to the DWORD bits is defined in section 2.2.1.2. The **BitStreamData** fields for consecutive BitStream pages contain consecutive segments of the BitStream.

End page signature (4 bytes): DWORD used for ensuring the validity of the page. The value MUST be nonzero and it MUST be equal to the value of the **Start page signature** field.

2.2.1.2 BitStream DWORD

The data in a **BitStream** is stored in segments of 32 bits each which are mapped to the DWORD (see [MS-DTYP]) in the **BitStreamData** field of the **BitStream page**. The first BitStream bit is mapped to the most significant DWORD bit. The full mapping is represented in the following table. The first row represents the BitStream position and the second row the DWORD bit range.

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

The same mapping is represented in the following table, using the network transfer order.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Bits 24 to 31								Bits 16 to 23								Bits 8 to 15								Bits 0 to 7							

Bits 24 to 31: Bits 24 to 31 of the BitStream segment are stored in the first byte of the DWORD. The high bit of this byte is mapped to the position 24.

Bits16 to 23: Bits 16 to 23 of the BitStream segment are stored in the second byte of the DWORD. The high bit of this byte is mapped to the position 16.

Bits 8 to 15: Bits 8 to 15 of the BitStream segment are stored in the third byte of the DWORD. The high bit of this byte is mapped to the position 8.

Bits 0 to 7: Bits 0 to 7 of the BitStream segment are stored in the fourth byte of the DWORD. The high bit of this byte is mapped to the position 0.

Example

The following BitStream segment starts at positions that are multiples of 32.

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

If this segment is read as a DWORD, it is equal to 0x05900018. The segment is stored in the file as the following 4 byte sequence: 0x18, 0x00, 0x90, 0x05.

2.2.1.3 BitStreamPosition

BitStreamPosition is a conceptual structure which is used in multiple components of the full-text index catalog to specify a location of a **BitStream field** or a structure in a **BitStream file**. The structure contains two unsigned integer values:

- **Page**: a 0-based index of the **BitStream page** which contains the first bit of the BitStream field. **Page** indexes are commonly stored as 4-byte integers.
- **Offset**: the bit position in the **BitStream** relative to the beginning of the page. The valid range for the **Offset** field value is 0 to 32,703.

2.2.2 BitStream Field Structures

This section defines a set of structures used for storing data in **BitStream files**.

The data in any BitStream file is stored as a sequence of **BitStream fields**. Each BitStream field MUST NOT exceed 32 bits in size. Successive BitStream fields occupy consecutive segments of the **BitStream**. The following table is a sample representation of a segment of the BitStream that includes several BitStream fields organized in a **BitStream field structure**.

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
Field1							Field2						Field3																		

Field1 (7 bits): The first field in the BitStream field structure is an unsigned integer which occupies the first 7 bits of a segment of the BitStream.

Field2 (6 bits): The second field in the BitStream field structure is an unsigned integer which occupies the next 6 bits of a segment of the BitStream.

Field3 (17 bits): The third field of the BitStream field structure is an unsigned integer which occupies the following 17 bits of the BitStream segment.

The bits of a BitStream field are mapped to the BitStream segments in big-endian order.

Example

The following table is an instantiation of the BitStream field structure defined in the preceding table with **Field1** set to 5, **Field2** set to 2 and **Field3** set to 6.

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
0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0		

2.2.2.1 BitCompress(K)

BitCompress(K) is a method of encoding 32-bit unsigned integer values to a **BitStream field structure**. The encoding attempts to save space by representing only the significant bits.

The format of the **BitCompress(K)** is represented in the following table.

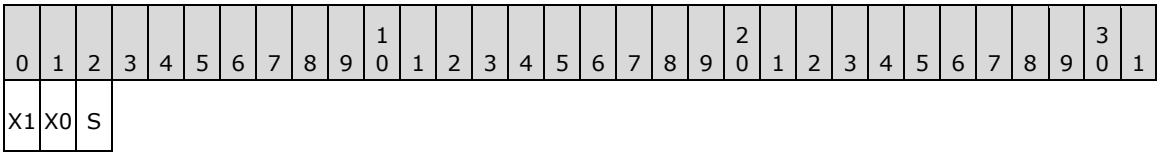
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FirstKBits (variable)																															
...																															
E	ExtraBits (variable)																														
...																															

FirstKBits (K bits): The size of first bit field in the structure is a parameter in the encoding method. The first field of the **BitCompress(K)** structure stores most significant bits of the integer value. Notation for the **BitCompress(K)** structure includes the value of K in parenthesis.

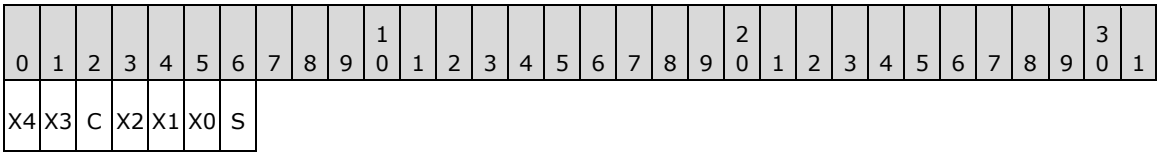
E (1 bit): If the **E** field is set to zero, the **ExtraBits** field MUST NOT be present, and the integer value MUST equal the **FirstKBits** field. If **E** is set to 1, the **ExtraBits** field MUST be present.

ExtraBits (variable): A BitStream field structure within the **BitCompress(K)** structure that stores the least significant bits of the integer value. The size, in bits, of this structure MUST be 3, 7, 12, 18, 25, 33, or 42. Depending on the size, one of the following structures is used:

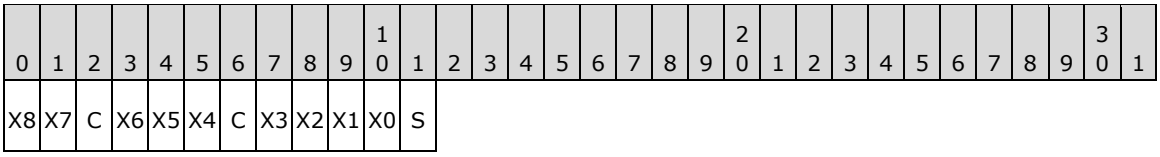
ExtraBits(2): 3-bit structure



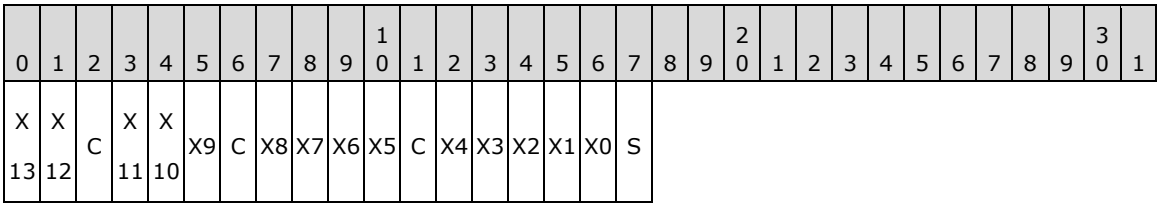
ExtraBits(5): 7-bit structure



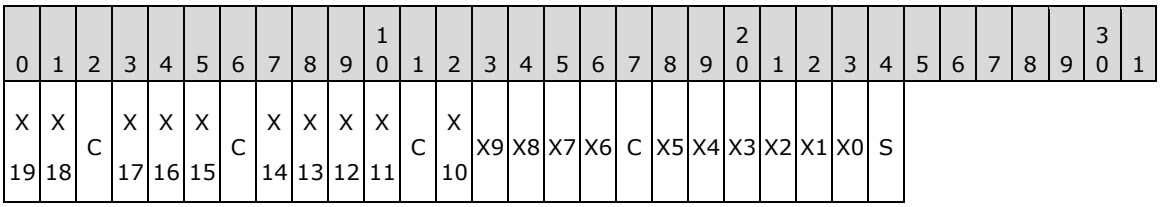
ExtraBits(9): 12-bit structure



ExtraBits(14): 18-bit structure



ExtraBits(20): 25-bit structure



ExtraBits(27): 33-bit structure

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
X26	X25	C	X24	X23	X22	C	X21	X20	X19	X18	C	X17	X16	X15	X14	X13	C	X12	X11	X10	X9	X8	X7	C	X6	X5	X4	X3	X2	X1	X0
S																															

ExtraBits(35): 42-bit structure

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
P	P	C	P	X31	X30	C	X29	X28	X27	X26	C	X25	X24	X23	X22	X21	C	X20	X19	X18	X17	X16	X15	C	X14	X13	X12	X11	X10	X9	X8
C	X7	X6	X5	X4	X3	X2	X1	X0	S																						

In the preceding structures:

X0, X1, … , X31 (1 bit each): Represent the bits of the integer value. X0 is the least significant bit of the integer value.

C (1 bit): Continuation bit which MUST be set to 1. It indicates that the structure is to be continued with additional fields.

S (1 bit): Stop bit which MUST be set to 0. It indicates that the structure does not contain subsequent fields.

P (1 bit): Padding bits which MUST be set to 0. The padding bits are used when the cumulated size of the BitStream field structure is more than 32 bits. In this case, padding bits are added to the left-side field(s) (**FirstKBits** field or **ExtraBits** field) so that the least significant bit X0 is always the last bit before the stop bit.

Examples:

The value 5 represented as BitCompress(7):

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

The value 0xCCC represented as BitCompress(7):

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

The value 0xFFFFFFFE represented as BitCompress(2):

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

2.2.2.2 PidCompress

The **PidCompress** **BitStream field structure** is used for encoding specific 32-bit unsigned integer values in **BitStream files**.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
C	PidBitCompress (variable)																														
...																															

C (1 bit): If C bit is 0, the integer value is assumed to be equal to 1. If C bit is 1, then the integer value is equal to the value stored in the **PidBitCompress** field.

PidBitCompress (variable): Stores the integer value in **BitCompress(4)** format as described in section 2.2.2.1. The field MUST NOT be present if the bit **C** is not set.

2.2.2.3 DocIDCountCompress

The **DocIDCountCompress** **BitStream field structure** is used to store a 32-bit unsigned integer value in **BitStream files**. The encoded value, which is stored using the **DocIDCountCompress** structure, is the integer value plus 1.

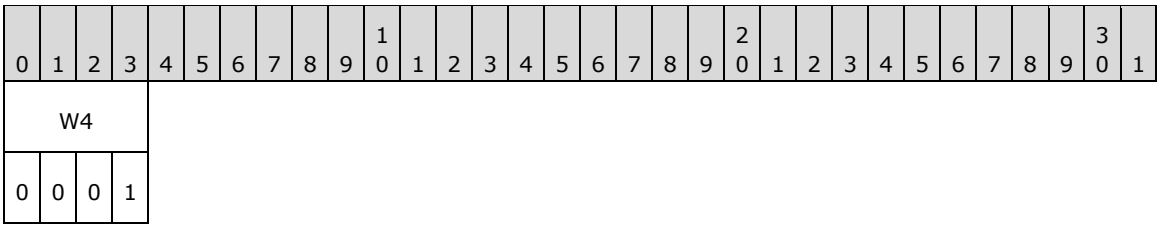
0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
W4				W8 (optional)								W32 (optional)																			
...																															

W4 (4 bits): Stores the encoded value if it is less than 16. W4 MUST be 0 if the encoded value is greater than 15. In this case the W8 field MUST be present.

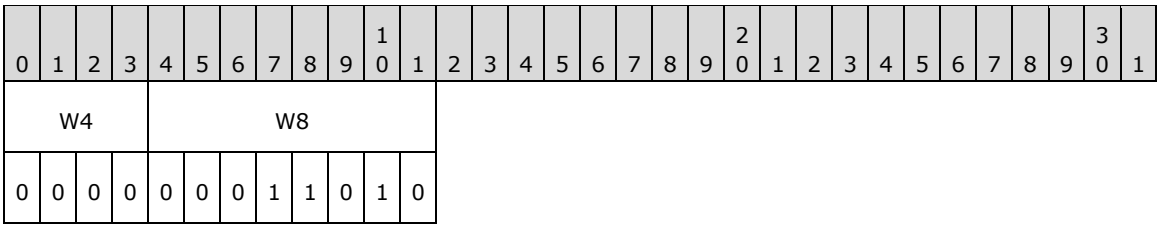
W8 (1 byte, optional): Stores the encoded value if it is between 16 and 255. The field MUST NOT be present if W4 is not equal to zero. W8 MUST be zero if the encoded value is greater than 255. In this case the W32 field MUST be present.

W32 (4 bytes, optional): Stores the encoded value if it is greater than 255. The field MUST NOT be present if either W4 or W8 is not 0.

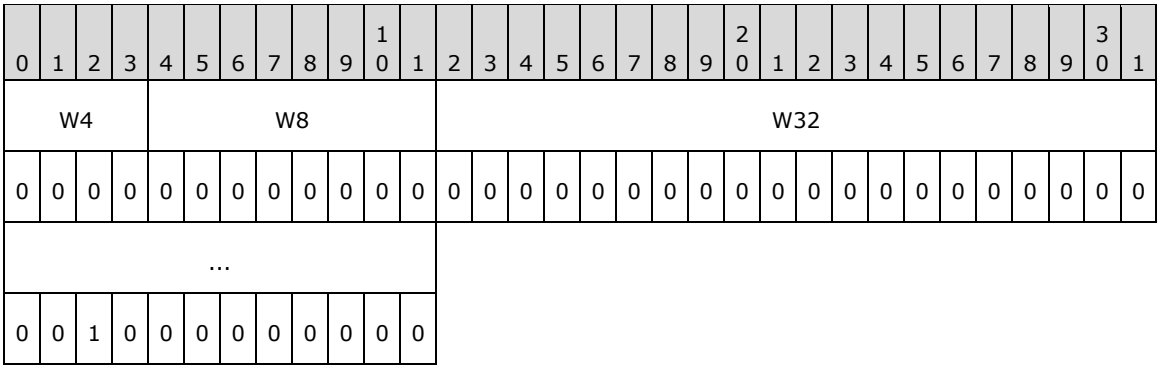
Examples



This **DocIDCountCompress** BitStream field structure encodes the integer 0. W4 is 1, W8 and W32 are not present, and therefore the integer value equals `W4 - 1 = 0`.



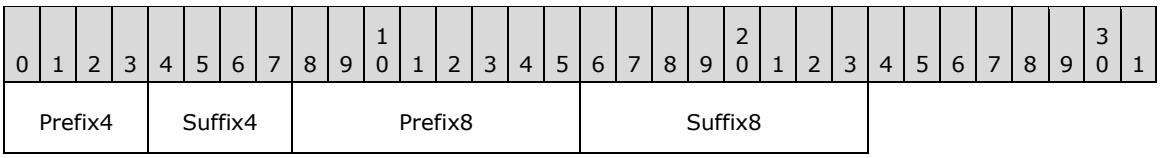
This **DocIDCountCompress** BitStream field structure encodes the integer 25. W4 is 0, W8 is present and equals 26, and W32 is not present, and therefore the integer value equals `W8 - 1 = 25`.



This **DocIDCountCompress** BitStream field structure encodes the integer 511. W4 is 0, W8 is 0, and W32 is present and equals 512, and therefore the integer equals `W32 - 1 = 511`.

2.2.2.4 PrefixSuffixCompress

The **PrefixSuffixCompress** **BitStream field structure** is used to store two integers with values in the range from zero through 129 that are used in a scope index record or a **content index record**: **prefix length**, **suffix length**.



Prefix4 (4 bits): Contains the prefix length. If both the **Prefix4** field and the **Suffix4** field are set to zero, **Prefix8** contains the length of the prefix.

Suffix4 (4 bits): Contains suffix length. If both **Prefix4** and **Suffix4** are set to zero, **Suffix8** contains the length of the suffix.

Prefix8 (1 byte): Contains the prefix length. The field MUST NOT be present if **Prefix4** or **Suffix4** is not set to zero.

Suffix8 (1 byte): Contains the suffix length. The field MUST NOT be present if **Prefix4** or **Suffix4** is not set to zero.

2.2.3 Index Keys

An index key references a **content index record** or a scope index record.

The index key consists of

- The **index key string**: A sequence of bytes with different meaning for each type of index key.
- The property identifier: An identifier of a property that is referenced by index key.

When ordering for index keys is required, index keys MUST be ordered using default sorting order, unless otherwise noted, as follows:

1. The index key string ascending.
2. The property identifier as a DWORD (see [MS-DTYP]) ascending.

2.2.3.1 String Normalization

The following algorithm defines a transformation of a string into a normalized **token**, for use in index keys.

The original string MUST be a string in **Unicode** format. See section 5 for the tables.

1. Each character from the original string is processed sequentially and is represented by a variable number of characters in the normalized token. For each character (WORD in little-endian notation, see [MS-DTYP]) from the original string that is present in Table 1 in the 'original' column, a sequence of WORDs from the 'Transformed' column in little-endian notation MUST be written to the normalized token. If 'Removed' is specified in the 'Transformed' column, a character MUST NOT be added to the normalized token. If a WORD from the original string is not in Table 1, the same WORD in big-endian notation MUST be written to the normalized token.
2. If the original string contains at least one character from Table 2 and the **content index key** and **DiacriticNormalizationMethod** in the Diacritic Settings file, as specified in section 2.16, defined for the current full-text index catalog is set to 3, a character 0x0000 MUST be written at the end of the normalized token; otherwise, go to step 5.
3. An integer K is defined which MUST be equal to the position of the last character from Table 2 in the original string.
4. Each character from the original string is processed a second time sequentially and represented by a variable number of characters added to the end of the normalized token from step 2. For each character in the original string that is present in Table 2 column 'original' WORD (see [MS-DTYP]) in little-endian notation, a BYTE (see [MS-DTYP]) from the 'Transformed' column MUST be added to the normalized token. If a WORD from the original string with position <=K is not in Table 2, the BYTE 0x02 MUST be added to the normalized token.
5. If the total length of the normalized token is greater than the maximum length allowed (defined for each index key), the minimum number of characters MUST be removed from the end of the original string so that the length of the normalized token upon iteration is shorter than or equal to the maximum length allowed. Once the characters are removed, normalization MUST be retried by repeating the algorithm from step 1.

2.2.3.2 Content

One **content index key** is stored in every **content index record**. It is constructed from a normalized token and property identifier.

The index key string for content index key MUST have length equal to 1 plus the length of the normalized token in bytes. The first byte of the index key string MUST be 0 followed by the normalized token.

The maximum length in bytes of the normalized token MUST be 128.

The token is normalized using the method defined in section 2.2.3.1.

2.2.3.3 BOF

A **beginning-of-file (BOF) key** references a **content index record** that contains the maximum occurrence for all items in a **full-text index component** for a given property. It is constructed from a property identifier value.

The index key string for BOF key MUST have length of 1 byte and be equal to 0x00.

2.2.3.4 EOF

An EOF key references a **content index record** that contains the maximum occurrence for all items in a **full-text index component** for given property. It is constructed from a property identifier value.

The index key string for EOF key MUST have length of 2 bytes and be equal to 0x7e, 0xff.

2.2.3.5 Max

A max key is the last key in a **full-text index component**, ordered by index key string and then property identifier.

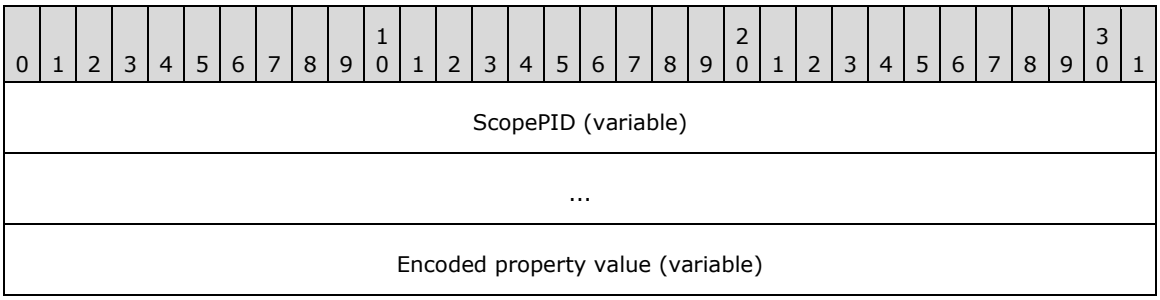
The index key string for the max key MUST have a length of 129 bytes with the first byte equal to "0x7f" and the remainder of the bytes equal to "0xff".

The property identifier for the max key MUST be ignored.

2.2.3.6 Basic Scope

A **basic scope index key** is an index key used to denote a **search scope** which contains all items which contain the same value for one property. It is stored in a scope index record. The property identifier for this index key MUST be 298.

The index key string encodes the value and the property identifier and has the following format.



…

ScopePID (variable): Stores the property identifier of the encoded property. The length in bytes MUST be 1 if the **ScopePID** field < 0x7D. If the **ScopePID** field >= 0x7D and the property type is a date/time (property type equals 4), the length in bytes MUST be 6, otherwise the length in bytes MUST be 5.

The first byte (byte 0) MUST be equal to the **ScopePID** field if the **ScopePID** field < 0x7D. If the **ScopePID** field >= 0x7D and the property type is a date/time, the first byte MUST be equal to 0x7D, otherwise the first byte MUST be equal to 0x7E.

If the first byte is 0x7D, the second byte (byte 1) MUST be 0x7E and the **ScopePID** field MUST be written to bytes 2 through 5 of the index key in big-endian order.

If the first byte is 0x7E, the **ScopePID** field MUST be written to bytes 1 through 4 of the index key in big-endian order.

Encoded property value (variable): Stores an encoded property value. Encoding type depends on the property type. The managed property types are converted to a string following the following rules:

- Signed 64-bit integer values (property type equals 2) MUST be treated as unsigned integer values and written as base 16 numbers to a string.
- Boolean values (property type equals 5) MUST be written as "ffffffff" if **true**; 0 if **false** to a string.
- String (property type equals 1) MUST NOT be changed. Note: The property **pidScopeSite** is a string.
- Coordinated Universal Time (UTC) date and time (property type equals 4) values MUST be represented using date components. There are four date components: year, month, day, hour. Each date/time property MUST have four basic scope index keys corresponding to each date component. Each date component has a component byte, which is a constant for that component, and a component string value which is derived from the original UTC date/time value. The first byte of the encoded property value for each date component basic scope index key MUST be the component constant. The component string value MUST be converted in base 10 to an unsigned integer and written in big-endian order starting from the second byte. For a year component, the first byte MUST be 0x59. The year component string value MUST be composed of the 4 digit year as a base 10 number written to a string. For a month component, the first byte MUST be 0x4D. The month component string value MUST be composed of the year component string value concatenated with the 2 digit month of the year as a base 10 number written to a string. For a day component, the first byte MUST be 0x44. The day component string value MUST be composed of the month component string value concatenated with the 2 digit day of the month as a base 10 number written to a string. For an hour component, the first byte MUST be 0x48. The hour component string value MUST be composed of the day component string value concatenated with the 2 digit hour in 24-hour format as a base 10 number written to a string.

The string is normalized using the method defined in section 2.2.3.1. The maximum length of the normalized token MUST be 128 bytes. If the length of the normalized token in bytes is less than or equal to 122, **Encoded property value** field MUST be equal to the normalized token.

If the length of the normalized token in bytes is greater than 122, **Encoded property value** field MUST be equal to bytes 14 through 29 of the normalized token, the last 16 bytes of the normalized token, and all 16 bytes of the MD5 (see [RFC1321]) for the normalized token, written sequentially.

If the length of the normalized token in bytes is greater than 122, **Encoded property value** field MUST have the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Semi-prefix (16 bytes)																															
...																															
...																															
Suffix (16 bytes)																															
...																															
...																															
MD5 (16 bytes)																															
...																															
...																															

Semi-prefix (16 bytes): Bytes 14 to 29 of normalized token.

Suffix (16 bytes): Last 16 bytes of normalized token

MD5 (16 bytes): MD5 digest value [RFC1321] of normalized token.

2.2.3.7 Compound Scope

A **compound scope index key** is an index key used to denote a search scope which contains all items which satisfy a condition referenced by **compound scopeID**. It is stored in a scope index record. The property identifier for this index key MUST be 0x7FFEFFF1 (**pidCompoundScope**).

The index key string encodes the **compound scopeID** as specified by the following table:

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
CompressedScopeID								FullSizeScopeID (optional)																							
...																															

CompressedScopeID (1 byte): Stores the value of the **compound scopeID** if it is smaller than 0x7E, in this case the field **FullSizeScopeID** field MUST NOT be present. If the **compound scopeID** is larger or equal to 0x7E, the value MUST be set to 0x7E and the field **FullSizeScopeID** field MUST be present.

FullSizeScopeID (4 bytes, optional): Stores compound scopeID in big-endian order if compound **scopeID** is greater than or equal to 0x7E.

Example

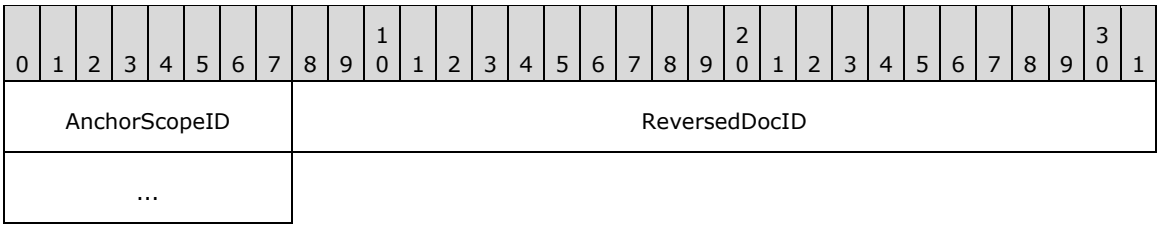
```
compound scopeID = 0x10, index key string = [0x10]
compound scopeID = 0x1234ff, index key string = [0x7e, 0x00, 0x12, 0x34, 0xff]
```

2.2.3.8 Anchor Scope

An **anchor scope index key** is an index key for a source item. A source item has links to target items. The collection of all target items is defined as a search scope for a given source item and is referenced by the anchor scope index key.

The property identifier for this index key MUST be 298.

The index key string encodes the **document identifier** as specified by the following table.



AnchorScopeID (1 byte): MUST be equal to 97.

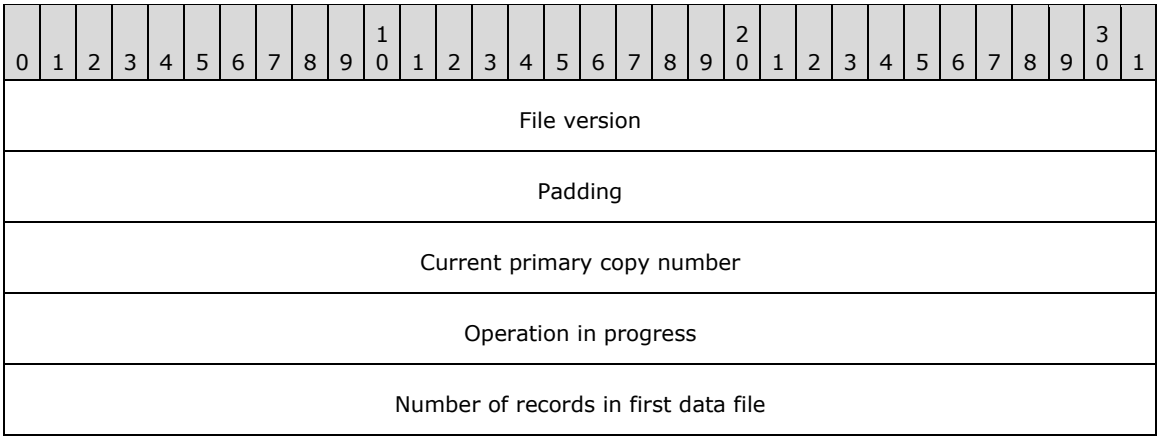
ReversedDocID (4 bytes): Stores document identifier in big-endian order.

2.2.4 Recoverable Storage File Format

The recoverable storage file format uses a basic transaction mechanism to store records. The size of each record in bytes MUST be a whole number. This format consists of a header file and 2 data files, each of which stores individual records. The header file stores structures required to maintain recoverable storage. Each data file stores individual records and the content of both data files is identical when the value of the **Operation in progress** field in the header file is 0x00000000. Of these data files, one is the primary data file and the other is a secondary data file. The information about which file is the primary data file is stored in the header. The data in the primary data file MUST be valid.

Integer values are recorded in little-endian except when stated otherwise.

2.2.4.1 Header File Format



Number of valid bytes in first data file
Number of unused bytes in first data file
...
Number of records in second data file
Number of valid bytes in second data file
Number of unused bytes in second data file
...
Signature 1
First data file user header (92 bytes)
...
...
Second data file user header (92 bytes)
...
...
Signature 2

File version (4 bytes): A 32-bit unsigned integer whose two higher bytes specify the file **format version** number. This MUST be either 0x00520000 or 0x00530000 or 0x00540000.<1>

Padding (4 bytes): The value of these 4 bytes is arbitrary, and MUST be ignored.

Current primary copy number (4 bytes): A 32-bit unsigned integer that specifies the data file which is the primary data file. If the first data file is the primary data file, the value of this field MUST be 0x00000000. If the second data file is the primary data file, the value of this field MUST be 0x00000001.

Operation in progress (4 bytes): A 32-bit unsigned integer that specifies whether the secondary data file contains valid data. This value MUST be less than or equal to 5. If the value of this field is 0x00000000, the data in secondary data file MUST be valid. If the value of this field is not 0x00000000, the data in the secondary data file MUST be ignored.

Number of records in first data file (4 bytes): A 32-bit unsigned integer that specifies the number of records stored in the first data file.

Number of valid bytes in first data file (4 bytes): A 32-bit unsigned integer that specifies the number of bytes in all the records stored in the first data file.

Number of unused bytes in first data file (8 bytes): A 64-bit unsigned integer that specifies the number of unused bytes present at the beginning of the first data file.

Number of records in second data file (4 bytes): A 32-bit unsigned integer that specifies the number of records stored in the second data file.

Number of valid bytes in second data file (4 bytes): A 32-bit unsigned integer that specifies the number of bytes in all the records stored in the second data file.

Number of unused bytes in second data file (8 bytes): A 64-bit unsigned integer that specifies the number of unused bytes present at the beginning of the second data file.

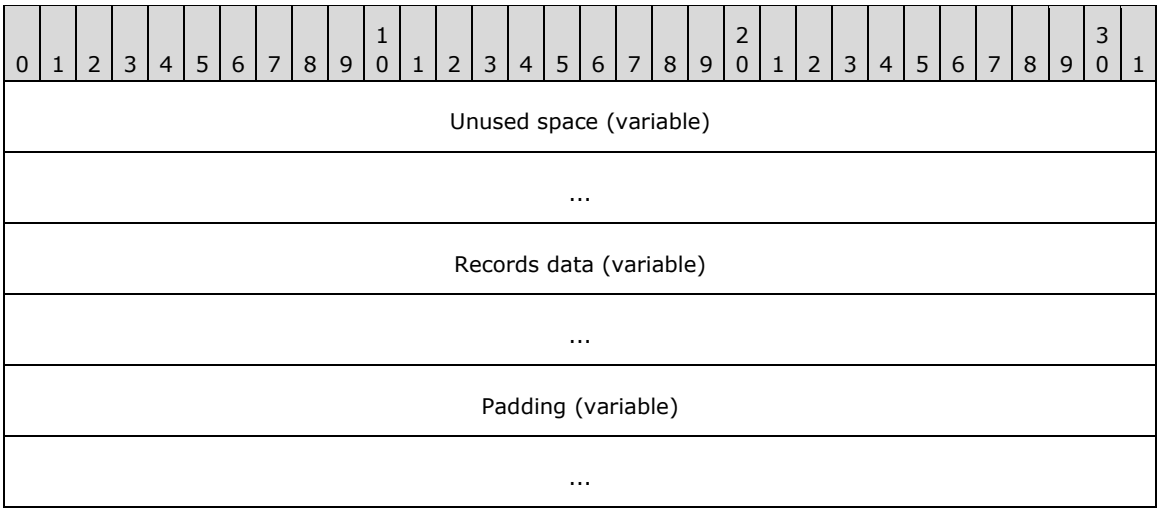
Signature 1 (4 bytes): A 32-bit unsigned integer that stores a signature for the file. This MUST be 0x46524853.

First data file user header (92 bytes): A block of 92 bytes in which the content and structure are defined by the file that is using the recoverable storage format to store extra data for the first data file.

Second data file user header (92 bytes): A block of 92 bytes in which the content and structure are defined by the file that is using the recoverable storage format to store extra data for the second data file.

Signature 2 (4 bytes): A 32-bit unsigned integer that stores a signature for the file. This MUST be 0x49524853.

2.2.4.2 Data File Format



Unused space (variable): An optional field that MUST be ignored. For the first data file, the size of this field is specified in the **Number of unused bytes in first data file** field in the header file**.** For the second data file, the size of this field is specified in the **Number of unused bytes in second data file** field in the header file.

Records data (variable): A list of records. The size and structure of these records depend on the implementation, although each record MUST contain a whole number of bytes. For the first data file, the number of records and the total size of all records are specified in the **Number of records in first data file** and the **Number of valid bytes in first data file** fields in the header file. For the second data file, the number of records and the total size of all records are specified in

the **Number of records in second data file** and the **Number of valid bytes in second data file** fields in the header file.

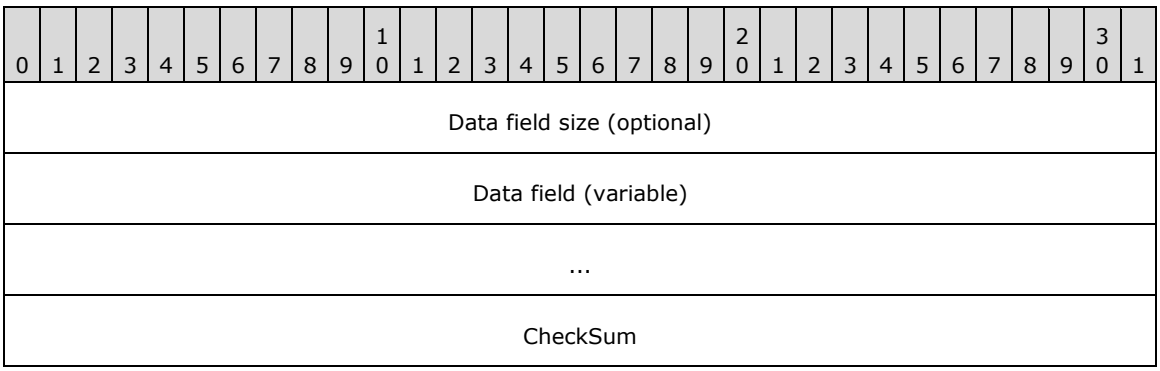
Padding (variable): An optional field that exists to ensure that the size of the data file, in bytes, is a multiple of 65536. The value of this field is arbitrary, and MUST be ignored.

2.2.5 CheckSummed Recoverable Storage File Format

The CheckSummed Recoverable Storage file format is an extension of the recoverable storage file format, as specified in section 2.2.4, and is used to provide data integrity validation. In the CheckSummed Recoverable Storage file format, every data record that is stored in the **Records data** field in the recoverable storage data files has the format of a CheckSummedRecord structure, as specified in section 2.2.5.1.

2.2.5.1 CheckSummedRecord Structure

A **CheckSummedRecord** stores fixed- and variable-sized data fields together with their checksum. Data field size is stored for variable-sized data fields. Data field size is not needed to correctly read a fixed-sized data field and is not stored for such fields.



Data field size (4 bytes, optional): A 32-bit unsigned integer that specifies the size of the **Data field** in bytes. This field MUST be present only for variable-sized data fields.

Data field (variable): The size and structure of this field depend on the file type.

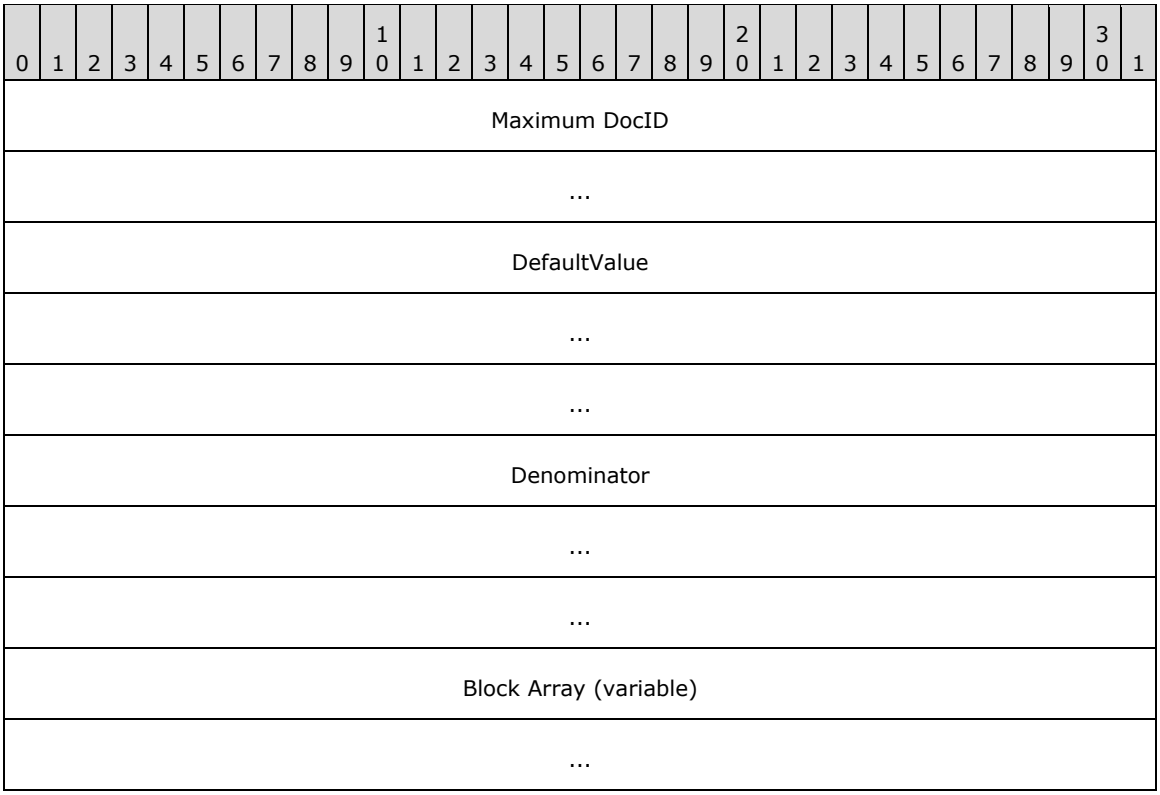
CheckSum (4 bytes): A 32-bit unsigned integer that specifies the checksum of the **Data field**. The value of this field is calculated in the following way: the **Data field** is split into 32-bit blocks. These blocks are added up as integers in little-endian bit ordering and the remainder (if any) is added as an integer in big-endian (with 32-bit overflow ignored in additions). The value of the field is the result of the previous calculation except when the result is 0. If the result is 0, the value of the field is 1. **Checksum** field MUST be recorded in little-endian. For example, a 15-byte long record (0x12, 0x34, 0x56, 0x78, 0x9a, 0xbc, 0xde, 0xf0, 0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd) will be split into three 32-bit blocks (0x12, 0x34, 0x56, 0x78), (0x9a, 0xbc, 0xde, 0xf0), (0x01, 0x23, 0x45, 0x67) which will be summed as integers in little-endian (0x78563412+0xf0debc9a+0x67452301=0xd07a13ad with overflow ignored) and the remainder (0x89, 0xab, 0xcd) is added as integer in big-endian (0xd07a13ad+0x89abcd=0xd103bf7a with overflow ignored) giving the final value of 0xd103bf7a for the checksum.

2.2.6 Sparse Array File Format

The sparse array file format is based on the CheckSummed Recoverable Storage file format, as specified in section 2.2.5, and is used to store an array of DWORDs (see [MS-DTYP]) or floats. This format stores consecutive duplicates as one value.

If the value of the **Number of records in first data file** field in the recoverable storage header file, as specified in section 2.2.4.1, is zero, the first data file is empty. If the value of the **Number of records in second data file** field in the recoverable storage header file is zero, the second data file is empty.

The format of the **CheckSummedRecord's Data field** from section 2.2.5 is defined in the following table. The **Unused space** field MUST have size zero.



Maximum DocID (8 bytes): Stores a fixed-sized **CheckSummedRecord** of a DWORD (see [MS-DTYP]). Represents the maximum **document identifier** for which data is recorded in the file.

DefaultValue (12 bytes): Stores a variable-sized CheckSummedRecord of a float.

If the sparse array stores DWORDs, the default value for an element of the sparse array is **DefaultValue** field divided by **Denominator** (the next field) and truncated to an unsigned long.

If the sparse array stores floats, the default value for an element is **DefaultValue** field divided by **Denominator** (the next field), truncated to an unsigned long and multiplied by **Denominator** field.

If the sparse array stores uncompressed floats, the default value for an element is **DefaultValue** field.

Denominator (12 bytes): Stores a variable-sized **CheckSummedRecord** of a float and is used only when the sparse array stores float values. In this case the value stored in the **SparseArrayBlock** is the actual float number divided by **Denominator** field and truncated to an unsigned long. When the sparse array stores DWORDs, the value is stored in the **SparseArrayBlockData** structure, as specified in section 2.2.6.2, and the **Denominator** field MUST be ignored.

Block Array (variable): An array of **SparseArrayBlock** objects. Each **SparseArrayBlock** object has two **CheckSummedRecords** as described in the following section.

2.2.6.1 SparseArrayBlock Structure

This is a compact way of representing a sequence of up to 256 DWORDs or floats.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Block number																															
...																															
SparseArrayBlock data (variable)																															
...																															

Block number (8 bytes): Stores a fixed size **CheckSummedRecord** of a DWORD. The value of this DWORD is used as a base for the **document identifiers** referred in the **SparseArrayBlock data** field. Each block stores data for a contiguous range of document identifiers whose 24 most significant bits are equal to the **Block number** field. If there is no **SparseArrayBlock** for a range of document identifiers the data for these document identifiers is equal to the default value, calculated as specified for the **DefaultValue** field.

SparseArrayBlock data (variable): Stores a variable-sized **CheckSummedRecord** of a **SparseArrayBlockData**.

2.2.6.2 SparseArrayBlockData Structure

The following table describes the **SparseArrayBlockData** structure.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Previous Bits 0								Bitmap 0								Previous Bits 1								Bitmap 1							
Previous Bits 2								Bitmap 2								Previous Bits 3								Bitmap 3							
Previous Bits 4								Bitmap 4								Previous Bits 5								Bitmap 5							
Previous Bits 6								Bitmap 6								Previous Bits 7								Bitmap 7							
Previous Bits 8								Bitmap 8								Previous Bits 9								Bitmap 9							
Previous Bits 10								Bitmap 10								Previous Bits 11								Bitmap 11							
Previous Bits 12								Bitmap 12								Previous Bits 13								Bitmap 13							
Previous Bits 14								Bitmap 14								Previous Bits 15								Bitmap 15							
Previous Bits 16								Bitmap 16								Previous Bits 17								Bitmap 17							
Previous Bits 18								Bitmap 18								Previous Bits 19								Bitmap 19							

Previous Bits 20	Bitmap 20	Previous Bits 21	Bitmap 21
Previous Bits 22	Bitmap 22	Previous Bits 23	Bitmap 23
Previous Bits 24	Bitmap 24	Previous Bits 25	Bitmap 25
Previous Bits 26	Bitmap 26	Previous Bits 27	Bitmap 27
Previous Bits 28	Bitmap 28	Previous Bits 29	Bitmap 29
Previous Bits 30	Bitmap 30	Previous Bits 31	Bitmap 31
Valarray (variable)			
...			

Previous bits(i) (1 byte): For each *i* from 1 to 31, **Previous Bits**(*i*) field is equal to the total number of bits set to 1 in the fields **Bitmap** (0) field through **Bitmap** (*i*-1) field. **Previous Bits**(0) field MUST be equal to 0.

Bitmap(i) (1 byte): Considering that the current **SparseArrayBlockData** structure belongs to a **SparseArrayBlock** structure with the **Block Number** field equal to *j*, if the bit *k* in **Bitmap** field *i* is set, this means that the value corresponding to document identifier `k + (i * 8) + (j * 256)` is different from the value corresponding to the document identifier `k + (i * 8) + (j * 256) - 1`. If the bit is not set, then the two values are identical. The first bit in **Bitmap**(0) field SHOULD be 1. If it's not, then all the elements before the first bit set to the default value.

Valarray (variable): This is an array of DWORDs (see [MS-DTYP]). It MUST contain as many elements as there are bits set in all the **Bitmap** fields of the **SparseArrayBlockData object**. To find the data associated with a document identifier *w*, go to the (w / 256)-th block in the sparse array file format and find the total number of bits set in **Bitmap**(0 to ((*w*& 0xFF) / 8) - 1) field (which is stored in **Previous Bits**((*w*& 0xFF) / 8)) field and then add the number of bits set in the first (*w*& 0x7) bits of the ((*w*& 0xFF)-th **Bitmap** Field. This number is the 1-based index in the **Valarray** field of the corresponding data being stored for this **document identifier**. If this value is zero, the data for this document identifier is the default value, calculated as specified for **DefaultValue**.

2.3 Content Index File Format

A **content index file** stores an **inverted index** that allows fast search for all items that contain a given term in a specific property of an item. Each distinct property of an item, such as title, author, main text, and so on, has a separate property identifier assigned to it. For each search query term, it is possible to define a **content index key** that is used to find information about this term in content index file.

A content index file stores a set of **content index records**. Each content index record is associated with a unique content index key and stores **document identifiers** of all items that contain the term used to create content index key in a part of item defined by property identifier. See the following diagram:

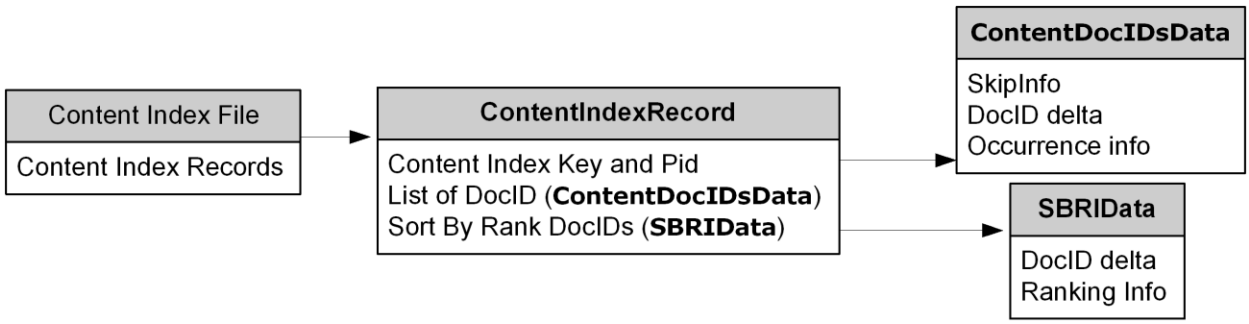


Figure 1: Basic structure of a content index file (version 0x52, 0x53)

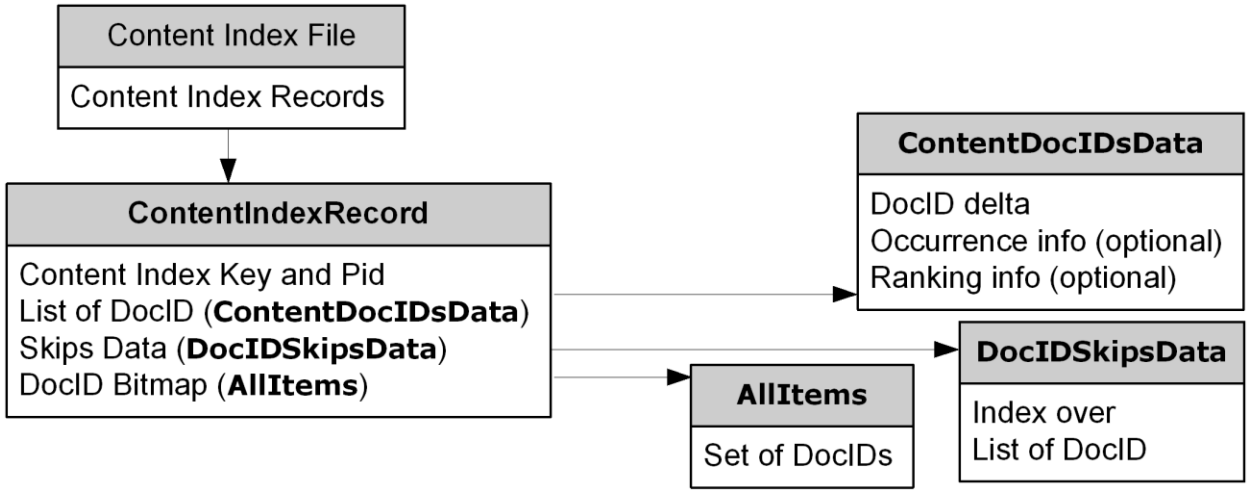


Figure 2: Basic structure of a content index file (version 0x54)

A content index file has two input parameters: **DocIDMax** and **format version**.

A content index file MUST contain: records with content index keys, one record with max key, records with EOF keys for all property identifiers that are used in at least one record with content index key, and one record with EOF key and property identifier equal to 0x7FFEFFFF. Content index records MUST be ordered by content index key in default index key sorted order.

A content index file which belongs to a **master index component** whose **format version** is equal to 0x53 or 0x54 MUST contain records with BOF keys for all property identifiers that are used in at least one record with content index key, one record with BOF key and property identifier equal to 0x7FFEFFFF.<2>

A content index file which belongs to an index component whose **format version** is less than 0x54 MUST NOT contain content index records with property identifier equal to 0x7ffeFFC8 or 0x7ffeFFC9. Content index record with property identifier equal to 0x7ffeFFC8 contains a list of items that are more likely to be relevant for a query that contains the term that is used to create the content key and for each item it contains a value that represents relative **rank** of an item for that term. Content index record with property identifier equal to 0x7ffeFFC9 MUST be present if content index record with property identifier equal to 0x7ffeFFC8 is present with same key and the record MUST contain a set of items that are less likely to be relevant for a query that contains the term that is used to create the content key.

2.3.1 ContentIndexRecord

A **content index record** encodes a **content index key** and a list of integers representing **document identifiers**. The document identifiers MUST be stored in increasing order as an incremental change from the previous document identifier. There MUST be no duplicates. For each document identifier, the position of all instances of the term associated with the content index key in the corresponding property of the item pointed to by the property identifier MUST be recorded in a list of occurrences. For content index records with a large number of document identifiers, an extra list of document identifiers is stored as necessary. This list MUST contain a subset of document identifiers for the current content index record that has the highest **rank** value for the current content index key / property identifier pair.

The content index key MUST be encoded as an incremental change from the previous content index key value in the **content index file**. **Prefix Length** MUST be equal to the number of bytes that are in the previous content index key. **Suffix Length** MUST be equal to the number of bytes that are different, and follow directly after the prefix bytes. For the first content index record in a content index file, **Prefix Length** MUST be zero. The total length of the current content index key MUST be equal to **Prefix Length** + **Suffix Length**.

The content index record format is defined in the following table. Each field is present unless specified otherwise.

Name	Size	Type
Link	20 bits	**BitStream field**
Prefix/Suffix Length	Variable	**PrefixSuffixCompress**
SuffixValue	Variable	**Suffixbyte**[suffix length]
Pid	Variable	**PidCompress**
DocIDCount	Variable	**DocIDCountCompress**
IsSBRIPresent	1 bit	BitStream field
SBRIOffset	32 bits	BitStream field
AverageDocIDbitcount	5 bits	BitStream field
logCDocIDs	5 bits	BitStream field
SkipsPage<3>	32 bits	BitStream field
SkipsOffset<4>	32 bits	BitStream field
IsCIXLinkPresent<5>	1 bit	BitStream field
CIXPage<6>	32 bits	BitStream field
CIXOffset<7>	32 bits	BitStream field
ContentDocIDsData	Variable	**ContentDocIDData[DocIDCount]**

Name	Size	Type
Padding_dword_align	Variable	BitStream field
SBRIData	Variable	**SBRIData[**n**]**
DocIDSkipCount<8>	Variable	**BitCompress(9)**
DocIDSkipsData<9>	Variable	**DocIDSkipData[DocIDSkipCount]**
AllItems	Variable	**AllItems**

Content index record fields:

Link (20 bits): Stores the size of the content index record in bits. The field value MUST be zero if the size of the content index record is greater than 2^20 bits or if the current record is the **max key**.

Prefix/Suffix Length (variable): Contains **Prefix Length** and **Suffix Length**. The sum of these 2 values MUST NOT exceed 129. **Prefix Length** MUST NOT exceed the sum of **Prefix Length** and **Suffix Length** for the previous content index record. **Prefix Length** MUST be zero for the first content index record in the content index file.

SuffixValue (variable): MUST contain **suffix length** bytes. Each byte MUST be read as a BitStream field (size 8 bits) from **BitStream**; these are the modified bytes from the previous content index key.

Pid (variable): MUST contain the value of the property identifier associated with the content index key.

DocIDCount (variable): MUST contain the total count of document identifiers in the content index key. MUST NOT be present if the current **index key** is the max key.

IsSBRIPresent (1 bit, optional):

- MUST NOT be set if **log2** (**DocIDCount**)* 1024 >= **DocIDCount**
- MUST NOT be present if the **format version** is 0x54.
- MUST NOT be set if the current content index record contains the EOF key.
- MUST be set only if **SBRIData** is present for the content index record.
- MUST NOT be present if the current index key is the Max key.
- MUST NOT be set if the current content index record contains the BOF key.<10>

SBRIOffset (32bits, optional): Number of DWORDs (see [MS-DTYP]) to skip in BitStream from the beginning of this field to the position in the BitStream at the beginning of **SBRIData** field. **SBRIOffset** MUST NOT be present if the **IsSBRIPresent** field bit is not set. BitStream MUST be aligned up to the nearest DWORD before reading the **SBRIData** field.

- MUST NOT be present if the **format version** is 0x54.
- MUST NOT be present if the current content index key is the max key.

AverageDocIDbitcount (5 bits): Defines the average number of bits to use for document identifier (1) storage. MUST NOT be present if the current index key is the max key.

logCDocIDs (5 bits, optional): Parameter that defines the frequency of the **DocID skips** and how many bits each DocID skip takes. No DocID skips are used for current content index record if the **logCDocIDs** field is zero.

- MUST NOT be present if the current index key is the max key.
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

SkipsPage (32 bits, optional):<11>

- 32-bit number of the page in the current content index file that contains the beginning of the DocID skips data for the current content index record.
- MUST NOT be present if the **logCDocIDs** field equals zero.
- MUST NOT be present if the **format version** is less than 0x54.
- MUST NOT be present if the current index key is the max key.
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

SkipsOffset (32 bits, optional):<12>

- 32-bit value of the offset on a page in the current content index file that contains the beginning of the DocID skips data for the current content index record.
- MUST NOT be present if the **logCDocIDs** field equals zero.
- MUST NOT be present if the **format version** is less than 0x54.
- MUST NOT be present if the current index key is the max key.
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

IsCIXLinkPresent (1 bit, optional):<13> If this bit is set, this content index record MUST contain a link to the document identifier information in the corresponding **.cix file**. MUST NOT be present if the **format version** is 0x52.

- MUST NOT be present if the current index key is the max key.
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

CIXPage (32 bits, optional):<14>

- MUST NOT be present if the **IsCIXLinkPresent** field bit is not set.
- MUST NOT be present if the **format version** is 0x52.
- MUST contain the 32-bit value of a page in the CIX file that contains the beginning of the index extension data for the current content index record.
- If the **CIXPage** field equals 0xffffffff, the CIX link is not valid and index extension information is not available for the current content index record.
- MUST NOT be present if the current index key is the max key.
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

CIXOffset (32 bits, optional):<15>

- MUST NOT be present if the **IsCIXLinkPresent** field bit is not set.
- MUST NOT be present if the **format version** is 0x52.

- MUST contain the 32-bit value of the offset on a page in the CIX file that contains the beginning of the index extension data for the current content index record.
- MUST NOT be present if the current index key is the max key.
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

ContentDocIDsData (Variable, optional): Stores document identifiers for the given content index key. Contains **DocIDCount ContentDocIDData** records numbered from zero to **DocIDCount** -1.

- MUST NOT be present if the current index key is the max key.
- MUST NOT be present if **Pid** equals 0x7ffeFFC9.

Padding_dword_align (variable):

- A variable length field to align the next field on 32-bit boundary.
- The value of this field is arbitrary, and MUST be ignored.
- MUST NOT be present if the **format version** is 0x54.
- MUST NOT be present if the **IsSBRIPresent** field is not set.
- MUST NOT be present if the current index key is the max key.
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

SBRIData (variable):

- This field MUST store the highest-ranked document identifiers sorted in ascending order by document identifier. The document identifier rank MUST be calculated as follows:
  - `fRank = 0.05*cOcc/ (0.25 + ( 0.75 * maxoccur / AvdlThisPid ))`
  - where *cOcc* is the total number of occurrences of the current search query term in an item for the current property identifier, *maxoccur* is a **Max Occurrence**, as defined in the **MaxOccBuckets** table, as specified in section 2.1.2, and *AvdlThisPid* is a **cAvgOcc** field, as defined in section 2.8.1, for the current property identifier.
- **SBRIData** field MUST contain (log2 (DocIDCount)* 1024) document identifiers with the maximum **fRank** of all document identifiers for this content index record.
- MUST NOT be present if the current index key is the max key.
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

DocIDSkipCount (Variable, optional)<16> (BitCompress(9)):

- Number of **DocIDSkipData** records for the current content index record.
- MUST NOT be present if the **logCDocIDs** field equals zero.
- MUST NOT be present if the **format version** is less than 0x54.
- MUST NOT be present if the current index key is the max key.
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

DocIDSkipsData (Variable, optional):<17>

- MUST NOT be present if the **logCDocIDs** field equals zero.

- MUST NOT be present if the **format version** is less than 0x54.
- Stores DocID skips for the current content index record. Contains **DocIDSkipCount** **DocIDSkipData** records numbered from zero to **DocIDSkipCount** -1. Each **DocIDSkipData** record defines the relative position of the document identifier in the **ContentDocIDsData[DocIDCount]** structure.
- MUST NOT be present if the current index key is the max key.
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

AllItems (Variable, optional):

- Contains a set of all document identifiers that are present in content index records with the same key but different property identifiers.
- MUST NOT be present if **Pid** is not equal to 0x7ffeFFC9.
- MUST NOT be present if the current content index record contains the EOF key.
- MUST NOT be present if the current content index record contains the BOF key.

The **AllItems** structure is defined by the following table:

Name	Size	Type
Version	4 bits	BitStream field
DocIDMask	256 bits	BitStream field
DocIdBitmapSize	32 bits	BitStream field
Padding	Variable	BitStream field
DocIdBitmap	**DocIdBitmapSize** bits	BitStream field

AllItems fields:

Version (4 bits): MUST be zero.

DocIDMask (256 bits): Each bit is numbered from zero to 255. The bit at position *N* MUST be set if there exists an item that is stored in the current content index record with a document identifier that has the low-order byte equal to *N*.

DocIdBitmapSize (32 bits):

- Contains the total number of bits in the **DocIdBitmap** field.
- MUST be equal to ((MaxBitMapId/256) * DocIdMaskDelta[256] + MaxBitMapIdDelta + 2). *MaxBitMapId* is the maximum value for the document identifier for the items stored in the current content index record. *MaxBitMapIdDelta* is the number of set bits in **DocIDMask** at positions less than the low-order byte of **MaxBitMapId.** *DocIdMaskDelta[256]* is the number of set bits in **DocIDMask**. The result of the division is rounded down before multiplication.

Padding (Variable, optional):

- A variable length field to align the next field on a 32-bit boundary.
- The value of this field MUST be ignored.

DocIdBitmap (Variable, optional):

- MUST contain **DocIdBitmapSize** bits.
- For each item stored in the current content index record, a bit at position ((DocId/256) * DocIdMaskDelta[256] + DocIdMaskDelta[N] + 1) MUST be set. *DocId* is the document identifier for the item. *N* equals the low-order byte of **DocId**. *DocIdMaskDelta[N]* is the number of set bits in **DocIDMask** at positions less than *N. DocIdMaskDelta[256]* is the number of set bits in **DocIDMask**. The result of the division is rounded down before multiplication.
- All other bits MUST NOT be set.

A **ContentDocIDData[n]** record is defined by the following table, where *n* is from zero to (DocIDCount -1).

Name	Size	Type
DocIDSkipbits	**logCDocIDs** + 6 bits	BitStream field
DocIDSkip	log2 (**DocIDMax**) bits	BitStream field
DocIDDelta	Variable	**BitCompress(AverageDocIDbitcount** field**+ 1)**
MaxDocIDOccBucket	7 bits	BitStream field
AllPropertyRank	12 bits	BitStream field
OccCount	Variable	**BitCompress(3)**
OccSkip	9 + log2 (**OccCount**/16) bits	BitStream field
Padding_dword_align	variable	BitStream field
OccsDelta	Variable	**BitCompress(7)[OccCount]**

ContentDocIDData[n] fields:

DocIDSkipbits (logCDocIDs + 6 bits , optional):

- MUST NOT be present if the **format version** is 0x54.
- The field MUST NOT be present if *n* is not a multiple of the **logCDocIDs** field ***4** or the **logCDocIDs** field is zero.
- The field MUST be zero if **DocIDCount** <= *n* + **logCDocIDs** *4.
- The field contains the number of bits from the beginning of the current record **ContentDocIDsData[n]** to the record **ContentDocIDsData[***n***+ logCDocIDs***4**].**
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

DocIDSkip (log2 (DocIDMax) bits , optional):

- MUST NOT be present if the **format version** is 0x54.
- The field MUST NOT be present if *n* is not a multiple of **logCDocIDs *4** or **logCDocIDs** is zero.

- The field MUST be zero if **DocIDCount** <= *n*+ **logCDocIDs** *4.
- The field contains a document identifier that is stored in **ContentDocIDsData**[*n*+ **logCDocIDs** *4**]** record. **DocIDMax** is a global parameter for the content index file.
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

DocIDDelta (Variable): MUST store the incremental value between the previous and current document identifiers. If the current document identifier is the first in **ContentDocIDsData**, the actual document identifier MUST be stored. The value returned by BitCompress(AverageDocIDbitcount + 1) **MUST** be incremented by 1 before it is used as **DocIDDelta**.

MaxDocIDOccBucket (7 bits, optional):

- MUST NOT be present if the current content index record contains the EOF key.
- **MaxDocIDOccBucket** MUST be the **MaxOccBucket** for a document identifier and property identifier.
- MUST NOT be present if the current content index record contains the BOF key.<18>
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

AllPropertyRank (12 bits, optional):

- Contains a 12 bit unsigned **integer** that defines the relative rank of an item with the current document identifier for the term defined by the key in the current content index record.
- MUST NOT be present if **Pid** is not equal to 0x7ffeFFC8.
- MUST NOT be present if the current content index record contains the EOF key.
- MUST NOT be present if the current content index record contains the BOF key.<19>

OccCount (Variable, optional):

- Stores the number of occurrences for the current document identifier.
- MUST NOT be present if the current content index record contains the EOF key.
- **OccCount** is assumed to be equal to "1" in all other references in this section if the current content index record contains the EOF key.
- MUST NOT be present if the current content index record contains the BOF key.<20>
- **OccCount** is assumed to be equal to "1" in all other references in this section if the current content index record contains the BOF key.<21>
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

OccSkip (9 + log2 (OccCount /16) bits, optional):

- Field MUST NOT be present if **OccCount** < 8.
- MUST store sum of size **Padding_dword_align** and **OccsDelta**[**OccCount**] in bits.
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

Padding_dword_align (Variable, optional):

- A variable-sized field to align **OccDelta[OccCount]** on a 32-bit boundary.
- The value of this field is arbitrary, and MUST be ignored.

- MUST NOT be present if **OccCount** < 8.
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

OccsDelta(Variable, optional):

- MUST store **OccCount** values encoded as a BitCompress(7), as specified in section 2.2.2.1.
- If the current index key is not an EOF key, **OccsDelta** MUST contain occurrences in the current item. The first value is equal to the first occurrence minus 1. Each subsequent value is equal to the difference between the current and the previous occurrence minus 1.
- If the current index key is not a BOF key, **OccsDelta** MUST contain occurrences in the current item. The first value is equal to the first occurrence minus 1. Each subsequent value is equal to the difference between the current and the previous occurrence minus 1.<22>
- MUST NOT be present if **Pid** equals 0x7ffeFFC8 or 0x7ffeFFC9.

Example:

Value_1 = Occurrence_1 - 1

Value_2 = Occurrence_2 - Occurrence_1 - 1

Value_3 = Occurrence_3 - Occurrence_2 – 1

- If the current index key is an EOF key and the property identifier is NOT 0x7FFEFFFF, **OccsDelta** MUST contain the maximum occurrence value for the current property identifier.
- If the current index key is an EOF key and the property identifier is 0x7FFEFFFF, **OccsDelta** MUST contain the sum of the maximum occurrence values for all property identifiers.
- If the current index key is a BOF key and the property identifier is NOT 0x7FFEFFFF, **OccsDelta** MUST contain the maximum occurrence value for the current property identifier.<23>
- If the current index key is a BOF key and the property identifier is 0x7FFEFFFF, **OccsDelta** MUST contain the sum of the maximum occurrence values for all property identifiers.<24>

A **SBRIData[n]** record is defined by the following table, where *n* is from zero to (log2 (DocIDCount)* 1024 -1).

Name	Size	Type
DocIDDelta	Variable	**BitCompress**(log2 (**DocIDMax** / (log2 (**DocIDCount**) * 1024)))
Rank	12 bits	BitStream field

SBRIData[n] fields:

DocIDDelta (Variable): MUST store the incremental value between the current document identifier and the previous document identifier. If the current document identifier is the first in **SBRIData,** the actual value MUST be stored. The value returned by BitCompress(7) MUST be incremented by 1 before it is used as **DocIDDelta.**

Rank (12 bits):

- Contains 12 bits of ranking information.
- If **fRank** for the current document identifier is >=1, the value of **Rank** MUST be equal to:

- `Min(0x7ff, (log(1.0 + ( fRank - 1.0 ) * dResolutionAdjust) / dLnDivider)) + 0x0fff`

- If **fRank** for the current document identifier is < 1, the value of **Rank** MUST be equal to:
  - `Min(0x7ff,(log(1.0 + ( 1.0/fRank - 1.0 ) * dResolutionAdjust) / dLnDivider))`
  - where **ResolutionAdjust** = 26612.566117305021291272917047288 and **dLnDivider** = 0.009950330853168082848215357544260 7.

A **DocIDSkipData [n]** record is defined by the following table, where *n* is from zero to **DocIDSkipCount** -1.

Name	Size	Type
DocIDDelta	Variable	**BitCompress(** log2**(logCDocIDs * 4) + AverageDocIDbitcount + 2)**
DocIDSkipOffsetDelta	Variable	**BitCompress(** min**(logCDocIDs +6, 32))**
IsDefaultDocIDSkip	1 bit	BitStream field
DocIdSkip	log2(**logCDocIDs** * 4) bits	BitStream field

DocIDSkipData [n] fields:

DocIDDelta (Variable):

- Contains incremental value between the document identifier for the previous **DocIDSkipData** and the current one.
- The value returned by **BitCompress**, as specified in section 2.2.2.1, MUST be incremented by 1 before it is used as DocIDDelta**.**
- MUST contain the actual document identifier if *n* equals zero.
- Document identifier MUST be present in one of **ContentDocIDData** records in the current content index record**.**

DocIDSkipOffsetDelta (Variable):

- If *n* is greater than zero, the field MUST contain the number of bits from the beginning of the **ContentDocIDsData[m]** record to the beginning of record **ContentDocIDsData[k]**, where **ContentDocIDsData[m]** stores the document identifier equal to the document identifier stored in **DocIDSkipData [n - 1]** and **ContentDocIDsData[k]** stores document identifier equal to the document identifier stored in **DocIDSkipData [n]**.
- If *n* equals zero, the field MUST contain the number of bits from the beginning of the **ContentDocIDsData[0]** record to the beginning of record **ContentDocIDsData[k]**, where **ContentDocIDsData[k]** stores the document identifier equal to the document identifier stored in **DocIDSkipData [n]**.

IsDefaultDocIDSkip (1 bit):

- If *n* is greater than zero, the field MUST be "1" if **k - m** equals **logCDocIDs** * 4, where **ContentDocIDsData[m]** stores the document identifier equal to the document identifier stored in **DocIDSkipData [n - 1]** and **ContentDocIDsData[k]** stores the document identifier equal to the document identifier stored in **DocIDSkipData [n]**.

- If *n* equals zero, the field MUST be "1" if **k** equals **logCDocIDs** * 4, where **ContentDocIDsData[k]** stores index data for the document identifier equal to the document identifier stored in **DocIDSkipData [n]**.
- The field MUST be zero in all other cases.

DocIdSkip (log2(logCDocIDs * 4) bits, optional):

- The field MUST NOT be present if **IsDefaultDocIDSkip** is "1".
- If *n* is greater than zero, the field MUST contain the value **k - m**, where **ContentDocIDsData[m]** stores index data for the document identifier equal to the document identifier stored in **DocIDSkipData [n - 1]** and **ContentDocIDsData[k]** stores the document identifier equal to the document identifier stored in **DocIDSkipData [n]**.
- If *n* equals zero, the field MUST contain **k**, where **ContentDocIDsData[k]** stores document identifier equal to the document identifier stored in **DocIDSkipData [n]**.

2.4 Scope Index File Format

A scope index file stores a set of scope index records. Each scope index record is associated with a unique **scope index key** and stores **document identifiers** for all items that belong to a specific set pointed to by the scope index key.

The set can include, for example, all items on a particular site, all items authored by a particular person, or all items that have a given extension, and can be used to limit the items returned by a **search query**.

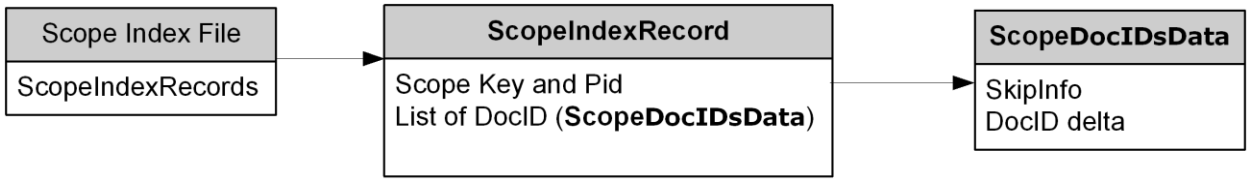


Figure 3: Basic structure of a scope index file

A **basic scope index** is a scope index file that MUST contain zero or more records with basic scope index keys or an anchor scope index key and one record with a **max key.**

A **compound scope index** is a scope index file that MUST contain zero or more records with compound scope index keys and one record with a max key.

Scope index records MUST be ordered by scope index key in default index key sorted order.

2.4.1 ScopeIndexRecord

ScopeIndexRecord MUST encode a scope index key and a list of integers representing **document identifiers**. The document identifiers MUST be stored in increasing order as an incremental change from the previous document identifier. There MUST be no duplicates.

The scope index key MUST be encoded as an incremental change from previous scope index key value in scope index file. Prefix length MUST equal the number of bytes that are the same as in previous scope index key. Suffix length MUST equal the number of bytes that are different and follow directly after prefix bytes. For the first **ScopeIndexRecord** in a scope index file**,** prefix length MUST be zero. The total length of current scope index key MUST equal prefix length + suffix length.

ScopeIndexRecord is defined by the following table:

Name	Size	Type
Link	20 bits	**BitStream field**
Prefix/Suffix Length	Variable	**PrefixSuffixCompress**
SuffixValue	Variable	Suffixbyte[suffix length]
Pid	Variable	**PidCompress**
DocIDCount	Variable	**DocIDCountCompress**
AverageDocIDbitcount	5 bits	BitStream field
logCDocIDs	5 bits	BitStream field
ScopeDocIDsData	Variable	**ScopeDocIDData[DocIDCount]**

ScopeIndexRecord fields:

Link (20 bits): Stores the size of the scope index record in bits. The field value MUST be 0 if size of scope index record is greater than 2^20 bits or if current record contains max key.

Prefix/SuffixLength (Variable): Contains prefix length and suffix length. The sum of these 2 values MUST NOT exceed 129. Prefix length MUST NOT exceed sum of prefix length and suffix length for previous scope index record**.** Prefix length MUST be 0 for first scope index record in scope index file**.**

SuffixValue (Variable, optional): MUST contain **suffix length** bytes. Each byte MUST be read as a BitStream field (size 8 bits) from **BitStream**; these are the modified bytes from the previous scope index key.

Pid (Variable): MUST contain the value of property identifier associated with scope index key.

DocIDCount (Variable, optional): MUST contain the total count of document identifiers in the scope index key. MUST NOT be present if the current index key is max key.

AverageDocIDbitcount (5 bits): Defines the average number of bits to use for document identifier storage. MUST NOT be present if the current index key is max key.

logCDocIDs (5 bits, optional): Parameter that defines frequency of DocID skips and how many bits each DocID skip takes. If **logCDocIDs** field is 0, DocID skips MUST NOT be used for current scope index record. MUST NOT be present if current index key is max key.

ScopeDocIDsData (Variable): Stores document identifiers for the given scope index key. Contains DocIDCount **ScopeDocIDData** field records numbered from 0 to DocIDCount -1. MUST NOT be present if the current index key is max key.

ScopeDocIDData record is defined by the following table, where n is from 0 to DocIDCount -1:

Name	Size	Type
DocIDSkipbits	**logCDocIDs** + 6 bits	BitStream field
DocIDSkip	log2 (**DocIDMax**) bits	BitStream field
DocIDDelta	Variable	**BitCompress(AverageDocIDbitcount** field **+ 1)**

ScopeDocIDData fields:

DocIDSkipbits (logCDocIDs + 6 bits ,optional):

- The field MUST NOT be present if n is not a multiple of **logCDocIDs** field ***4** or if **logCDocIDs** field is zero.
- The field MUST be present if n is a multiple of **logCDocIDs** field***4**.
- The field MUST be zero if **DocIDCount** field <= n + **logCDocIDs** field *4.
- The field contains the number of bits from the beginning of current record **ScopeDocIDsData[n]** to the record **ScopeDocIDsData[n+ logCDocIDs** field *4**].**

DocIDSkip (log2 (DocIDMax) bits, optional):

- The field MUST NOT be present if n is not a multiple of **logCDocIDs** field ***4** or **logCDocIDs** field is zero.
- The field MUST be present if n is a multiple of **logCDocIDs** field ***4.**
- The field MUST be zero if **DocIDCount** field <= n+ **logCDocIDs** field *4.
- The field contains a document identifier that is stored in **ScopeDocIDsData[n+ logCDocIDs***4**]** record. **DocIDMax** is a global parameter for the scope index file.

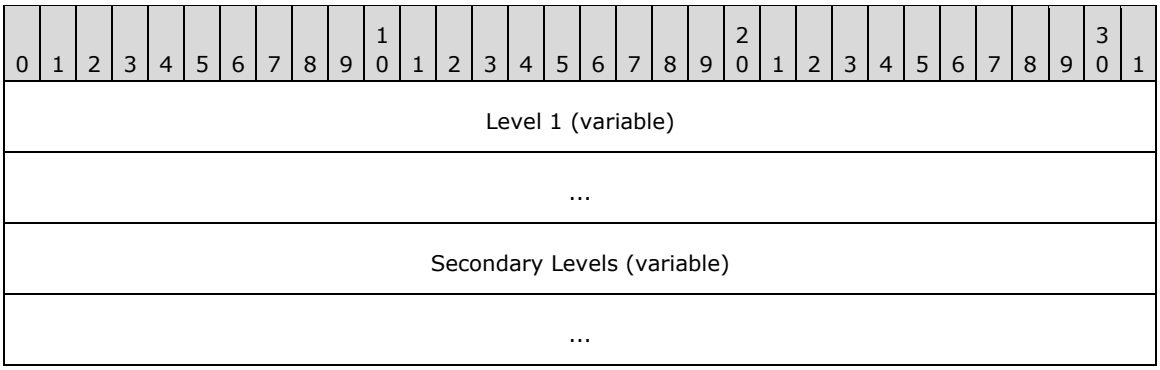
DocIDDelta (Variable): MUST store the incremental value between the current document identifier and the previous one. If the current document identifier is the first in **ScopeDocIDsData** field**,** the actual document identifier MUST be stored. The value returned by BitCompress(AverageDocIDbitcount + 1) MUST be incremented by 1 before it is used as **DocIDDelta**.

2.5 Index Directory File Format

A file in the index directory file format is always associated with a content index, a basic scope index, or a compound scope index. This format consists of several segments, each of which stores a list of index keys selected from the **content index file** that it is associated with. These segments represent levels of lookup data structures. Each segment is a sorted list of **records**. Consecutive records are consolidated into fixed-sized batches called **index directory pages**.

A lookup into an **index directory file** produces an index key and a position in the associated index file.

2.5.1 File Layout



Level 1 (variable): An array of index directory pages. The index directory records stored in these pages contain index keys from the associated index and their position in the index. The total size of level 1 MUST be a multiple of 4096 bytes.

Secondary Levels (variable): Sequence of **index directory levels**. The index directory records stored in **Secondary Levels** provide lookup to preceding index directory levels. MUST be a multiple of 4096 bytes. The Secondary Levels are not present in the following cases:

- When **Level 1** contains only one **index directory page**.
- When the index directory file belongs to a **full-text index component** that is being created by an in-progress merge process, as specified in section 2.9.

Each index directory level stores a sorted list of index directory records, split in segments which are stored in pages of 4,096 bytes (4K) each. The list of index directory records is sorted based on the index key using the sort criteria defined in section 2.2.3.

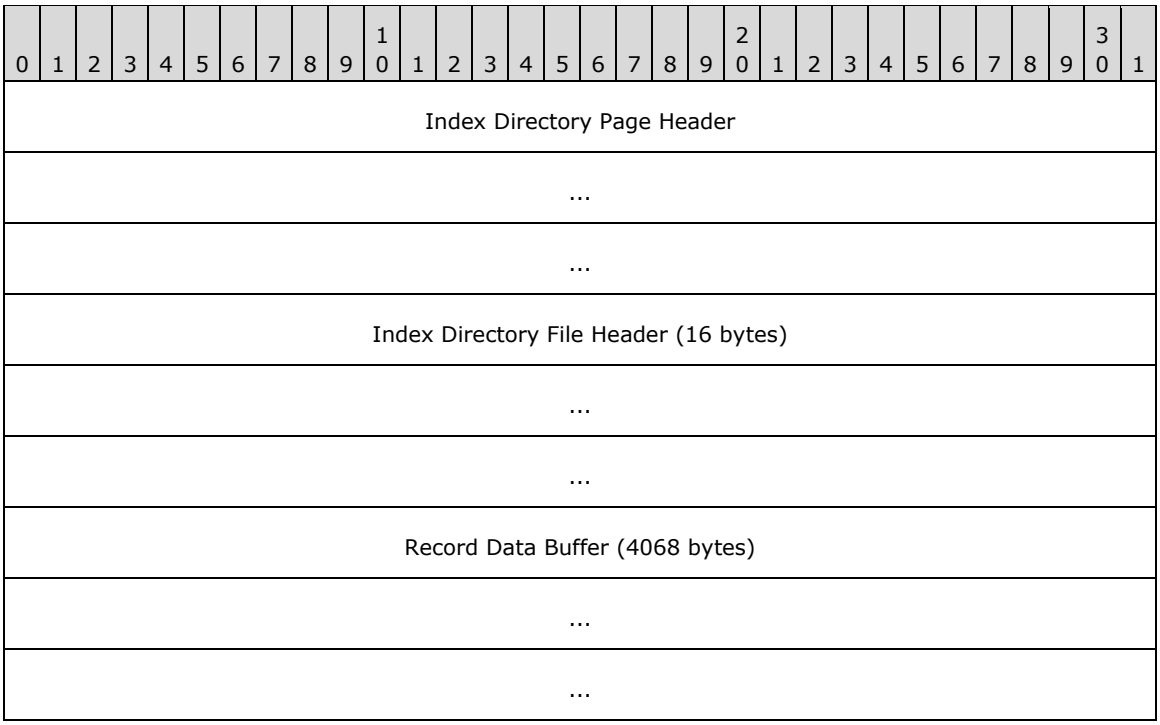
The size of each level is determined by the number of pages that are needed for storing the list of index directory records selected for that level. The list of index directory records included in a level is defined by the following rules:

- The index directory records stored in Level 1 correspond to **content index record** or scope index record from the associated index file. Level 1 MUST include one record for each **BitStream page** of the associated **content index file** or scope index file which contains the beginning of at least one of the associated records. Unless the index directory file belongs to a full-text index component that is being created by an in-progress merge process, an extra index directory record is appended which contains a **Max** key with property identifier = pidMaximum (0x7FFFFFFF).
- For all successive levels (*n)*, the total count of index directory records is equal to the number of index directory pages present in level *n-1*. Each index directory record on level *n* stores the index key of the first index directory record of the corresponding index directory page on level *n-1*.

Unless the index directory file belongs to a full-text index component that is being created by an in-progress merge process, the last level of the index directory file MUST contain only one page and MUST be the only level of the index directory file which contains only one page.

2.5.2 First Page Structure

The structure defined in this section applies only to the first page of the index directory file which is always the first page of Level 1.



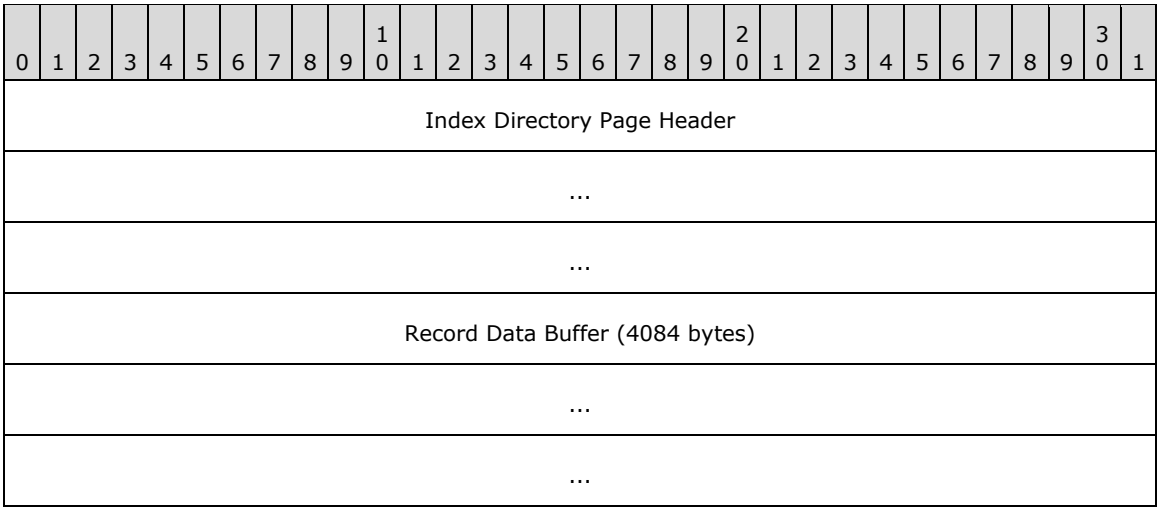
Index Directory Page Header (12 bytes): A 12-byte structure containing information that applies to the content of this page. The structure is defined in section 2.5.4.

Index Directory File Header (16 bytes): A 16-byte structure containing information that applies to the content of the entire file. The structure is defined in section 2.5.5.

Record Data Buffer (4068 bytes): A 4068-byte buffer in which the index directory records are stored. The format of this field is defined in section 2.5.6.

2.5.3 Page Structure

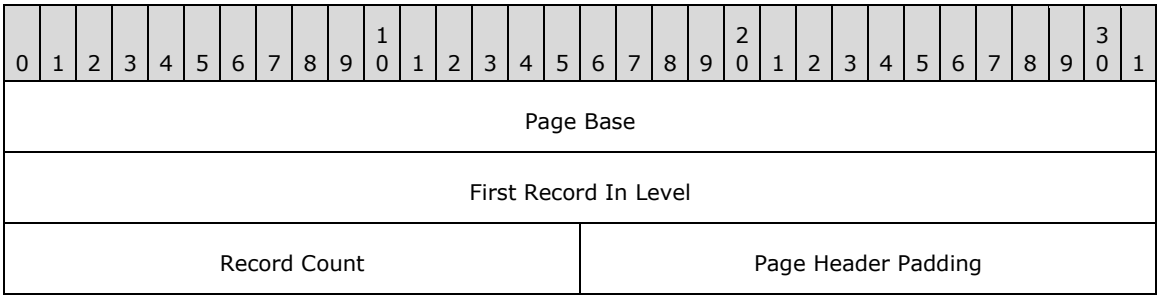
The structure defined in this section applies to all index directory pages except the first page of Level 1.



Index Directory Page Header (12 bytes): A 12-byte structure containing information that applies to the content of this page. The structure is defined in section 2.5.4.

Record Data Buffer (4084 bytes): A 4084-byte buffer in which the index directory records are stored. The format of this field is defined in section 2.5.6.

2.5.4 Page Header Structure



Page Base (4 bytes): A 32-bit unsigned integer. For the pages of level 1 this field specifies the base value that needs to be added to the **BitStreamPage** stored in each index directory record included in this page to obtain the absolute value of the **page** component of **BitStreamPosition** of the index key associated with the index directory record. If this structure is included in index directory pages of secondary levels the **Page Base** field MUST be set to the 0-based index of the first page of the previous level.

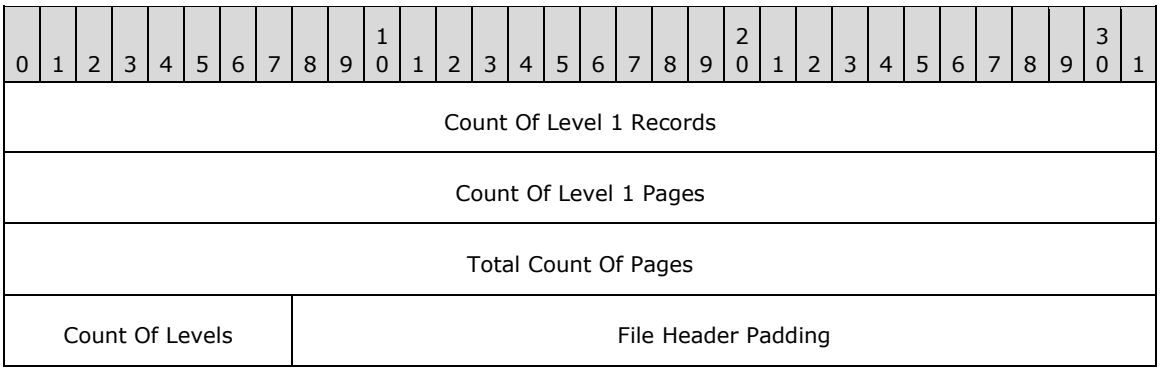
First Record In Level (4 bytes): A 32-bit unsigned integer that specifies a zero-based index of the first index directory record included in this page, relative to the beginning of the current level.

Record Count (2 bytes): A 16-bit unsigned integer that specifies the count of index directory records stored in this page.

Page Header Padding (2 bytes): The value of these 2 bytes is arbitrary and MUST be ignored.

2.5.5 File Header Structure

This structure appears in the index directory file once, in the first page, in a position subsequent to the page header.



Count Of Level 1 Records (4 bytes): The total number of records stored in Level 1 pages.

Count Of Level 1 Pages (4 bytes): The total number of Level 1 pages.

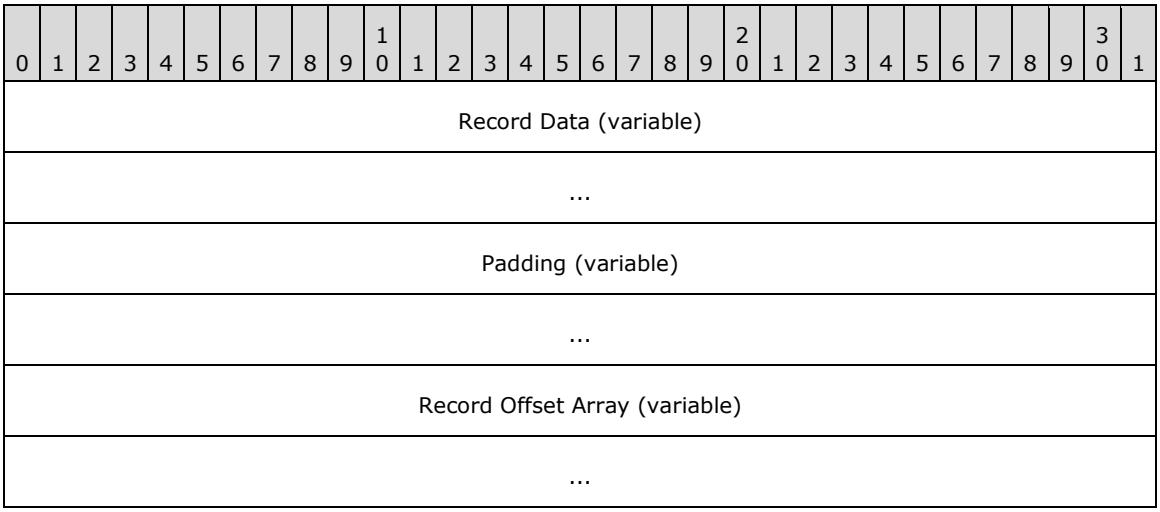
Total Count Of Pages (4 bytes): The total number of pages across all the levels.

Count Of Levels (1 bytes): The total number of index directory levels.

File Header Padding (3 bytes): The value of these 3 bytes is arbitrary, and MUST be ignored.

2.5.6 Record Buffer Structure

The structure defined in this section defines how the index directory records are organized within an index directory page. The size of this structure is 4068 bytes when it appears in the first index directory page. For all the other pages the size of this structure is 4084 bytes.



Record Data (variable): A field in which the index directory records are stored. The size of this field is determined by the maximum number of records that can be fit into the page. The index directory records are stored in this buffer sequentially, without any padding or special alignment.

Padding (variable): The value of this field is arbitrary, and MUST be ignored.

Record Offset Array (variable): An array of 16-bit unsigned integers. The number of elements is equal to the number of index directory records stored in the **Record Data** field. Each value of the array represents the offset in bytes for an index directory record stored in **Record Data**, relative to beginning of the page. The record offsets are stored in this array in reverse order, which means that the first value stored in this array corresponds to the last index directory record stored in **Record Data**. The value of the last element of this array, which corresponds to the first index directory record stored in **Record Data**, is the offset in page of the structure defined in this section minus the record buffer. Subsequently the value of the last element of **Record Offset Array** MUST be set to the following:

- 28 for the first index directory page.
- 12 for all the other pages.

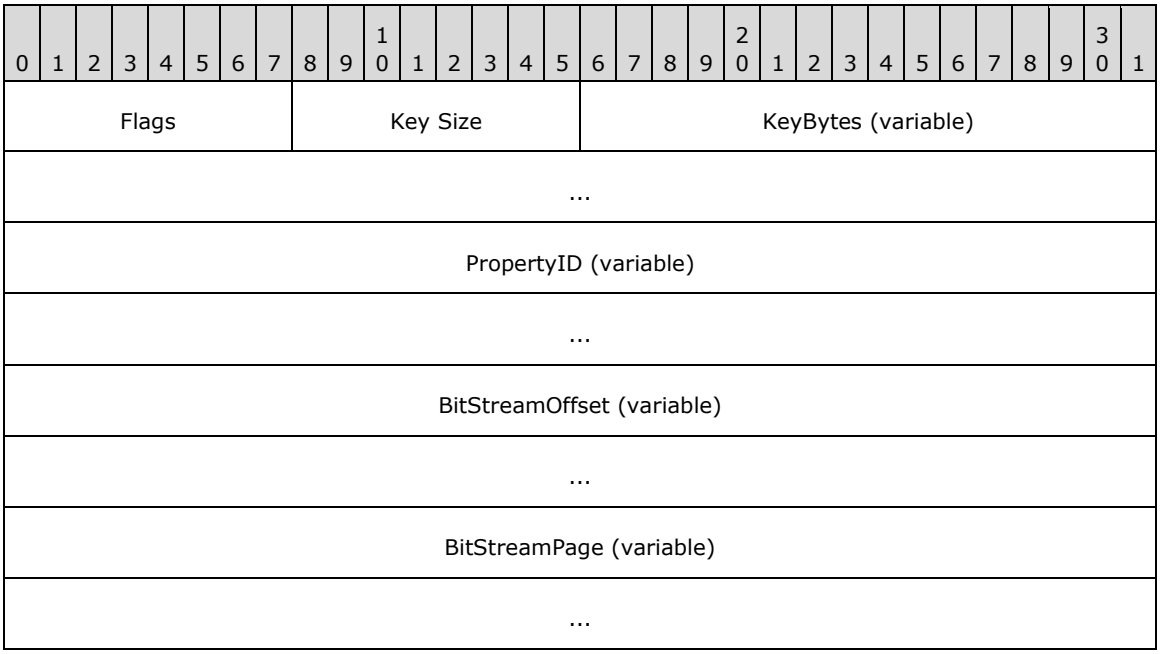
2.5.7 Record Structure

The index directory record is a variable length structure and is stored in the **Record Data** field of the 2.5.6, without any padding or record alignment.

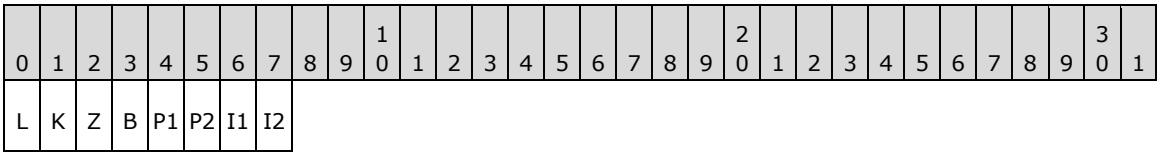
The data elements which are represented in the index directory record structure are:

- An index key (see section 2.2.3): The structure includes separate fields for representing the data components of an index key: index key string and property identifier**.** The list of index directory records for each level of the index directory files is sorted in ascending order of the index key. The details of the sort criteria are defined in section 2.2.3.

- **A key position**: A **BitStreamPosition** that points to the record which contain the same index key in the associated **content index file** or scope index file which contains the same index key. The **key position** field is stored only in the index directory records which make up the Level 1 of the index directory file.



Flags (1 byte): This field determines the size of other fields in this structure.



L (1 bit): Indicates whether the record contains the **BitStreamOffset** field and the **BitStreamPage** field. Its value MUST be zero if the record does not contain **BitStreamOffset** and **BitStreamPage** fields, and "1" otherwise.

K (1 bit): Indicates whether the **KeyBytes** field is stored in a compressed mode. Its value MUST be 0 if the index key string is stored uncompressed in the **KeyBytes** field, and 1 otherwise.

Z (1 bit): If set, it indicates that the representation of the **KeyBytes** field does not include the first byte of the index key string. The value of this byte is assumed to be 0.

B (1 bit): Selector for the size of **BitStreamOffset** field.

- 0 - **BitStreamOffset** field is stored as a 2-byte integer.
- 1 - **BitStreamOffset** field is stored as a 1-byte integer.

P1, P2 (1 bit each): 2 bits which specify the size of the **BitStreamPage** field as defined in the following table.

P1	P2	Size of BitStreamPage field
0	0	1 byte

P1	P2	Size of BitStreamPage field
0	1	2 bytes
1	0	4 bytes
1	1	Undefined. This bit combination is not valid and it MUST NOT be used.

I1, I2 (1 bit each): 2 bits that specify the size of the **PropertyID** field as defined in the following table.

I1	I2	Size of PropertyID field
0	0	1 Byte
0	1	2 Bytes
1	0	4 Bytes
1	1	0 Bytes. The PropertyID field is not present in the record.

KeySize (1 byte): A single byte unsigned integer which specifies the size of **KeyBytes** field. The value of this field MUST be less than or equal to 129.

KeyBytes (variable): Array of bytes that stores the content of index key string component of the index key**.** When possible, the representation of this field is compressed by skipping bytes with 0 values. The flags **Z** and **K** define which bytes of the index key string are skipped and assumed to be 0.

- If both **Z** and **K** are 0, KeyBytes stores integrally the index key string.
- If **Z** is 1, the first byte of the index key string is not included in **KeyBytes** field and it is assumed to be 0.
- If **K** is 1, the second byte and then every other byte of the index key string is not included in **KeyBytes** field and it is assumed to be 0. This compression method is specific to the **content index keys** which represent a token composed of only characters which belong to Unicode range 0 to 255.

Examples

Index key string	K	Z	KeyBytes stored in Index Directory Record
0x0, 0x0, 0x61, 0x0, 0x62, 0x0, 0x63	1	1	0x61, 0x62, 0x63
0x0, 0xe, 0x02, 0x0e, 0x32, 0xe, 0x27	0	1	0xe, 0x02, 0x0e, 0x32, 0xe, 0x27
0x7E, 0xFF	0	0	0x7E, 0xFF

PropertyID (variable): An unsigned integer value specifying the property identifier of the index key. The size of this field MUST be 1, 2 or 4 bytes. The actual size is determined by the value of bits **I1** and **I2** of the **Flags** field.

If both **I1** and **I2** are set to 1, this field is not present, and the value for the property identifier to be used in the index key is 4096.

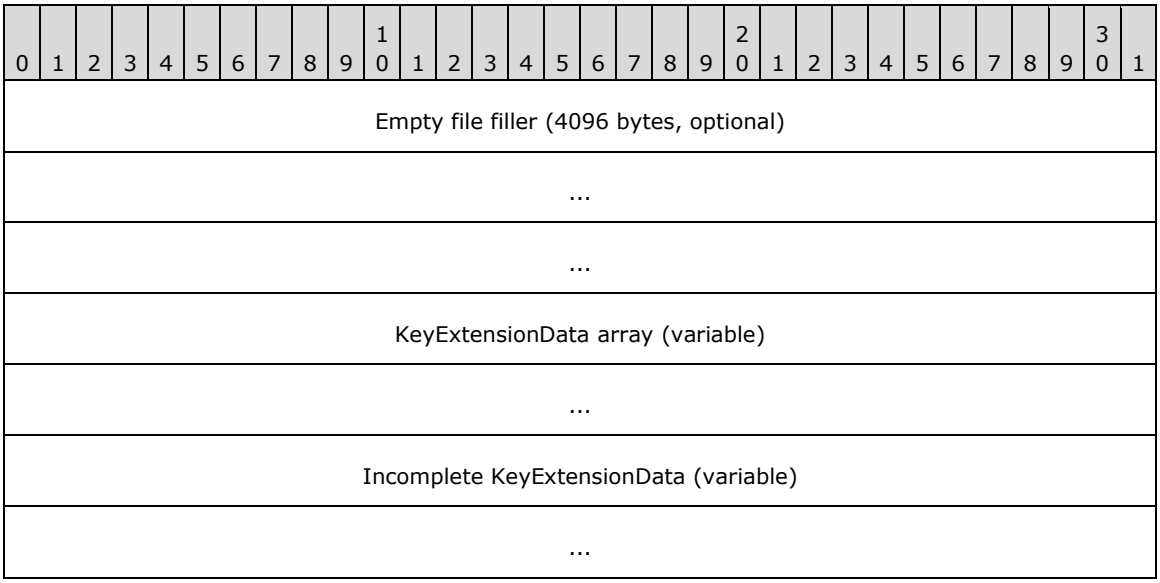
BitStreamOffset (variable): The size of this field can be either 1 or 2 bytes and is determined by the value of bit **B** of the **Flags** field. The field is an unsigned integer value representing the **Offset** part of the BitStreamPosition, which locates the index key in the associated content index file or scope index file**.**

BitStreamPage (variable): Unsigned integer field. The size of field can be 1, 2 or 4 bytes and is determined by the value of bits **P1** and **P2** of **Flags** field. The value of this field added on top of **Page Base** field stored in the current Index Directory Page Header gives the **Page** part of the BitStreamPosition which locates the index key in the associated content index file or scope index file.

2.6 Content Index Extension File Format

The **content index extension (.cix) file** format is an extension of the bitStream file format, as specified in section 2.2.1, and is used to store compressed **document identifiers** and corresponding **OccCounts** or **MaxOccBuckets** for some **content index keys**, as specified in section 2.3.1. The bit ordering for this file format is the same as described in section 2.2.1.

CIX files are used for auxiliary storage and MUST correspond to a **content index file** that stores the content index keys.



Empty file filler (4096 bytes, optional): An empty **BitStream page** with valid **start page signature** and **end page signature** fields. This field MUST exist and be ignored if the size of the **KeyExtensionData array** field is set to zero and the size of the **Incomplete KeyExtensionData** field is set to zero. This field MUST NOT exist if the size of the **KeyExtensionData array** field is greater than zero or the size of the **Incomplete KeyExtensionData** field is greater than zero.

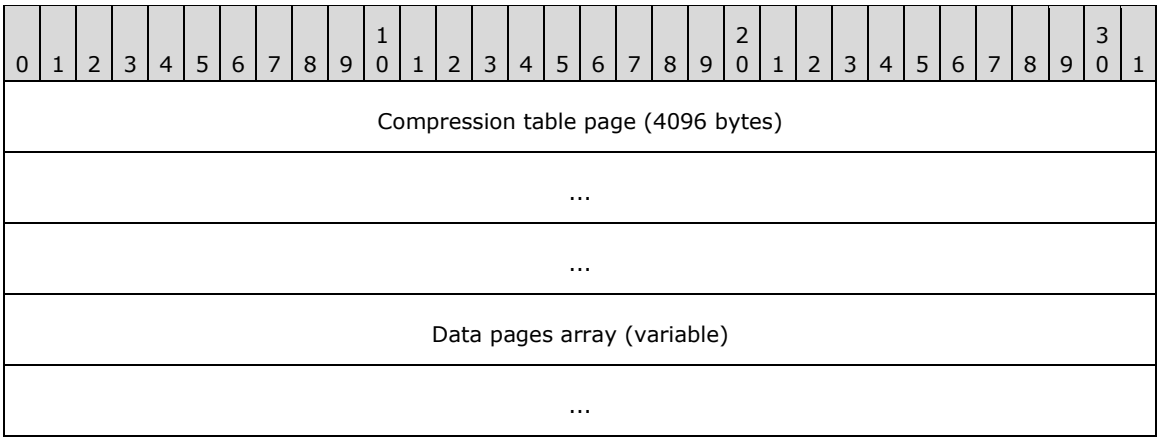
KeyExtensionData array (variable): An array of **KeyExtensionData** structure, as specified in section 2.6.1.

Incomplete KeyExtensionData (variable): MUST be ignored.

2.6.1 KeyExtensionData Structure

The **KeyExtensionData** structure stores compressed **document identifiers** and corresponding **OccCount** or **MaxOccBucket** information for one **content index key**. The **KeyExtensionData** structures corresponding to BOF keys and EOF keys store the **MaxOccBucket** for the property identifier specified by the content index key. OccCount is stored for other content index keys.

The structure MUST be aligned on a 4-kilobyte (4096-byte) page boundary.

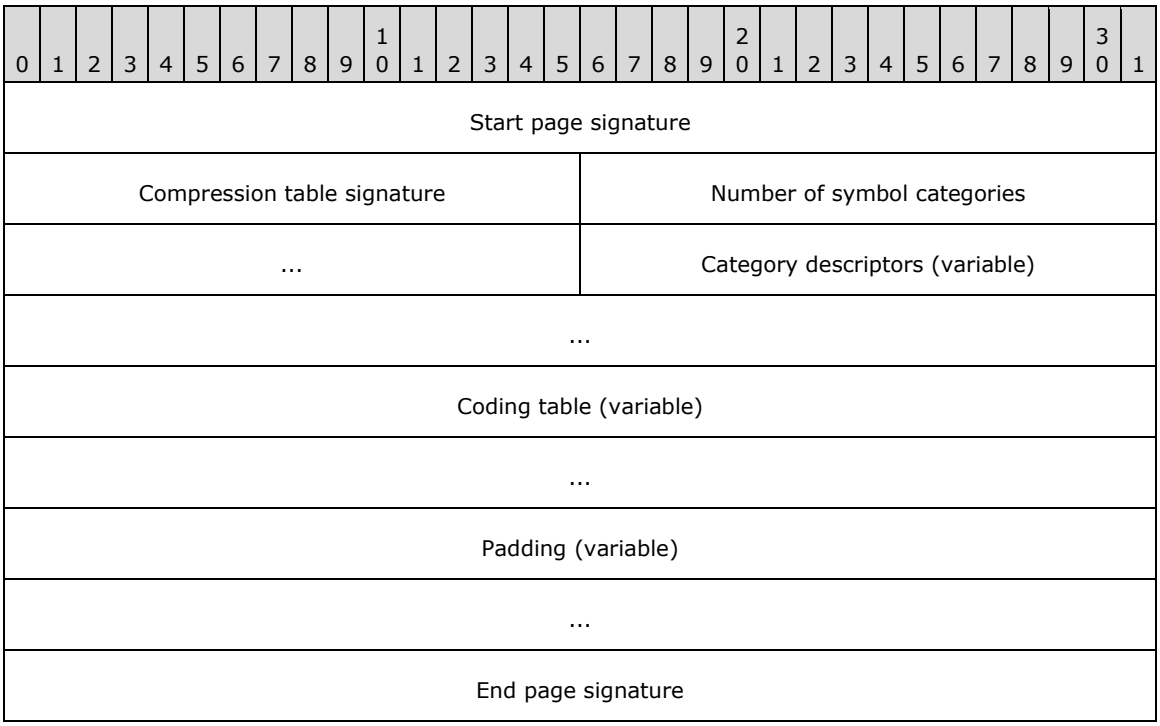


Compression table page (4096 bytes): An **ExtensionCompressionTablePage** structure, as specified in section 2.6.1.1, that contains compression parameters for the content index key.

Data pages array (variable): An array of **ExtensionDataPage** structures, as specified in section 2.6.1.2, that contain the compressed data.

2.6.1.1 ExtensionCompressionTablePage Structure

The **ExtensionCompressionTablePage** structure stores settings needed to decode the **ExtensionDataPage** structures. It MUST be aligned on a 4-kilobyte (4096-byte) page boundary.



Start page signature (4 bytes): The start page signature of the **BitStream page**.

Compression table signature (2 bytes): A 16-bit integer that MUST be equal to 0x4b52.

Number of symbol categories (4 bytes): The number of records in the **Category descriptors** field. This MUST be equal to 0x00000005.

Category descriptors (variable): An array of **SymbolCategory** structures, as specified in section 2.6.1.1.1. The number of objects in this array is specified in the N**umber of symbol categories** field.

Coding table (variable): An array of **CodingTableEntry** structures, as specified in section 2.6.1.1.2, that defines the bit sequence used for compression of **document identifiers** and **OccCounts** as specified in section 2.3.1. The number of objects in this array is the sum of values of the **Number of symbols** fields in the elements of the C**ategory descriptors** field. The coding table MUST NOT contain a bit sequence that is a prefix of another bit sequence.

Padding (variable): A field that exists to ensure that the total structure size is 4096 bytes. The value of this field is arbitrary and MUST be ignored.

End page signature (4 bytes): The **End page signature** of the BitStream page.

2.6.1.1.1 SymbolCategory Structure

Every **Symbol Category** structure defines a set of symbols. All symbols are assigned a value in order from 0 to the total number of symbols minus 1, starting from the first category and ending with the last category. In every category, the smallest symbol value is the **Base symbol value**. Therefore, the **Base symbol value** of the first category is 0, and the **Base symbol value** for every other category equals the **Base symbol value** of the previous category plus the number of symbols in the previous category.

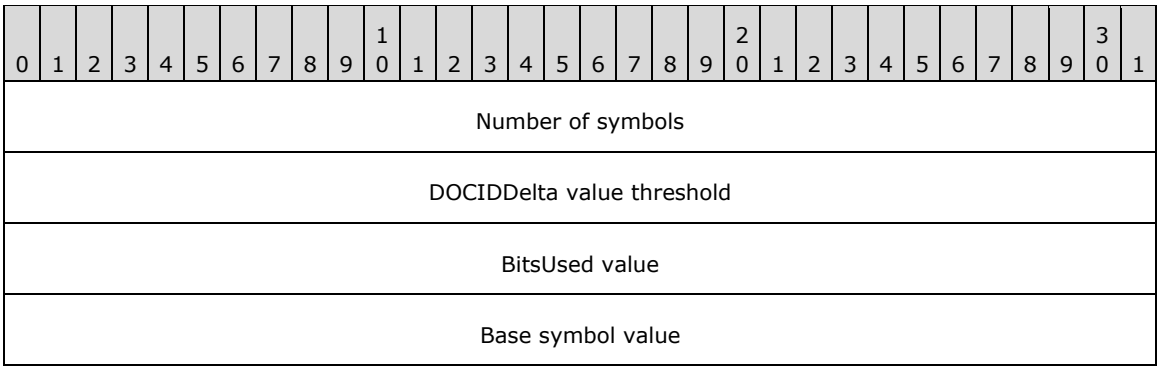
For every category, all symbols with values greater than or equal to the **Base symbol value** plus **DocIDDelta value threshold** are **category special symbols**. The **category special symbol** with the smallest value is the first special symbol.

The **Coding table** array in the ExtensionCompressionTablePage structure stores the code bit sequences for all symbols in order of increasing value.

For every item containing the **content index key** the DocIDDelta value is encoded using the defined symbols in the **DOCID bit stream** field and the corresponding **OccCount** or **MaxOccBucket** is stored in the **OccCount bit stream** array in the **ExtensionDataPage**, as specified in section 2.6.1.2. The **BitsUsed value** in the symbol category structure is the number of bits used to store the corresponding element in the **OccCount bit stream** array.

For every non-special symbol the corresponding DocIDDelta value equals the difference of the symbol value and the **Base symbol value**. For special symbols, the DocIDDelta value is stored after the symbol bit sequence in the **DOCID bit stream** field using 16 bits for the first special symbol and 32 bits for other special symbols.

The format of a **SymbolCategory** structure is as follows.



Number of symbols (4 bytes): The number of symbols in this category. This value MUST be equal to 0x00000082.

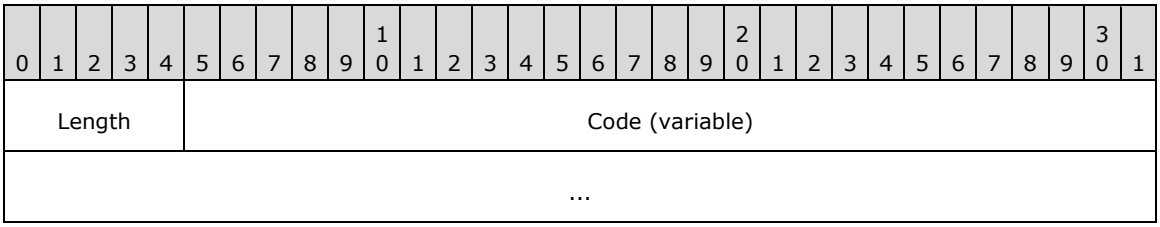
DOCIDDelta value threshold (4 bytes): DocIDDelta values greater than or equal to this threshold are replaced with a special symbol. This value MUST be equal to 0x00000080.

BitsUsed value (4 bytes): The number of bits used to record the corresponding element in the O**ccCountbit stream** of the **ExtensionDataPage**. If this value is zero, the element is not stored in the array and its value is the same as the value for the previous **document identifier**.

Base symbol value (4 bytes): The base symbol value of category. This MUST be equal to the **Base symbol value** of the previous category plus the **Number of symbols** field in the previous category (zero for the first category).

2.6.1.1.2 CodingTableEntry Structure

Each entry in the coding table has the following format.

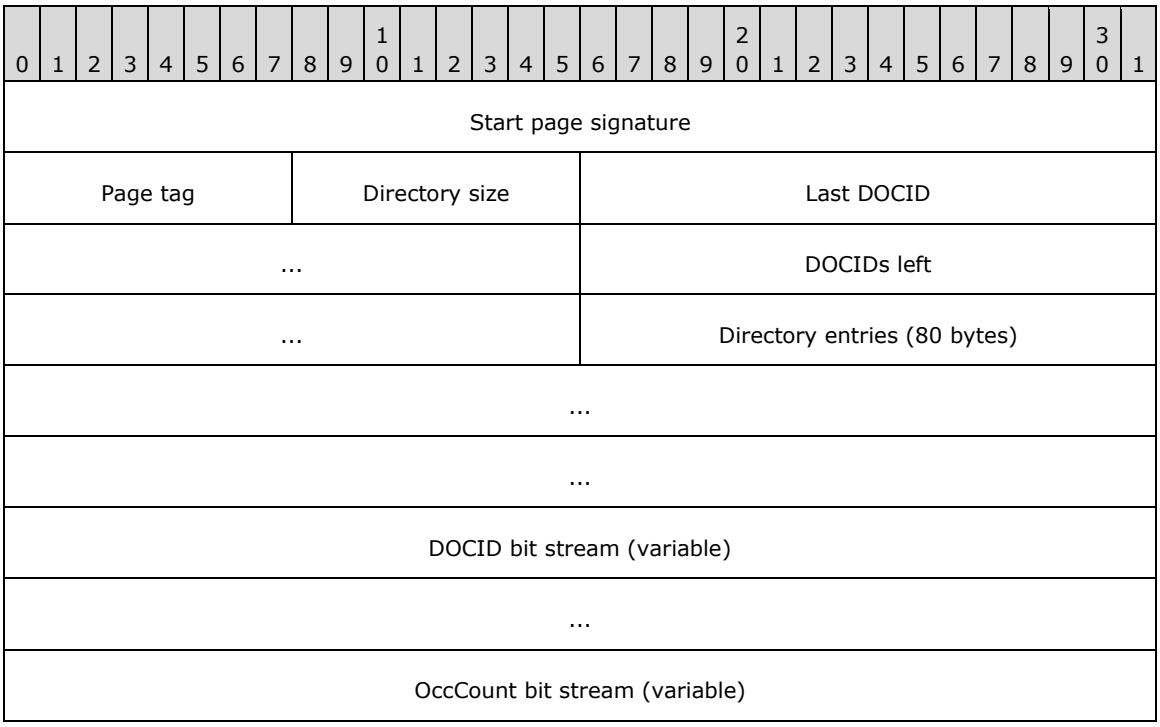


Length (5 bits): The length of **Code** field in bits.

Code (variable): Bit sequence used to compress the symbol.

2.6.1.2 ExtensionDataPage Structure

The object MUST be aligned on a 4-kilobyte (4096 bytes) page boundary.



...
Padding (variable)
...
End page signature

Start page signature (4 bytes): The **Start page signature** of **BitStream page**.

Page tag (1 byte): Last page identifier. This value MUST be equal to 0x4c for the last data page in a key, 0x50 for the remaining data pages.

Directory size (1 byte): Number of valid entries in **Directory entries** field. This MUST be less than or equal to 8 and greater than or equal to 1.

Last DOCID (4 bytes): The last **document identifier** in this page.

DOCIDs left (4 bytes): The number of document identifiers left in the key including all document identifiers in this page.

Directory entries (80 bytes): An array of 8 INTREFEENCE: (**DirectoryEntry Structure** section 2.6.1.2.1) objects storing page bookmarks.

DOCID bit stream (variable): An array of **EncodedDOCIDDelta** structure objects, as specified in section 2.6.1.2.2. The number of objects equals the number of document identifiers in the page.

OccCount bit stream (variable): An array of integer values corresponding to the document identifiers stored in this page. The size of each element of the array is defined by the corresponding **EncodedDOCIDDelta** structure object in the **DOCID bit stream** array. If the **content index key** is a BOF key or an EOF key, the values represent the **MaxOccBucket** values. If the content index key is not a BOF key and is not an EOF key, the values represent the **OccCount** values.

Padding (variable): A field that exists to ensure that the object size is 4096 bytes. The value for this field is arbitrary, and MUST be ignored.

End page signature (4 bytes): The **End page signature** of the BitStream page.

2.6.1.2.1 DirectoryEntry Structure

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
DOCID																															
cDocIDsInPage																				DocIDOffset											
occOffset																															

DOCID (4 bytes): The value of **document identifier** which the bookmark points to.

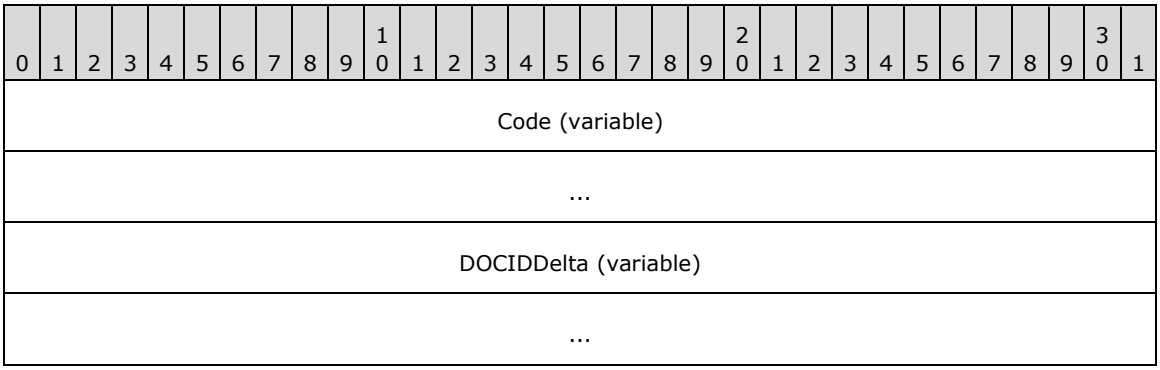
cDocIDsInPage (2 bytes): The number of document identifiers in the page, up to the **DOCID** value (not including the **DOCID** itself).

DocIDOffset (2 bytes): The offset in bits in the **ExtensionDataPage** object from the beginning of **Page tag** to the **EncodedDOCIDDelta** structure object corresponding to this document identifier.

occOffset (2 bytes): The offset in bits in the **ExtensionDataPage** object from the **Page tag** field to the element in the **OccCount bit stream** array that corresponds to this document identifier. The element for document identifiers that are pointed to by a directory entry MUST be recorded in full and MUST NOT occupy zero bits (even if the value is the same for previous document identifier).

2.6.1.2.2 EncodedDOCIDDelta Structure

The **EncodedDOCIDDelta** structure stores the encoded DocIDDelta and the number of bits used to store the corresponding element in the **OccCount bit stream** of the **ExtensionDataPage**. The value of the **Code** field corresponds to a symbol according to the **Coding table** stored in the **ExtensionCompressionTablePage**.



Code (variable): Bit sequence for symbol corresponding to the **document identifier**. The size of this field MUST equal the value of the **Length** field in the corresponding **CodingTableEntry** element of the **Coding table** array in the **ExtensionCompressionTablePage**.

DOCIDDelta (variable): Uncompressed **DocIDDelta** value for the document identifier. This field only exists if **Code** field defines a special symbol. The size of this field MUST be 2 bytes if **Code** field defines the first special symbol for the category, or 4 bytes if the **Code** field defines a special symbol other than the first special symbol.

2.7 Document Set Files

Document set files contain a list of the indexed items represented by a 32-bit **document identifier**. Each item also has freshness information; an item is marked as either fresh or outdated. An item is marked as fresh if no other **content index file** contains a more recent version of the contents of the item, and is marked as outdated otherwise.

The system uses three different file schemes to store the list of document identifiers and freshness information:

- List document set, as specified in section 2.7.1.
- Bitmap document set, as specified in section 2.7.2.
- Indexed bitmap document set, as specified in section 2.7.3.

The guidelines in the following table establish which schema to use.

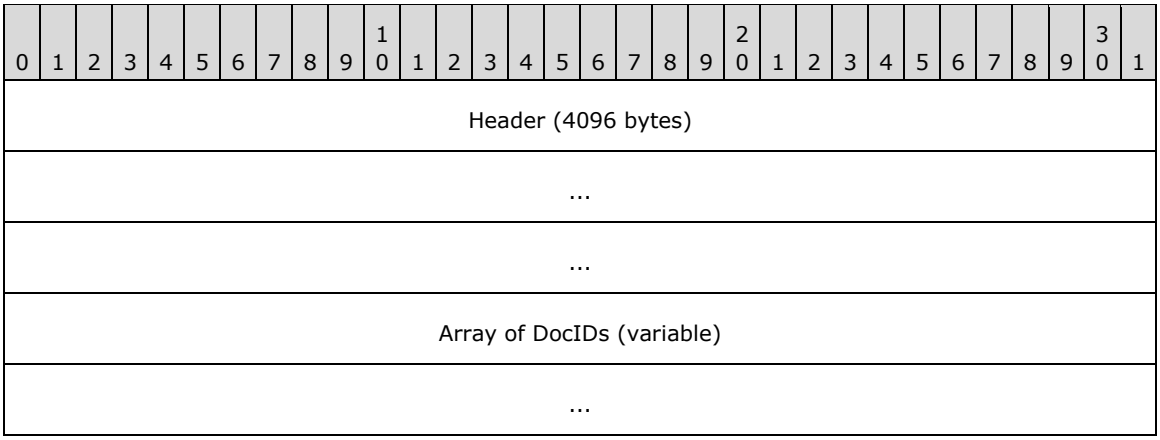
Document Set Schema	Number of DocIDs	Density
List document set scheme	Low	Low
Bitmap document scheme	Any	High
Indexed bitmap document set scheme	High	Low

Where density of the list of document identifiers is related to the maximum and minimum document identifiers. If the value of **Maximum DocIDValue- Minimum DocID Value** fields is approximately the number of document identifiers the list has high density, otherwise the list has low density.

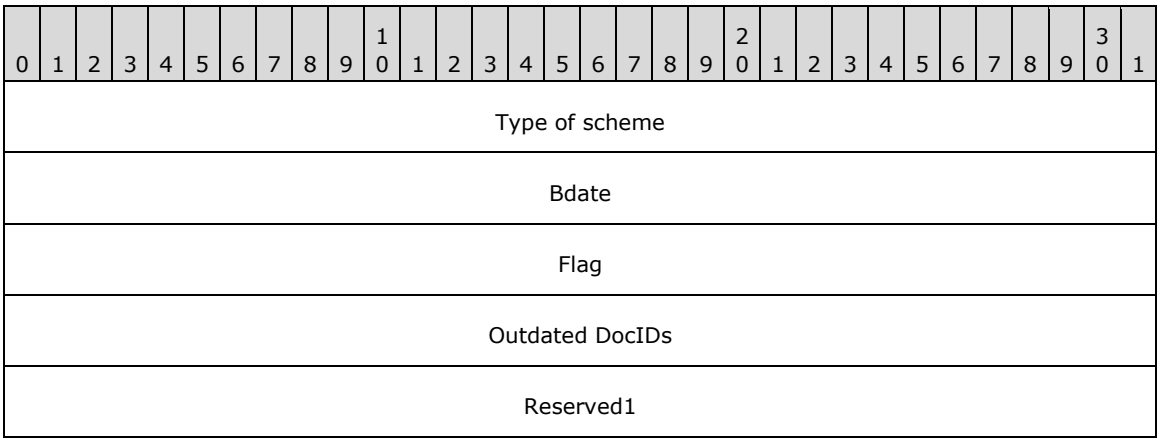
Each document set file scheme contains a file with a .wid extension, called a WID file. In addition, the indexed bitmap document set contains a file with a .wsb extension, called the WSB file.

2.7.1 List Document Set

The list document set scheme is efficient when iterative operations across **document identifiers** are necessary. In the list document set scheme the WID file contains a header and stores the document identifiers as a list. The following is a high-level representation of the format of the file.



Header (4096 bytes): The header is in the following format.



Number of Hint Pages
Hint page size
Number of DocIDs
Minimum DocID Value
Maximum DocID Value
Number of DocIDs Delta
Reserved2 (2004 bytes)
...
...
Hint Array (variable)
...
Reserved3 (variable)
...

Type of scheme (4 bytes): A 32-bit unsigned integer. Value MUST be 0x00000001.

Bdate (4 bytes): A 32-bit unsigned integer assigned during the creation of the file which is used to indicate order of file creation. The larger the number, the more recent the file.

Flag (4 bytes): A 32-bit unsigned integer. The most significant bit of this integer MUST be set to zero if all instances of the items in the file are outdated in all older files (that is, all files with a lower **Bdate** field). Otherwise, the most significant bit of the integer MUST be set to 1. Other bits MUST be ignored.

Outdated DocIDs **(4 bytes):** A 32-bit unsigned integer which represents the count of outdated document identifiers (1) in the file. This integer is used for estimation purposes to determine the efficient document identifiers (1) representation format during further merges. This value SHOULD be within 10% of the correct value. If the integer is not within this range, performance could be affected.

Reserved1 (4 bytes): The value of these 4 bytes is arbitrary, and MUST be ignored.

Number of Hint Pages (4 bytes): A 32-bit unsigned integer which determines how many entries are in the Hint Array. The value MUST NOT exceed 512. A value of zero indicates there are not enough document identifiers (1) to warrant this optimization.

Hint page size (4 bytes): A 32-bit unsigned integer. Number of document identifiers (1) in each Hint Page. The size of the last hint page is variable. A value of zero indicates there are not enough document identifiers (1) to warrant this optimization.

Number of DocIDs (4 bytes): A 32-bit unsigned integer which is the total number of document identifiers (1) stored in the file.

Minimum DocID Value, Maximum DocID Value (4 bytes each): Two 32-bit unsigned integers. Recorded at the time of file creation, no updates, used to check the density of the list of document identifiers.

Number of DocIDs Delta (4 bytes): A 32-bit unsigned integer which is the number of outdated DocIDs at the moment of file creation.

Reserved2 (2004 bytes): The value of these 2,004 bytes is arbitrary, and MUST be ignored.

Hint Array (variable): An array of 32-bit integers. Contains the first document identifier for every hint page. The most significant bit of each document identifier in the array MUST be set to 1 if any of the document identifiers on the corresponding hint page are outdated.

Hint Pages is a structural concept that is used to organize document identifiers (1) in the file. The array of document identifiers (1) stored in the file is split into hint pages. The first document identifier (1) of each page is used as a marker of the entire hint page.

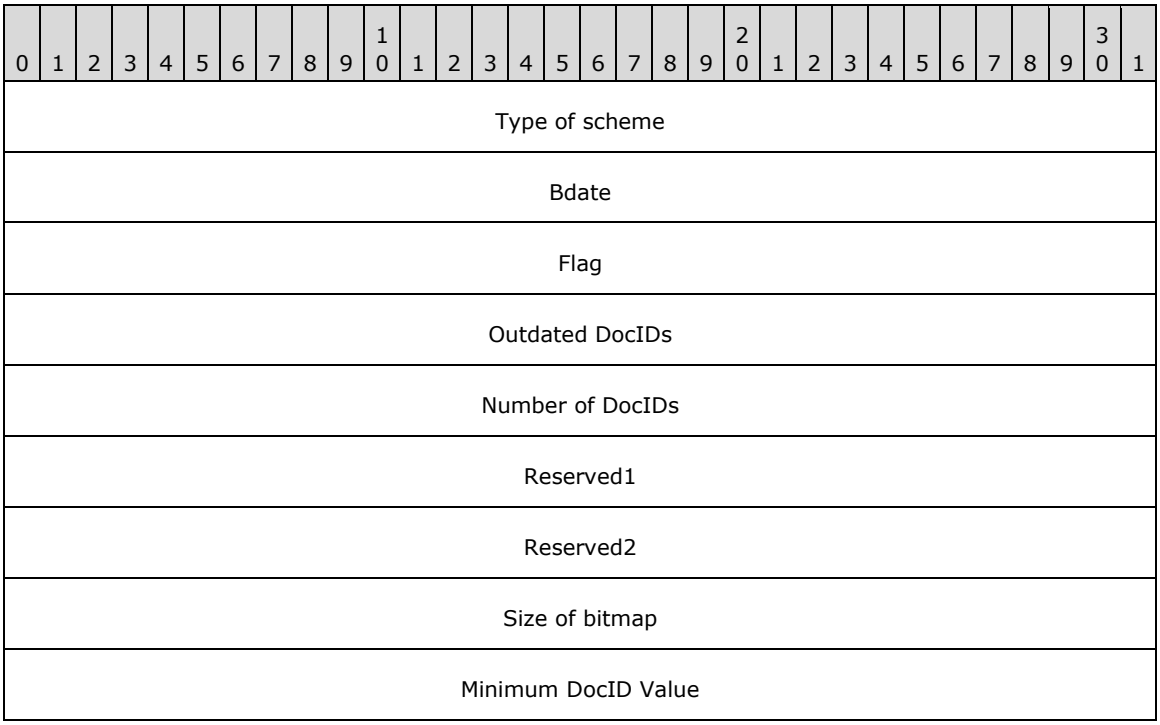
Reserved3 (variable): The value of this field is arbitrary, and MUST be ignored.

Array of DocIDs (variable): Array of 32-bit integers. The list of document identifiers (1) sorted by increasing value. Each document identifier (1) has a size of 4 bytes. The most significant bit is set to 1 if the item is outdated, and set to 0 if the item is fresh.

2.7.2 Bitmap Document Set

In bitmap document set scheme the WID file contains a header and stores the freshness information about the items as a plain bitmap.

The following is a high-level representation of the format of the file.



Maximum DocID Value
Number of DocIDs Delta
Reserved3 (4052 bytes)
...
...
Bitmap (variable)
...

Type of scheme (4 bytes): A 32-bit unsigned integer. Value MUST be 0x00000003.

Bdate (4 bytes): A 32-bit unsigned integer assigned during the creation of the file which is used to indicate order of file creation. The larger the number, the more recent the file.

Flag (4 bytes): A 32-bit unsigned integer. The most significant bit of this integer MUST be set to zero if all instances of the items in the file are outdated in all older files (that is, all files with a lower **Bdate** field). Otherwise, the most significant bit of the integer MUST be set to 1. Other bits MUST be ignored.

Outdated DocIDs (4 bytes): A 32-bit unsigned integer which represents the count of outdated **document identifiers** in the file. This integer is used for estimation purposes to determine the efficient document identifiers representation format during further merges. This value SHOULD be within 10% of the correct value. If the integer is not within this range, performance could be affected.

Number of DocIDs (4 bytes): A 32-bit unsigned integer which is the total number of document identifiers stored in the file.

Reserved1 (4 bytes): The value of these 4 bytes is arbitrary, and MUST be ignored.

Reserved2 (4 bytes): MUST be 0.

Size of bitmap (4 bytes): A 32-bit unsigned integer. Size of bitmap in bytes divided by 4. The field is used to calculate range of documents which can be stored in the map, which is **Minimum Doc ID** field to `Minimum Doc ID field + Size of bitmap field * 4 * 8`.

Minimum DocID Value, Maximum DocID Value (4 bytes each): Two 32-bit unsigned integers. Recorded at the time of file creation, no updates, used to check the density of the list of document identifiers.

Number of DocIDs Delta (4 bytes): A 32-bit unsigned integer which is the number of outdated DocIDs at the moment of file creation.

Reserved3 (4052 bytes): MUST be ignored.

Bitmap (Size of bitmap times 4 bytes): The following table shows the format of the bitmap, which stores the freshness information about the items.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Array of Masks (Size of bitmap * 4 bytes)																															
...																															

To store freshness information in the bitmap file the normalized document identifiers are used; that is, the **Minimum DocID** field rounded down to the nearest multiple of 32 is subtracted from each document identifier. Each normalized document identifier is split into two parts. The value of the 27 most significant bits corresponds to the mask number. The value of the 5 least significant bits of each document identifier, shifted left 1 bit in this 32-bit mask, defines the bit which is used to store the freshness information for the specific document identifier.

If an item is not in the full-text index catalog or is outdated the corresponding bit in the mask MUST be 0. If the item is fresh then the corresponding bit in the mask MUST be 1.

2.7.3 Indexed Bitmap Document Set

In indexed bitmap document set scheme the WID file contains a header. The freshness information about the items is stored in a WID and WSB file. The corresponding WID file and WSB file have the same name.

The following is a high level representation of the format of the WID file.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type of scheme																															
Bdate																															
Flag																															
Outdated DocIDs																															
Number of DocIDs																															
Reserved1																															
Reserved2																															
SizeOfH1																															
Maximum DocID Value																															
Minimum DocID Value																															
Number of DocIDs Delta																															
Reserved3 (4052 bytes)																															

...
...
H1 (SizeOfH1 field value *4 bytes)
...

Type of scheme (4 bytes): A 32-bit unsigned integer. The value MUST be 0x00000002.

Bdate (4 bytes): A 32-bit unsigned integer assigned during the creation of the file which is used to indicate order of file creation, the bigger the number the more recent the file.

Flag (4 bytes): A 32-bit unsigned integer. The most significant bit of this integer MUST be set to zero if all instances of the items in the file are outdated in all older files (that is, all files with a lower **Bdate** field). Otherwise, the most significant bit of the integer MUST be set to 1. Other bits MUST be ignored.

Outdated DocIDs (4 bytes): A 32-bit unsigned integer representing approximate count of outdated **document identifiers** in the file. This integer is used for estimation purposes to determine the efficient document identifiers representation format during further merges. This value SHOULD be within 10% of the correct value. If the integer is not within this range, performance could be reduced.

Number of DocIDs (4 bytes): A 32-bit unsigned integer which is the total number of document identifiers stored in the file.

Reserved1 (4 bytes): The value of these 4 bytes is arbitrary, and MUST be ignored.

Reserved2 (4 bytes): A 32-bit unsigned integer. MUST be 0 for this file type.

SizeOfH1 (4 bytes): A 32-bit unsigned integer. This value is the size in bytes of **H1** divided by 4.

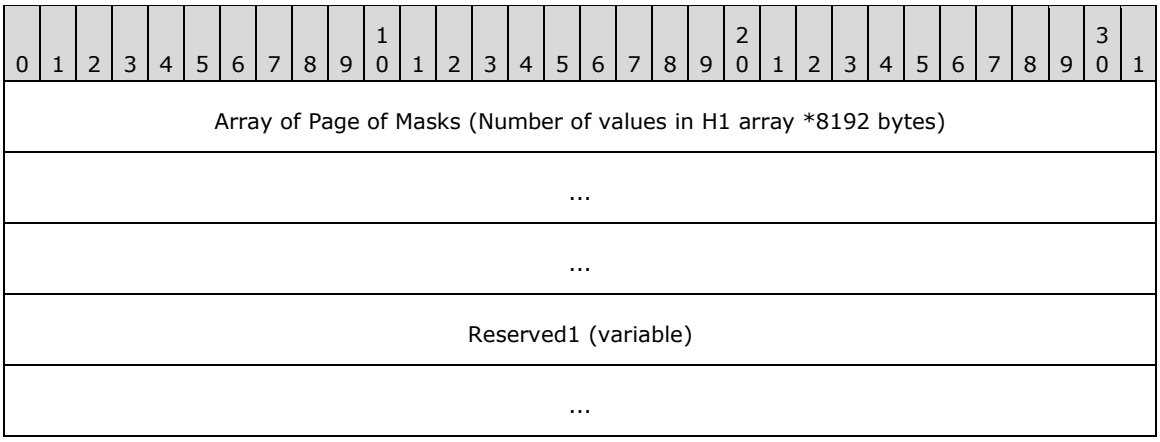
Maximum DocIDValue, Minimum DocID Value (4 bytes each): Two 32-bit unsigned integers. Recorded at the time of file creation, no updates, used to check the density of the list of document identifiers.

Number of DocIDs Delta (4 bytes): A 32-bit unsigned integer which is the number of outdated DocIDs at the moment of file creation.

Reserved3 (4052 bytes): The value of these 4052 bytes is arbitrary, and MUST be ignored.

H1 (SizeOfH1 field value times 4 bytes): Array of 16-bit values, in ascending order. Each entry corresponds to the value of the 16 most significant bits of the document identifiers. There MUST NOT be duplicates. Each entry refers to a Page of Masks in the corresponding WSB file. The most significant bit of the corresponding entry in this array is set to 1 if any of the document identifiers on the corresponding bitmap page are outdated. If there are an odd number of values in the **H1** array, the last 16 most significant bits in the array is 0.

The WSB file stores an array of 8-kilobyte blocks. Each block stores freshness information for 65,536 items identified as successive document identifiers. Each 8-kilobyte block in the WSB file is a page of masks. The index of the 16 most significant bits of a document identifier in H1 equals the index of page of masks in the WSB file.



Array of Page of Masks (Number of values in H1 array times 8192 bytes): The 16 least significant bits of each document identifier is split in two parts and used to identify the bit which stores the freshness information of the item. The value of the 11 most significant bits of 16 least significant bits corresponds to the mask number. The value of the 5 least significant bits of each document identifier corresponds to the position in this 32-bit mask. A zero in a mask indicates no item or an outdated item.

Reserved1 (variable): MUST be set to zero and ignored.

The size of a WSB file is always a multiple of 64 kilobytes.

2.8 Average Document Length File Format

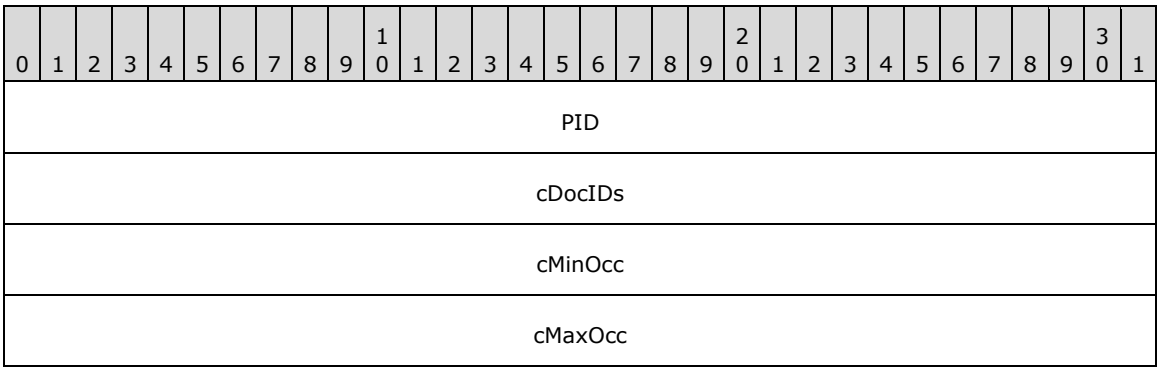
The average document length file format is an extension of the CheckSummed Recoverable Storage file format, as specified in section 2.2.5, and is used to store statistics for Properties of a set of items. The average document length file format uses fixed-sized CAVDLItem structures, as specified in section 2.8.1, as **Data Fields** in the **CheckSummedRecord** of the **CheckSummed** recoverable storage file format.

A file that implements the average document length file format MUST contain one CAVDLItem structure for every unique property encountered in the set of items.

This file format is used for AVDL files and AVDL backup files.

2.8.1 CAVDLItem Structure

The CAVDLItem structure stores statistics for a single property.



cAvgOcc
Padding
cOcc
...
cTerms
...

PID (4 bytes): A 32-bit unsigned integer that specifies the property identifier of the property whose statistics are enumerated in this structure.

cDocIDs (4 bytes): A 32-bit unsigned integer that specifies the number of items that contain the property.

cMinOcc (4 bytes): A 32-bit unsigned integer that specifies the lowest number of tokens in the property value across all items that contain the property.

cMaxOcc (4 bytes): A 32-bit unsigned integer that specifies the highest number of tokens in the property value across all items that contain the property.

cAvgOcc (4 bytes): A 32-bit unsigned integer that specifies the average number of tokens (rounded down) in the property value across all items that contain the property.

Padding (4 bytes): The value for these 4 bytes is arbitrary, and MUST be ignored.

cOcc (8 bytes): A 64-bit unsigned integer that specifies the total number of tokens in the property values across all items.

cTerms (8 bytes): A 64-bit unsigned integer that specifies the number of distinct tokens in the property values across all items.

2.9 Merge Process

A merge process combines data from several source **full-text index components** into one target full-text index component. There are two types of merge processes: shadow merge process and master merge process.

The result of a shadow merge process is a shadow index component. The result of a master merge process is a new **master index component** and its corresponding AVDL backup file.

If a master merge process is in progress and there is a master index component in the catalog, it MUST be one of the source full-text index components for the master merge. If a master index component participates in a merge process, it MUST be a master merge process.

2.10 Merge Log File Format

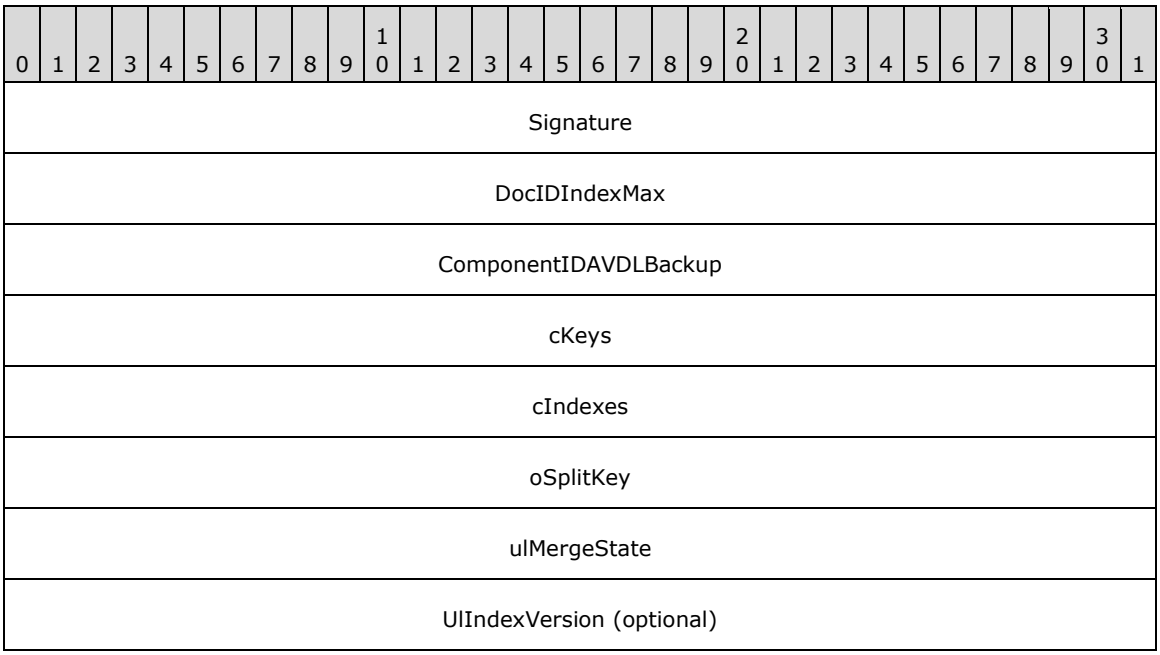
The merge log file format is an extension of the recoverable storage file format, as specified in section 2.2.4, and is used to store merge process information, as specified in section 2.9.

A file that implements the merge log file format identifies the type of the merge process and the **full-text index components** participating in the merge process: a master merge process creates and

uses a master merge log file, and a shadow merge process creates and uses a shadow merge log file. It also identifies the AVDL files participating in the merge, if any.

2.10.1 User Header Format

A file implementing the merge log file format stores information in the **First data file user header** field and the **Second data file user header** field in the recoverable storage header file, as specified in section 2.2.4.1. The structure of that information is as follows.



Signature (4 bytes): A 32-bit unsigned integer that specifies the signature of the merge log user header. This MUST be 0x44484c4d.

DocIDIndexMax (4 bytes): A 32-bit unsigned integer that specifies the MaxDocID value for target **full-text index component**.

ComponentIDAVDLBackup (4 bytes): A 32-bit unsigned integer that specifies the **ComponentID** of AVDL backup file that stores statistics for Properties of the items in the target **content index file**. Statistics kept in AVDL backup file cover data up to the **split key** identified by the **split key descriptor** field in the merge log data file.

cKeys (4 bytes): A 32-bit unsigned integer that specifies the number of **content index keys** present in the target content index file up to the split key identified by the **Split key descriptor** field in the merge log data file.

cIndexes (4 bytes): A 32-bit unsigned integer that specifies the number of source full-text index components participating in this merge.

oSplitKey (4 bytes): A 32-bit unsigned integer that specifies value of the offset (in bytes) of the **Split key descriptor** field from the beginning of the **Merge log signature** field in the merge log data file.

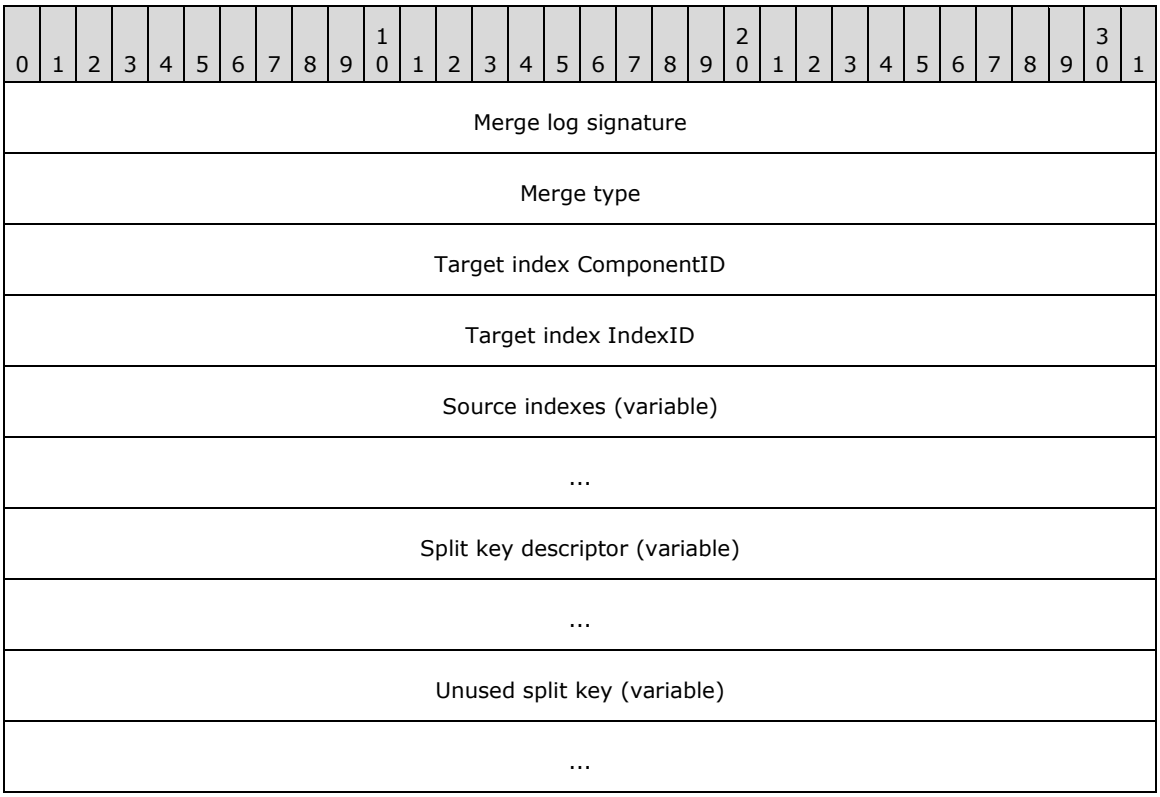
ulMergeState (4 bytes): A 32-bit unsigned integer that specifies the stage of merge. It MUST be one of the values from the following table.

Value	Description
0x00000000	Document set files merge in progress.
0x00000001	Document set files merge is complete.
0x00000002	Content index files merge in progress.

ulIndexVersion (4 bytes, optional): A 32-bit unsigned integer whose 2 higher bytes specify the **format version** of the target full-text index component. This MUST be 0x00520000 or 0x00530000 or 0x00540000. This field is only present when the value of the **Merge log signature** field in the merge log data file is "Extended shadow merge log file" or "Extended master merge log file". If this field is missing, the **format version** of the target full-text index component is 0x0052.<25>

2.10.2 File Content

Every field is a record in the recoverable storage file format, as specified in section 2.2.4, except for the **source indexes**, which is an array whose members are individual records.



Merge log signature (4 bytes): A 32-bit unsigned integer that identifies the type of merge log file. This MUST be one of the values from the following table.

Value	Description
0x474C4D53	Shadow merge log file
0x474C4D4D	Master merge log file
0x4C4D5356<26>	Extended shadow merge log file

Value	Description
0x4C4D4D56<27>	Extended master merge log file

Merge type (4 bytes): A 32-bit unsigned integer that identifies the type of merge. This MUST be one of the values from the following table.

Value	Description
0x00000002	Shadow merge
0x00000003	Master merge

The value of the **Merge type** field MUST be "Shadow merge" if the value of the **Merge log signature** field is "Shadow merge log file" and MUST be "Master merge" if the value of the **Merge log signature** field is "Master merge log file".

The value of the **Merge type** field MUST be "Shadow merge" if the value of the **Merge log signature** field is "Extended shadow merge log file" and MUST be "Master merge" if the value of the **Merge log signature** field is "Extended master merge log file".

Target index ComponentID (4 bytes): A 32-bit unsigned integer that MUST be equal to the value of **Target index IndexID** field.

Target index IndexID (4 bytes): A 32-bit unsigned integer that specifies the **index identifier** of the target **full-text index component**.

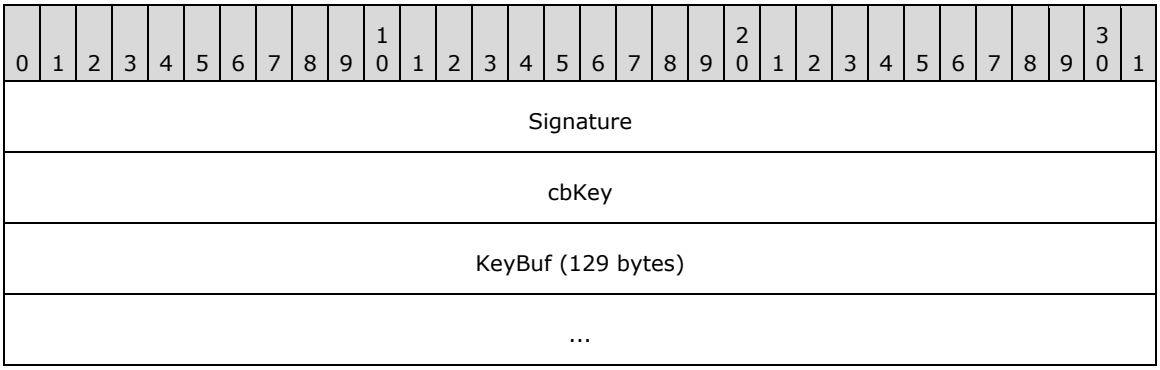
Source indexes (variable): An array of 32-bit unsigned integers that specify the index identifiers of source full-text index components. Every index identifier is counted as a separate record in the recoverable storage data file. The number of **source indexes** is specified in the **cIndexes** field of the merge log user header, as specified in section 2.10.1.

Split key descriptor (variable): A **CMergeSplitKey** structure, as specified in section 2.10.3. that stores information about progress of the merge process, as specified in section 2.9.

Unused split key (variable): A **CMergeSplitKey** structure that MUST be ignored. It MUST only be present when the value of the **Merge type** field is "Master merge".

2.10.3 CMergeSplitKey Structure

This structure identifies the **content index key** whose data was fully written in the target **content index files** and its location in the target content index file. This content index key, subsequently referred to as the split key, is an indication of the merge progress during the third merge phase (**Content Index merge in progress**).



...	
Padding	PID
...	Start page
...	Start offset
...	End page
...	End offset
...	Extension end page
...	Extension end offset
...	

Signature (4 bytes): A 32-bit unsigned integer that specifies the signature of this structure. This MUST be 0x4b53474d.

cbKey (4 bytes): A 32-bit unsigned integer that specifies the number of bytes in use in the **KeyBuf** buffer.

KeyBuf (129 bytes): A 129 byte buffer that contains the text of the split key. Unused bytes MUST be ignored (number of bytes in use is specified in the **cbKey** field).

Padding (3 bytes): A 24-bit field used to align the **PID** field to a 32-bit boundary. The value of these 3 bytes is arbitrary, and MUST be ignored.

PID (4 bytes): A 32-bit unsigned integer that specifies the property identifier of the split key.

Start page (4 bytes): The **Page** part of the **BitStreamPosition** that points to the beginning of the **content index record** of the split key in the target content index file.

Start offset (4 bytes): The **Offset** part of the BitStreamPosition that points to the beginning of the content index record of the split key in the target content index file.

End page (4 bytes): The **Page** part of the BitStreamPosition that points to the end of the content index record of the split key in the target content index file.

End offset (4 bytes): The **Offset** part of the BitStreamPosition that points to the first bit after the end of the content index record of the split key in the target content index file.

Extension end page (4 bytes, optional): The **Page** part of the BitStreamPosition that points to the first bit after the end of the **KeyExtensionData** field of the split key in the target CIX file. This field MUST be present only when the **format version** of the target **full-text index component** is greater than or equal to 0x0053.<28>

Extension end offset (4 bytes, optional): The **Offset** part of the BitStreamPosition that points to the first bit after the end of the **KeyExtensionData** field of the split key in the target CIX file. This field MUST be present only when the **format version** of the target full-text index component is greater than or equal to 0x0053.<29>

2.11 Query-Independent Rank Files

Query-Independent Rank files use the sparse array file format to store a float value for each **document identifier** in the index. This value is combined with query-dependent data to compute the rank of the item for each search query.

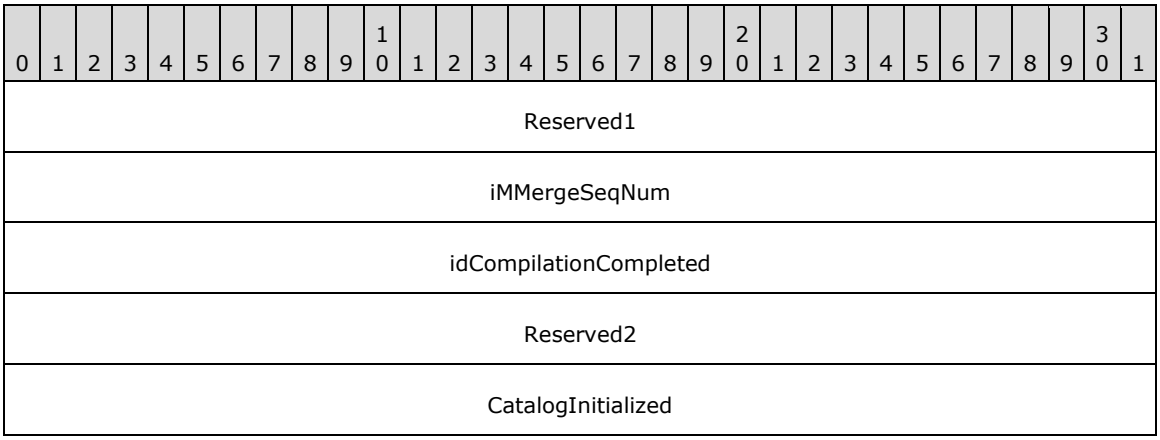
2.12 Detected Language Files

Detected language files use the sparse array file format to store, for each **document identifier** in the index, the identifier of the language of the corresponding item, as detected by the index server.

2.13 Index Table File Format

The index table file format is an extension of the CheckSummed Recoverable Storage file format, as specified in section 2.2.5. For every group of files in a full-text index catalog, a fixed-sized **CIndexRecord** is stored in the data files of a file implementing the index table file format. The user header fields of recoverable storage header file, as specified in section 2.2.4.1, are used to store values that apply to the entire full-text index catalog.

2.13.1 User Header



Reserved1 (4 bytes): The value of these 4 bytes is arbitrary, and MUST be ignored.

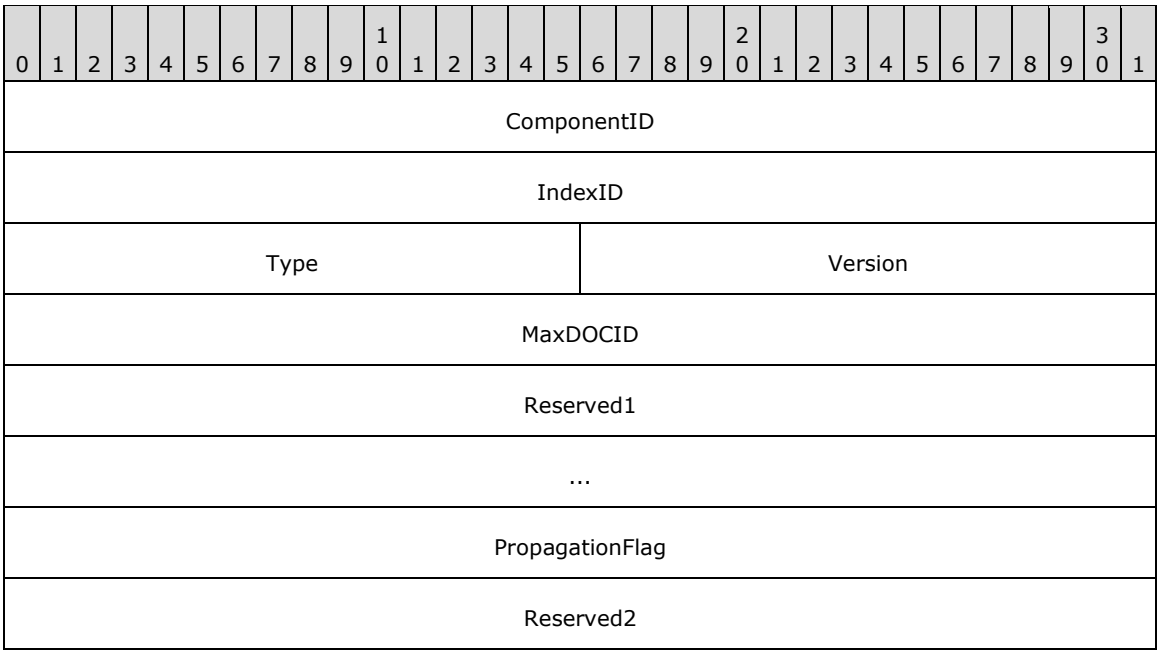
iMMergeSeqNum (4 bytes): A 32-bit unsigned integer that specifies the number of master merges processes that have occurred on the full-text index catalog.

idCompilationCompleted (4 bytes): A 32-bit unsigned integer that specifies the current **search scope compilation identifier**. Every **full-text index component** MUST have a compound scope index with this search scope compilation identifier.

Reserved2 (4 bytes): The value of these 4 bytes is arbitrary, and MUST be ignored.

CatalogInitialized (4 bytes): A 32-bit unsigned integer that specifies whether the **index table file** was initialized and contains correct data. This MUST be 0x00000000 if the index table file is new and empty and 0x00000001 if index table file was already initialized.

2.13.2 CIndexRecord



ComponentID (4 bytes): A 32-bit unsigned integer whose value depends on the **Type** field and is described in section 2.13.3.

IndexID (4 bytes): A 32-bit unsigned integer whose value depends on the **Type** field and is described in section 2.13.3. Every record that describes a **full-text index component** MUST have an **IndexID** field whose value is unique across all records describing full-text index component.

Type (2 bytes): A 16-bit unsigned integer that specifies the type of this record. This value MUST be in the **IndexType** enumeration, as defined in section 2.13.3.

Version (2 bytes): A 16-bit unsigned integer that specifies the **format version** of files described by the **CIndexRecord**. This MUST be 0x0052 or 0x0053 or 0x0054.<30>

MaxDOCID (4 bytes): A 32-bit unsigned integer whose value depends on the **Type** field and is described in section 2.13.3.

Reserved1 (8 bytes): The value of these 8 bytes is arbitrary, and MUST be ignored.

PropagationFlag (4 bytes): This MUST be 0x00000000 or 0x00008000, and MUST be ignored.

Reserved2 (4 bytes): The value of these 4 bytes is arbitrary, and MUST be ignored.

2.13.3 IndexType Enumeration

```
enum IndexType
{
  itMaster         = 0x0000,
  itShadow         = 0x0001,
  itZombie         = 0x0002,
  itDeleted        = 0x0003,
  itPartition      = 0x0004,
  itKeyList        = 0x0005,
  itNewMaster      = 0x0006,
  itAvdlLog        = 0x0007,
  itAvdlLogBackup1 = 0x0009,
  itAvdlLogBackup2 = 0x000A,
  itShadowMergeLog = 0x000B,
  itMasterMergeLog = 0x000C
```

```
};
```

itMaster: The **CIndexRecord** describes a **master index component**. The value of the **ComponentID** field MUST be equal to the value of the **IndexID** field. The **IndexID** field specifies the index identifier of the full-text index component and MUST be greater than or equal to 0x00010001, and less than or equal to 0x000100ff. **MaxDOCID** field specifies the **MaxDocID** value of the full-text index component. There MUST NOT be more than one **CIndexRecord** of this type in the index table file.

itShadow: The **CIndexRecord** describes a shadow full-text index component. It has the same restrictions as a master index component except that the number of such **CIndexRecords** in the index table file is not limited.

itZombie: The **CIndexRecord** describes a full-text index component that SHOULD be ignored and deleted. It has the same restrictions as a shadow index component. This type indicates that the content of such a full-text index component was merged into other files.

itDeleted: The **CIndexRecord** was deleted and can be reused. The value of the **IndexID** field MUST be 0xffff0000. There are no files on disk associated with this CIndexRecord.

itPartition: A special reserved type of **CIndexRecord**. The values of the **ComponentID**, **IndexID** and **MaxDocID** fields of this record MUST be 0x00000000, 0x00010000 and 0x00000000 respectively. There MUST be one record of this type in the index table file. There are no files on disk represented by this record.

itKeyList: The **CIndexRecord** is used to store the number of **content index keys** in the current master index component. The values of the **ComponentID** and **IndexID** fields of this record MUST be 0x00000001 and 0xfffe0001 respectively. **MaxDocID** field specifies the number of keys in current full-text index catalog. There MUST be one **CIndexRecord** of this type in the index table file if there is a record corresponding to a master index component, and there MUST NOT be **CIndexRecords** of this type in the index table file otherwise. There are no files on disk represented by this record.

itNewMaster: The **CIndexRecord** describes a new master index component that will replace the current master index component once the master merge process, as specified in section 2.9, is complete. It has the same restrictions as the master full-text index component.

itAvdlLog: The **CIndexRecord** describes the AVDL file that stores statistics for properties of items in the current master index component. The **ComponentID** field value MUST be 0x00010007 or 0x00020007. The values of the **IndexID** and **MaxDocID** fields MUST be 0x00010000 and 0x00000000 respectively. There MUST be one CIndexRecord of this type in the index table file.

itAvdlLogBackup1: The **CIndexRecord** describes the first average document length backup file. At any moment during the master merge process, one of the AVDL backup files stores AVDL statistics for properties of items in the new master index component. The **ComponentID** of the currently used AVDL backup file is stored in the **ComponentIDAVDLBackup** field of the merge log user header, as specified in section 2.10.1. The **ComponentID** field specifies the **ComponentID** of the AVDL backup file and its value MUST be 0x00010008. The values of the **IndexID** and **MaxDocID** fields MUST be 0x00010000 and 0x00000000 respectively. There MUST be one record of this type in the index table file.

itAvdlLogBackup2: The **CIndexRecord** describes the second AVDL backup file. At any moment during the master merge process, one of the AVDL backup files stores average document length statistics for properties of items in the new master index component. The **ComponentID** of the currently used AVDL backup file is stored in the **ComponentIDAVDLBackup** field of the merge log user header, as specified in section 2.10.1. The **ComponentID** field specifies the ComponentID of the AVDL backup file and its

value MUST be 0x00020008. The values of the **IndexID** and **MaxDocID** fields MUST be 0x00010000 and 0x00000000 respectively. There MUST be one record of this type in the index table file.

itShadowMergeLog: The **CIndexRecord** describes a shadow merge log file, as specified in section 2.10, and the target full-text index component of that shadow merge process. The **ComponentID** field specifies the **ComponentID** value of the shadow merge log file. The 2 lower bytes of the **ComponentID** field MUST be 0x0000 and the 2 higher bytes of the **ComponentID** field MUST be equal to the 2 lower bytes of the index identifier of the target full-text index component. The **IndexID** field specifies the index identifier of the target shadow index component and MUST have the same restrictions as index identifier of **itShadow** CIndexRecord. The value of the **MaxDocID** field MUST be 0x00000000.

itMasterMergeLog: The **CIndexRecord** describes a master merge log file, as specified in section 2.10,. The values of the **ComponentID**, and **MaxDocID** fields have the same restrictions as those in record corresponding to the **itShadowMergeLog**. The value of the **IndexID** field MUST be 0x10000. There MUST be one record of this type whenever there is an **itNewMaster CIndexRecord** in the table.

2.14 Click Distance File

The click distance file uses the appropriate Content Index file format, as specified in section 2.3, to store specific data used in the rank.

A click distance file stores the click distance value for every **document identifier** present in the full-text index catalog. The click distance value is calculated using the minimum number of links that need to be followed to create a path between the list of **authority pages** and the item represented by this document identifier on the web graph.

The encoding of the file uses the same **MaxDocID** value as the master index of the same full-text index catalog and the **format version** is always 0x52.

The click distance file contains two **content index records**. The first content index record has the **content index key** with the index key string the same as BOF key and property identifier =96 (**pidClickDistance**). This record is used for storing 2 values:

- **MaxClickDistance**: The maximum click distance value stored in the file.
- **AverageClickDistance**: The average of the click distance values stored in the file.

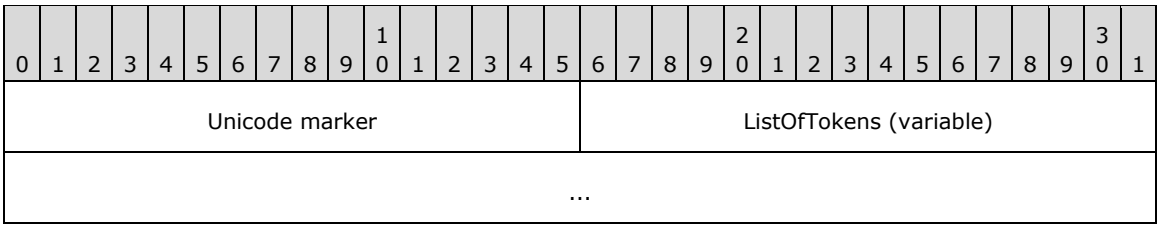
These 2 values are stored as occurrence values for 2 document identifiers**.** The document identifier values MUST be ignored by the reader and SHOULD be set by the writer to 1 and 2 respectively. The **MaxDocIDOccBucket** field MUST be ignored.<31>

The second content index record has the content index key with the index key string the same as the EOF key and property identifier =96 (**pidClickDistance**). This record lists all of the document identifiers used in the current full-text index catalog. For each document identifier, there is one occurrence value, which is the click distance value for that document identifier.

2.15 Index Lexicon File

The index lexicon file is a text file using Unicode encoding which lists the most frequent tokens which appear in the **content index file** of a master full-text index component of the current full-text index catalog. It is used by the query server to determine alternative spelling variants for the tokens encountered in the received queries.

In a binary representation, the format of the file is as follows.

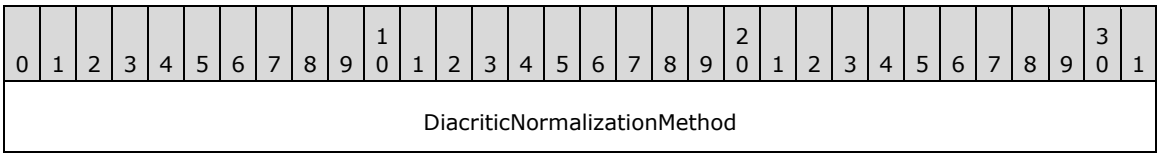


Unicode marker (2 bytes): A 2 byte field specific to the text files which use the Unicode encoding. The values of the bytes MUST be 0xFF followed by 0xFE.

ListOfTokens (variable): Array of Unicode characters representing the list of the most frequent tokens in the catalog. The tokens are separated by the new line characters and each token is composed of 1 to 64 non-space characters.

2.16 Diacritic Settings File

The diacritic settings file is a binary file which contains a single 4-byte integer.



DiacriticNormalizationMethod (4 bytes): DWORD (see [MS-DTYP]) that specifies the character normalization method used for the index keys stored in the current full-text index catalog. The value of this field MUST be one of the values in the following table.

Value	Meaning
1	The index keys were generated using the normalization method insensitive to the character diacritics.
3	The index keys were generated using the normalization method sensitive to the character diacritics.

2.17 Full-Text Index Component

A full-text index component is a set of files that contain all of the index keys extracted from a set of items. Each full-text index component is identified based on an index identifier.

The index identifier is a numeric value in the range from 65,537 to 65,791 (in hexadecimal 0x10001 to 0x100FF). The index identifier is assigned to every full-text index component by the index server. The index identifier for each full-text index component MUST be unique within the search scope of a full-text index catalog.

The individual files that belong to the same full-text index component MUST be identified based on a naming convention in which all file names derive from the index identifier. The naming convention for the files that make up a full-text index component is defined in section 2.17.1.

The following input parameters need to be known to read or write a full-text index component. For full-text index components in full-text index catalogs, these values are defined in corresponding **CIndexRecord** structures in the catalog **Index Table** file.

- **DocIDMax:** A **document identifier** value that is guaranteed to be greater than or equal to any document identifier value of any document in the document set represented by the full-text index component.
- **Format version:** Determines several format variations for the subcomponents of the full-text index component. MUST be 0x52 or 0x53 or 0x54.

The following table enumerates the files that make up a full-text index component.

Component name	Format	Example
Content Index	Content Index File Format (section 2.3)	Section 3.1.5
Content Index Extension (optional)<32>	Content Index Extension File Format (section 2.6)	Section 3.1.6.2
Content Index Directory	Index Directory File Format (section 2.5)	Section 3.1.6.1
Basic Scope Index	Scope Index File Format (section 2.4)	Section 3.1.4
Basic Scope Index Directory	Index Directory File Format (section 2.5)	Section 3.1.3
Compound Scope Index	Scope Index File Format (section 2.4)	Section 3.1.2
Compound Scope Index Directory	Index Directory File Format (section 2.5)	Section 3.1.1
Document Set	Document Set Files Format (section 2.7)	Section 3.1.7

Content Index: A **content index file** that contains **content index keys** generated from the words extracted from the properties of the indexed items. The parameters **DocIDMax** and **format version**, as specified in section 2.17, determine the representation of this component.

Content Index Extension (optional): A CIX file associated with the full-text index component. This file is not present if the version is 0x52.<33>

Content Index Directory: An index directory file associated with the full-text index component.

Basic Scope Index: A scope index file that contains records with either basic scope index keys or anchor scope index keys.

Basic Scope Index Directory: An index directory file associated with the basic scope index.

Compound Scope Index: A scope index file for which the sort keys are compound scope index keys.

Compound Scope Index Directory: Index directory file associated with the **compound scope index**.

Document Set: Several files associated with the full-text index component.

2.17.1 Naming Convention for the Full-Text Index Component Files

The format of the file name for the full-text index component is specified in the following table.

Component name	File name extension	File name format
Content Index	.CI	*XXXXXXXX*.CI
Content Index Extension<34>	.CIX	*XXXXXXXX*.CIX
Content Index Directory	.DIR	*XXXXXXXX*.DIR
Basic Scope Index	.BSI	*XXXXXXXX*.BSI
Basic Scope Index Directory	.BSD	*XXXXXXXX*.BSD
Compound Scope Index	.CSI	*XXXXXXXX.YYYYYYYY*.CSI
Compound Scope Index Directory	.CSD	*XXXXXXXX.YYYYYYYY*.CSD
Document Set	.WID .WSB	*XXXXXXXX*.WID *XXXXXXXX*.WSB

Where:

XXXXXXXX- is the hexadecimal representation of the index identifier.

YYYYYYYY- is the hexadecimal representation of the search scope compilation identifier.

Example:

The following table lists the files that make up the full-text index component with index identifier = 65547 and search scope compilation identifier = 28.

File name
0001001B.CI
0001001B.CIX<35>
0001001B.DIR
0001001B.BSI
0001001B.BSD
0001001B.0000001C.CSI
0001001B.0000001C.CSD
0001001B.WID

2.18 Full-Text Index Catalog

A full-text index catalog is a collection of files placed in the same directory. These files contain the data necessary for resolving full-text queries against all documents crawled by the **search application**.

Each search application operates with 3 full-text index catalogs

- Main catalog, as specified in section 2.18.1.
- Anchor text catalog, as specified in section 2.18.2.

- Active anchor text catalog, as specified in section 2.18.3.

The following files MUST be present in any full-text index catalog.

Diacritic settings: The file SETTINGS.DIA has the diacritic settings file format, as specified in section 3.1.13, and stores the diacritic setting for the full-text index catalog.

QIR file<36>: A set of files that has the query-independent rank file format, as specified in section 3.1.10. These files contain query independent values for a property for each document. Each set of files correspond to one property. The filenames are CiQR????.000, CiQR????.001 and CiQR????.002 for the header, first and second data files respectively. The last 4 characters of file names MUST be equal to the hexadecimal value of the property identifier for the property.

Example: For a property with a property identifier equal to "172", the filenames are: CiQR00AC.000, CiQR00AC.001, and CiQR00AC.002.

Detected languages file<37>: A set of files that has the detected languages file format, as specified in section 3.1.9. The filenames are CiDL0000.000, CiDL0000.001, and CiDL0000.002 for the header, first, and second data files respectively.

Index table: A set of files with the index table file format, as specified in section 3.1.11. The index table enumerates all remaining files in the full-text index catalog, unless specified otherwise. Filenames are INDEX.000, INDEX.001, and INDEX.002 for the header, first and second data files respectively.

The following components MUST be included in the full-text index catalog if they are referenced by the catalog index table file. The file names corresponding to the shadow merge log file, as specified in master merge log file, AVDL file and backup AVDL file are composed of the log prefix (mentioned in the following list) and the 2 higher bytes of **ComponentID** recorded in hexadecimal representation (4 digits). The extensions for these files are ".000" for the header, ".001" for the first and ".002" for the second data files.

Master index component: A full-text index component referenced by an **itMaster** CIndexRecord, as specified in section 2.13.3. There MUST be no more than one master full-text index component in a full-text index catalog.

Shadow index component: Full-text index components referenced by **itShadow** CIndexRecords, as specified in section 2.13.3. There MUST be exactly one full-text index component for each itShadow CIndexRecord in the **Index table.**

Interrupted shadow merges: A set of files referenced by an **itShadowMergeLog** CIndexRecord, as specified in section 2.13.3, that includes an incomplete full-text index component and a shadow merge log file, whose log prefix is "CiMG".

Interrupted master merge: A set of files referenced by **itMasterMergeLog**, as specified in section 2.13.3, and **itNewMaster** CIndexRecords. It includes an incomplete full-text index component and a master merge log file whose log prefix is "CiMG".

AVDL file: An AVDL file referenced by an **itAvdlLog** CIndexRecord. The AVDL file log prefix is "CiAD".

AVDL backup files: AVDL files referenced by **itAvdlLogBackup1** and **itAvdlLogBackup2** CIndexRecords. AVDL backup files log prefix is "CiAB".

The following components are included in the full-text index catalog and they are not referenced by the index table file.

Lexicon file: The file NLGINDEXLEXICON.LEX has the index lexicon file format, as specified in section 2.15. This file MUST be present if there is a master index component or a master merge log file with split key bigger than the minimal **content index key** in the catalog.

Click distance: The file 00CD00CD.ci has the click distance file format, as specified in section 2.14. This file MUST only be present in the anchor text catalog, as specified in section 2.18.2 and the active anchor text catalog, as specified in section 2.18.3, when the active anchor text catalog is not empty.

If **content index file** which belongs to a **master index component** whose **format version** is equal to 0x54 contains **content index records** with property identifier equal to 0x7ffeFFC8 and 0x7ffeFFC9 then a QIR file with property identifier equals 0xAC MUST be present in full-text index catalog. For each **document identifier** in master index component this QIR file MUST store an uncompressed float value. This value defines importance of the item for any query it might be retrieved for.

2.18.1 Main Catalog

Main catalog is a full-text index catalog whose full-text index components contain the data extracted from all the properties that were designated to be placed in the full text index by the **metadata schema**.

The **content index files** in the main catalog MUST contain **content index keys** generated from the words extracted from the Properties of the indexed items. The Properties that are included are the ones that are marked "FullTextQueriable" as defined in [MS-QSSWS] section 3.1.1.3.

The basic scope index files in the main catalog MUST contain the basic scope index keys for all values of the Properties designated to be placed in the scope index files by the metadata schema. The Properties that are included are the ones which are marked "Scopable" as defined in [MS-QSSWS] section 3.1.1.3.

In addition, the basic scope index files store basic scope index keys generated from string values for the pidSiteScope property. For each item, these keys record all generated string values for folders in the URL of the item.

Example:

For an item "http://server/folder/document.htm", the string values "server", "http://server", and "http://server/folder" will be generated.

The compound scope index files MUST contain one scope index record for each compound scope index defined in the search application with a compound scope index key constructed from **compound scopeID**.

2.18.2 Anchor Text Catalog

Anchor text catalog is a full-text index catalog that contains the data extracted from links between items.

The **anchor text** for each link is considered a property on the target item and MUST be stored in the **content index files** with property identifier equal to 10. The content index files MAY<38> contain other records with property identifiers not equal to 10 that contain extra information associated with documents.

For each full-text index component, the basic scope index file MUST contain scope index records for every indexed item which has a link to an item in that full-text index component. These scope index records have the following information:

- **Key**: The record contains an anchor scope index key, which MUST encode the DWORD, as specified in [MS-DTYP], value equal to the **document identifier** of the source item.
- **List of document identifiers**: The record MUST contain document identifiers of all target items in this full-text index component.

There MUST NOT be other scope index records in the basic scope index files.

In an anchor text catalog, all compound scope index files, QIR files, as specified in section 3.1.10, and detected languages files, as specified in section 3.1.9, MUST NOT contain any data and MUST be ignored.

2.18.3 Active Anchor Text Catalog

Active anchor text catalog contains the same data as the anchor text catalog, as specified in section 2.18.2, but it MUST NOT contain any shadow index components.

3 Structure Examples

3.1 Full-text Index Catalog

The following table lists an example file set found in a full-text index catalog. Further details about each individual file are found in subsequent sections. The CIX file in this example is documented in section 3.2.

File name
00010006.0000000A.csd
00010006.0000000A.csi
00010006.bsd
00010006.bsi
00010006.ci
00010006.cix
00010006.dir
00010006.wid
00010006.wsb
CiAB0001.000
CiAB0001.001
CiAB0001.002
CiAB0002.000
CiAB0002.001
CiAB0002.002
CiAD0002.000
CiAD0002.001
CiAD0002.002
CiDL0000.000
CiDL0000.001
CiDL0000.002
CiQR0000.000
CiQR0000.001
CiQR0000.002
INDEX.000
INDEX.001
INDEX.002

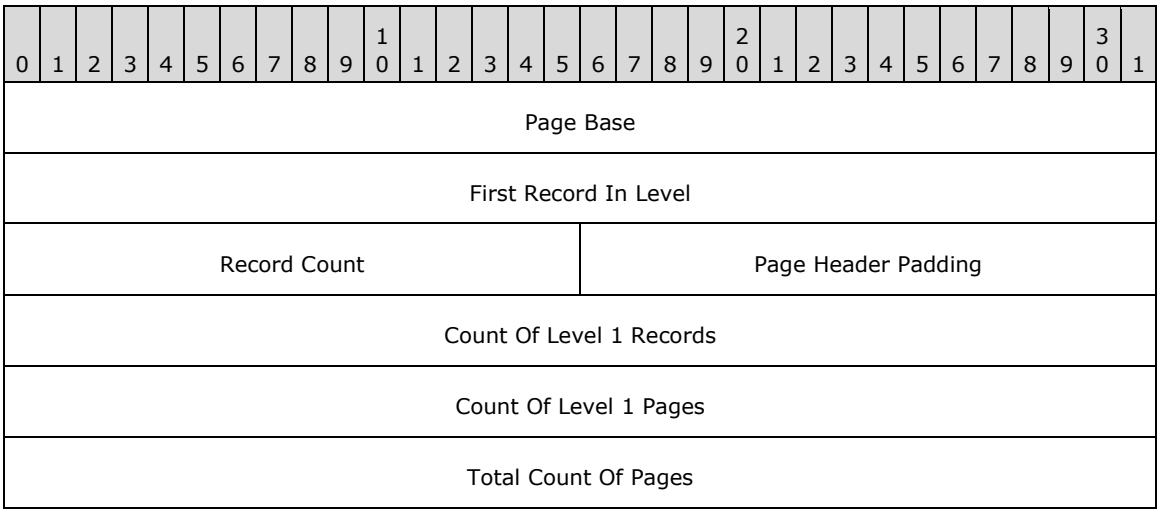
File name
NLGINDEXLEXICON.LEX
SETTINGS.DIA

3.1.1 Compound Scope Index Directory

The following file is 000100006.0000000A.csd in the example full-text index catalog and stores a compound scope index directory in the index directory file format, as specified in section 2.5.

```
0000    00 00 00 00 00 00 00 00 - 02 00 00 00 02 00 00 00
0010    01 00 00 00 01 00 00 00 - 01 00 00 00 90 81 7f ff
0020    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff ff
0030    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff ff
0040    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff ff
0050    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff ff
0060    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff ff
0070    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff ff
0080    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff ff
0090    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff 01
00a0    00 00 92 81 7f ff ff ff - ff ff ff ff ff ff ff ff
00b0    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff ff
00c0    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff ff
00d0    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff ff
00e0    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff ff
00f0    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff ff
0100    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff ff
0110    ff ff ff ff ff ff ff ff - ff ff ff ff ff ff ff ff
0120    ff ff ff ff ff ff ff ff - 7f 00 00 00 00 00 00 00
0130    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
0140    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
…
0fd0    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
0fe0    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
0ff0    00 00 00 00 00 00 00 00 - 00 00 00 00 a2 00 1c 00
```

The following table shows the Index Directory file header, the first 16 bytes of the example at address 0000-0010. The **Page Base**, **First Record In Level**, **Record Count**, and **Page Header Padding** fields comprise the Index Directory Page Header.



Count Of Levels	Padding

Page Base (4 bytes): Set to 00 00 00 00.

First Record In Level (4 bytes): Set to 00 00 00 00.

Record Count (2 bytes): For two records, set to 02 00.

Page Header Padding (2 bytes): Set to 00 00.

Count Of Level 1 Records (4 bytes): Set to 02 00 00 00.

Count Of Level 1 Pages (4 bytes): For one page, set to 01 00 00 00.

Total Count Of Pages (4 bytes): For one page, set to 01 00 00 00.

Count Of Levels (1 byte): For one level, set to 01.

Padding (3 bytes): Set to 00 00 00.

The following table shows the Record Data Buffer (4068 bytes) at address 0010-0ff0.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Flags								Key Size								Key Bytes (129 bytes)																	
...																															
...																															
Property ID								BitStream Offset								BitStream Page								Flags							
Key Size								Key Bytes (129 bytes)																							
...																															
...																															
Offset								Page (3798 bytes)																							
...																															
...																															
Record Offset Array																															

Flags (1 byte): Set to 90.

Key Size (1 byte): For **Key Size** 129, set to 81.

Key Bytes (129 bytes): Begins with 7f and ends with ff at address 0010-0090.

A - Property ID (1 byte): Set to 01.

B - BitStream Offset (1 byte): Set to 00.

C - BitStream Page (1 byte): Set to 00.

Flags (1 byte): Set to 92.

Key Size (1 byte): For **Key Size** 129, set to 81.

Key Bytes (129 bytes): Begins and ends with 7f at address 00a0-0120.

Offset (1 byte): Set to 00.

Page (3798 bytes): Begins and ends with 00 at address 0120-0ff0.

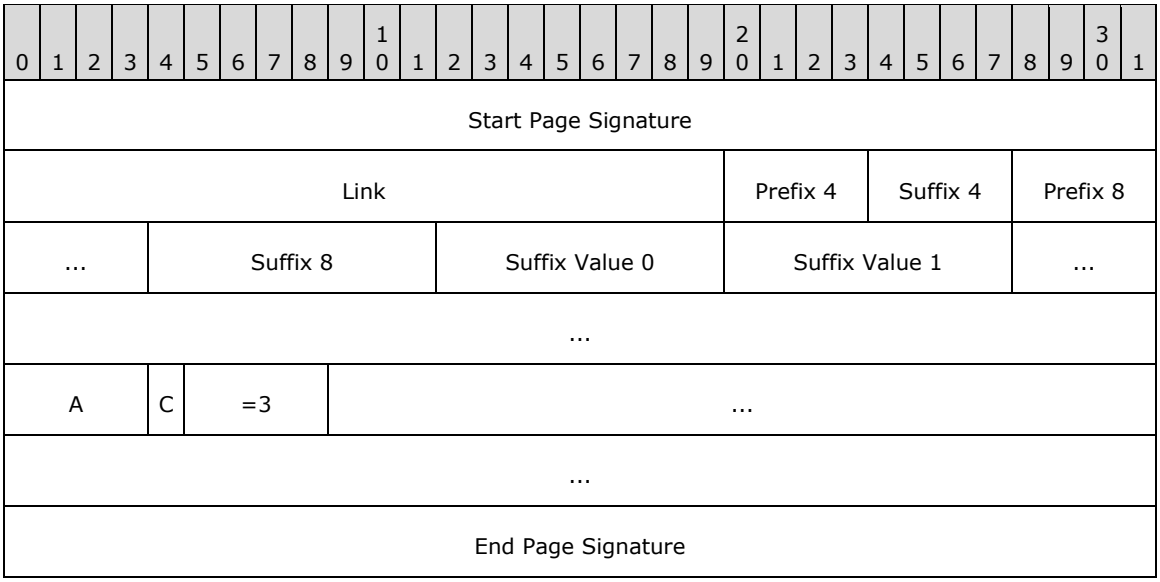
Record Offset Array (4 bytes): Set to a2 00 1c 00.

3.1.2 Compound Scope Index

The following file is 000100006.0000000A.csi in the example full-text index catalog and stores a compound scope index in the scope index file format, as specified in section 2.4.

```
0000    02  00  00  00  00  00  00  00 - ff  ff  17  08  ff  ff  ff  ff
0010    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0020    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0030    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0040    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0050    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0060    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0070    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0080    ff  ff  ff  ff  ff  ff  ff  ff - 1c  f1  ff  ff  00  00  00  00
0090    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00a0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
…
0fd0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0fe0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0ff0    00  00  00  00  00  00  00  00 - 00  00  00  00  02  00  00  00
```

To illustrate file format each 4 bytes are reversed and written in binary form in the following bit table.



Start Page Signature (4 bytes): Set to 00000000000000000000000000000010.

Link (20 bits): Set to 00000000000000000000.

Prefix 4 (4 bits): Set to 0000.

Suffix 4 (4 bits): Set to 0000.

Prefix 8 (1 byte): Set to 00000000.

Suffix 8 (1 byte): Set to 129 (10000001).

Suffix Value 0 (1 byte): Set to 01111111.

Suffix Value 1 (1 byte): Set to 11111111.

... (variable): Continuation.

A - Suffix Value128 (4 bits): Set to 1111.

C (1 bit): Set to 0.

DocID Count (4 bits): For a count of 3, set to 0010.

... (variable): Continuation.

End Page Signature (4 bytes): Set to 00000000000000000000000000000010.

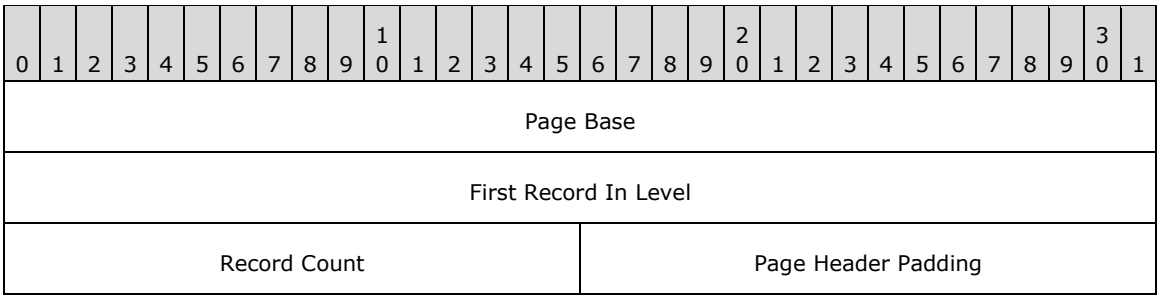
3.1.3 Basic Scope Index Directory

The following file is 000100006.bsd in the example full-text index catalog and stores a basic scope index directory in the index directory file format, as specified in section 2.5.

```
0000    00  00  00  00  00  00  00  00 - 02  00  00  00  02  00  00  00
0010    01  00  00  00  01  00  00  00 - 01  00  00  00  d1  1e  55  66
0020    69  6c  65  3a  2f  2f  63  6f - 37  34  35  2d  31  39  35  2f
0030    66  69  6c  65  73  74  6f  63 - 72  61  77  6c  2a  01  00  00
0040    92  81  7f  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0050    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0060    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0070    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0080    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0090    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
00a0    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
00b0    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
00c0    ff  ff  7f  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00d0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00e0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
…
0fd0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0fe0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0ff0    00  00  00  00  00  00  00  00 - 00  00  00  00  40  00  1c  00
```

The preceding example consists of one level and has the following structure.

Index Directory Page Header (12 bytes at address 0000)



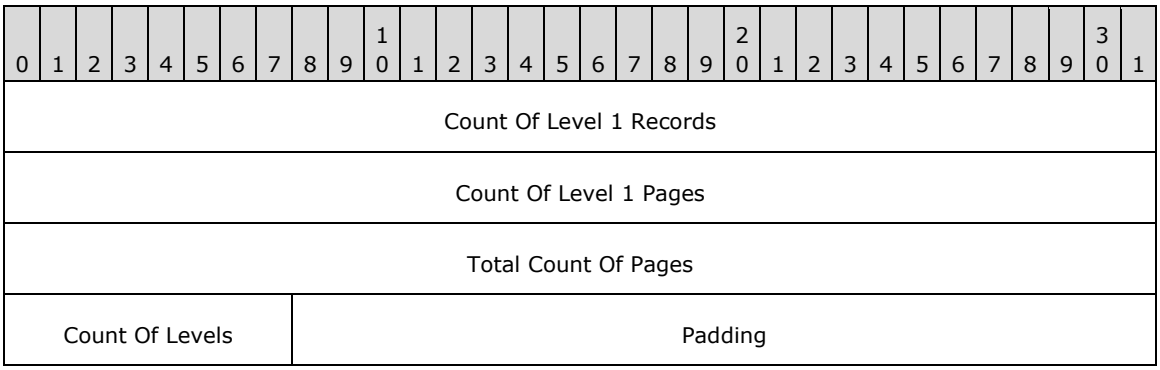
Page Base (4 bytes): Set to 00 00 00 00.

First Record In Level (4 bytes): Set to 00 00 00 00.

Record Count (2 bytes): Set to 02 00.

Page Header Padding (2 bytes): Set to 00 00.

Index Directory File Header (16 bytes at address 0000-0010)



Count Of Level 1 Records (4 bytes): Set to 02 00 00 00.

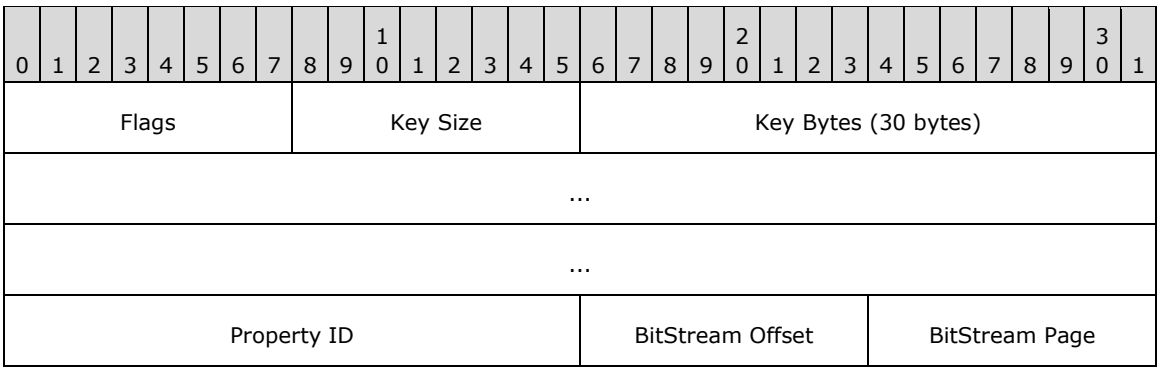
Count Of Level 1 Pages (4 bytes): Set to 01 00 00 00.

Total Count Of Pages (4 bytes): Set to 01 00 00 00.

Count of Levels (1 byte): Set to 01.

Padding (3 bytes): Set to 00 00 00.

Record Data Buffer (4068 bytes at address 0010-0ff0.)



Flags	Key Size	Key Bytes (129 bytes)
...		
...		
Property ID		
BitStream Offset	BitStream Page	...
...		
Record Offset Array		

Flags (1 byte): Set to d1.

Key Size (1 byte): Set to 1e for **Key Size** of 30.

Key Bytes (30 bytes): Set to 55 66 69 6c 65 3a 2f 2f 63 6f 37 34 35 2d 31 39 35 2f 66 69 6c 65 73 74 6f 63 72 61 77 6c.

Property ID (2 bytes): Set to 2a 01.

BitStream Offset (1 byte): Set to 00.

BitStream Page (1 byte): Set to 00.

Flags (1 byte): Set to 92.

Key Size (1 byte): Set to 81

Key Bytes (129 bytes): Starts with 7f and ends with 7f 00.

Property ID (4 bytes): Set to 00 00 00 00.

BitStream Offset (1 byte): Set to 00.

BitStream Page (1 byte): Set to 00.

... : Continuation.

Record Offset Array (4 bytes): Set to 40 00 1c 00.

3.1.4 Basic Scope Index

The following file is 000100006.bsi in the example full-text index catalog and stores a basic scope index in the scope index file format, as specified in section 2.4.

```
0000    06  00  00  00  00  50  3c  00 - 06  50  b5  03  06  90  06  60
0010    03  50  06  c0  02  f0  02  a0 - 06  30  06  f0  03  70  03  f0
0020    02  50  03  40  03  10  03  d0 - 02  50  03  90  06  60  06  f0
0030    06  c0  06  90  07  30  07  50 - 06  f0  06  40  06  20  07  30
0040    06  70  07  10  c8  04  da  cc - 14  b1  02  10  00  00  00  00
0050    00  00  00  00  00  40  c8  0a - 00  00  00  00  2b  00  00  00
0060    00  00  00  31  00  00  00  00 - 04  ad  00  00  00  00  00  00
0070    00  00  00  00  00  00  00  00 - 00  00  00  00  80  42  01  00
0080    80  af  09  00  80  37  80  31 - 00  1a  80  1b  80  16  80  1a
```

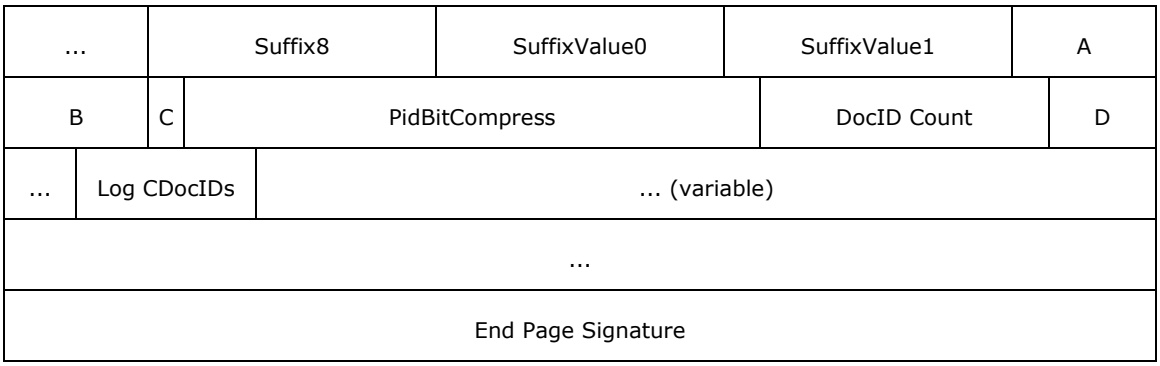
```
0090    80  1c  80  18  26  d0  e6  1a - 88  15  80  40  00  00  00  a0
00a0    00  00  00  00  00  42  56  00 - 00  00  00  00  01  00  00  00
00b0    00  00  88  59  00  00  00  00 - 68  05  00  00  00  00  00  20
00c0    00  00  00  00  00  00  00  00 - 00  00  00  00  94  0b  00  00
00d0    98  7d  08  00  b0  01  a4  01 - e8  00  94  01  bc  00  bc  00
00e0    bc  01  8c  01  d0  00  dc  00 - b4  00  d4  00  e4  00  c4  00
00f0    81  36  d7  00  ac  00  04  32 - 00  00  00  45  00  00  00  00
0100    10  b2  02  00  00  00  00  00 - 00  00  00  00  00  40  cc  0a
0110    00  00  00  00  2b  00  00  00 - 00  00  00  41  00  00  00  00
0120    00  00  00  00  00  00  00  00 - 57  00  00  00  40  23  04  a0
0130    c0  0c  e0  05  80  0d  20  0d - 60  0e  a0  0c  e0  0d  80  0e
0140    40  0e  60  0c  e0  0e  20  0c - 09  b4  99  0d  62  05  20  90
0150    00  00  00  28  00  00  00  00 - 80  90  15  00  00  00  00  00
0160    00  00  00  00  00  00  62  56 - 00  00  00  00  5a  01  00  00
0170    00  00  00  08  00  00  00  00 - 00  00  00  00  00  00  00  00
0180    20  02  00  00  00  14  3b  00 - 00  73  00  2f  00  62  00  75
0190    00  6f  00  66  00  64  00  6c - cd  72  00  65  e0  84  26  a0
01a0    19  6c  6a  77  00  00  00  a0 - 00  00  00  00  d3  00  00  00
01b0    00  00  00  0c  00  00  00  00 - 00  00  00  00  00  00  00  00
01c0    00  00  00  00  05  02  00  00 - 00  00  7e  05  a0  cd  05  01
01d0    2b  00  81  4c  00  00  40  11 - 00  00  00  00  84  ac  00  00
01e0    00  00  00  00  00  00  00  00 - 00  10  b3  02  00  00  00  00
01f0    0a  00  00  00  00  00  40  d0 - 00  00  00  00  00  00  00  00
0200    00  00  00  00  15  00  00  00 - c9  20  00  a8  03  30  03  d8
0210    03  60  03  48  03  98  03  28 - 03  78  03  a0  03  90  03  18
0220    03  b8  03  08  64  02  6d  66 - 8a  58  01  08  00  00  00  00
0230    00  00  00  00  00  20  64  05 - 00  00  00  00  15  00  00  00
0240    00  00  80  98  00  00  00  00 - 82  56  00  00  00  00  00  00
0250    00  00  00  00  00  00  00  00 - 00  00  00  00  44  79  00  00
0260    13  68  f3  7f  c4  0a  40  20 - 00  00  00  50  00  00  00  00
0270    00  21  2b  00  00  00  00  00 - 00  00  00  00  00  00  c4  ac
0280    00  00  00  00  b4  02  00  00 - 00  00  00  10  00  00  00  00
0290    00  00  00  00  00  00  00  00 - 00  00  00  00  ff  02  01  00
02a0    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
02b0    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
02c0    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
02d0    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
02e0    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
02f0    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0300    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0310    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  fe  ff  ff  ff
0320    00  00  80  23  00  00  00  00 - 00  00  00  00  00  00  00  00
0330    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0340    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
…
0fd0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0fe0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0ff0    00  00  00  00  00  00  00  00 - 00  00  00  00  06  00  00  00
```

Remarks:

This file is in the scope index file format.

To illustrate file format, each 4 bytes are reversed and written in binary form in the following bit table.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>20</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>30</td><td>1</td></tr>
<tr><td colspan="32">Start Page Signature</td></tr>
<tr><td colspan="20">Links</td><td colspan="4">Prefix4</td><td colspan="4">Suffix4</td><td colspan="4">Prefix8</td></tr>
</table>



Start Page Signature (4 bytes): Set to 00000000000000000000000000000110.

Links (20 bits): Set to 00000000001111000101.

Prefix4 (4 bits): Set to 0000.

Suffix4 (4 bits): Set to 0000.

Prefix8 (1 byte): Set to 00000000.

Suffix8 (1 byte): Set to 59 (00111011).

SuffixValue0 (1 byte): Set to 01010101.

SuffixValue1 (1 byte): Set to 00000000.

A - SuffixValue2 (4 bits): Set to 0110.

B - Suffix Value58 (4 bits): Set to 1100.

C (1 bit): Set to 1.

PidBitCompress (2 bytes): Set to 1001 1 01 1 010 00000.

DocID Count (1 byte): Set to 10011001.

D - Average DocID bitcount (5 bits): Set to 00000.

Log CDocIDs (5 bits): Set to 01000.

... (variable): Continuation.

End Page Signature (4 bytes): Set to 00000000000000000000000000000110.

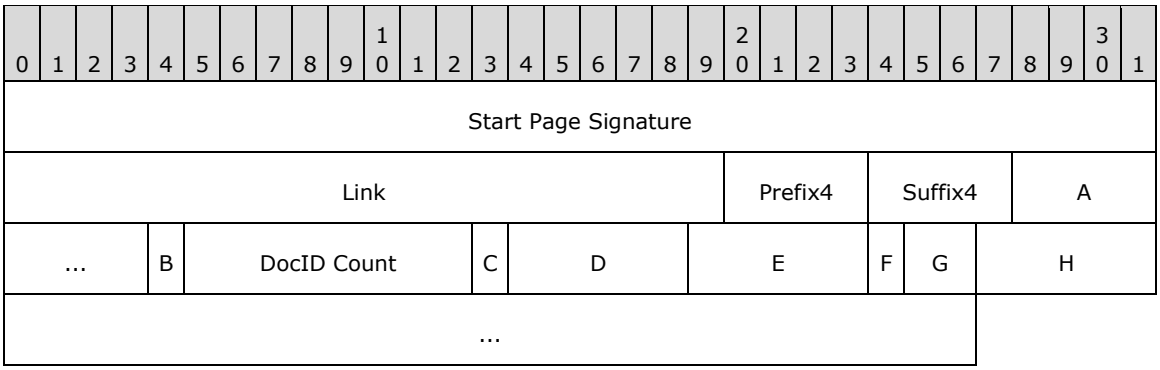
3.1.5 Content Index File

The following file is 000100006.ci in the example full-text index catalog and stores a **content index file** in the content index file format, as specified in section 2.3.

```
0000    06  00  00  00  10  a0  68  00 - 80  80  4b  00  04  6c  24  2b
0010    10  40  00  01  40  00  01  04 - 00  01  04  10  01  04  10  40
0020    04  10  40  00  10  40  00  01 - 40  00  01  04  00  01  04  10
0030    01  04  10  40  15  10  40  00 - 04  10  40  64  10  40  00  01
0040    40  00  81  04  00  01  04  10 - 01  04  10  40  04  10  40  00
0050    10  40  00  01  40  00  01  04 - 00  01  04  10  01  04  10  40
0060    99  55  40  00  04  10  40  00 - 10  40  00  01  40  00  01  04
0070    00  01  04  10  01  04  10  40 - 04  10  40  00  10  40  00  01
0080    40  00  01  04  00  01  04  10 - 01  04  10  40  01  84  56  01
```

```
0090    04 10 40 00 10 40 00 01 - 40 00 01 04 00 01 04 10
00a0    01 04 10 40 04 10 40 00 - 10 40 00 01 40 00 01 04
00b0    00 01 04 10 00 04 10 40 - 01 04 00 00 04 10 40 00
00c0    10 40 00 01 40 00 01 04 - 00 01 04 10 01 04 10 40
00d0    04 10 40 00 60 1a 00 01 - 01 99 40 42 a0 88 55 00
00e0    01 04 10 20 04 10 40 00 - 10 40 00 01 40 00 01 04
00f0    00 01 04 10 01 04 10 40 - 04 10 40 00 10 40 00 01
0100    40 00 01 04 01 01 04 10 - 00 01 42 56 01 04 10 40
0110    02 10 40 00 10 40 00 01 - 40 00 01 04 00 01 04 10
0120    01 04 10 40 04 10 40 00 - 10 40 00 01 40 00 01 04
0130    59 05 04 10 00 01 04 88 - 01 04 10 40 04 10 40 00
0140    10 40 00 01 40 00 01 04 - 00 01 04 10 01 04 10 40
0150    04 10 40 00 10 40 00 01 - 40 00 01 04 20 68 15 10
0160    00 01 04 10 01 04 10 40 - 04 10 40 00 10 40 00 01
0170    40 00 01 04 00 01 04 10 - 01 04 10 40 04 10 40 00
0180    10 40 00 01 40 00 01 04 - 40 00 00 00 00 01 04 10
0190    01 04 10 40 04 10 40 00 - 10 40 00 01 40 00 01 04
01a0    00 01 04 10 01 04 10 40 - 42 60 1a 00 00 01 99 e0
01b0    80 a0 88 55 00 03 0c 30 - 03 0c 30 c0 0c 30 c0 00
01c0    30 c0 00 03 c0 00 03 0c - 00 03 0c 30 03 0c 30 c0
01d0    0c 30 c0 00 30 c0 00 03 - 56 01 03 0c c0 00 03 42
01e0    00 03 0c 30 03 0a 30 c0 - 0c 30 c0 00 30 c0 00 03
01f0    c0 00 03 0c 00 03 0c 30 - 03 0c 30 c0 0c 30 c0 00
0200    30 c0 00 03 88 59 05 0c - c0 00 03 0c 00 03 0c 30
0210    03 0c 30 c0 0c 30 c0 00 - 38 e0 80 03 e0 80 03 0e
0220    80 03 0e 38 03 0e 38 e0 - 0e 38 e0 80 38 e0 80 03
0230    38 20 68 15 e0 80 03 0e - 80 03 0e 38 03 0e 38 e0
0240    0e 38 e0 80 38 e0 80 03 - e0 80 03 0e 80 03 0e 38
0250    03 0e 38 e0 0e 38 e0 80 - 00 e0 80 03 38 e0 00 00
0260    e0 80 03 0e 80 03 0e 38 - 03 0e 38 e0 0e 38 e0 80
0270    38 e0 80 03 e0 80 03 0e - 80 03 0e 38 d0 43 6c 1a
…
4fd0    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
4fe0    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
4ff0    00 00 00 00 00 00 00 00 - 00 00 00 06 00 00 00
```

To illustrate file format each 4 bytes are reversed and written in binary form in the following bit table.



Start Page Signature (4 bytes): Set to 00000000000000000000000000000110.

Link (20 bits): Set to 00000000011010001010.

Prefix4 (4 bits): Set to 0000.

Suffix4 (4 bits): Set to 0001.

A - SuffixValue0 (1 byte): Set to 00000000.

B - C (1 bit): Set to 0.

DocID Count (1 byte): Set to 151 (10010111).

C - IsSBRIPresent (1 bit): Set to 0.

D - AverageDocIDbitcount (5 bits): Set to 00000.

E - LogCDocIDs (5 bits): Set to 01000.

F - Is CIXLink Present (1 bit): Set to 0.

G - DocID Delta0 (2 bits): Set to 00.

H - End Page Signature (4 bytes): Set to 00000000000000000000000000000110.

3.1.6 Index Directory

The following file is 000100006.dir in the example full-text index catalog and stores an index directory in the index directory file format, as specified in section 2.5.

```
0000    00  00  00  00  00  00  00  00 - 06  00  00  00  06  00  00  00
0010    01  00  00  00  01  00  00  00 - 01  00  00  00  f0  00  01  00
0020    00  e0  09  66  69  6c  65  6e - 61  6d  65  31  02  4a  03  01
0030    f0  0e  66  69  6c  65  6e  61 - 6d  65  37  33  2e  74  78  74
0040    02  39  02  f0  0e  74  65  6d - 70  66  69  6c  65  32  39  2e
0050    74  78  74  02  60  03  e0  03 - 74  78  74  38  bb  05  04  92
0060    81  7f  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0070    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0080    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0090    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
00a0    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
00b0    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
00c0    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
00d0    ff  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
00e0    ff  ff  ff  ff  ff  7f  00  00 - 00  00  00  00  00  00  00  00
00f0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0100    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
…
0fd0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0fe0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0ff0    00  00  00  00  00  00  00  00 - 00  00  00  00  22  00  1c  00
```

This example has the same structure as 00010006.0000000A.csd and 00010006.bsd.

3.1.6.1 Content Index Record

This is a standalone example of two **content index record**s with **property identifiers** equal to 0x7ffeFFC8 and 0x7ffeFFC9. This example is not related to a full-text index catalog described in other examples.

Assumptions:

These content index records are written sequentially into a **content index file** with file version equal to 0x54. Previous content index records in content index file contained data about term "office" with property identifiers 1 and 2. The following two content index records contain data about term "office". Maximum **document identifier** for the current content index file is 300.

Previous content index records contained the following data:

property identifier = 1

- docid: 1; Occurrences:2,3;
- docid: 2; Occurrences: 1;
- docid: 3; Occurrences: 1;
- docid: 258; Occurrences: 1;
- docid: 262; Occurrences: 1.

property identifier = 2

- docid: 3; Occurrences: 1;
- docid: 261; Occurrences: 1.

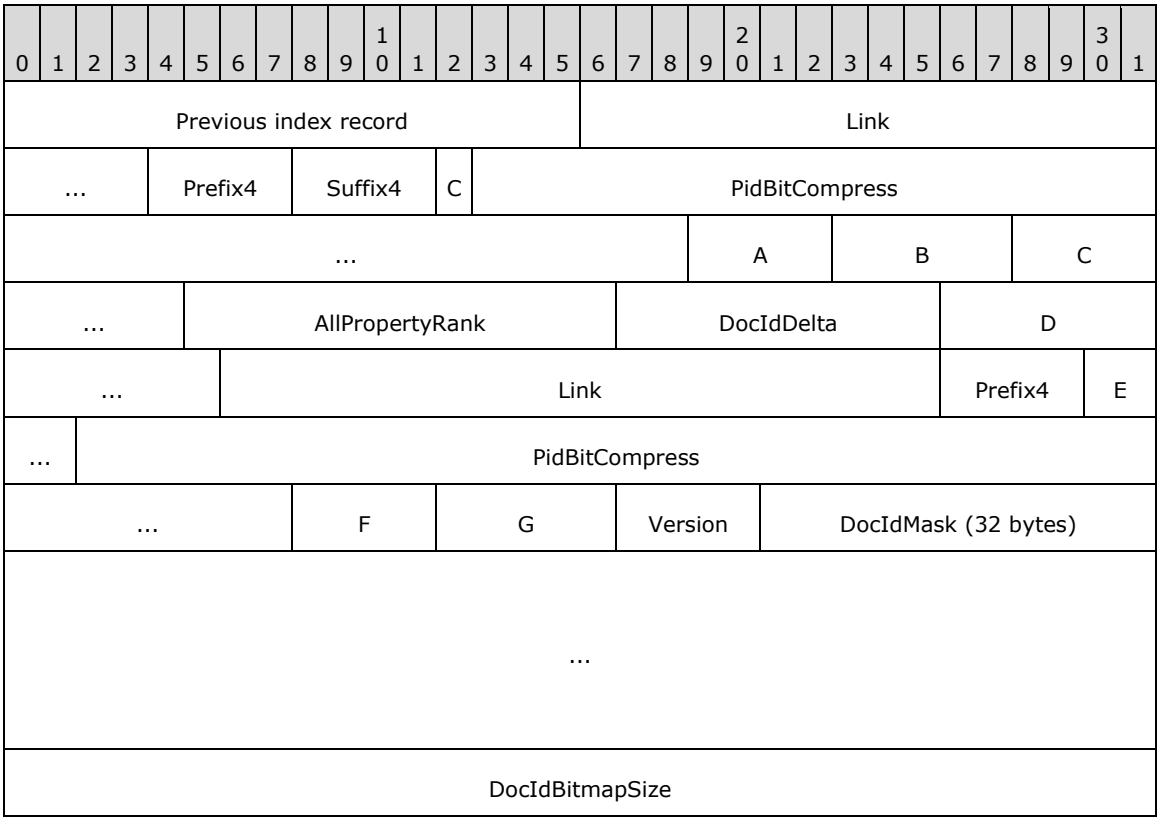
In this example:

property identifier = 0x7ffeFFC8

- docid: 1;
- docid: 3.

property identifier = 0x7ffeFFC9

- docid: 2;
- docid: 258;
- docid: 262.



H	Next Content Index Record
...	

Previous Content Index Record: Continued from previous content index record.

Link (20 bits): For a value of 169, set to 00000000000010101001.

Prefix4 (4 bits): Set to 0110.

Suffix4 (4 bits): Set to 0000.

C (1 bit): Set to 1.

PidBitCompress (38 bits): For a value of 0x7ffeFFC8, set to 11111111111111111101111111111110010000.

A - DocID Count (4 bits): For a doc count of 2, set to 0001.

B - Average DocID bitcount (5 bits): For a bit count of 8, set to 00111.

C - DocIdDelta (9 bits): For a delta of 1, set to 000000000.

AllPropertyRank (12 bits): Set to 110000000000.

DocIdDelta (9 bits): For a delta of 2, set to 000000010.

D - AllPropertyRank (12 bits): Set to 101000000000.

Link (20 bits): For a link of 372, set to 00000000000101110100.

Prefix4 (4 bits): Set to 0110.

E - Suffix4 (4 bits): Set to 0000.

PidBitCompress (38 bits): For a value of 0x7ffeFFC9, set to 11111111111111111101111111111110010010.

F - DocID Count (4 bits): Set to 0001.

G - Average DocID bitcount (5 bits): Set to 00110.

Version (4 bits): Set to 0000.

DocIdMask (32 bytes): Set to the following:

```
00100010000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
```

DocIdBitmapSize (4 bytes): For a size of 5, set to 00000000000000000000000000000101.

H - DocIdBitmap (5 bits): Set to 10110.

Next Content Index Record: Beginning of the next content index record.

3.1.6.2 Content Index Record with Skips

In this example, a **content index record** for the term "office", with a **property identifier** of 2, follows a content index record for the same term with a property identifier of 1. The file version of **content index file** is 0x54 and maximum **document identifier** is 300. It contains information about 7 items with document identifiers 1, 5, 8, 9, 10, 16, 32, each with one occurrence equal to 1. Content index record contains 2 skips. The first one points at ContentDocIDData[2], the second one points at ContentDocIDData[6]. The values of **SkipsPage** and **SkipsOffset** are not specified because they depend on actual position within content index file. The content index record is represented as **BitStream**.

Previous index record				Link	
xxxxxxxxxxxxxxxxxxxxxxxxxx				000000	
Link		**Prefix4**	**Suffix4**	**C**	**PidBitCompress = 2**
00000010101001		0110	0000	1	00100
DocID Count=6	**Average DocID bitcount = 2**		**logCDocIDs**		
0101	00010		00001		
SkipsPage					
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx					
SkipsOffset					
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx					
IsCIXLinkPresent	**DocIdDelta = 1 (0+ 1)**	**MaxDocIDOccBucket**	**OccCount**	**OccsDelta[0]**	
0	0000	0000011	0010	00000000	
DocIdDelta = 4 (3+ 1)		**MaxDocIDOccBucket**	**OccCount**	**OccsDelta[0]**	
0110		0001011	0010	00000000	
DocIdDelta = 3 (2+ 1)		**MaxDocIDOccBucket**	**OccCount**	**OccsDelta[0]**	
0100		0000111	0010	00000000	
DocIdDelta = 1 (0+ 1)		**MaxDocIDOccBucket**	**OccCount**	**OccsDelta[0]**	
0000		0000001	0010	00000000	
DocIdDelta = 1 (0+ 1)		**MaxDocIDOccBucket**	**OccCount**	**OccsDelta[0]**	
0000		0001001	0010	00000000	
DocIdDelta = 6 (5+ 1)		**MaxDocIDOccBucket**	**OccCount**	**OccsDelta[0]**	
1010		0011001	0010	00000000	
DocIdDelta = 16 (15+ 1)		**MaxDocIDOccBucket**	**OccCount**	**OccsDelta[0]**	
0111110		0011011	0010	00000000	
DocIDSkipCount =2	**DocIDDelta = 5 (4+ 1)**	**DocIDSkipOffsetDelta = 46**	**IsDefaultDocIDSkip**	**DocIdSkip = 2**	
0000000100	00001010	01011100	0	010	
DocIDDelta = 11 (10+ 1)		**DocIDSkipOffsetDelta = 92**	**IsDefaultDocIDSkip**	**Next index record**	
00010100		10111000	1	xxxxxxxxxxxxx	

Figure 4: Content index record with skips

3.1.7 Document Set Files

The following two files store a document set file, as specified in section 2.7, in the example full-text index catalog in the indexed bitmap document set scheme, as specified in section 2.7.3.

This is the 000100006.wid file in the example set.

```
0000    02  00  00  00  05  00  00  00 - 00  00  00  00  00  00  00  00
0010    98  00  00  00  00  00  00  00 - 00  00  00  00  01  00  00  00
0020    02  00  00  00  99  00  00  00 - 00  00  00  00  00  00  00  00
0030    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0040    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
…
ffe0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
fff0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
```

The preceding file has the following structure.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type of scheme																															
Bdate																															
Flag																															
Outdated DocIDs																															
Number of DocIDs																															
Reserved1																															
Reserved2																															
SizeOfH1																															
Minimum DocID Value																															
Maximum DocID Value																															
Number of DocIDs Delta																															
Reserved3 (4052 bytes)																															
...																															
H1 (4 bytes)																															

Type of scheme (4 bytes): Set to 02 00 00 00 for `Scheme = 2`.

Bdate (4 bytes): Set to 05 00 00 00.

Flag (4 bytes): Set to 00 00 00 00.

Outdated DocIDs (4 bytes): Set to 00 00 00 00.

Number of DocIDs (4 bytes): Set to 98 00 00 00.

Reserved1 (4 bytes): Set to 00 00 00 00.

Reserved2 (4 bytes): Set to 00 00 00 00.

SizeOfH1 (4 bytes): Set to 01 00 00 00.

Minimum DocID Value (4 bytes): Set to 02 00 00 00.

Maximum DocID Value (4 bytes): Set to 99 00 00 00.

Number of DocIDs Delta (4 bytes): Set to 00 00 00 00.

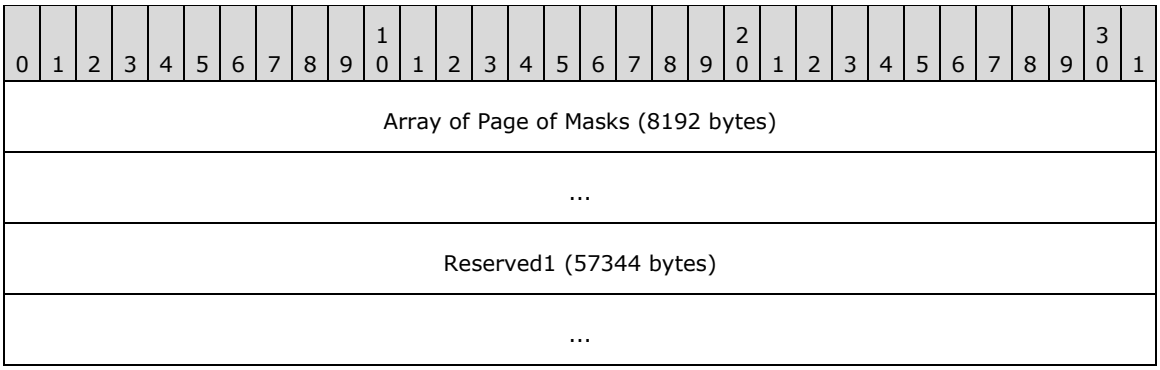
Reserved3 (4052 bytes): Set to all zeros from address 002c through 1000.

H1 (4 bytes): Set to 00 00 00 00.

This is the 000100006.wsb file in the example set.

```
0000    fc  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
0010    ff  ff  ff  03  00  00  00  00 - 00  00  00  00  00  00  00  00
0020    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
…
ffe0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
fff0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
```

The preceding file has the following structure.



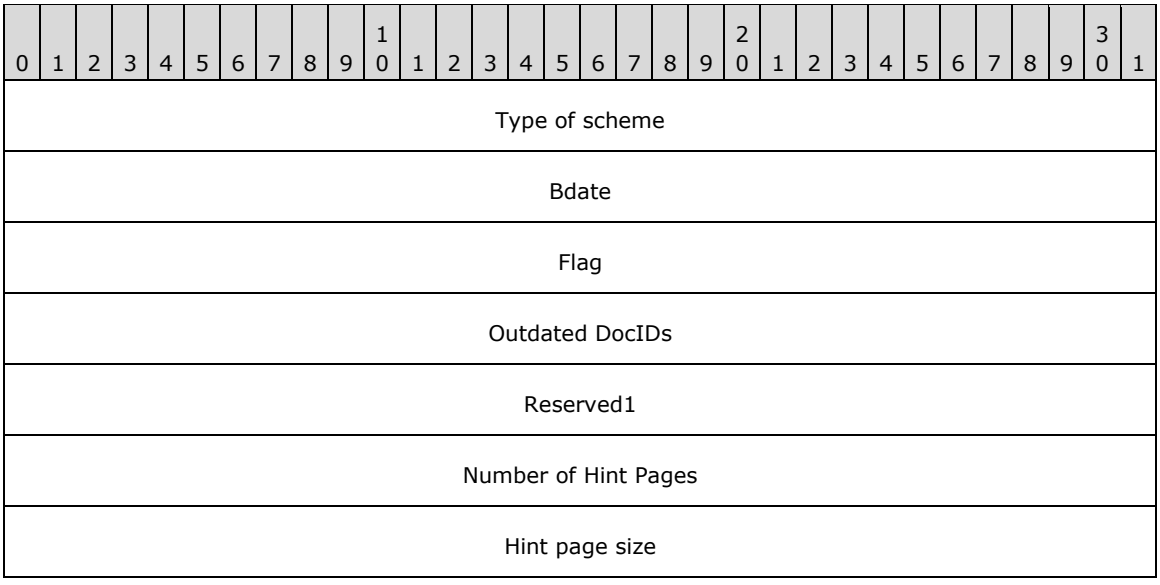
Array of Page of Masks (8192 bytes): Bits corresponding to document identifiers (1) present in the document set file.

Reserved1 (57344 bytes): Set to all zeros.

The same example in List Document Set format, as specified in section 2.7.1, 000100006.wid in the example set:

```
0000    01  00  00  00  05  00  00  00 - 00  00  00  00  00  00  00  00
0010    98  00  00  00  00  00  00  00 - 00  00  00  00  98  00  00  00
0020    02  00  00  00  99  00  00  00 - 00  00  00  00  00  00  00  00
0030    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0040    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
…
0fe0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0ff0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
1000    02  00  00  00  03  00  00  00 - 04  00  00  00  05  00  00  00
1010    06  00  00  00  07  00  00  00 - 08  00  00  00  09  00  00  00
```

```
1020    0a 00 00 00 0b 00 00 00 - 0c 00 00 00 0d 00 00 00
1030    0e 00 00 00 0f 00 00 00 - 10 00 00 00 11 00 00 00
1040    12 00 00 00 13 00 00 00 - 14 00 00 00 15 00 00 00
1050    16 00 00 00 17 00 00 00 - 18 00 00 00 19 00 00 00
1060    1a 00 00 00 1b 00 00 00 - 1c 00 00 00 1d 00 00 00
1070    1e 00 00 00 1f 00 00 00 - 20 00 00 00 21 00 00 00
1080    22 00 00 00 23 00 00 00 - 24 00 00 00 25 00 00 00
1090    26 00 00 00 27 00 00 00 - 28 00 00 00 29 00 00 00
10a0    2a 00 00 00 2b 00 00 00 - 2c 00 00 00 2d 00 00 00
10b0    2e 00 00 00 2f 00 00 00 - 30 00 00 00 31 00 00 00
10c0    32 00 00 00 33 00 00 00 - 34 00 00 00 35 00 00 00
10d0    36 00 00 00 37 00 00 00 - 38 00 00 00 39 00 00 00
10e0    3a 00 00 00 3b 00 00 00 - 3c 00 00 00 3d 00 00 00
10f0    3e 00 00 00 3f 00 00 00 - 40 00 00 00 41 00 00 00
1100    42 00 00 00 43 00 00 00 - 44 00 00 00 45 00 00 00
1110    46 00 00 00 47 00 00 00 - 48 00 00 00 49 00 00 00
1120    4a 00 00 00 4b 00 00 00 - 4c 00 00 00 4d 00 00 00
1130    4e 00 00 00 4f 00 00 00 - 50 00 00 00 51 00 00 00
1140    52 00 00 00 53 00 00 00 - 54 00 00 00 55 00 00 00
1150    56 00 00 00 57 00 00 00 - 58 00 00 00 59 00 00 00
1160    5a 00 00 00 5b 00 00 00 - 5c 00 00 00 5d 00 00 00
1170    5e 00 00 00 5f 00 00 00 - 60 00 00 00 61 00 00 00
1180    62 00 00 00 63 00 00 00 - 64 00 00 00 65 00 00 00
1190    66 00 00 00 67 00 00 00 - 68 00 00 00 69 00 00 00
11a0    6a 00 00 00 6b 00 00 00 - 6c 00 00 00 6d 00 00 00
11b0    6e 00 00 00 6f 00 00 00 - 70 00 00 00 71 00 00 00
11c0    72 00 00 00 73 00 00 00 - 74 00 00 00 75 00 00 00
11d0    76 00 00 00 77 00 00 00 - 78 00 00 00 79 00 00 00
11e0    7a 00 00 00 7b 00 00 00 - 7c 00 00 00 7d 00 00 00
11f0    7e 00 00 00 7f 00 00 00 - 80 00 00 00 81 00 00 00
1200    82 00 00 00 83 00 00 00 - 84 00 00 00 85 00 00 00
1210    86 00 00 00 87 00 00 00 - 88 00 00 00 89 00 00 00
1220    8a 00 00 00 8b 00 00 00 - 8c 00 00 00 8d 00 00 00
1230    8e 00 00 00 8f 00 00 00 - 90 00 00 00 91 00 00 00
1240    92 00 00 00 93 00 00 00 - 94 00 00 00 95 00 00 00
1250    96 00 00 00 97 00 00 00 - 98 00 00 00 99 00 00 00
1260    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
1270    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
…
ffe0    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
fff0    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
```



Number of DocIDs
Minimum DocID Value
Maximum DocID Value
Number of DocIDs Delta
Reserved2 (2004 bytes)
...
Reserved3 (2048 bytes)
…
Array of DocIDs (608 bytes)
...

Type of scheme (4 bytes): Set to 01 00 00 00 for `Type of Scheme = 1`.

Bdate (4 bytes): Set to 05 00 00 00.

Flag (4 bytes): Set to 00 00 00 00.

Outdated DocIDs (4 bytes): Set to 00 00 00 00.

Reserved 1 (4 bytes): Set to 98 00 00 00.

Number of Hint Pages (4 bytes): Set to 00 00 00 00.

Hint page size (4 bytes): Set to 00 00 00 00.

Number of DocIDs (4 bytes): Set to 98 00 00 00.

Minimum DocID Value (4 bytes): Set to 02 00 00 00.

Maximum DocID Value (4 bytes): Set to 99 00 00 00.

Number of DocIDs Delta (4 bytes): Set to 00 00 00 00.

Reserved2 (2004 bytes): Set to all zeros from address 0020 through 0800.

Hint Array (0 bytes): The field is missing.

Reserved3 (2048 bytes): Set to all zeros from address 0800 through 1000.

Array of DocIDs (608 bytes): Array of 152 of document identifiers (1).

The same example in Bitmap Document Set format, as specified in section 2.7.2, 000100006.wid in the example set:

```
0000    03  00  00  00  05  00  00  00 – 00  00  00  00  00  00  00  00
```

```
0010    98  00  00  00  00  00  00  00 - 00  00  00  00  05  00  00  00
0020    02  00  00  00  99  00  00  00 - 00  00  00  00  00  00  00  00
0030    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0040    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
…
0fe0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0ff0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
1000    fc  ff  ff  ff  ff  ff  ff  ff - ff  ff  ff  ff  ff  ff  ff  ff
1010    ff  ff  ff  03  00  00  00  00 - 00  00  00  00  00  00  00  00
1020    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
…
ffe0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
fff0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type of scheme																															
Bdate																															
Flag																															
Outdated DocIDs																															
Number of DocIDs																															
Reserved1																															
Reserved2																															
Size of bitmap																															
Minimum DocID Value																															
Maximum DocID Value																															
Number of DocIDs Delta																															
Reserved3 (4052 bytes)																															
...																															
Bitmap (20 bytes)																															
…																															

Type of scheme (4 bytes): Set to 03 00 00 00 for `Type of Scheme = 3`.

Bdate (4 bytes): Set to 05 00 00 00.

Flag (4 bytes): Set to 00 00 00 00.

Outdated DocIDs (4 bytes): Set to 00 00 00 00.

Reserved1 (4 bytes): Set to 98 00 00 00.

Reserved2 (4 bytes): Set to 00 00 00 00.

Size of bitmap (4 bytes): Set to 05 00 00 00.

Number of DocIDs (4 bytes): Set to 98 00 00 00.

Minimum DocID Value (4 bytes): Set to 02 00 00 00.

Maximum DocID Value (4 bytes): Set to 99 00 00 00.

Number of DocIDs Delta (4 bytes): Set to 00 00 00 00.

Reserved3 (4052 bytes): Set to all zeros from address 0020 through 1000.

Bitmap (20 bytes): Bits corresponding to document identifiers (1) present in the document set file.

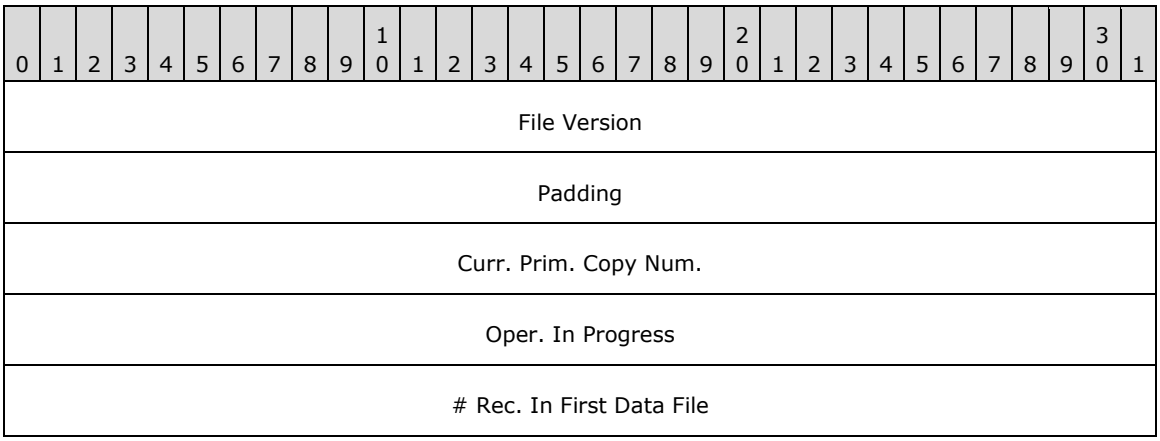
3.1.8 Average Document Length Files

The following example AVDL backup files are part of the example full-text index catalog and are stored in the Average Document Length File format, as specified in section 2.8. Additional AVDL files found in the full-text index catalog have the same structure as these AVDL backup files.

This is the CiAB0002.000 file in the example set.

```
0000    00  00  52  00  00  00  00  00 - 01  00  00  00  00  00  00  00
0010    09  00  00  00  8c  01  00  00 - 00  00  00  00  00  00  00  00
0020    09  00  00  00  8c  01  00  00 - 00  00  00  00  00  00  00  00
0030    53  48  52  46  00  00  00  00 - 00  00  00  00  00  00  00  00
0040    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0050    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0060    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0070    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0080    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0090    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00a0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00b0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00c0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00d0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00e0    00  00  00  00  00  00  00  00 - 00  00  00  00  53  48  52  49
```

The preceding file has the following structure:



Val. Bytes In First File
Unused bytes in First Data File
...
Rec. In Sec. Data File
Val. Bytes In Sec. File
Unused bytes in Sec. Data File
...
Signature 1
First Data File Header & Second Data File Header (variable)
...
Signature 2

File Version (4 bytes): Set to 00 00 52 00.

Padding (4 bytes): Set to 00 00 00 00.

Curr. Prim. Copy Num. (4 bytes): Set to 01 00 00 00.

Oper. In Progress (4 bytes): Set to 00 00 00 00.

 # Rec. In First Data File (4 bytes): Set to 09 00 00 00.

 # Val. Bytes In First File (4 bytes): Set to 8c 01 00 00.

 # Unused bytes in First Data File (8 bytes): Set to 00 00 00 00 00 00 00 00.

 # Rec. In Sec. Data File (4 bytes): Set to 09 00 00 00.

 # Val. Bytes In Sec. File (4 bytes): Set to 8c 01 00 00.

 # Unused bytes in Sec. Data File (8 bytes): Set to 00 00 00 00 00 00 00 00.

Signature 1 (4 bytes): Set to 53 48 52 46.

First Data File Header & Second Data File Header (variable): Set to all zeros.

Signature 2 (4 bytes): Set to 53 48 52 46.

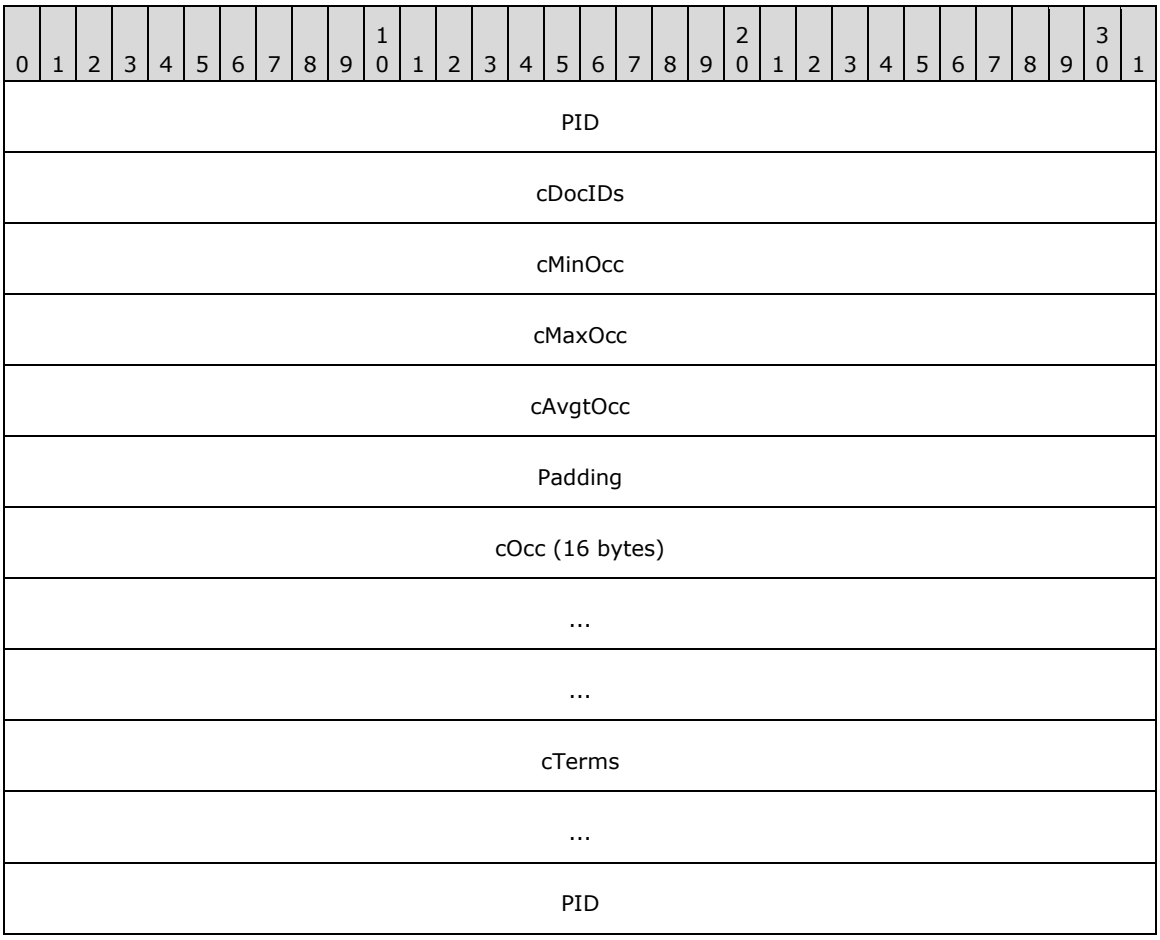
This is the CiAB0002.001 file in the example set:

```
0000    01  00  00  00  96  00  00  00 - 02  00  00  00  02  00  00  00
0010    02  00  00  00  cc  cc  cc  cc - 2c  01  00  00  00  00  00  00
0020    98  00  00  00  00  00  00  00 - 2d  cf  cc  cc  02  00  00  00
0030    98  00  00  00  01  00  00  00 - 02  00  00  00  01  00  00  00
0040    cc  cc  cc  cc  2e  01  00  00 - 00  00  00  00  2f  01  00  00
```

```
0050    00  00  00  00  c7  cf  cc  cc - 07  00  00  00  98  00  00  00
0060    04  00  00  00  07  00  00  00 - 06  00  00  00  cc  cc  cc  cc
0070    d8  03  00  00  00  00  00  00 - 9c  00  00  00  00  00  00  00
0080    f0  d1  cc  cc  38  00  00  00 - 98  00  00  00  01  00  00  00
0090    02  00  00  00  01  00  00  00 - cc  cc  cc  cc  2e  01  00  00
00a0    00  00  00  00  99  00  00  00 - 00  00  00  00  67  cf  cc  cc
00b0    3c  00  00  00  98  00  00  00 - 00  00  00  00  02  00  00  00
00c0    01  00  00  00  cc  cc  cc  cc - 2c  01  00  00  00  00  00  00
00d0    02  00  00  00  00  00  00  00 - d1  ce  cc  cc  62  00  00  00
00e0    96  00  00  00  04  00  00  00 - 04  00  00  00  04  00  00  00
00f0    cc  cc  cc  cc  58  02  00  00 - 00  00  00  00  00  00  00  00
0100    00  00  00  00  28  d0  cc  cc - 05  01  00  00  98  00  00  00
0110    00  00  00  00  00  00  00  00 - 00  00  00  00  cc  cc  cc  cc
0120    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0130    69  ce  cc  cc  2f  01  00  00 - 98  00  00  00  01  00  00  00
0140    03  00  00  00  02  00  00  00 - cc  cc  cc  cc  7a  01  00  00
0150    00  00  00  00  00  00  00  00 - 00  00  00  00  13  d0  cc  cc
0160    ff  ff  fe  7f  98  00  00  00 - 0d  00  00  00  1d  00  00  00
0170    1b  00  00  00  cc  cc  cc  cc - 3a  10  00  00  00  00  00  00
0180    cd  01  00  00  00  00  00  00 - af  df  cb  4c  00  00  00  00
0190    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
01a0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00

ffe0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
fff0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
```

The preceding file has the following structure.



cDocIDs
cMinOcc
... (variable)
...

PID (4 bytes): Set to 01 00 00 00.

cDocIDs (4 bytes): Set to 96 00 00 00.

cMinOcc (4 bytes): Set to 02 00 00 00.

cMaxOcc (4 bytes): Set to 02 00 00 00.

cAvgtOcc (4 bytes): Set to 02 00 00 00.

Padding (4 bytes): set to cc cc cc cc.

cOcc (16 bytes): Set to 2c 01 00 00 00 00 00 00 98 00 00 00 00 00 00 00.

cTerms (8 bytes): Set to 2d cf cc cc 02 00 00 00.

PID (4 bytes): Set to 98 00 00 00.

cDocIDs (4 bytes): Set to 01 00 00 00.

cMinOcc (4 bytes): Set to 02 00 00 00.

... (variable): Continuation.

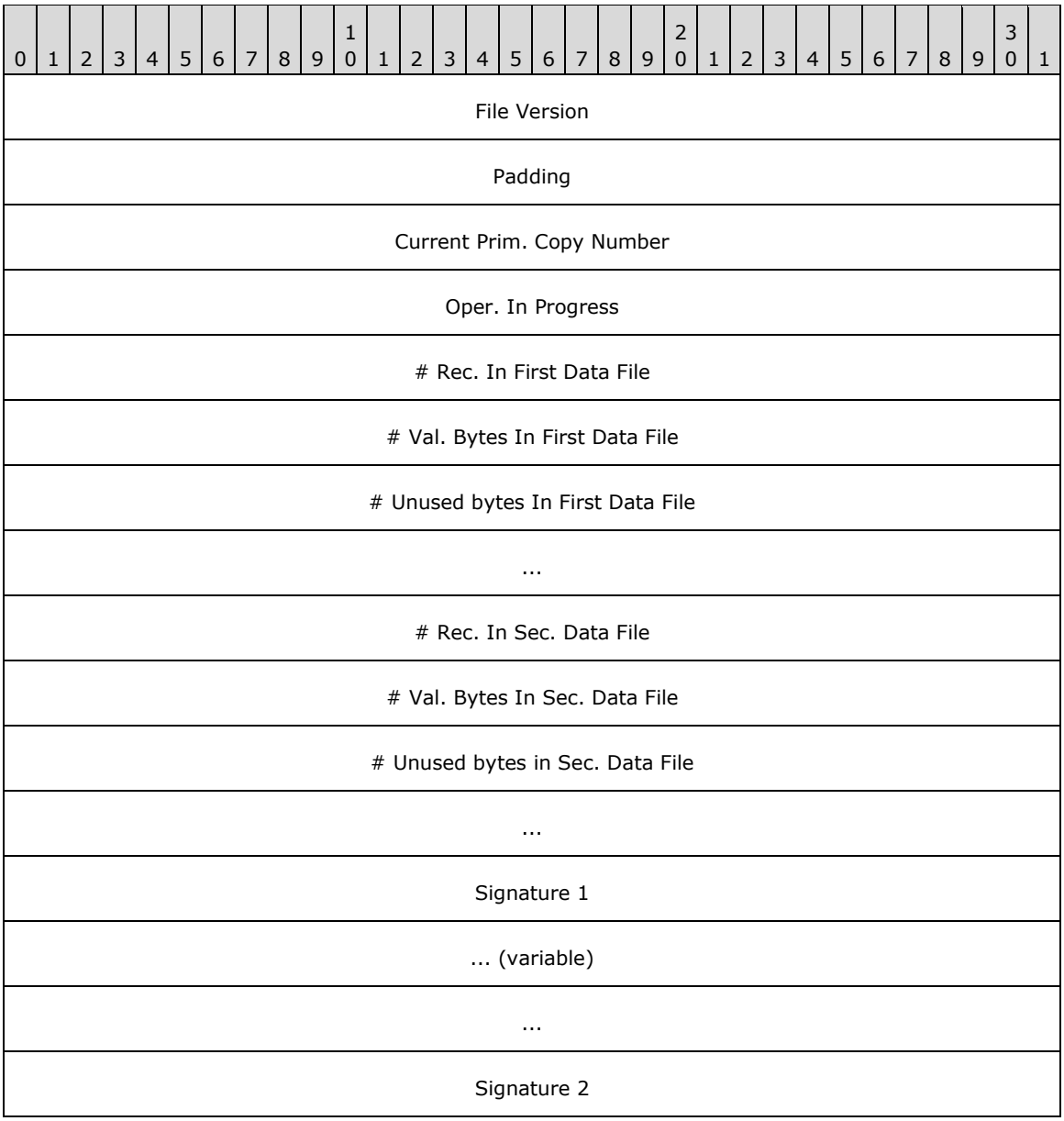
3.1.9 Detected Language Files

The following three example detected language files, as specified in section 2.12, are part of the example full-text index catalog.

This is the CiDL0000.000 file in the example set:

```
0000    00  00  52  00  00  00  00  00 - 01  00  00  00  00  00  00  00
0010    03  00  00  00  20  00  00  00 - 00  00  00  00  00  00  00  00
0020    03  00  00  00  20  00  00  00 - 00  00  00  00  00  00  00  00
0030    53  48  52  46  00  00  00  00 - 00  00  00  00  00  00  00  00
0040    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0050    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0060    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0070    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0080    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0090    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00a0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00b0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00c0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00d0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00e0    00  00  00  00  00  00  00  00 - 00  00  00  00  53  48  52  49
```

The preceding header file has the following structure:



File Version (4 bytes): Set to 00 00 52 00.

Padding (4 bytes): Set to 00 00 00 00.

Current Prim. Copy Number (4 bytes): Set to 01 00 00 00.

Oper. In Progress (4 bytes): Set to 00 00 00 00.

 # Rec. In First Data File (4 bytes): Set to 03 00 00 00.

 # Val. Bytes In First Data File (4 bytes): Set to 20 00 00 00.

 # Unused bytes In First Data File (8 bytes): Set to 00 00 00 00 00 00 00 00.

 # Rec. In Sec. Data File (4 bytes): Set to 03 00 00 00.

 # Val. Bytes In Sec. Data File (4 bytes): Set to 20 00 00 00.

 # Unused bytes in Sec. Data File (8 bytes): Set to 00 00 00 00 00 00 00 00.

Signature 1 (4 bytes): Set to 53 48 52 46.

... (variable): Continuation.

Signature 2 (4 bytes): Set to 53 48 52 49.

This is the CiDL0000.001 file in the example set:

```
0000    ff  ff  ff  ff  ff  ff  ff  ff - 04  00  00  00  00  00  00  00
0010    01  00  00  00  04  00  00  00 - 00  00  80  3f  00  00  80  3f
0020    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0030    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
…
ffe0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
fff0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
```

The preceding data file has the following structure.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Maximum DocID value																															
...																															
Default Value																															
...																															
...																															
Denominator (variable)																															
...																															

Maximum DocID value (8 bytes): Set to ff ff ff ff ff ff ff ff.

Default Value (12 bytes): Set to 04 00 00 00 00 00 00 00 00 00 00 00.

Denominator (variable): Begins with 04 00 00 00 00 00 80 3f 00 00 80 3f.

3.1.10 Query-Independent Rank Files

The following three example Query-Independent Rank files, as specified in section 2.11, are part of the example full-text index catalog.

This is the CiQR0000.000 file in the example set:

```
0000    00  00  53  00  00  00  00  00 - 01  00  00  00  00  00  00  00
0010    05  00  00  00  8c  00  00  00 - 00  00  00  00  00  00  00  00
0020    05  00  00  00  8c  00  00  00 - 00  00  00  00  00  00  00  00
0030    53  48  52  46  00  00  00  00 - 00  00  00  00  00  00  00  00
```

```
0040    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
0050    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
0060    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
0070    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
0080    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
0090    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
00a0    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
00b0    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
00c0    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
00d0    00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
00e0    00 00 00 00 00 00 00 00 - 00 00 00 53 48 52 49
```

The preceding header file has the following structure.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
File Version																															
Padding																															
Current Prim. Copy Number																															
Oper. In Progress																															
# Rec. In First Data File																															
# Val. Bytes In First Data File																															
# Unused bytes In First Data File																															
...																															
# Rec. In Sec. Data File																															
# Val. Bytes In Sec. Data File																															
# Unused bytes In Sec. Data File																															
...																															
Signature 1																															
... (variable)																															
...																															
Signature 2																															

File Version (4 bytes): Set to 00 00 53 00.

Padding (4 bytes): Set to 00 00 00 00.

Current Prim. Copy Number (4 bytes): Set to 01 00 00 00.

Oper. In Progress (4 bytes): Set to 00 00 00 00.

 # Rec. In First Data File (4 bytes): Set to 05 00 00 00.

 # Val. Bytes In First Data File (4 bytes): Set to 8c 00 00 00.

 # Unused bytes In First Data File (8 bytes): Set to 00 00 00 00 00 00 00 00.

 # Rec. In Sec. Data File (4 bytes): Set to 05 00 00 00.

 # Val. Bytes In Sec. Data File (4 bytes): Set to 8c 00 00 00.

 # Unused bytes In Sec. Data File (8 bytes): Set to 00 00 00 00 00 00 00 00.

Signature 1 (4 bytes): Set to 53 48 52 46.

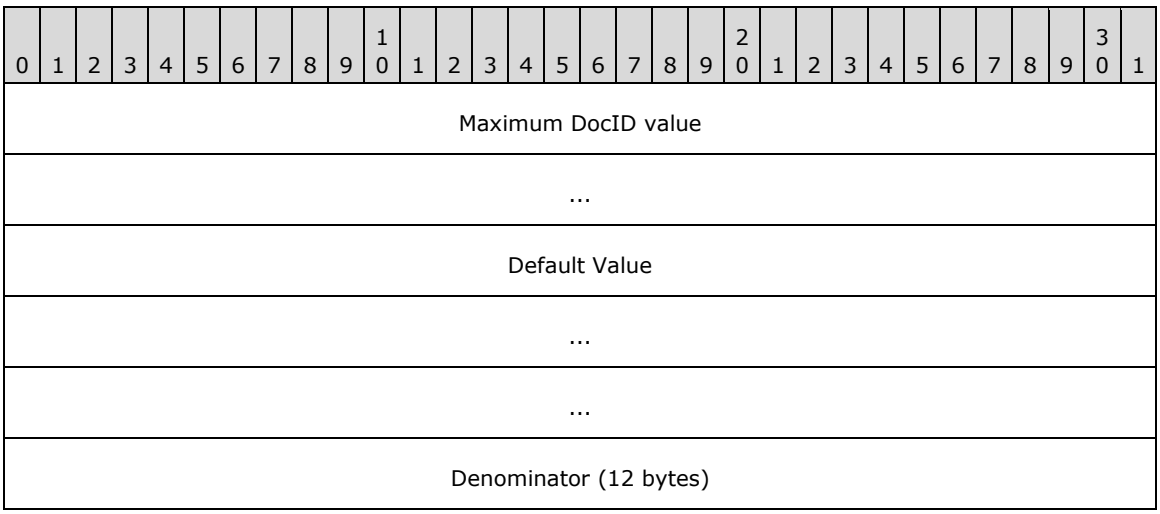
... (variable): Continuation.

Signature 2 (4 bytes): Set to 53 48 52 49.

This is the CiQR0000.001 file in the example set:

```
0000    9a  00  00  00  9a  00  00  00 - 04  00  00  00  a9  f4  07  42
0010    a9  f4  07  42  04  00  00  00 - 95  bf  d6  33  95  bf  d6  33
0020    00  00  00  00  01  00  00  00 - 5c  00  00  00  00  0d  03  00
0030    03  00  03  00  03  00  03  06 - 05  00  05  00  05  00  05  80
0040    06  00  06  00  06  00  06  00 - 06  00  06  00  06  00  06  00
0050    06  00  06  04  07  00  07  00 - 07  00  07  00  07  00  07  00
0060    07  00  07  00  07  00  07  00 - 07  00  07  00  60  4c  42  14
0070    80  2a  91  7a  80  d7  bf  70 - 80  b2  0d  79  80  d7  bf  70
0080    80  06  7c  6f  60  4c  42  14 - 98  38  7a  f7  00  00  00  00
0090    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00a0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
…
ffe0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
fff0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
```

The preceding data file has the following structure:



...
...
Block Number 1
...
Data File Size
Data Field (12 bytes)
...
...
Check Sum
...
...

Maximum DocID value (8 bytes): Set to 9a 00 00 00 9a 00 00 00.

Default Value (12 bytes): Set to 04 00 00 00 a9 f4 07 42 a9 f4 07 42.

Denominator (12 bytes): Set to 04 00 00 00 95 bf d6 33 95 bf d6 33.

Block Number 1 (8 bytes): Set to 00 00 00 00 01 00 00 00.

Data File Size (4 bytes): Set to 5c 00 00 00.

Data Field (12 bytes): Set to 00 0d 03 00 80 06 7c 6f 60 4c 42 14.

Check Sum (4 bytes): Set to 98 38 7a f7.

... (variable): Continuation.

3.1.11 Index Table File

The following three example index table files are part of the example full-text index catalog and are stored in the index table file format, as specified in section 2.13.

This is the INDEX.000 file in the example set:

```
0000    00  00  53  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0010    0b  00  00  00  8c  01  00  00 - 00  00  00  00  00  00  00  00
0020    0b  00  00  00  8c  01  00  00 - 00  00  00  00  00  00  00  00
0030    53  48  52  46  00  00  00  00 - 01  00  00  00  0a  00  00  00
0040    00  00  00  00  01  00  00  00 - 00  00  00  00  00  00  00  00
0050    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0060    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0070    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
```

```
0080    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0090    00  00  00  00  01  00  00  00 - 0a  00  00  00  00  00  00  00
00a0    01  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00b0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00c0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00d0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00e0    00  00  00  00  00  00  00  00 - 00  00  00  00  53  48  52  49
```

The preceding header file has the following structure.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
File Version																															
Padding																															
Current Prim. Copy Number																															
Oper. In Progress																															
# Rec. In First Data File																															
# Val. Bytes In First Data File																															
# Unused bytes In First Data File																															
...																															
# Rec. In Sec. Data File																															
# Val. Bytes In Sec. Data File																															
# Unused bytes In Sec. Data File																															
...																															
Signature 1																															
Reserved 1																															
idCompilationCompleted																															
Reserved2																															
CatalogInitialized																															
... (variable)																															

...
Signature 2

File Version (4 bytes): Set to 00 00 53 00.

Padding (4 bytes): Set to 00 00 00 00.

Current Prim. Copy Number (4 bytes): Set to 00 00 00 00.

Oper. In Progress (4 bytes): Set to 00 00 00 00.

 # Rec. In First Data File (4 bytes): Set to 0b 00 00 00.

 # Val. Bytes In First Data File (4 bytes): Set to 8c 01 00 00.

 # Unused bytes In First Data File (8 bytes): Set to 00 00 00 00 00 00 00 00.

 # Rec. In Sec. Data File (4 bytes): Set to 0b 00 00 00.

 # Val. Bytes In Sec. Data File (4 bytes): Set to 8c 01 00 00.

 # Unused bytes In Sec. Data File (8 bytes): Set to 00 00 00 00 00 00 00 00.

Signature 1 (4 bytes): Set to 53 48 52 46.

Reserved 1 (4 bytes): Set to 00 00 00 00.

idCompilationCompleted (4 bytes): Set to 01 00 00 00.

Reserved2 (4 bytes): Set to 00 00 00 00.

CatalogInitialized (4 bytes): Set to 01 00 00 00.

... (variable): Continuation.

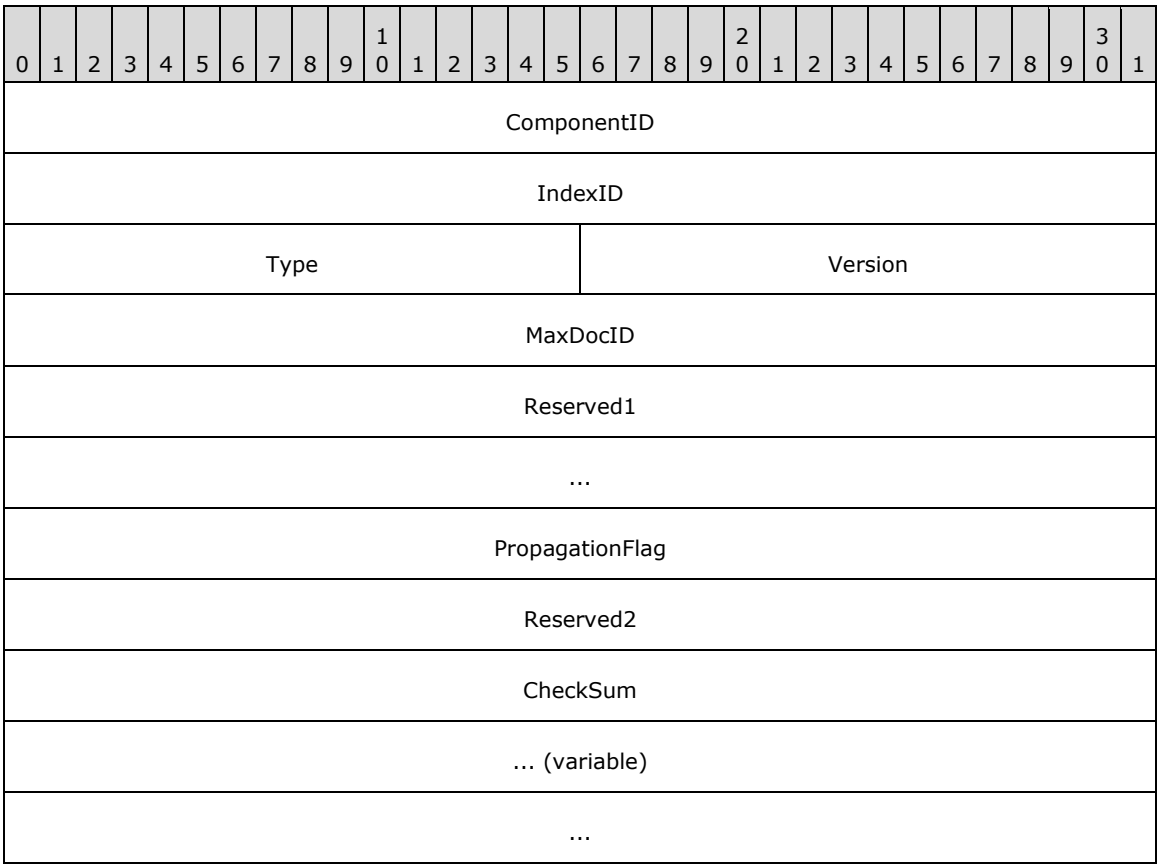
Signature 2: Set to 53 48 52 49.

This is the INDEX.001 file in the example set:

```
0000    00  00  00  00  00  00  01  00 - 04  00  53  00  00  00  00  00
0010    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0020    04  00  54  00  07  00  02  00 - 00  00  01  00  07  00  53  00
0030    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0040    00  00  00  00  0e  00  56  00 - 08  00  01  00  00  00  01  00
0050    09  00  53  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0060    00  00  00  00  00  00  00  00 - 11  00  55  00  08  00  02  00
0070    00  00  01  00  0a  00  53  00 - 00  00  00  00  00  00  00  00
0080    00  00  00  00  00  00  00  00 - 00  00  00  00  12  00  56  00
0090    01  00  01  00  00  00  ff  ff - 03  00  53  00  02  00  00  00
00a0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00b0    06  00  53  00  02  00  01  00 - 00  00  ff  ff  03  00  53  00
00c0    49  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
00d0    00  00  00  00  4e  00  53  00 - 03  00  01  00  00  00  ff  ff
00e0    03  00  53  00  4e  00  00  00 - 00  00  00  00  00  00  00  00
00f0    00  00  00  00  00  00  00  00 - 54  00  53  00  04  00  01  00
0100    00  00  ff  ff  03  00  53  00 - 95  00  00  00  00  00  00  00
0110    00  00  00  00  00  00  00  00 - 00  00  00  00  9c  00  53  00
0120    05  00  01  00  00  00  ff  ff - 03  00  53  00  99  00  00  00
0130    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
0140    a1  00  53  00  06  00  01  00 - 06  00  01  00  00  00  53  00
0150    99  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
```

```
0160    00  00  00  00  a5  00  55  00 - 01  00  00  00  01  00  fe  ff
0170    05  00  53  00  10  03  00  00 - 00  00  00  00  00  00  00  00
0180    00  00  00  00  00  00  00  00 - 17  03  51  00  00  00  00  00
0190    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
01a0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
…
ffe0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
fff0    00  00  00  00  00  00  00  00 - 00  00  00  00  00  00  00  00
```

The preceding data file has the following structure.



ComponentID (4 bytes): Set to 00 00 00 00.

IndexID (4 bytes): Set to 00 00 01 00.

Type (2 bytes): Set to 04 00.

Version (2 bytes): Set to 53 00.

MaxDocID (4 bytes): Set to 00 00 00 00.

Reserved1 (8 bytes): Set to 00 00 00 00 00 00 00 00.

PropagationFlag (4 bytes): Set to 00 00 00 00.

Reserved2 (4 bytes): Set to 00 00 00 00.

CheckSum (4 bytes): Set to 04 00 54 00.

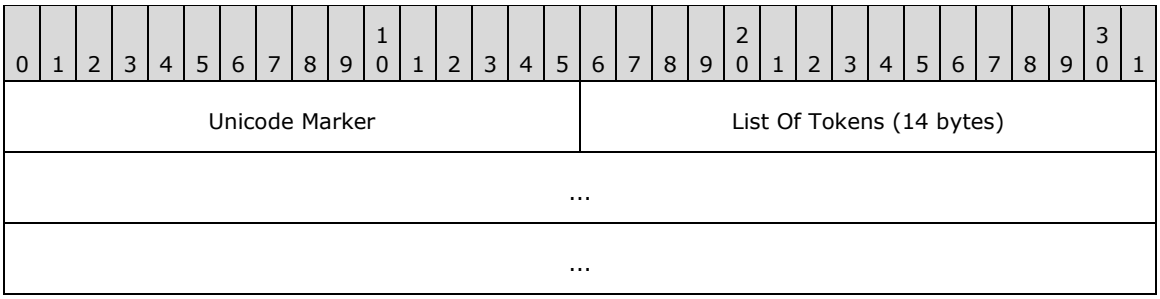
... (variable): Continuation.

3.1.12 Index Lexicon File

The following file is NLGINDEXLEXICON.LEX in the example full-text index catalog is an example of an index lexicon file, as specified in section 2.15.

```
0000    ff  fe  66  00  6f  00  6f  00 - 0d  00  0a  00  74  00  65  00
0010    6d  00  70  00  0d  00  0a  00 -
```

The preceding file has the following structure.



Unicode Marker (2 bytes): Set to ff fe.

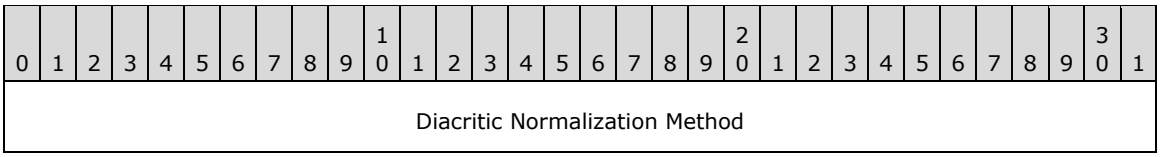
List Of Tokens (14 bytes): Set to 66 00 6f 00 0d 00 0a 00 74 00 65 00.

3.1.13 Diacritic Settings File

The following file is SETTINGS.DIA in the example full-text index catalog and stores a diacritic settings file, as specified in section 2.16.

```
0000    01  00  00  00
```

The preceding file has the following structure.



Diacritic Normalization Method (4 bytes): Set to 01 00 00 00.

3.2 CIX File

This is an example of the layout of a CIX file.

The **content index record** corresponding to the BOF key whose property identifier equals 1 indicates that the CIX file contains a **KeyExtensionData** structure, as specified in section 2.6.1, for this **content index key**. In addition, it indicates that this **KeyExtensionData** structure starts at page zero with bit offset zero.

3.2.1 Physical File on Disk

The CIX file in focus is not empty and not in a merge process, as specified in section 2.9, so the **Empty file filler** and the **Incomplete KeyExtensionData** fields are not present as described in the content index extension file format, as specified in section 2.6.

Offset	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	02	00	00	00	00	00	52	4B	00	00	05	00	00	00	82	00
0010	00	00	80	00	00	00	00	00	00	00	00	00	00	00	82	00
0020	00	00	80	00	00	00	03	00	00	00	82	00	00	00	82	00
0030	00	00	80	00	00	00	07	00	00	00	04	01	00	00	82	00
0040	00	00	80	00	00	00	0A	00	00	00	86	01	00	00	82	00
0050	00	00	80	00	00	00	18	00	94	61	08	02	B0	D2	61	09
0060	0B	55	D8	9A	36	26	6C	15	C9	0E	2B	18	A9	C2	E4	84
0070	8B	30	B1	61	1A	2B	7F	58	12	16	23	EC	87	E1	0E	43
0080	C2	80	B1	66	B0	91	61	73	0C	48	18	9A	5B	1D	36	36
…																
0FF0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1000	04	00	00	00	00	00	02	50	01	00	8A	12	00	00	A1	86
1010	00	00	04	00	59	19	D0	02	2B	0D	00	00	CF	12	27	0D
1020	DD	DD	0C	62	DD	DD	DD	DD	36	F7	DD	DD	DD	DD	DD	DD
1030	DD	DD	DD	DD	DD	DD	36	F7	DD	DD	DD	DD	36	F7	DD	DD
1040	DD	DD	DD	DD	DD	DD	DD	DD	DD	DD	36	F7	DD	DD	DD	DD
1050	36	F7	DD	DD	DD	DD	DD	DD	DD	DD	DD	DD	2B	2E	36	F7
1060	F7	F7	F7	ED	6E	FF	EF	7F	AD	FF	ED	F7	DB	FF	FD	BB
1070	BF	FB	FB	DF	FD	EB	FB	7D	5F	F7	7E	5F	EB	FB	FB	5F
…																
1320	BB	FF	BE	D5	FB	F5	FF	7D	5F	B7	DE	AF	C3	FE	FF	DF
1330	58	24	16	89	8B	C6	A1	91	71	58	34	12	0A	87	45	21
1340	21	B1	28	1C	1C	0A	87	C4	45	A2	90	58	28	2C	0A	8D
…																

Figure 5: Physical file on disk

According to the **BitStream** file format, as specified in section 2.2.1, in the following sections, each 4-byte segments reversed to get a continuous **BitStream**.

3.2.2 ExtensionCompressionTablePage

The first **BitStream page** of the **KeyExtensionData** structure, as specified in section 2.6.1, contains the **ExtensionCompressionTablePage** structure, as specified in section 2.6.1.1, with the data necessary to uncompress the data pages. This data occupies bytes from 0x0000 to 0x0fff inclusive.

3.2.2.1 Page start, symbol category descriptors

Bytes from 0x0000 to 0x059 contain signatures and the **SymbolCategory** structures, as specified in section 2.6.1.1.1, describing the symbols used for compression of data for the **content index key**. Five symbol categories are described in the following section. The first category contains symbols with values from 0 to 0x81, the second category contains symbols from 0x82 to 0x103, and so on. The numbers of bits used to store corresponding elements in the **OccCount bit stream** field of **ExtensionDataPages** are 0, 3, 7, 10 and 24 for the first, second, third, fourth and fifth categories, respectively. The number of bits used to store the element for the first category is 0 because the value is not changed from the previous **document identifier**.

Offset	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	Page signature 0x2				Signature 0x4B52		5 symbol categories				0x82 symbols in the first category				0x80 threshold	
	00	00	00	02	4B	52	00	00	00	05	00	00	00	82	00	00
0010	…		0 bits used in **OccCount bit stream**				Base symbol value is 0				0x82 symbols in the second category				0x80 threshold	
	00	80	00	00	00	00	00	00	00	00	00	00	00	82	00	00
0020	…		3 bits used in **OccCount bit stream**				Base symbol value is 0x82				Third symbol category descriptor					
	00	80	00	00	00	03	00	00	00	82	00	00	00	82	00	00
0030	…										Symbol category descriptors					
	00	80	00	00	00	07	00	00	01	04	00	00	00	82	00	00
0040	…															
	00	80	00	00	00	0A	00	00	01	86	00	00	00	82	00	00
0050	…										Coding table (see next section)					
	00	80	00	00	00	18	00	00	02	08	61	94	09	61	D2	B0

Figure 6: Page start, symbol category descriptors

3.2.2.2 Coding Table

The coding table is stored from the byte offset 0x005A. This table contains the bit sequences for all 650 symbols used in the compression in ascending order of symbol values. For convenience, the following data is expanded in bits and the top row contains bit offsets.

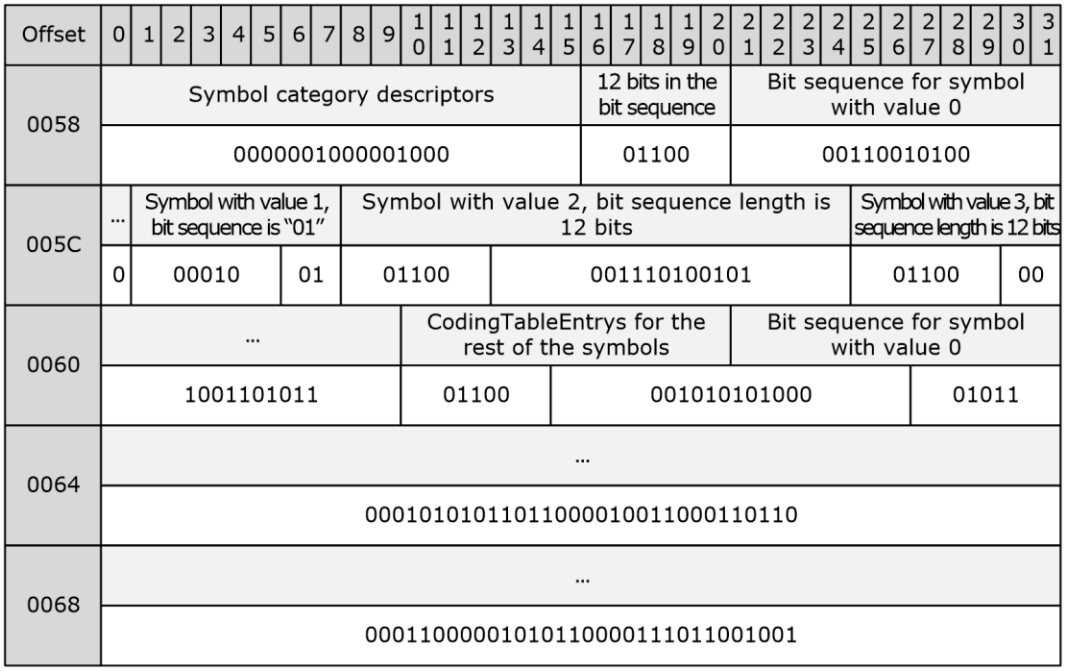
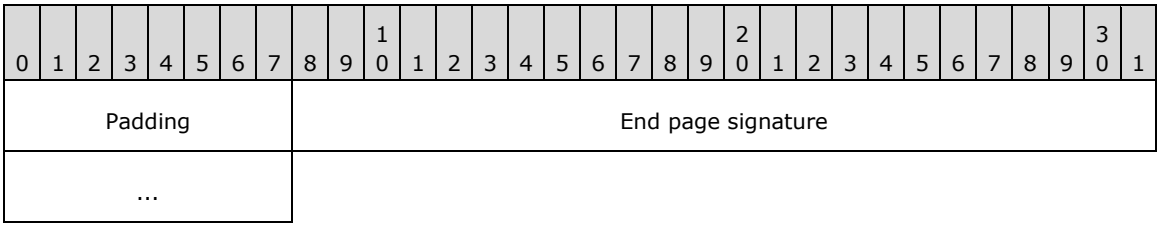


Figure 7: Coding table expanded in bits with Offsets in top row

3.2.2.3 End of Page

The last 4 bytes in the page contain the end-page signature which is the same as the start-page signature.



Padding (1 byte): Set to 00000000.

End page signature (4 bytes): Set to 00000002.

3.2.3 ExtensionDataPage

Bytes from 0x1000 to 0x1fff inclusively contain the first **ExtensionDataPage** structure, as specified in section 2.6.1.2, with encoded **document identifier** data. Because the data is stored for the BOF key, the **OccCount bit stream** field stores the values of **MaxOccBuckets** for corresponding documents.

3.2.3.1 Page start, page directory

The bytes from 0x1000 to 0x105D inclusively contain the **start page signature**, a page tag that indicates that the data page is not the last one in the **KeyExtensionData** structure, as specified in section 2.6.1, and the page directory. The page directory has 2 valid and 6 unused bookmarks.

The first directory bookmark points to the first **document identifier** in the page whose position in the **DOCID bit stream** field is 0x105E (page tag position is 0x1004 plus 0x2D0 bits=0x5A bytes offset) and position in the **OccCount bit stream** is 0x132F and 1 bit (page tag position is 0x1004 plus 0x1959 bits = 0x32B bytes and 1 bit).

Offset	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
1000	Start page signature 0x4								Not the last page		Directory size is 2		Last document identifier on the page 0x128A			
	00		00		00		04		50		02		00		00	
1008	...				0x186A1 document identifiers remaining in the key								The first bookmark points to document identifier 0x4			
	12		8A		00		01		86		A1		00		00	
1010	...				0 document identifiers in the page before this one				The document identifier offset is 0x2D0 bits				The **OccCount bit stream** offset is 0x1959 bits			
	00		04		00		00		02		D0		19		59	
1018	The second bookmark points to document identifier 0xD2B								0xD27 document identifiers in the page before this one				The document identifier offset is 0x12CF bits			
	00		00		0D		2B		0D		27		12		CF	
1020	The **OccCount bit stream** offset is 0x620C bits				Unused bookmarks											
	62		0C		DD		DD		DD		DD		DD		DD	
1028	...															
	DD		DD		F7		36		DD		DD		DD		DD	

Figure 8: Start page signature and page directory

3.2.3.2 DOCID Bit Stream

The **DOCID bit stream** starts from the byte offset 0x105E. For convenience, the following data is expanded in bits and the top row contains bit offsets.

<table>
<tr><td>Offset</td><td>+0</td><td>+1</td><td>+2</td><td>+3</td><td>+4</td><td>+5</td><td>+6</td><td>+7</td><td>+8</td><td>+9</td><td>+A</td><td>+B</td><td>+C</td><td>+D</td><td>+E</td><td>+F</td></tr>
<tr><td rowspan="2">105E</td><td colspan="11">C1</td><td colspan="2">C2</td><td colspan="2">C3</td><td>C4</td></tr>
<tr><td colspan="11">00101110001</td><td colspan="2">01</td><td colspan="2">01</td><td>1</td></tr>
<tr><td rowspan="2">1060</td><td colspan="16">10 codes for symbols with value 0x105 and 3 codes for symbols with value 0x1, DocIDDeltas are 1, document identifiers are 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, MaxOccBuckets are the same or use 7 bits in the OccCount bit stream</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td colspan="2">01</td><td>1</td><td colspan="2">01</td><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="2">01</td><td>1</td><td>1</td></tr>
<tr><td rowspan="2">1062</td><td colspan="16">DOCID bit stream continued</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="2">01</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="2">01</td><td>1</td><td>1</td></tr>
<tr><td rowspan="2">1064</td><td colspan="16">…</td></tr>
<tr><td colspan="2">01</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="2">01</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td rowspan="2">1066</td><td colspan="16">…</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="2">01</td><td>1</td><td>1</td><td colspan="2">01</td><td>1</td><td>1</td><td>0</td></tr>
</table>

Figure 9: DOCID bit stream expanded in bits with Offsets in top row

C1: Code for symbol with value 0x108 which belongs to the third category. **Base symbol value** is 0x104 and the **BitsUsed** is 7 for the category thus corresponding DocIDDelta is 4, document identifier is 4 and **MaxOccBucket** for the **document identifier** is stored in the **OccCount bit stream** using 7 bits.

C2: Code for symbol with value 0x1 which belongs to the first category. **Base symbol value** is 0x0 and the **BitsUsed** is 0 for the category thus corresponding DocIDDelta is 1, document identifier is 5 and **MaxOccBucket** for the document identifier is the same as for previous document identifier.

C3: Code for symbol with value 0x1 which belongs to the first category. **Base symbol value** is 0x0 and the **BitsUsed** is 0 for the category thus corresponding DocIDDelta is 1, document identifier is 6 and **MaxOccBucket** for the document identifier is the same as for previous document identifier.

C4: Code for symbol with value 0x105 which belongs to the third category. **Base symbol value** is 0x104 and the **BitsUsed** is 7 for the category thus corresponding DocIDDelta is 1, document identifier is 7 and **MaxOccBucket** for the document identifier is stored in the **OccCount bit stream** using 7 bits.

3.2.3.3 OccCount Bit Stream

The **OccCount bit stream** starts from the second bit of the byte with offset 0x132F and contains the **MaxOccBuckets** for the corresponding **document identifiers** because the data is for the BOF key.

Offset	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
132C	**DOCID bit stream**																									MaxOccBuckets for document identifiers 4, 5, and 6 are 0x83							
	11011111111111111111111101																									1000011						
1330	MaxOccBucket for document identifier 7 is 0x84								MaxOccBucket for document identifier 8 is 0x85								MaxOccBucket for document identifier 9 is 0x84								MaxOccBucket for document identifiers 10 and 11 are 0x85								…			
	1000100								1000101								1000100									1000101								1000			
1334	**OccCount bit stream** continued																															
	10010001101000011100011010001011																															
1338	…																															
	00010010001101000101100001110001																															

Figure 10: OccCount bit stream

4 Security Considerations

None.

5 Appendix A: Character Normalization Tables

Table 1

Original	Transformed
0x0 - 0x1f	REMOVED
0x41	0x6100
0x42	0x6200
0x43	0x6300
0x44	0x6400
0x45	0x6500
0x46	0x6600
0x47	0x6700
0x48	0x6800
0x49	0x6900
0x4a	0x6a00
0x4b	0x6b00
0x4c	0x6c00
0x4d	0x6d00
0x4e	0x6e00
0x4f	0x6f00
0x50	0x7000
0x51	0x7100
0x52	0x7200
0x53	0x7300
0x54	0x7400
0x55	0x7500
0x56	0x7600
0x57	0x7700
0x58	0x7800
0x59	0x7900
0x5a	0x7a00
0xaa	0x6100
0xb2	0x3200

Original	Transformed
0xb3	0x3300
0xb9	0x3100
0xba	0x6f00
0xc0 - 0xc5	0x6100
0xc6	0x6100 0x6500
0xc7	0x6300
0xc8 - 0xcb	0x6500
0xcc - 0xcf	0x6900
0xd0	0x6400
0xd1	0x6e00
0xd2 - 0xd8	0x6f00
0xd9 - 0xdc	0x7500
0xdd	0x7900
0xde	0x7400 0x6800
0xdf	0x7300 0x7300
0xe0 - 0xe5	0x6100
0xe6	0x6100 0x6500
0xe7	0x6300
0xe8 - 0xeb	0x6500
0xec - 0xef	0x6900
0xf0	0x6400
0xf1	0x6e00
0xf2 - 0xf8	0x6f00
0xf9 - 0xfc	0x7500
0xfd	0x7900
0xfe	0x7400 0x6800
0xff	0x7900
0x100 - 0x105	0x6100
0x106 - 0x10d	0x6300
0x10e - 0x111	0x6400
0x112 - 0x11b	0x6500
0x11c - 0x123	0x6700

Original	Transformed
0x124 - 0x127	0x6800
0x128 - 0x131	0x6900
0x132, 0x133	0x6900 0x6a00
0x134, 0x135	0x6a00
0x136 - 0x138	0x6b00
0x139 - 0x142	0x6c00
0x143 - 0x149	0x6e00
0x14a	0x4b01
0x14c - 0x151	0x6f00
0x152, 0x153	0x6f00 0x6500
0x154 - 0x159	0x7200
0x15a - 0x161	0x7300
0x162 - 0x165	0x7400
0x166	0x6701
0x168 - 0x173	0x7500
0x174, 0x175	0x7700
0x176 - 0x178	0x7900
0x179 - 0x17e	0x7a00
0x180 - 0x185	0x6200
0x186 - 0x188	0x6300
0x18a - 0x18d	0x6400
0x18e - 0x190	0x6500
0x191, 0x192	0x6600
0x193, 0x194	0x6700
0x195	0x6800
0x196, 0x197	0x6900
0x198, 0x199	0x6b00
0x19a, 0x19b	0x6c00
0x19c	0x6d00
0x19d, 0x19e	0x6e00
0x19f - 0x1a3	0x6f00
0x1a4, 0x1a5	0x7000

Original	Transformed
0x1a6	0x7200
0x1a7 - 0x1aa	0x7300
0x1ab - 0x1ae	0x7400
0x1af - 0x1b1	0x7500
0x1b2	0x7600
0x1b3, 0x1b4	0x7900
0x1b5, 0x1b6	0x7a00
0x1b7 - 0x1ba	0x9202
0x1bb	0x3200
0x1bc	0xbd01
0x1bf	0x7700
0x1c4 - 0x1c6	0x6400 0x7a00
0x1c7 - 0x1c9	0x6c00 0x6a00
0x1ca - 0x1cc	0x6e00 0x6a00
0x1cd, 0x1ce	0x6100
0x1cf, 0x1d0	0x6900
0x1d1, 0x1d2	0x6f00
0x1d3 - 0x1dc	0x7500
0x1dd	0x6500
0x1de - 0x1e1	0x6100
0x1e2, 0x1e3	0x6100 0x6500
0x1e4 - 0x1e7	0x6700
0x1e8, 0x1e9	0x6b00
0x1ea - 0x1ed	0x6f00
0x1ee, 0x1ef	0x9202
0x1f0	0x6a00
0x1f1 - 0x1f3	0x6400 0x7a00
0x1f4, 0x1f5	0x6700
0x1fa, 0x1fb	0x6100
0x1fc, 0x1fd	0x6100 0x6500
0x1fe, 0x1ff	0x6f00
0x200 - 0x203	0x6100

Original	Transformed
0x204 - 0x207	0x6500
0x208 - 0x20b	0x6900
0x20c - 0x20f	0x6f00
0x210 - 0x213	0x7200
0x214 - 0x217	0x7500
0x218, 0x219	0x7300
0x21a, 0x21b	0x7400
0x220 - 0x24f	REMOVED
0x250 - 0x252	0x6100
0x253	0x6200
0x254, 0x255	0x6300
0x256, 0x257	0x6400
0x258 - 0x25e	0x6500
0x25f	0x6a00
0x260 - 0x264	0x6700
0x265, 0x266	0x6800
0x268 - 0x26a	0x6900
0x26b - 0x26e	0x6c00
0x26f - 0x271	0x6d00
0x272 - 0x274	0x6e00
0x275 - 0x277	0x6f00
0x278	0x7000
0x279 - 0x281	0x7200
0x282	0x7300
0x283	0x6500
0x284	0x6a00
0x285, 0x286	0x6500
0x287, 0x288	0x7400
0x289, 0x28a	0x7500
0x28b, 0x28c	0x7600
0x28d	0x7700
0x28e, 0x28f	0x7900

Original	Transformed
0x290, 0x291	0x7a00
0x293	0x9202
0x294 - 0x296	0xbe01
0x297	0x6300
0x299	0x6200
0x29a	0x6500
0x29b	0x6700
0x29c	0x6800
0x29d	0x6a00
0x29e	0x6b00
0x29f	0x6c00
0x2a0	0x7100
0x2a1, 0x2a2	0xbe01
0x2a4	0x6400
0x2a5	0xa302
0x2a7, 0x2a8	0x7400
0x2ae, 0x2af	REMOVED
0x2b0, 0x2b1	0x6800
0x2b2	0x6a00
0x2b3 - 0x2b6	0x7200
0x2b7	0x7700
0x2b8	0x7900
0x2b9 - 0x2c5	REMOVED
0x2c6	0x5e00
0x2c8	REMOVED
0x2c9	0xaf00
0x2ca	0xb400
0x2cb	0x6000
0x2cc - 0x2d7	REMOVED
0x2d8	0xc702
0x2de, 0x2df	REMOVED
0x2e0	0x6700

Original	Transformed
0x2e1	0x6c00
0x2e2	0x7300
0x2e3	0x7800
0x2e5 - 0x383	REMOVED
0x386	0xb103
0x387	REMOVED
0x388	0xb503
0x389	0xb703
0x38a	0xb903
0x38b	REMOVED
0x38c	0xbf03
0x38d	REMOVED
0x38e	0xc503
0x38f	0xc903
0x390	0xb903
0x391	0xb103
0x392	0xb203
0x393	0xb303
0x394	0xb403
0x395	0xb503
0x396	0xb603
0x397	0xb703
0x398	0xb803
0x399	0xb903
0x39a	0xba03
0x39b	0xbb03
0x39c	0xbc03
0x39d	0xbd03
0x39e	0xbe03
0x39f	0xbf03
0x3a0	0xc003
0x3a1	0xc103

Original	Transformed
0x3a2	REMOVED
0x3a3	0xc303
0x3a4	0xc403
0x3a5	0xc503
0x3a6	0xc603
0x3a7	0xc703
0x3a8	0xc803
0x3a9	0xc903
0x3aa	0xb903
0x3ab	0xc503
0x3ac	0xb103
0x3ad	0xb503
0x3ae	0xb703
0x3af	0xb903
0x3b0	0xc503
0x3c2	0xc303
0x3ca	0xb903
0x3cb	0xc503
0x3cc	0xbf03
0x3cd	0xc503
0x3ce	0xc903
0x3cf	REMOVED
0x3d0	0xb203
0x3d1	0xb803
0x3d2 - 0x3d4	0xc503
0x3d5	0xc603
0x3d6	0xc003
0x3d8, 0x3d9	REMOVED
0x3da	0xdb03
0x3dc	0xdd03
0x3de	0xdf03
0x3e0	0xe103

Original	Transformed
0x3e2	0xe303
0x3e4	0xe503
0x3e6	0xe703
0x3e8	0xe903
0x3ea	0xeb03
0x3ec	0xed03
0x3ee	0xef03
0x3f0	0xba03
0x3f1	0xc103
0x3f2	0xc303
0x3f4 - 0x3ff	REMOVED
0x401	0x3504
0x402	0x5204
0x403	0x3304
0x404	0x5404
0x405	0x5504
0x406	0x5604
0x407	0x5704
0x408	0x5804
0x409	0x5904
0x40a	0x5a04
0x40b	0x5b04
0x40c	0x3a04
0x40e	0x5e04
0x40f	0x5f04
0x410	0x3004
0x411	0x3104
0x412	0x3204
0x413	0x3304
0x414	0x3404
0x415	0x3504
0x416	0x3604

Original	Transformed
0x417	0x3704
0x418	0x3804
0x419	0x3904
0x41a	0x3a04
0x41b	0x3b04
0x41c	0x3c04
0x41d	0x3d04
0x41e	0x3e04
0x41f	0x3f04
0x420	0x4004
0x421	0x4104
0x422	0x4204
0x423	0x4304
0x424	0x4404
0x425	0x4504
0x426	0x4604
0x427	0x4704
0x428	0x4804
0x429	0x4904
0x42a	0x4a04
0x42b	0x4b04
0x42c	0x4c04
0x42d	0x4d04
0x42e	0x4e04
0x42f	0x4f04
0x451	0x3504
0x453	0x3304
0x45c	0x3a04
0x460	0x6104
0x463	0x3d04
0x464	0x6504
0x466	0x6704

Original	Transformed
0x468	0x6904
0x46a	0x6b04
0x46c	0x6d04
0x46e	0x6f04
0x470	0x7104
0x472	0x7304
0x474 - 0x477	0x7504
0x478	0x7904
0x47a	0x7b04
0x47c	0x7d04
0x47e	0x7f04
0x480	0x8104
0x483 - 0x48b	REMOVED
0x490 - 0x495	0x3304
0x496, 0x497	0x3604
0x498, 0x499	0x3704
0x49a - 0x4a1	0x3a04
0x4a2 - 0x4a5	0x3d04
0x4a6, 0x4a7	0x3f04
0x4a8, 0x4a9	0x3e04
0x4aa, 0x4ab	0x4104
0x4ac, 0x4ad	0x4204
0x4ae, 0x4af	0x4304
0x4b0	0xb104
0x4b2, 0x4b3	0x4504
0x4b4, 0x4b5	0x4604
0x4b6 - 0x4b9	0x4704
0x4ba, 0x4bb	0x3d04
0x4bc - 0x4bf	0x3504
0x4c0	0x5604
0x4c1, 0x4c2	0x3604
0x4c3, 0x4c4	0x3a04

Original	Transformed
0x4c5, 0x4c6	REMOVED
0x4c7, 0x4c8	0x3d04
0x4c9, 0x4ca	REMOVED
0x4cb, 0x4cc	0x4704
0x4cd - 0x4cf	REMOVED
0x4d8, 0x4d9	0x3504
0x4e8	0xe904
0x4f6 - 0x530	REMOVED
0x531	0x6105
0x532	0x6205
0x533	0x6305
0x534	0x6405
0x535	0x6505
0x536	0x6605
0x537	0x6705
0x538	0x6805
0x539	0x6905
0x53a	0x6a05
0x53b	0x6b05
0x53c	0x6c05
0x53d	0x6d05
0x53e	0x6e05
0x53f	0x6f05
0x540	0x7005
0x541	0x7105
0x542	0x7205
0x543	0x7305
0x544	0x7405
0x545	0x7505
0x546	0x7605
0x547	0x7705
0x548	0x7805

Original	Transformed
0x549	0x7905
0x54a	0x7a05
0x54b	0x7b05
0x54c	0x7c05
0x54d	0x7d05
0x54e	0x7e05
0x54f	0x7f05
0x550	0x8005
0x551	0x8105
0x552	0x8205
0x553	0x8305
0x554	0x8405
0x555	0x8505
0x556	0x8605
0x557 - 0x5cf	REMOVED
0x5db	0xda05
0x5de	0xdd05
0x5e0	0xdf05
0x5e4	0xe305
0x5e6	0xe505
0x5eb - 0x5ef	REMOVED
0x5f0	0xd505 0xd505
0x5f1	0xd505 0xd905
0x5f2	0xd905 0xd905
0x5f5 - 0x620	REMOVED
0x622, 0x623	0x2706
0x624	0x2106
0x625	0x2706
0x626	0x2106
0x62a	0x2906
0x63b - 0x640	REMOVED
0x64a	0x4906

Original	Transformed
0x656 - 0x65f	REMOVED
0x660	0x3000
0x661	0x3100
0x662	0x3200
0x663	0x3300
0x664	0x3400
0x665	0x3500
0x666	0x3600
0x667	0x3700
0x668	0x3800
0x669	0x3900
0x66d - 0x66f	REMOVED
0x670 - 0x675	0x2706
0x6a9, 0x6aa	0x4306
0x6dd - 0x6ef	REMOVED
0x6f0	0x3000
0x6f1	0x3100
0x6f2	0x3200
0x6f3	0x3300
0x6f4	0x3400
0x6f5	0x3500
0x6f6	0x3600
0x6f7	0x3700
0x6f8	0x3800
0x6f9	0x3900
0x6fd - 0x904	REMOVED
0x929	0x2809
0x931	0x3009
0x934	0x3309
0x93a - 0x957	REMOVED
0x958	0x1509
0x959	0x1609

Original	Transformed
0x95a	0x1709
0x95b	0x1c09
0x95c	0x2109
0x95d	0x2209
0x95e	0x2b09
0x95f	0x2f09
0x966	0x3000
0x967	0x3100
0x968	0x3200
0x969	0x3300
0x96a	0x3400
0x96b	0x3500
0x96c	0x3600
0x96d	0x3700
0x96e	0x3800
0x96f	0x3900
0x971 - 0x9db	REMOVED
0x9dc	0xa109
0x9dd	0xa209
0x9de	REMOVED
0x9df	0xaf09
0x9e4, 0x9e5	REMOVED
0x9e6	0x3000
0x9e7	0x3100
0x9e8	0x3200
0x9e9	0x3300
0x9ea	0x3400
0x9eb	0x3500
0x9ec	0x3600
0x9ed	0x3700
0x9ee	0x3800
0x9ef	0x3900

Original	Transformed
0x9fb - 0xa31	REMOVED
0xa33	0x320a
0xa34	REMOVED
0xa36	0x380a
0xa37 - 0xa58	REMOVED
0xa59	0x160a
0xa5a	0x170a
0xa5b	0x1c0a
0xa5d	REMOVED
0xa5e	0x2b0a
0xa5f - 0xa65	REMOVED
0xa66	0x3000
0xa67	0x3100
0xa68	0x3200
0xa69	0x3300
0xa6a	0x3400
0xa6b	0x3500
0xa6c	0x3600
0xa6d	0x3700
0xa6e	0x3800
0xa6f	0x3900
0xa75 - 0xabc	REMOVED
0xac4	0xc30a
0xac6 - 0xadf	REMOVED
0xae0	0x8b0a
0xae1 - 0xae5	REMOVED
0xae6	0x3000
0xae7	0x3100
0xae8	0x3200
0xae9	0x3300
0xaea	0x3400
0xaeb	0x3500

Original	Transformed
0xaec	0x3600
0xaed	0x3700
0xaee	0x3800
0xaef	0x3900
0xaf0 - 0xb5b	REMOVED
0xb5c	0x210b
0xb5d	0x220b
0xb5e	REMOVED
0xb5f	0x2f0b
0xb62 - 0xb65	REMOVED
0xb66	0x3000
0xb67	0x3100
0xb68	0x3200
0xb69	0x3300
0xb6a	0x3400
0xb6b	0x3500
0xb6c	0x3600
0xb6d	0x3700
0xb6e	0x3800
0xb6f	0x3900
0xb71 - 0xbe6	REMOVED
0xbe7	0x3100
0xbe8	0x3200
0xbe9	0x3300
0xbea	0x3400
0xbeb	0x3500
0xbec	0x3600
0xbed	0x3700
0xbee	0x3800
0xbef	0x3900
0xbf3 - 0xc65	REMOVED
0xc66	0x3000

Original	Transformed
0xc67	0x3100
0xc68	0x3200
0xc69	0x3300
0xc6a	0x3400
0xc6b	0x3500
0xc6c	0x3600
0xc6d	0x3700
0xc6e	0x3800
0xc6f	0x3900
0xc70 - 0xcdf	REMOVED
0xce0	0xb00c
0xce2 - 0xce5	REMOVED
0xce6	0x3000
0xce7	0x3100
0xce8	0x3200
0xce9	0x3300
0xcea	0x3400
0xceb	0x3500
0xcec	0x3600
0xced	0x3700
0xcee	0x3800
0xcef	0x3900
0xcf0 - 0xd65	REMOVED
0xd66	0x3000
0xd67	0x3100
0xd68	0x3200
0xd69	0x3300
0xd6a	0x3400
0xd6b	0x3500
0xd6c	0x3600
0xd6d	0x3700
0xd6e	0x3800

Original	Transformed
0xd6f	0x3900
0xd70 - 0xecf	REMOVED
0xed0	0x3000
0xed1	0x3100
0xed2	0x3200
0xed3	0x3300
0xed4	0x3400
0xed5	0x3500
0xed6	0x3600
0xed7	0x3700
0xed8	0x3800
0xed9	0x3900
0xeda - 0x109f	REMOVED
0x10a0	0xd010
0x10a1	0xd110
0x10a2	0xd210
0x10a3	0xd310
0x10a4	0xd410
0x10a5	0xd510
0x10a6	0xd610
0x10a7	0xd710
0x10a8	0xd810
0x10a9	0xd910
0x10aa	0xda10
0x10ab	0xdb10
0x10ac	0xdc10
0x10ad	0xdd10
0x10ae	0xde10
0x10af	0xdf10
0x10b0	0xe010
0x10b1	0xe110
0x10b2	0xe210

Original	Transformed
0x10b3	0xe310
0x10b4	0xe410
0x10b5	0xe510
0x10b6	0xe610
0x10b7	0xe710
0x10b8	0xe810
0x10b9	0xe910
0x10ba	0xea10
0x10bb	0xeb10
0x10bc	0xec10
0x10bd	0xed10
0x10be	0xee10
0x10bf	0xef10
0x10c0	0xf010
0x10c1	0xf110
0x10c2	0xf210
0x10c3	0xf310
0x10c4	0xf410
0x10c5	0xf510
0x10c6 - 0x1dff	REMOVED
0x1e00, 0x1e01	0x6100
0x1e02 - 0x1e07	0x6200
0x1e08, 0x1e09	0x6300
0x1e0a - 0x1e13	0x6400
0x1e14 - 0x1e1d	0x6500
0x1e1e, 0x1e1f	0x6600
0x1e20, 0x1e21	0x6700
0x1e22 - 0x1e2b	0x6800
0x1e2c - 0x1e2f	0x6900
0x1e30 - 0x1e35	0x6b00
0x1e36 - 0x1e3d	0x6c00
0x1e3e - 0x1e43	0x6d00

Original	Transformed
0x1e44 - 0x1e4b	0x6e00
0x1e4c - 0x1e53	0x6f00
0x1e54 - 0x1e57	0x7000
0x1e58 - 0x1e5f	0x7200
0x1e60 - 0x1e69	0x7300
0x1e6a - 0x1e71	0x7400
0x1e72 - 0x1e7b	0x7500
0x1e7c - 0x1e7f	0x7600
0x1e80 - 0x1e89	0x7700
0x1e8a - 0x1e8d	0x7800
0x1e8e, 0x1e8f	0x7900
0x1e90 - 0x1e95	0x7a00
0x1e96	0x6800
0x1e97	0x7400
0x1e98	0x7700
0x1e99	0x7900
0x1e9a	0x6100
0x1e9c - 0x1e9f	REMOVED
0x1ea0 - 0x1eb7	0x6100
0x1eb8 - 0x1ec7	0x6500
0x1ec8 - 0x1ecb	0x6900
0x1ecc - 0x1ee3	0x6f00
0x1ee4 - 0x1ef1	0x7500
0x1ef2 - 0x1ef9	0x7900
0x1efa - 0x202f	REMOVED
0x2045	0x5b00
0x2046	0x5d00
0x2047 - 0x206f	REMOVED
0x2070	0x3000
0x2071 - 0x2073	REMOVED
0x2074	0x3400
0x2075	0x3500

Original	Transformed
0x2076	0x3600
0x2077	0x3700
0x2078	0x3800
0x2079	0x3900
0x207a	0x2b00
0x207b	0x2d00
0x207c	0x3d00
0x207d	0x2800
0x207e	0x2900
0x207f	0x6e00
0x2080	0x3000
0x2081	0x3100
0x2082	0x3200
0x2083	0x3300
0x2084	0x3400
0x2085	0x3500
0x2086	0x3600
0x2087	0x3700
0x2088	0x3800
0x2089	0x3900
0x208a	0x2b00
0x208b	0x2d00
0x208c	0x3d00
0x208d	0x2800
0x208e	0x2900
0x208f - 0x20ff	REMOVED
0x2102, 0x2103	0x6300
0x2107	0x6500
0x2109	0x6600
0x210a	0x6700
0x210b - 0x210f	0x6800
0x2110, 0x2111	0x6900

Original	Transformed
0x2112, 0x2113	0x6c00
0x2115	0x6e00
0x2118, 0x2119	0x7000
0x211a	0x7100
0x211b - 0x211d	0x7200
0x2124, 0x2125	0x7a00
0x2126, 0x2127	0xc903
0x2128	0x7a00
0x2129	0xc903
0x212b	0x6100
0x212c	0x6200
0x212d	0x6300
0x212e - 0x2130	0x6500
0x2131, 0x2132	0x6600
0x2133	0x6d00
0x2134	0x6f00
0x2135	0xd005
0x2136	0xd105
0x2137	0xd205
0x2138	0xd305
0x213a - 0x2152	REMOVED
0x215f	0x3100
0x2160	0x7021
0x2161	0x7121
0x2162	0x7221
0x2163	0x7321
0x2164	0x7421
0x2165	0x7521
0x2166	0x7621
0x2167	0x7721
0x2168	0x7821
0x2169	0x7921

Original	Transformed
0x216a	0x7a21
0x216b	0x7b21
0x216c	0x7c21
0x216d	0x7d21
0x216e	0x7e21
0x2184 - 0x218f	REMOVED
0x219a	0x9021
0x219b	0x9221
0x219c	0x9021
0x219d	0x9221
0x219e	0x9021
0x219f	0x9121
0x21a0	0x9221
0x21a1	0x9321
0x21a2	0x9021
0x21a3	0x9221
0x21a4	0x9021
0x21a5	0x9121
0x21a6	0x9221
0x21a7	0x9321
0x21a8	0x9521
0x21a9	0x9021
0x21aa	0x9221
0x21ab	0x9021
0x21ac	0x9221
0x21ad, 0x21ae	0x9421
0x21af	0x9321
0x21b0	0x9021
0x21b1	0x9221
0x21b2	0x9021
0x21b3	0x9221
0x21b4	0x9321

Original	Transformed
0x21b5	0x9021
0x21b8	0x9621
0x21b9	0x9421
0x21ba	0xb621
0x21bb	0xb721
0x21bc, 0x21bd	0x9021
0x21be, 0x21bf	0x9121
0x21c0, 0x21c1	0x9221
0x21c2, 0x21c3	0x9321
0x21c4	0x9421
0x21c5	0x9521
0x21c6	0x9421
0x21c7	0x9021
0x21c8	0x9121
0x21c9	0x9221
0x21ca	0x9321
0x21cb, 0x21cc	0x9421
0x21cd	0x9021
0x21ce	0x9421
0x21cf	0x9021
0x21d0	0x9221
0x21d1	0x9121
0x21d2	0x9221
0x21d3	0x9321
0x21d4	0x9421
0x21d5	0x9521
0x21d6	0x9621
0x21d7	0x9721
0x21d8	0x9821
0x21d9	0x9921
0x21da	0x9021
0x21db	0x9221

Original	Transformed
0x21dc	0x9021
0x21dd	0x9221
0x21de	0x9121
0x21df	0x9321
0x21e0	0x9021
0x21e1	0x9121
0x21e2	0x9221
0x21e3	0x9321
0x21e4	0x9021
0x21e5	0x9221
0x21e6	0x9021
0x21e7	0x9121
0x21e8	0x9221
0x21e9	0x9321
0x21ea	0x9121
0x21eb - 0x21ff	REMOVED
0x2295	0x2b00
0x2296	0x1222
0x2298	0x1522
0x2299	0x1922
0x229a	0x1822
0x229b	0x1722
0x229c	0x3d00
0x229e	0x2b00
0x229f	0x1222
0x22a0	0x9722
0x22a1	0x1922
0x22f2 - 0x245f	REMOVED
0x2460	0x3100
0x2461	0x3200
0x2462	0x3300
0x2463	0x3400

Original	Transformed
0x2464	0x3500
0x2465	0x3600
0x2466	0x3700
0x2467	0x3800
0x2468	0x3900
0x2474	0x3100
0x2475	0x3200
0x2476	0x3300
0x2477	0x3400
0x2478	0x3500
0x2479	0x3600
0x247a	0x3700
0x247b	0x3800
0x247c	0x3900
0x247d	0x6924
0x247e	0x6a24
0x247f	0x6b24
0x2480	0x6c24
0x2481	0x6d24
0x2482	0x6e24
0x2483	0x6f24
0x2484	0x7024
0x2485	0x7124
0x2486	0x7224
0x2487	0x7324
0x2488	0x3100
0x2489	0x3200
0x248a	0x3300
0x248b	0x3400
0x248c	0x3500
0x248d	0x3600
0x248e	0x3700

Original	Transformed
0x248f	0x3800
0x2490	0x3900
0x2491	0x6924
0x2492	0x6a24
0x2493	0x6b24
0x2494	0x6c24
0x2495	0x6d24
0x2496	0x6e24
0x2497	0x6f24
0x2498	0x7024
0x2499	0x7124
0x249a	0x7224
0x249b	0x7324
0x249c	0x6100
0x249d	0x6200
0x249e	0x6300
0x249f	0x6400
0x24a0	0x6500
0x24a1	0x6600
0x24a2	0x6700
0x24a3	0x6800
0x24a4	0x6900
0x24a5	0x6a00
0x24a6	0x6b00
0x24a7	0x6c00
0x24a8	0x6d00
0x24a9	0x6e00
0x24aa	0x6f00
0x24ab	0x7000
0x24ac	0x7100
0x24ad	0x7200
0x24ae	0x7300

Original	Transformed
0x24af	0x7400
0x24b0	0x7500
0x24b1	0x7600
0x24b2	0x7700
0x24b3	0x7800
0x24b4	0x7900
0x24b5	0x7a00
0x24b6	0x6100
0x24b7	0x6200
0x24b8	0x6300
0x24b9	0x6400
0x24ba	0x6500
0x24bb	0x6600
0x24bc	0x6700
0x24bd	0x6800
0x24be	0x6900
0x24bf	0x6a00
0x24c0	0x6b00
0x24c1	0x6c00
0x24c2	0x6d00
0x24c3	0x6e00
0x24c4	0x6f00
0x24c5	0x7000
0x24c6	0x7100
0x24c7	0x7200
0x24c8	0x7300
0x24c9	0x7400
0x24ca	0x7500
0x24cb	0x7600
0x24cc	0x7700
0x24cd	0x7800
0x24ce	0x7900

Original	Transformed
0x24cf	0x7a00
0x24d0	0x6100
0x24d1	0x6200
0x24d2	0x6300
0x24d3	0x6400
0x24d4	0x6500
0x24d5	0x6600
0x24d6	0x6700
0x24d7	0x6800
0x24d8	0x6900
0x24d9	0x6a00
0x24da	0x6b00
0x24db	0x6c00
0x24dc	0x6d00
0x24dd	0x6e00
0x24de	0x6f00
0x24df	0x7000
0x24e0	0x7100
0x24e1	0x7200
0x24e2	0x7300
0x24e3	0x7400
0x24e4	0x7500
0x24e5	0x7600
0x24e6	0x7700
0x24e7	0x7800
0x24e8	0x7900
0x24e9	0x7a00
0x24ea	0x3000
0x24eb - 0x24ff	REMOVED
0x2501	0x25
0x2503	0x225
0x2504, 0x2505	0x25

Original	Transformed
0x2506, 0x2507	0x225
0x2508, 0x2509	0x25
0x250a, 0x250b	0x225
0x250d - 0x250f	0xc25
0x2511 - 0x2513	0x1025
0x2515 - 0x2517	0x1425
0x2519 - 0x251b	0x1825
0x251d - 0x2523	0x1c25
0x2525 - 0x252b	0x2425
0x252d - 0x2533	0x2c25
0x2535 - 0x253b	0x3425
0x253d - 0x254b	0x3c25
0x254c, 0x254d	0x25
0x254e, 0x254f	0x225
0x2550	0x25
0x2551	0x225
0x2552 - 0x2554	0xc25
0x2555 - 0x2557	0x1025
0x2558 - 0x255a	0x1425
0x255b - 0x255d	0x1825
0x255e - 0x2560	0x1c25
0x2561 - 0x2563	0x2425
0x2564 - 0x2566	0x2c25
0x2567 - 0x2569	0x3425
0x256a - 0x256c	0x3c25
0x256d	0xc25
0x256e	0x1025
0x256f	0x1825
0x2570	0x1425
0x2578	0x7425
0x2579	0x7525
0x257a	0x7625

Original	Transformed
0x257b	0x7725
0x257c	0x25
0x257d	0x225
0x257e	0x25
0x257f	0x225
0x2584	0x8025
0x2589	0x8725
0x258a	0x8625
0x258b	0x8525
0x258c	0x8025
0x258d	0x8325
0x258e	0x8225
0x258f	0x8125
0x2590	0x8025
0x2591 - 0x2593	0x8825
0x2594, 0x2595	0x8125
0x2596 - 0x259f	REMOVED
0x25a1 - 0x25ab	0xa025
0x25ad	0xac25
0x25af	0xae25
0x25b1	0xb025
0x25b3 - 0x25b5	0xb225
0x25b6 - 0x25b9	0xef25
0x25bb	0xba25
0x25bd - 0x25bf	0xbc25
0x25c1 - 0x25c3	0xc025
0x25c5	0xc425
0x25c7, 0x25c8	0xc625
0x25cc - 0x25d9	0xcb25
0x25e6	0xd825
0x25e7 - 0x25eb	0xa025
0x25ec - 0x25ee	0xb225

Original	Transformed
0x25f0 - 0x2619	REMOVED
0x261a	0x9021
0x261b	0x9221
0x261c	0x9021
0x261d	0x9121
0x261e	0x9221
0x261f	0x9321
0x2670 - 0x2775	REMOVED
0x2776	0x3100
0x2777	0x3200
0x2778	0x3300
0x2779	0x3400
0x277a	0x3500
0x277b	0x3600
0x277c	0x3700
0x277d	0x3800
0x277e	0x3900
0x277f	0x6924
0x2780	0x3100
0x2781	0x3200
0x2782	0x3300
0x2783	0x3400
0x2784	0x3500
0x2785	0x3600
0x2786	0x3700
0x2787	0x3800
0x2788	0x3900
0x2789	0x6924
0x278a	0x3100
0x278b	0x3200
0x278c	0x3300
0x278d	0x3400

Original	Transformed
0x278e	0x3500
0x278f	0x3600
0x2790	0x3700
0x2791	0x3800
0x2792	0x3900
0x2793	0x6924
0x2794	0x9221
0x2795 - 0x2797	REMOVED
0x2798	0x9821
0x2799	0x9221
0x279a	0x9721
0x279b - 0x27af	0x9221
0x27b0	REMOVED
0x27b1 - 0x27b3	0x9221
0x27b4	0x9821
0x27b5	0x9221
0x27b6	0x9721
0x27b7	0x9821
0x27b8	0x9221
0x27b9	0x9721
0x27ba - 0x27be	0x9221
0x27bf - 0x2e7f	REMOVED
0xf900	0x488c
0xf901	0xf466
0xf902	0xca8e
0xf903	0xc88c
0xf904	0xd16e
0xf905	0x324e
0xf906	0xe553
0xf907, 0xf908	0x9c9f
0xf909	0x5159
0xf90a	0xd191

Original	Transformed
0xf90b	0x8755
0xf90c	0x4859
0xf90d	0xf661
0xf90e	0x6976
0xf90f	0x857f
0xf910	0x3f86
0xf911	0xba87
0xf912	0xf888
0xf913	0x8f90
0xf914	0x26a
0xf915	0x1b6d
0xf916	0xd970
0xf917	0xde73
0xf918	0x3d84
0xf919	0x6a91
0xf91a	0xf199
0xf91b	0x824e
0xf91c	0x7553
0xf91d	0x46b
0xf91e	0x1b72
0xf91f	0x2d86
0xf920	0x1e9e
0xf921	0x505d
0xf922	0xeb6f
0xf923	0xcd85
0xf924	0x6489
0xf925	0xc962
0xf926	0xd881
0xf927	0x1f88
0xf928	0xca5e
0xf929	0x1767
0xf92a	0x6a6d

Original	Transformed
0xf92b	0xfc72
0xf92d	0x864f
0xf92e	0xb751
0xf92f	0xde52
0xf930	0xc464
0xf931	0xd36a
0xf932	0x1072
0xf933	0xe776
0xf934	0x180
0xf935	0x686
0xf936	0x5c86
0xf937	0xef8d
0xf938	0x3297
0xf939	0x6f9b
0xf93a	0xfa9d
0xf93b	0x8c78
0xf93c	0x7f79
0xf93d	0xa07d
0xf93e	0xc983
0xf93f	0x493
0xf940	0x7f9e
0xf941	0xd68a
0xf942	0xdf58
0xf943	0x45f
0xf944	0x607c
0xf945	0x7e80
0xf946	0x6272
0xf947	0xca78
0xf948	0xc28c
0xf949	0xf796
0xf94a	0xd858
0xf94b	0x625c

Original	Transformed
0xf94c	0x136a
0xf94d	0xda6d
0xf94e	0xf6f
0xf94f	0x2f7d
0xf950	0x377e
0xf951	0x4b96
0xf952	0xd252
0xf953	0x8b80
0xf954	0xdc51
0xf955	0xcc51
0xf956	0x1c7a
0xf957	0xbe7d
0xf958	0xf183
0xf959	0x7596
0xf95a	0x808b
0xf95b	0xcf62
0xf95c	0x26a
0xf95d	0xfe8a
0xf95e	0x394e
0xf95f	0xe75b
0xf960	0x1260
0xf961	0x8773
0xf962	0x7075
0xf963	0x1753
0xf964	0xfb78
0xf965	0xbf4f
0xf966	0xa95f
0xf967	0xd4e
0xf968	0xcc6c
0xf969	0x7865
0xf96a	0x227d
0xf96b	0xc353

Original	Transformed
0xf96c	0x5e58
0xf96d	0x177
0xf96e	0x4984
0xf96f	0xaa8a
0xf970	0xba6b
0xf971	0xb08f
0xf972	0x886c
0xf973	0xfe62
0xf974	0xe582
0xf975	0xa063
0xf976	0x6575
0xf977	0xae4e
0xf978	0x6951
0xf97a	0x8168
0xf97b	0xe77c
0xf97c	0x6f82
0xf97d	0xd28a
0xf97e	0xcf91
0xf97f	0xf552
0xf980	0x4254
0xf981	0x7359
0xf982	0xec5e
0xf983	0xc565
0xf984	0xfe6f
0xf985	0x2a79
0xf986	0xad95
0xf987	0x6a9a
0xf988	0x979e
0xf989	0xce9e
0xf98a	0x9b52
0xf98b	0xc666
0xf98c	0x776b

Original	Transformed
0xf98d	0x628f
0xf98e	0x745e
0xf98f	0x9061
0xf990	0x62
0xf991	0x9a64
0xf992	0x236f
0xf993	0x4971
0xf994	0x8974
0xf996	0xf47d
0xf997	0x6f80
0xf998	0x268f
0xf999	0xee84
0xf99a	0x2390
0xf99b	0x4a93
0xf99c	0x1752
0xf99d	0xa352
0xf99e	0xbd54
0xf99f	0xc870
0xf9a0	0xc288
0xf9a1	0xaa8a
0xf9a2	0xc95e
0xf9a3	0xf55f
0xf9a4	0x7b63
0xf9a5	0xae6b
0xf9a6	0x3e7c
0xf9a7	0x7573
0xf9a8	0xe44e
0xf9a9	0xf956
0xf9aa	0xe75b
0xf9ab	0xba5d
0xf9ac	0x1c60
0xf9ad	0xb273

Original	Transformed
0xf9ae	0x6974
0xf9af	0x9a7f
0xf9b0	0x4680
0xf9b1	0x3492
0xf9b2	0xf696
0xf9b3	0x4897
0xf9b4	0x1898
0xf9b5	0x8b4f
0xf9b6	0xae79
0xf9b7	0xb491
0xf9b8	0xb896
0xf9b9	0xe160
0xf9ba	0x864e
0xf9bb	0xda50
0xf9bc	0xee5b
0xf9bd	0x3f5c
0xf9be	0x9965
0xf9bf	0x26a
0xf9c0	0xce71
0xf9c1	0x4276
0xf9c2	0xfc84
0xf9c3	0x7c90
0xf9c4	0x8d9f
0xf9c5	0x8866
0xf9c6	0x2e96
0xf9c7	0x8952
0xf9c8	0x7b67
0xf9c9	0xf367
0xf9ca	0x416d
0xf9cb	0x9c6e
0xf9cc	0x974
0xf9cd	0x5975

Original	Transformed
0xf9ce	0x6b78
0xf9cf	0x107d
0xf9d0	0x5e98
0xf9d1	0x6d51
0xf9d2	0x2e62
0xf9d3	0x7896
0xf9d4	0x2b50
0xf9d5	0x195d
0xf9d6	0xea6d
0xf9d7	0x2a8f
0xf9d8	0x8b5f
0xf9d9	0x4461
0xf9da	0x1768
0xf9db	0x8773
0xf9dc	0x8696
0xf9dd	0x2952
0xf9de	0xf54
0xf9df	0x655c
0xf9e0	0x1366
0xf9e1	0x4e67
0xf9e2	0xa868
0xf9e3	0xe56c
0xf9e4	0x674
0xf9e5	0xe275
0xf9e6	0x797f
0xf9e8	0xe188
0xf9e9	0xcc91
0xf9ea	0xe296
0xf9eb	0x3f53
0xf9ec	0xba6e
0xf9ed	0x1d54
0xf9ee	0xd071

Original	Transformed
0xf9ef	0x9874
0xf9f0	0xfa85
0xf9f2	0x579c
0xf9f3	0x9f9e
0xf9f4	0x9767
0xf9f5	0xcb6d
0xf9f6	0xe881
0xf9f7	0xcb7a
0xf9f8	0x207b
0xf9f9	0x927c
0xf9fa	0xc072
0xf9fb	0x9970
0xf9fc	0x588b
0xf9fd	0xc04e
0xf9fe	0x3683
0xf9ff	0x3a52
0xfa00	0x752
0xfa01	0xa65e
0xfa02	0xd362
0xfa03	0xd67c
0xfa04	0x855b
0xfa05	0x1e6d
0xfa06	0xb466
0xfa07	0x3b8f
0xfa08	0x4c88
0xfa09	0x4d96
0xfa0a	0x8b89
0xfa0b	0xd35e
0xfa10	0x5a58
0xfa12	0x7466
0xfa15	0xde51
0xfa16	0x6c8c

Original	Transformed
0xfa17	0xca76
0xfa19	0x5e79
0xfa1a	0x6579
0xfa1b	0x8f79
0xfa1c	0x5697
0xfa1d	0xbe7c
0xfa1e	0xbd7f
0xfa22	0xf88a
0xfa25	0x3890
0xfa26	0xfd90
0xfa2a	0xef98
0xfa2b	0xfc98
0xfa2c	0x2899
0xfa2d	0xb49d
0xfa2e - 0xfaff	REMOVED
0xfb00	0x6600 0x6600
0xfb01	0x6600 0x6900
0xfb02	0x6600 0x6c00
0xfb03	0x6600 0x6600 0x6900
0xfb04	0x6600 0x6600 0x6c00
0xfb05	0x7f01 0x7400
0xfb06	0x7300 0x7400
0xfb07 - 0xfe2f	REMOVED
0xfe30	0x2520
0xfe31	0x1420
0xfe32	0x1320
0xfe33, 0xfe34	0x5f00
0xfe35	0x2800
0xfe36	0x2900
0xfe37	0x7b00
0xfe38	0x7d00
0xfe39	0x1430

Original	Transformed
0xfe3a	0x1530
0xfe3b	0x1030
0xfe3c	0x1130
0xfe3d	0xa30
0xfe3e	0xb30
0xfe3f	0x830
0xfe40	0x930
0xfe41	0xc30
0xfe42	0xd30
0xfe43	0xe30
0xfe44	0xf30
0xfe45 - 0xfe48	REMOVED
0xfe49 - 0xfe4c	0x3e20
0xfe4d - 0xfe4f	0x5f00
0xfe50	0x2c00
0xfe51	0x130
0xfe52	0x2e00
0xfe53	REMOVED
0xfe54	0x3b00
0xfe55	0x3a00
0xfe56	0x3f00
0xfe57	0x2100
0xfe58	0x1420
0xfe59	0x2800
0xfe5a	0x2900
0xfe5b	0x7b00
0xfe5c	0x7d00
0xfe5d	0x1430
0xfe5e	0x1530
0xfe5f	0x2300
0xfe60	0x2600
0xfe61	0x2a00

Original	Transformed
0xfe62	0x2b00
0xfe63	0x2d00
0xfe64	0x3c00
0xfe65	0x3e00
0xfe66	0x3d00
0xfe67	REMOVED
0xfe69	0x2400
0xfe6a	0x2500
0xfe6b	0x4000
0xfe6c - 0xfe6f	REMOVED
0xfe70, 0xfe71	0x4b06
0xfe72	0x4c06
0xfe73	REMOVED
0xfe74	0x4d06
0xfe75	REMOVED
0xfe76, 0xfe77	0x4e06
0xfe78, 0xfe79	0x4f06
0xfe7a, 0xfe7b	0x5006
0xfe7c, 0xfe7d	0x5106
0xfe7e, 0xfe7f	0x5206
0xfe80	0x2106
0xfe81 - 0xfe84	0x2706
0xfe85, 0xfe86	0x2106
0xfe87, 0xfe88	0x2706
0xfe89 - 0xfe8c	0x2106
0xfe8d, 0xfe8e	0x2706
0xfe8f - 0xfe92	0x2806
0xfe93 - 0xfe98	0x2906
0xfe99 - 0xfe9c	0x2b06
0xfe9d - 0xfea0	0x2c06
0xfea1 - 0xfea4	0x2d06
0xfea5 - 0xfea8	0x2e06

Original	Transformed
0xfea9, 0xfeaa	0x2f06
0xfeab, 0xfeac	0x3006
0xfead, 0xfeae	0x3106
0xfeaf, 0xfeb0	0x3206
0xfeb1 - 0xfeb4	0x3306
0xfeb5 - 0xfeb8	0x3406
0xfeb9 - 0xfebc	0x3506
0xfebd - 0xfec0	0x3606
0xfec1 - 0xfec4	0x3706
0xfec5 - 0xfec8	0x3806
0xfec9 - 0xfecc	0x3906
0xfecd - 0xfed0	0x3a06
0xfed1 - 0xfed4	0x4106
0xfed5 - 0xfed8	0x4206
0xfed9 - 0xfedc	0x4306
0xfedd - 0xfee0	0x4406
0xfee1 - 0xfee4	0x4506
0xfee5 - 0xfee8	0x4606
0xfee9 - 0xfeec	0x4706
0xfeed, 0xfeee	0x4806
0xfeef - 0xfef4	0x4906
0xfefd - 0xff00	REMOVED
0xff01	0x2100
0xff02	0x2200
0xff03	0x2300
0xff04	0x2400
0xff05	0x2500
0xff06	0x2600
0xff07	0x2700
0xff08	0x2800
0xff09	0x2900
0xff0a	0x2a00

Original	Transformed
0xff0b	0x2b00
0xff0c	0x2c00
0xff0d	0x2d00
0xff0e	0x2e00
0xff0f	0x2f00
0xff10	0x3000
0xff11	0x3100
0xff12	0x3200
0xff13	0x3300
0xff14	0x3400
0xff15	0x3500
0xff16	0x3600
0xff17	0x3700
0xff18	0x3800
0xff19	0x3900
0xff1a	0x3a00
0xff1b	0x3b00
0xff1c	0x3c00
0xff1d	0x3d00
0xff1e	0x3e00
0xff1f	0x3f00
0xff20	0x4000
0xff21	0x6100
0xff22	0x6200
0xff23	0x6300
0xff24	0x6400
0xff25	0x6500
0xff26	0x6600
0xff27	0x6700
0xff28	0x6800
0xff29	0x6900
0xff2a	0x6a00

Original	Transformed
0xff2b	0x6b00
0xff2c	0x6c00
0xff2d	0x6d00
0xff2e	0x6e00
0xff2f	0x6f00
0xff30	0x7000
0xff31	0x7100
0xff32	0x7200
0xff33	0x7300
0xff34	0x7400
0xff35	0x7500
0xff36	0x7600
0xff37	0x7700
0xff38	0x7800
0xff39	0x7900
0xff3a	0x7a00
0xff3b	0x5b00
0xff3c	0x5c00
0xff3d	0x5d00
0xff3e	0x5e00
0xff3f	0x5f00
0xff40	0x6000
0xff41	0x6100
0xff42	0x6200
0xff43	0x6300
0xff44	0x6400
0xff45	0x6500
0xff46	0x6600
0xff47	0x6700
0xff48	0x6800
0xff49	0x6900
0xff4a	0x6a00

Original	Transformed
0xff4b	0x6b00
0xff4c	0x6c00
0xff4d	0x6d00
0xff4e	0x6e00
0xff4f	0x6f00
0xff50	0x7000
0xff51	0x7100
0xff52	0x7200
0xff53	0x7300
0xff54	0x7400
0xff55	0x7500
0xff56	0x7600
0xff57	0x7700
0xff58	0x7800
0xff59	0x7900
0xff5a	0x7a00
0xff5b	0x7b00
0xff5c	0x7c00
0xff5d	0x7d00
0xff5e	0x7e00
0xff5f, 0xff60	REMOVED
0xff61	0x230
0xff62	0xc30
0xff63	0xd30
0xff64	0x130
0xff65	0xfb30
0xff66	0xf230
0xff67	0xa130
0xff68	0xa330
0xff69	0xa530
0xff6a	0xa730
0xff6b	0xa930

Original	Transformed
0xff6c	0xe330
0xff6d	0xe530
0xff6e	0xe730
0xff6f	0xc330
0xff70	0xfc30
0xff71	0xa230
0xff72	0xa430
0xff73	0xa630
0xff74	0xa830
0xff75	0xaa30
0xff76	0xab30
0xff77	0xad30
0xff78	0xaf30
0xff79	0xb130
0xff7a	0xb330
0xff7b	0xb530
0xff7c	0xb730
0xff7d	0xb930
0xff7e	0xbb30
0xff7f	0xbd30
0xff80	0xbf30
0xff81	0xc130
0xff82	0xc430
0xff83	0xc630
0xff84	0xc830
0xff85	0xca30
0xff86	0xcb30
0xff87	0xcc30
0xff88	0xcd30
0xff89	0xce30
0xff8a	0xcf30
0xff8b	0xd230

Original	Transformed
0xff8c	0xd530
0xff8d	0xd830
0xff8e	0xdb30
0xff8f	0xde30
0xff90	0xdf30
0xff91	0xe030
0xff92	0xe130
0xff93	0xe230
0xff94	0xe430
0xff95	0xe630
0xff96	0xe830
0xff97	0xe930
0xff98	0xea30
0xff99	0xeb30
0xff9a	0xec30
0xff9b	0xed30
0xff9c	0xef30
0xff9d	0xf330
0xff9e	0x9b30
0xff9f	0x9c30
0xffa0	0x6431
0xffa1	0x3131
0xffa2	0x3231
0xffa3	0x3331
0xffa4	0x3431
0xffa5	0x3531
0xffa6	0x3631
0xffa7	0x3731
0xffa8	0x3831
0xffa9	0x3931
0xffaa	0x3a31
0xffab	0x3b31

Original	Transformed
0xffac	0x3c31
0xffad	0x3d31
0xffae	0x3e31
0xffaf	0x3f31
0xffb0	0x4031
0xffb1	0x4131
0xffb2	0x4231
0xffb3	0x4331
0xffb4	0x4431
0xffb5	0x4531
0xffb6	0x4631
0xffb7	0x4731
0xffb8	0x4831
0xffb9	0x4931
0xffba	0x4a31
0xffbb	0x4b31
0xffbc	0x4c31
0xffbd	0x4d31
0xffbe	0x4e31
0xffbf - 0xffc1	REMOVED
0xffc2	0x4f31
0xffc3	0x5031
0xffc4	0x5131
0xffc5	0x5231
0xffc6	0x5331
0xffc7	0x5431
0xffc8, 0xffc9	REMOVED
0xffca	0x5531
0xffcb	0x5631
0xffcc	0x5731
0xffcd	0x5831
0xffce	0x5931

Original	Transformed
0xffcf	0x5a31
0xffd0, 0xffd1	REMOVED
0xffd2	0x5b31
0xffd3	0x5c31
0xffd4	0x5d31
0xffd5	0x5e31
0xffd6	0x5f31
0xffd7	0x6031
0xffd8, 0xffd9	REMOVED
0xffda	0x6131
0xffdb	0x6231
0xffdc	0x6331
0xffdd - 0xffdf	REMOVED
0xffe0	0xa200
0xffe1	0xa300
0xffe2	0xac00
0xffe3	0xaf00
0xffe4	0xa600
0xffe5	0xa500
0xffe6	0xa920
0xffe7	REMOVED
0xffe8	0x225
0xffe9	0x9021
0xffea	0x9121
0xffeb	0x9221
0xffec	0x9321
0xffed	0xa025
0xffee	0xcb25
0xffef - 0xffff	REMOVED

Table 2

Original	Transformed
0xaa - 0xba	0x3
0xc0	0xf
0xc1	0xe
0xc2	0x12
0xc3	0x19
0xc4	0x13
0xc5	0x1a
0xc7	0x1c
0xc8	0xf
0xc9	0xe
0xca	0x12
0xcb	0x13
0xcc	0xf
0xcd	0xe
0xce	0x12
0xcf	0x13
0xd0	0x68
0xd1	0x19
0xd2	0xf
0xd3	0xe
0xd4	0x12
0xd5	0x19
0xd6	0x13
0xd8	0x21
0xd9	0xf
0xda	0xe
0xdb	0x12
0xdc	0x13
0xdd	0xe
0xe0	0xf
0xe1	0xe
0xe2	0x12

Original	Transformed
0xe3	0x19
0xe4	0x13
0xe5	0x1a
0xe7	0x1c
0xe8	0xf
0xe9	0xe
0xea	0x12
0xeb	0x13
0xec	0xf
0xed	0xe
0xee	0x12
0xef	0x13
0xf0	0x68
0xf1	0x19
0xf2	0xf
0xf3	0xe
0xf4	0x12
0xf5	0x19
0xf6	0x13
0xf8	0x21
0xf9	0xf
0xfa	0xe
0xfb	0x12
0xfc	0x13
0xfd	0xe
0xff	0x13
0x100, 0x101	0x17
0x102, 0x103	0x15
0x104, 0x105	0x1b
0x106, 0x107	0xe
0x108, 0x109	0x12
0x10a, 0x10b	0x10

Original	Transformed
0x10c - 0x10f	0x14
0x110, 0x111	0x1e
0x112, 0x113	0x17
0x114, 0x115	0x15
0x116, 0x117	0x10
0x118, 0x119	0x1b
0x11a, 0x11b	0x14
0x11c, 0x11d	0x12
0x11e, 0x11f	0x15
0x120, 0x121	0x10
0x122, 0x123	0x1c
0x124, 0x125	0x12
0x126, 0x127	0x1e
0x128, 0x129	0x19
0x12a, 0x12b	0x17
0x12c, 0x12d	0x15
0x12e, 0x12f	0x1b
0x130	0x10
0x131	0x3
0x134, 0x135	0x12
0x136, 0x137	0x1c
0x138	0x3
0x139, 0x13a	0xe
0x13b, 0x13c	0x1c
0x13d, 0x13e	0x14
0x13f, 0x140	0x11
0x141, 0x142	0x1f
0x143, 0x144	0xe
0x145, 0x146	0x1c
0x147, 0x148	0x14
0x149	0x48
0x14c, 0x14d	0x17

Original	Transformed
0x14e, 0x14f	0x15
0x150, 0x151	0x1d
0x154, 0x155	0xe
0x156, 0x157	0x1c
0x158, 0x159	0x14
0x15a, 0x15b	0xe
0x15c, 0x15d	0x12
0x15e, 0x15f	0x1c
0x160, 0x161	0x14
0x162, 0x163	0x1c
0x164, 0x165	0x14
0x166, 0x167	0x1e
0x168, 0x169	0x19
0x16a, 0x16b	0x17
0x16c, 0x16d	0x15
0x16e, 0x16f	0x1a
0x170, 0x171	0x1d
0x172, 0x173	0x1b
0x174 - 0x177	0x12
0x178	0x13
0x179, 0x17a	0xe
0x17b, 0x17c	0x10
0x17d, 0x17e	0x14
0x180	0x1e
0x181	0x43
0x182, 0x183	0x68
0x184, 0x185	0x87
0x186	0x7d
0x187, 0x188	0x43
0x189	0x4
0x18a	0x43
0x18b, 0x18c	0x1e

Original	Transformed
0x18d	0x7c
0x18e	0x7d
0x18f	0x7e
0x190	0x7b
0x191	0x43
0x192	0x7b
0x193	0x43
0x194 - 0x196	0x7b
0x197	0x1e
0x198, 0x199	0x43
0x19a	0x1e
0x19b	0x20
0x19c	0x7b
0x19d	0x43
0x19e	0x7b
0x19f	0x20
0x1a0, 0x1a1	0x52
0x1a2, 0x1a3	0x7c
0x1a4, 0x1a5	0x43
0x1a6	0x7b
0x1a7, 0x1a8	0x87
0x1a9	0x7c
0x1aa	0x7f
0x1ab	0x57
0x1ac, 0x1ad	0x43
0x1ae	0x59
0x1af, 0x1b0	0x52
0x1b1, 0x1b2	0x7b
0x1b3, 0x1b4	0x43
0x1b5, 0x1b6	0x1e
0x1b8, 0x1b9	0x7c
0x1ba	0x7d

Original	Transformed
0x1bb - 0x1bd	0x3
0x1bf	0x7b
0x1c4 - 0x1c6	0x2 0x14
0x1cd - 0x1d4	0x14
0x1d5, 0x1d6	0x28
0x1d7, 0x1d8	0x1f
0x1d9, 0x1da	0x25
0x1db, 0x1dc	0x20
0x1dd	0x7d
0x1de, 0x1df	0x28
0x1e0, 0x1e1	0x25
0x1e2, 0x1e3	0x17 0x17
0x1e4, 0x1e5	0x1e
0x1e6 - 0x1e9	0x14
0x1ea, 0x1eb	0x1b
0x1ec, 0x1ed	0x30
0x1ee - 0x1f0	0x14
0x1f4, 0x1f5	0xe
0x1fa, 0x1fb	0x26
0x1fc, 0x1fd	0xe 0xe
0x1fe, 0x1ff	0x2b
0x200, 0x201	0x44
0x202, 0x203	0x46
0x204, 0x205	0x44
0x206, 0x207	0x46
0x208, 0x209	0x44
0x20a, 0x20b	0x46
0x20c, 0x20d	0x44
0x20e, 0x20f	0x46
0x210, 0x211	0x44
0x212, 0x213	0x46
0x214, 0x215	0x44

Original	Transformed
0x216, 0x217	0x46
0x218 - 0x21b	0x16
0x250	0x7b
0x251	0x7c
0x252	0x7d
0x253	0x43
0x254	0x7b
0x255	0x7c
0x256	0x59
0x257	0x43
0x258	0x80
0x259	0x7e
0x25a	0x7f
0x25b	0x7b
0x25c	0x81
0x25d	0x82
0x25e	0x83
0x25f	0x1e
0x260	0x43
0x261	0x7d
0x263	0x7b
0x264, 0x265	0x7c
0x266, 0x267	0x43
0x268	0x1e
0x269	0x7b
0x26b	0x19
0x26c	0x7b
0x26d	0x59
0x26e	0x7c
0x26f	0x7b
0x270	0x7c
0x271, 0x272	0x43

Original	Transformed
0x273	0x59
0x275	0x20
0x276	0x7b
0x277	0x7e
0x278, 0x279	0x7b
0x27a	0x7c
0x27b	0x7d
0x27c	0x7e
0x27d	0x43
0x27e	0x7f
0x27f	0x80
0x281	0x7b
0x282	0x43
0x283	0x7c
0x284	0x43
0x285	0x84
0x286	0x85
0x287	0x7b
0x288	0x59
0x289	0x1e
0x28a, 0x28b	0x7b
0x28c, 0x28d	0x7c
0x28e	0x7b
0x290	0x59
0x291 - 0x293	0x7e
0x294	0x3
0x295	0x4
0x296	0x5
0x297	0x88
0x29a	0x86
0x29b	0x43
0x29d, 0x29e	0x7b

Original	Transformed
0x2a0	0x43
0x2a1	0x6
0x2a2	0xb
0x2a3, 0x2a4	0x7b
0x2a5	0x7c
0x2a6	0x7b
0x2a7	0x7c
0x2a8	0x7d
0x2b0	0x7e
0x2b1	0x7f
0x2b2	0x7e
0x2b3	0x81
0x2b4	0x82
0x2b5	0x83
0x2b6	0x84
0x2b7, 0x2b8	0x7e
0x2c6 - 0x2d8	0x3
0x2e0 - 0x2e3	0x7e
0x386 - 0x38f	0x5
0x390	0x16
0x3aa, 0x3ab	0x13
0x3ac - 0x3af	0x5
0x3b0	0x16
0x3ca, 0x3cb	0x13
0x3cc - 0x3ce	0x5
0x3d0, 0x3d1	0x3
0x3d2	0x1b
0x3d3	0x44
0x3d4	0x13
0x3d5 - 0x3f1	0x3
0x3f2	0x4
0x401	0x13

Original	Transformed
0x403 - 0x40c	0xe
0x451	0x13
0x453 - 0x45c	0xe
0x463	0x3
0x476, 0x477	0x46
0x482	0x37
0x490, 0x491	0x3
0x492, 0x493	0x1a
0x494, 0x495	0x5
0x496, 0x497	0x7
0x498, 0x499	0x1a
0x49a, 0x49b	0x17
0x49c, 0x49d	0x9
0x49e, 0x49f	0x4
0x4a0, 0x4a1	0xa
0x4a2, 0x4a3	0x7
0x4a4, 0x4a5	0x8
0x4a6 - 0x4a9	0x5
0x4aa, 0x4ab	0x1a
0x4ac, 0x4ad	0x7
0x4ae, 0x4af	0xb
0x4b2, 0x4b3	0x17
0x4b4, 0x4b5	0x3
0x4b6	0x8
0x4b7	0x7
0x4b8 - 0x4bb	0x9
0x4bc, 0x4bd	0x5
0x4be, 0x4bf	0x17
0x4c0	0x3
0x4c1, 0x4c2	0x15
0x4c3, 0x4c4	0x5
0x4c7, 0x4c8	0xa

Original	Transformed
0x4cb, 0x4cc	0xd
0x4d8, 0x4d9	0x1e
0x622	0x9
0x623	0xa
0x624	0x5
0x625	0xb
0x626	0x7
0x627 - 0x648	0x8
0x649	0x5
0x64a	0x7
0x670	0xc
0x671	0xd
0x672	0xe
0x673	0xf
0x675	0x10
0x67e - 0x95f	0x8
0x9dc - 0x9df	0x6
0x9f4 - 0x9f9	0x3c
0xa33 - 0xa5e	0x4
0xac4 - 0xae0	0x5
0xb5c - 0xb5f	0x6
0xbf0 - 0xbf2	0x40
0xce0	0x4
0x1e00, 0x1e01	0x5a
0x1e02, 0x1e03	0x10
0x1e04, 0x1e05	0x58
0x1e06, 0x1e07	0x55
0x1e08, 0x1e09	0x28
0x1e0a, 0x1e0b	0x10
0x1e0c, 0x1e0d	0x58
0x1e0e, 0x1e0f	0x55
0x1e10, 0x1e11	0x1c

Original	Transformed
0x1e12, 0x1e13	0x60
0x1e14, 0x1e15	0x24
0x1e16, 0x1e17	0x23
0x1e18, 0x1e19	0x60
0x1e1a, 0x1e1b	0x63
0x1e1c, 0x1e1d	0x2f
0x1e1e, 0x1e1f	0x10
0x1e20, 0x1e21	0x17
0x1e22, 0x1e23	0x10
0x1e24, 0x1e25	0x58
0x1e26, 0x1e27	0x13
0x1e28, 0x1e29	0x1c
0x1e2a, 0x1e2b	0x61
0x1e2c, 0x1e2d	0x63
0x1e2e, 0x1e2f	0x1f
0x1e30, 0x1e31	0xe
0x1e32, 0x1e33	0x58
0x1e34, 0x1e35	0x55
0x1e36, 0x1e37	0x58
0x1e38, 0x1e39	0x6e
0x1e3a, 0x1e3b	0x55
0x1e3c, 0x1e3d	0x60
0x1e3e, 0x1e3f	0xe
0x1e40, 0x1e41	0x10
0x1e42, 0x1e43	0x58
0x1e44, 0x1e45	0x10
0x1e46, 0x1e47	0x58
0x1e48, 0x1e49	0x55
0x1e4a, 0x1e4b	0x60
0x1e4c, 0x1e4d	0x25
0x1e4e, 0x1e4f	0x2a
0x1e50, 0x1e51	0x24

Original	Transformed
0x1e52, 0x1e53	0x23
0x1e54, 0x1e55	0xe
0x1e56 - 0x1e59	0x10
0x1e5a, 0x1e5b	0x58
0x1e5c, 0x1e5d	0x6e
0x1e5e, 0x1e5f	0x55
0x1e60, 0x1e61	0x10
0x1e62, 0x1e63	0x58
0x1e64, 0x1e65	0x1d
0x1e66, 0x1e67	0x22
0x1e68, 0x1e69	0x68
0x1e6a, 0x1e6b	0x10
0x1e6c, 0x1e6d	0x58
0x1e6e, 0x1e6f	0x55
0x1e70, 0x1e71	0x60
0x1e72, 0x1e73	0x59
0x1e74, 0x1e75	0x63
0x1e76, 0x1e77	0x5a
0x1e78, 0x1e79	0x25
0x1e7a, 0x1e7b	0x28
0x1e7c, 0x1e7d	0x19
0x1e7e, 0x1e7f	0x58
0x1e80, 0x1e81	0xf
0x1e82, 0x1e83	0xe
0x1e84, 0x1e85	0x13
0x1e86, 0x1e87	0x10
0x1e88, 0x1e89	0x58
0x1e8a, 0x1e8b	0x10
0x1e8c, 0x1e8d	0x13
0x1e8e, 0x1e8f	0x10
0x1e90, 0x1e91	0x12
0x1e92, 0x1e93	0x58

Original	Transformed
0x1e94 - 0x1e96	0x55
0x1e97	0x13
0x1e98, 0x1e99	0x1a
0x1e9a	0x69
0x1ea0, 0x1ea1	0x59
0x1ea2, 0x1ea3	0x43
0x1ea4, 0x1ea5	0x1e
0x1ea6, 0x1ea7	0x1f
0x1ea8, 0x1ea9	0x55
0x1eaa, 0x1eab	0x29
0x1eac, 0x1ead	0x6a
0x1eae, 0x1eaf	0x21
0x1eb0, 0x1eb1	0x22
0x1eb2, 0x1eb3	0x58
0x1eb4, 0x1eb5	0x2c
0x1eb6, 0x1eb7	0x6d
0x1eb8, 0x1eb9	0x58
0x1eba, 0x1ebb	0x43
0x1ebc, 0x1ebd	0x19
0x1ebe, 0x1ebf	0x1e
0x1ec0, 0x1ec1	0x1f
0x1ec2, 0x1ec3	0x55
0x1ec4, 0x1ec5	0x29
0x1ec6, 0x1ec7	0x6a
0x1ec8, 0x1ec9	0x43
0x1eca - 0x1ecd	0x58
0x1ece, 0x1ecf	0x43
0x1ed0, 0x1ed1	0x1e
0x1ed2, 0x1ed3	0x1f
0x1ed4, 0x1ed5	0x55
0x1ed6, 0x1ed7	0x29
0x1ed8, 0x1ed9	0x6a

Original	Transformed
0x1eda, 0x1edb	0x60
0x1edc, 0x1edd	0x61
0x1ede, 0x1edf	0x95
0x1ee0, 0x1ee1	0x69
0x1ee2, 0x1ee3	0xaa
0x1ee4, 0x1ee5	0x58
0x1ee6, 0x1ee7	0x43
0x1ee8, 0x1ee9	0x60
0x1eea, 0x1eeb	0x61
0x1eec, 0x1eed	0x95
0x1eee, 0x1eef	0x69
0x1ef0, 0x1ef1	0xaa
0x1ef2, 0x1ef3	0xf
0x1ef4, 0x1ef5	0x58
0x1ef6, 0x1ef7	0x44
0x1ef8, 0x1ef9	0x19
0x2045, 0x2046	0x1c
0x2102	0x3
0x2103	0x4
0x2107	0x64
0x2109	0x4
0x210a	0x3
0x210b	0x5
0x210c	0x4
0x210d, 0x210e	0x3
0x210f	0x68
0x2110	0x4
0x2111	0x5
0x2112, 0x2113	0x4
0x2115	0x3
0x2118	0x4
0x2119, 0x211a	0x3

Original	Transformed
0x211b	0x4
0x211c	0x5
0x211d - 0x2124	0x3
0x2125	0x63
0x2126	0x3
0x2127, 0x2128	0x5
0x2129	0x4
0x212b	0x1a
0x212c	0x4
0x212d, 0x212e	0x5
0x212f, 0x2130	0x4
0x2131	0x5
0x2132	0x3
0x2133, 0x2134	0x4
0x215f	0x3
0x2160 - 0x2182	0x47
0x2190 - 0x2199	0x3
0x219a, 0x219b	0x4
0x219c, 0x219d	0x5
0x219e	0x6
0x219f	0x4
0x21a0	0x6
0x21a1	0x4
0x21a2, 0x21a3	0x7
0x21a4	0x8
0x21a5	0x5
0x21a6	0x8
0x21a7	0x5
0x21a8	0x4
0x21a9, 0x21aa	0x9
0x21ab, 0x21ac	0xa
0x21ad	0x4

Original	Transformed
0x21ae	0x5
0x21af	0x6
0x21b0, 0x21b1	0xb
0x21b2, 0x21b3	0xc
0x21b4	0x7
0x21b5	0xd
0x21b6, 0x21b7	0x3
0x21b8	0x4
0x21b9	0x6
0x21ba, 0x21bb	0x4
0x21bc	0xe
0x21bd	0xf
0x21be	0x6
0x21bf	0x7
0x21c0	0xd
0x21c1	0xe
0x21c2	0x8
0x21c3	0x9
0x21c4	0x7
0x21c5	0x6
0x21c6	0x8
0x21c7	0x10
0x21c8	0x8
0x21c9	0xf
0x21ca	0xa
0x21cb	0x9
0x21cc	0xa
0x21cd	0x11
0x21ce	0xb
0x21cf	0x12
0x21d0	0x10
0x21d1	0x9

Original	Transformed
0x21d2	0x11
0x21d3	0xb
0x21d4	0xc
0x21d5, 0x21d6	0x5
0x21d7 - 0x21d9	0x4
0x21da	0x13
0x21db	0x12
0x21dc	0x14
0x21dd	0x13
0x21de	0xa
0x21df	0xc
0x21e0	0x15
0x21e1	0xb
0x21e2	0x14
0x21e3	0xd
0x21e4	0x16
0x21e5	0x15
0x21e6	0x17
0x21e7	0xc
0x21e8	0x16
0x21e9	0xe
0x21ea	0xd
0x226a, 0x226b	0x3
0x2295 - 0x229d	0xee
0x229e - 0x22a1	0xef
0x2469 - 0x2473	0xee
0x247d - 0x2487	0xf3
0x2491 - 0x249b	0xf4
0x249c - 0x24b5	0xf3
0x24b6 - 0x24e9	0xee
0x2500	0x3
0x2501	0x4

Original	Transformed
0x2502	0x3
0x2503	0x4
0x2504	0x5
0x2505	0x6
0x2506	0x5
0x2507	0x6
0x2508	0x7
0x2509	0x8
0x250a	0x7
0x250b	0x8
0x250c	0x3
0x250d	0x4
0x250e	0x5
0x250f	0x6
0x2510	0x3
0x2511	0x4
0x2512	0x5
0x2513	0x6
0x2514	0x3
0x2515	0x4
0x2516	0x5
0x2517	0x6
0x2518	0x3
0x2519	0x4
0x251a	0x5
0x251b	0x6
0x251c	0x3
0x251d	0x4
0x251e	0x5
0x251f	0x6
0x2520	0x7
0x2521	0x8

Original	Transformed
0x2522	0x9
0x2523	0xa
0x2524	0x3
0x2525	0x4
0x2526	0x5
0x2527	0x6
0x2528	0x7
0x2529	0x8
0x252a	0x9
0x252b	0xa
0x252c	0x3
0x252d	0x4
0x252e	0x5
0x252f	0x6
0x2530	0x7
0x2531	0x8
0x2532	0x9
0x2533	0xa
0x2534	0x3
0x2535	0x4
0x2536	0x5
0x2537	0x6
0x2538	0x7
0x2539	0x8
0x253a	0x9
0x253b	0xa
0x253c	0x3
0x253d	0x4
0x253e	0x5
0x253f	0x6
0x2540	0x7
0x2541	0x8

Original	Transformed
0x2542	0x9
0x2543	0xa
0x2544	0xb
0x2545	0xc
0x2546	0xd
0x2547	0xe
0x2548	0xf
0x2549	0x10
0x254a	0x11
0x254b	0x12
0x254c	0x9
0x254d	0xa
0x254e	0x9
0x254f	0xa
0x2550, 0x2551	0xb
0x2552	0x7
0x2553	0x8
0x2554	0x9
0x2555	0x7
0x2556	0x8
0x2557	0x9
0x2558	0x7
0x2559	0x8
0x255a	0x9
0x255b	0x7
0x255c	0x8
0x255d	0x9
0x255e	0xb
0x255f	0xc
0x2560	0xd
0x2561	0xb
0x2562	0xc

Original	Transformed
0x2563	0xd
0x2564	0xb
0x2565	0xc
0x2566	0xd
0x2567	0xb
0x2568	0xc
0x2569	0xd
0x256a	0x13
0x256b	0x14
0x256c	0x15
0x256d - 0x2570	0xa
0x2571 - 0x2577	0x3
0x2578 - 0x257b	0x4
0x257c, 0x257d	0xc
0x257e, 0x257f	0xd
0x2580 - 0x2583	0x3
0x2584	0x4
0x2585 - 0x2588	0x3
0x2589 - 0x258b	0x4
0x258c	0x5
0x258d - 0x258f	0x4
0x2590	0x6
0x2591	0x4
0x2592	0x5
0x2593	0x6
0x2594	0x5
0x2595	0x6
0x25a0	0x3
0x25a1	0x4
0x25a2	0x5
0x25a3	0x6
0x25a4	0x7

Original	Transformed
0x25a5	0x8
0x25a6	0x9
0x25a7	0xa
0x25a8	0xb
0x25a9	0xc
0x25aa	0xd
0x25ab	0xe
0x25ac	0x3
0x25ad	0x4
0x25ae	0x3
0x25af	0x4
0x25b0	0x3
0x25b1	0x4
0x25b2	0x3
0x25b3	0x4
0x25b4	0x5
0x25b5	0x6
0x25b6	0x3
0x25b7	0x4
0x25b8	0x5
0x25b9	0x6
0x25ba	0x3
0x25bb	0x4
0x25bc	0x3
0x25bd	0x4
0x25be	0x5
0x25bf	0x6
0x25c0	0x3
0x25c1	0x4
0x25c2	0x5
0x25c3	0x6
0x25c4	0x3

Original	Transformed
0x25c5	0x4
0x25c6	0x3
0x25c7	0x4
0x25c8	0x5
0x25c9 - 0x25cb	0x3
0x25cc	0x4
0x25cd	0x5
0x25ce	0x6
0x25cf	0x7
0x25d0	0x8
0x25d1	0x9
0x25d2	0xa
0x25d3	0xb
0x25d4	0xc
0x25d5	0xd
0x25d6 - 0x25d8	0x3
0x25d9	0xe
0x25da - 0x25e5	0x3
0x25e6	0x4
0x25e7	0xf
0x25e8	0x10
0x25e9	0x11
0x25ea	0x12
0x25eb	0x13
0x25ec	0x7
0x25ed	0x8
0x25ee	0x9
0x261a	0x3c
0x261b, 0x261c	0x3d
0x261d - 0x261f	0x3c
0x277f - 0x2793	0xee
0x2794	0x17

Original	Transformed
0x2798	0x5
0x2799	0x18
0x279a	0x5
0x279b	0x19
0x279c	0x1a
0x279d	0x1b
0x279e	0x1c
0x279f	0x1d
0x27a0	0x1e
0x27a1	0x1f
0x27a2	0x20
0x27a3	0x21
0x27a4	0x22
0x27a5	0x23
0x27a6	0x24
0x27a7	0x25
0x27a8	0x26
0x27a9	0x27
0x27aa	0x28
0x27ab	0x29
0x27ac	0x2a
0x27ad	0x2b
0x27ae	0x2c
0x27af	0x2d
0x27b1	0x2e
0x27b2	0x2f
0x27b3	0x30
0x27b4	0x6
0x27b5	0x31
0x27b6	0x6
0x27b7	0x7
0x27b8	0x32

Original	Transformed
0x27b9	0x7
0x27ba	0x33
0x27bb	0x34
0x27bc	0x35
0x27bd	0x36
0x27be	0x37
0xfe81, 0xfe82	0x9
0xfe83, 0xfe84	0xa
0xfe85, 0xfe86	0x5
0xfe87, 0xfe88	0xb
0xfe89 - 0xfe8c	0x7
0xfe8d - 0xfeee	0x8
0xfeef, 0xfef0	0x5
0xfef1 - 0xfef4	0x7
0xffe8 - 0xffee	0x3

6 Appendix B: Product Behavior

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

- Microsoft Office SharePoint Server 2007
- Microsoft SharePoint Server 2010

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

<1> Section 2.2.4.1: Prior to the Office SharePoint Server 2007 Infrastructure Update, the value for this field is set to 0x00520000. As of the Office SharePoint Server 2007 Infrastructure Update, the value is set to either 0x00520000 or 0x00530000. As of SharePoint Server 2010, Microsoft SharePoint Server 2013 and Microsoft SharePoint Server 2016, the value is set to either 0x00520000 or 0x00530000 or 0x00540000.

<2> Section 2.3: This functionality was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<3> Section 2.3.1: This field was added in SharePoint Server 2010.

<4> Section 2.3.1: This field was added in SharePoint Server 2010.

<5> Section 2.3.1: This field was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<6> Section 2.3.1: This field was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<7> Section 2.3.1: This field was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<8> Section 2.3.1: This field was added in SharePoint Server 2010.

<9> Section 2.3.1: This field was added in SharePoint Server 2010.

<10> Section 2.3.1: This field was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<11> Section 2.3.1: This field was added in SharePoint Server 2010.

<12> Section 2.3.1: This field was added in SharePoint Server 2010.

<13> Section 2.3.1: This field was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<14> Section 2.3.1: This field was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<15> Section 2.3.1: This field was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<16> Section 2.3.1: This field was added in SharePoint Server 2010.

<17> Section 2.3.1: This field was added in SharePoint Server 2010.

<18> Section 2.3.1: The BOF key was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<19> Section 2.3.1: The BOF key was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<20> Section 2.3.1: The BOF key was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<21> Section 2.3.1: The BOF key was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<22> Section 2.3.1: This functionality was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<23> Section 2.3.1: This functionality was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<24> Section 2.3.1: This functionality was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<25> Section 2.10.1: This functionality was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<26> Section 2.10.2: This value was added as a part of the Office SharePoint Server 2007 Infrastructure Update.

<27> Section 2.10.2: This value was added as a part of the Office SharePoint Server 2007 Infrastructure Update.

<28> Section 2.10.3: This functionality was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<29> Section 2.10.3: This functionality was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<30> Section 2.13.2: 0x0053 value was added as a part of the Office SharePoint Server 2007 Infrastructure Update. 0x0054 value was added in SharePoint Server 2010.

<31> Section 2.14: This field is not part of the structure after the Office SharePoint Server 2007 Infrastructure Update.

<32> Section 2.17: This file was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<33> Section 2.17: This field was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<34> Section 2.17.1: This file was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<35> Section 2.17.1: This file was added as part of the Office SharePoint Server 2007 Infrastructure Update.

<36> Section 2.18: This file was removed in SharePoint Server 2010.

<37> Section 2.18: This file was removed in SharePoint Server 2010.

<38> Section 2.18.2: 1 In Office SharePoint Server 2007 and SharePoint Server 2010, the presence or lack of these records does not affect behavior.

7 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.

Section	Description	Revision class
2.2.4.1 Header File Format	Updated product behavior note for File version.	Major

8 Index

A

Active anchor text catalog
  full-text index 81
Anchor scope index key 28
Anchor text catalog
  full-text catalog 80
Applicability 11
Average document length file format 66
Average document length files example 101

B

Basic scope index directory example 86
Basic scope index example 88
Basic scope index key 25
BitCompress(K) field structure 19
Bitmap document set 62
BitStream DWORD 18
BitStream field structures 19
  BitCompress(K) 19
  DocIDCountCompress 22
  PidCompress 22
  PrefixSuffixCompress 23
BitStream file format 17
BitStream page structure 17
BitStreamPosition 18
BOF index key 25

C

CAVDLItem structure 66
Change tracking 203
Character normalization tables 121
CheckSummed recoverable storage file format 31
CheckSummedRecord structure 31
CIX File example 113
Click distance file 75
CMergeSplitKey structure 70
Coding table example 115
CodingTableEntry structure 57
Compound scope index directory example 83
Compound scope index example 85
Compound scope index key 27
Constants
  MaxOccBuckets table 12
  property identifier 12
Content index
  DirectoryEntry 58
Content index extension
  CodingTableEntry 57
  EncodedDOCIDDelta 59
  ExtensionCompressionTablePage 55
  ExtensionDataPage 57
  KeyExtensionData structure 54
  SymbolCategory 56
Content index file example 90
Content index file format (section 2.3 34, section 2.6 54)
Content index key 25
Content index record example 92
Content index record with skips example 95
ContentIndexRecord 36

D

Data file format
  recoverable storage 30
Details
  BitStream DWORD 18
  BitStream page structure 17
  BitStreamPosition 18
Detected language files 72
Detected language files example 104
Diacritic setting file 76
Diacritic settings file example 113
DirectoryEntry structure 58
DOCID bit stream example 117
DocIDCountCompress field structure 22
Document set files 59
Document set files example 95

E

EncodedDOCIDDelta structure 59
End of page example 116
EOF index key 25
Examples
  average document length files 101
  basic scope index 88
  basic scope index directory 86
  CIX File 113
  coding table 115
  compound scope index 85
  compound scope index directory 83
  content index file 90
  content index record 92
  content index record with skips 95
  detected language files 104
  diacritic settings file 113
  DOCID bit stream 117
  document set files 95
  end of page 116
  ExtensionCompressionTablePage 115
  ExtensionDataPage 116
  Full-text Index Catalog 82
  index directory 92
  index lexicon file 113
  index table file 109
  OccCount bit stream 118
  page-start page directory 116
  page-start symbol category descriptors 115
  physical file on disk 114
  query-independent rank files 106
ExtensionCompressionTablePage example 115
ExtensionCompressionTablePage structure 55
ExtensionDataPage example 116
ExtensionDataPage structure 57

F

Fields - vendor-extensible 11

File content
merge log 69
Full-text index
active anchor text catalog 81
anchor text catalog 80
main catalog 80
Full-text index catalog 78
Full-text Index Catalog example 82
Full-text index component 76
naming conventions 77

G

Glossary 7

I

Implementer - security considerations 120
Index directory
file header structure 50
file layout 47
first page structure 48
page header structure 50
page structure 49
record buffer structure 51
record structure 51
Index directory example 92
Index directory file format 47
Index keys
anchor scope 28
basic scope 25
BOF 25
compound scope 27
content 25
EOF 25
Max 25
string normalization 24
structures 24
Index lexicon file 75
Index lexicon file example 113
Index table
CIndexRecord 72
Indextype enumeration 73
user header 72
Index table file example 109
Index table file format 72
Indexed bitmap document set 64
Indextype enumeration 73
Informative references 10
Introduction 7

K

KeyExtensionData structure 54

L

List document set 60
Localization 11

M

Main catalog
full-text index 80
Max index key 25
Merge log
CMergeSplitKey structure 70
file content 69
user header format 68
Merge log file format 67
Merge process 67

N

Naming conventions
full-text index component 77
Normative references 10

O

OccCount bit stream example 118
Overview (synopsis) 10

P

Page-start page directory example 116
Page-start symbol category descriptors example 115
Physical file on disk example 114
PidCompress field structure 22
PrefixSuffixCompress field structure 23
Product behavior 200

Q

Query-independent rank files 72
Query-independent rank files example 106

R

Recoverable storage
Data file format 30
header file format 28
Recoverable storage file format 28
References 10
informative 10
normative 10
Relationship to protocols and other structures 10

S

Scope index file format 45
ScopeIndexRecord 45
Security - implementer considerations 120
Sparse array file format 31
SparseArrayBlock structure 33
SparseArrayBlockData structure 33
String normalization
Index keys 24
Structure
DirectoryEntry 58
Structures
anchor scope index key 28
average document length file format 66
basic scope index key 25
BitCompress(K) 19
bitmap document set 62
BitStream DWORD 18
BitStream field 19

BitStream file format 17
BitStream page structure 17
BitStreamPosition 18
BOF index key 25
CAVDLItem 66
CheckSummed recoverable storage file format 31
CheckSummedRecord 31
CIndexRecord 72
click distance file 75
CMergeSplitKey 70
CodingTableEntry 57
compound scope index key 27
content index file format (section 2.3 34, section 2.6 54)
content index key 25
ContentIndexRecord 36
detected language files 72
diacritic setting file 76
DocIDCountCompress 22
document set files 59
EncodedDOCIDDelta 59
EOF index key 25
ExtensionCompressionTablePage 55
ExtensionDataPage 57
full-text index catalog 78
full-text index component 76
index directory file format 47
index keys 24
index lexicon file 75
index table file format 72
indexed bitmap document set 64
Indextype enumeration 73
KeyExtensionData 54
list document set 60
Max index key 25
MaxOccBuckets table 12
merge log file format 67
merge process 67
PidCompress 22
PrefixSuffixCompress 23
property identifier 12
query-independent rank files 72
recoverable storage file format 28
scope index file format 45
ScopeIndexRecord 45
sparse array file format 31
SparseArrayBlock 33
SparseArrayBlockData 33
SymbolCategory 56
user header 72
user header format 68
SymbolCategory structure 56

T

Tracking changes 203

V

Vendor-extensible fields 11
Versioning 11