

python / cpython Public

<> Code Issues 5k+ Pull requests 2.1k Actions Projects Security and q

# Commit 333d4a6



serhiy-storchaka authored on Oct 7, 2025 · 37 / 41 · Verified

[3.13] [gh-139700](#): Check consistency of the zip64 end of central directory record ([GH-139702](#)) ([GH-139708](#))

Support records with "zip64 extensible data" if there are no bytes prepended to the ZIP file.  
(cherry picked from commit [162997b](#))

3.13 (AcreationOS-Linux/python#2, #139708) · v3.13.13 ... v3.13.10

1 parent [527623e](#) commit 333d4a6

3 files changed

+113 -23

Top

- ✓ Lib
  - ✓ test/test\_zipfile
    - test\_core.py
  - ✓ zipfile
    - \_\_init\_\_.py
- ✓ Misc/NEWS.d/next/Security
  - 2025-10-07-19-31-34.gh-issue-139700.vNHU1O.rst



Lib/test/test\_zipfile/test\_core.py



```

..... @@ -884,6 +884,8 @@ def make_zip64_file(
884      884          self, file_size_64_set=False, file_size_extra=False,
```

```

885 885         compress_size_64_set=False, compress_size_extra=False,
886 886         header_offset_64_set=False, header_offset_extra=False,
887 887 +         extensible_data=b'',
888 888 +         end_of_central_dir_size=None, offset_to_end_of_central_dir=None,
887 889     ):
888 890         """Generate bytes sequence for a zip with (incomplete) zip64 data.
889 891
⋮
⋮
@@ -937,6 +939,12 @@ def make_zip64_file(
937 939
938 940         central_dir_size = struct.pack('<Q', 58 + 8 *
len(central_zip64_fields))
939 941         offset_to_central_dir = struct.pack('<Q', 50 + 8 *
len(local_zip64_fields))
942 942 +         if end_of_central_dir_size is None:
943 943 +             end_of_central_dir_size = 44 + len(extensible_data)
944 944 +         if offset_to_end_of_central_dir is None:
945 945 +             offset_to_end_of_central_dir = (108
946 946 +                 + 8 * len(local_zip64_fields)
947 947 +                 + 8 * len(central_zip64_fields))
940 948
941 949         local_extra_length = struct.pack("<H", 4 + 8 *
len(local_zip64_fields))
942 950         central_extra_length = struct.pack("<H", 4 + 8 *
len(central_zip64_fields))
⋮
⋮
@@ -965,14 +973,17 @@ def make_zip64_file(
965 973         + filename
966 974         + central_extra
967 975         # Zip64 end of central directory
968 968 -         + b"PK\x06\x06,\x00\x00\x00\x00\x00\x00\x00-\x00-"
969 969 -         + b"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00"
976 976 +         + b"PK\x06\x06"
977 977 +         + struct.pack('<Q', end_of_central_dir_size)
978 978 +         + b"-\x00-
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00"
970 979         + b"\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00"
971 980         + central_dir_size
972 981         + offset_to_central_dir
982 982 +         + extensible_data

```

```

973  983          # Zip64 end of central directory locator
974  -          + b"PK\x06\x07\x00\x00\x00\x001\x00\x00\x00\x00\x00\x00\x01"
975  -          + b"\x00\x00\x00"
          984  +          + b"PK\x06\x07\x00\x00\x00\x00"
          985  +          + struct.pack('<Q', offset_to_end_of_central_dir)
          986  +          + b"\x01\x00\x00\x00"
976  987          # end of central directory
977  988          + b"PK\x05\x06\x00\x00\x00\x00\x01\x00\x01\x00:\x00\x00\x002\x00"
978  989          + b"\x00\x00\x00\x00"
          ⚡
          ⚡
          @@ -1003,6 +1014,7 @@ def test_bad_zip64_extra(self):
1003 1014          with self.assertRaises(zipfile.BadZipFile) as e:
1004 1015              zipfile.ZipFile(io.BytesIO(missing_file_size_extra))
1005 1016              self.assertIn('file size', str(e.exception).lower())
          1017  +
          self.assertTrue(zipfile.is_zipfile(io.BytesIO(missing_file_size_extra)))
1006 1018
1007 1019          # zip64 file size present, zip64 compress size present, one field in
1008 1020          # extra, expecting two, equals missing compress size.
          ⚡
          @@ -1014,6 +1026,7 @@ def test_bad_zip64_extra(self):
1014 1026          with self.assertRaises(zipfile.BadZipFile) as e:
1015 1027              zipfile.ZipFile(io.BytesIO(missing_compress_size_extra))
1016 1028              self.assertIn('compress size', str(e.exception).lower())
          1029  +
          self.assertTrue(zipfile.is_zipfile(io.BytesIO(missing_compress_size_extra)))
1017 1030
1018 1031          # zip64 compress size present, no fields in extra, expecting one,
1019 1032          # equals missing compress size.
          ⚡
          @@ -1023,6 +1036,7 @@ def test_bad_zip64_extra(self):
1023 1036          with self.assertRaises(zipfile.BadZipFile) as e:
1024 1037              zipfile.ZipFile(io.BytesIO(missing_compress_size_extra))
1025 1038              self.assertIn('compress size', str(e.exception).lower())
          1039  +
          self.assertTrue(zipfile.is_zipfile(io.BytesIO(missing_compress_size_extra)))
1026 1040
1027 1041          # zip64 file size present, zip64 compress size present, zip64 header
1028 1042          # offset present, two fields in extra, expecting three, equals
          missing
          ⚡
          @@ -1037,6 +1051,7 @@ def test_bad_zip64_extra(self):
1037 1051          with self.assertRaises(zipfile.BadZipFile) as e:

```

```

1038 1052         zipfile.ZipFile(io.BytesIO(missing_header_offset_extra))
1039 1053         self.assertIn('header offset', str(e.exception).lower())
1054 +
self.assertTrue(zipfile.is_zipfile(io.BytesIO(missing_header_offset_extra)))
1040 1055
1041 1056         # zip64 compress size present, zip64 header offset present, one field
1042 1057         # in extra, expecting two, equals missing header offset
@@ -1049,6 +1064,7 @@ def test_bad_zip64_extra(self):
1049 1064         with self.assertRaises(zipfile.BadZipFile) as e:
1050 1065             zipfile.ZipFile(io.BytesIO(missing_header_offset_extra))
1051 1066             self.assertIn('header offset', str(e.exception).lower())
1067 +
self.assertTrue(zipfile.is_zipfile(io.BytesIO(missing_header_offset_extra)))
1052 1068
1053 1069         # zip64 file size present, zip64 header offset present, one field in
1054 1070         # extra, expecting two, equals missing header offset
@@ -1061,6 +1077,7 @@ def test_bad_zip64_extra(self):
1061 1077         with self.assertRaises(zipfile.BadZipFile) as e:
1062 1078             zipfile.ZipFile(io.BytesIO(missing_header_offset_extra))
1063 1079             self.assertIn('header offset', str(e.exception).lower())
1080 +
self.assertTrue(zipfile.is_zipfile(io.BytesIO(missing_header_offset_extra)))
1064 1081
1065 1082         # zip64 header offset present, no fields in extra, expecting one,
1066 1083         # equals missing header offset
@@ -1072,6 +1089,63 @@ def test_bad_zip64_extra(self):
1072 1089         with self.assertRaises(zipfile.BadZipFile) as e:
1073 1090             zipfile.ZipFile(io.BytesIO(missing_header_offset_extra))
1074 1091             self.assertIn('header offset', str(e.exception).lower())
1092 +
self.assertTrue(zipfile.is_zipfile(io.BytesIO(missing_header_offset_extra)))
1093 +
1094 +         def test_bad_zip64_end_of_central_dir(self):
1095 +             zipdata = self.make_zip64_file(end_of_central_dir_size=0)
1096 +             with self.assertRaisesRegex(zipfile.BadZipFile, 'Corrupt.*record'):
1097 +                 zipfile.ZipFile(io.BytesIO(zipdata))
1098 +             self.assertFalse(zipfile.is_zipfile(io.BytesIO(zipdata)))
1099 +
1100 +             zipdata = self.make_zip64_file(end_of_central_dir_size=100)
1101 +             with self.assertRaisesRegex(zipfile.BadZipFile, 'Corrupt.*record'):

```

```
1102 +         zipfile.ZipFile(io.BytesIO(zipdata))
1103 +         self.assertFalse(zipfile.is_zipfile(io.BytesIO(zipdata)))
1104 +
1105 +         zipdata = self.make_zip64_file(offset_to_end_of_central_dir=0)
1106 +         with self.assertRaisesRegex(zipfile.BadZipFile, 'Corrupt.*record'):
1107 +             zipfile.ZipFile(io.BytesIO(zipdata))
1108 +         self.assertFalse(zipfile.is_zipfile(io.BytesIO(zipdata)))
1109 +
1110 +         zipdata = self.make_zip64_file(offset_to_end_of_central_dir=1000)
1111 +         with self.assertRaisesRegex(zipfile.BadZipFile, 'Corrupt.*locator'):
1112 +             zipfile.ZipFile(io.BytesIO(zipdata))
1113 +         self.assertFalse(zipfile.is_zipfile(io.BytesIO(zipdata)))
1114 +
1115 +     def test_zip64_end_of_central_dir_record_not_found(self):
1116 +         zipdata = self.make_zip64_file()
1117 +         zipdata = zipdata.replace(b"PK\x06\x06", b'\x00'*4)
1118 +         with self.assertRaisesRegex(zipfile.BadZipFile, 'record not found'):
1119 +             zipfile.ZipFile(io.BytesIO(zipdata))
1120 +         self.assertFalse(zipfile.is_zipfile(io.BytesIO(zipdata)))
1121 +
1122 +         zipdata = self.make_zip64_file(
1123 +             extensible_data=b'\xca\xfe\x04\x00\x00\x00data')
1124 +         zipdata = zipdata.replace(b"PK\x06\x06", b'\x00'*4)
1125 +         with self.assertRaisesRegex(zipfile.BadZipFile, 'record not found'):
1126 +             zipfile.ZipFile(io.BytesIO(zipdata))
1127 +         self.assertFalse(zipfile.is_zipfile(io.BytesIO(zipdata)))
1128 +
1129 +     def test_zip64_extensible_data(self):
1130 +         # These values are what is set in the make_zip64_file method.
1131 +         expected_file_size = 8
1132 +         expected_compress_size = 8
1133 +         expected_header_offset = 0
1134 +         expected_content = b"test1234"
1135 +
1136 +         zipdata = self.make_zip64_file(
1137 +             extensible_data=b'\xca\xfe\x04\x00\x00\x00data')
1138 +         with zipfile.ZipFile(io.BytesIO(zipdata)) as zf:
1139 +             zinfo = zf.infolist()[0]
1140 +             self.assertEqual(zinfo.file_size, expected_file_size)
1141 +             self.assertEqual(zinfo.compress_size, expected_compress_size)
```

```

1142 +         self.assertEqual(zinfo.header_offset, expected_header_offset)
1143 +         self.assertEqual(zf.read(zinfo), expected_content)
1144 +         self.assertTrue(zipfile.is_zipfile(io.BytesIO(zipdata)))
1145 +
1146 +         with self.assertRaisesRegex(zipfile.BadZipFile, 'record not found'):
1147 +             zipfile.ZipFile(io.BytesIO(b'prepending' + zipdata))
1148 +         self.assertFalse(zipfile.is_zipfile(io.BytesIO(b'prepending' +
zipdata)))

```

1075 1149

1076 1150 def test\_generated\_valid\_zip64\_extra(self):

1077 1151 # These values are what is set in the make\_zip64\_file method.



Lib/zipfile/\_\_init\_\_.py



```
@@ -245,41 +245,57 @@ def is_zipfile(filename):
```

245 245 else:

246 246 with open(filename, "rb") as fp:

247 247 result = \_check\_zipfile(fp)

248 - except OSError:

248 + except (OSError, BadZipFile):

249 249 pass

250 250 return result

251 251

252 252 def \_EndRecData64(fpin, offset, endrec):

253 253 """

254 254 Read the ZIP64 end-of-archive records and use that to update endrec

255 255 """

256 - try:

257 - fpin.seek(offset - sizeEndCentDir64Locator, 2)

258 - except OSError:

259 - # If the seek fails, the file is not large enough to contain a ZIP64

256 + offset -= sizeEndCentDir64Locator

257 + if offset &lt; 0:

258 + # The file is not large enough to contain a ZIP64

260 259 # end-of-archive record, so just return the end record we were given.

261 260 return endrec

262 -

261 + fpin.seek(offset)

263 262 data = fpin.read(sizeEndCentDir64Locator)

264 263 if len(data) != sizeEndCentDir64Locator:

265	-	return endrec
264	+	raise OSError("Unknown I/O error")
266	265	sig, diskno, reloff, disks = struct.unpack(structEndArchive64Locator, data)
267	266	if sig != stringEndArchive64Locator:
268	267	return endrec
269	268	
270	269	if diskno != 0 or disks > 1:
271	270	raise BadZipFile("zipfiles that span multiple disks are not supported")
272	271	
273	-	# Assume no 'zip64 extensible data'
274	-	fpin.seek(offset - sizeEndCentDir64Locator - sizeEndCentDir64, 2)
272	+	offset -= sizeEndCentDir64
273	+	if reloff > offset:
274	+	raise BadZipFile("Corrupt zip64 end of central directory locator")
275	+	# First, check the assumption that there is no prepended data.
276	+	fpin.seek(reloff)
277	+	extrasz = offset - reloff
275	278	data = fpin.read(sizeEndCentDir64)
276	279	if len(data) != sizeEndCentDir64:
277	-	return endrec
280	+	raise OSError("Unknown I/O error")
281	+	if not data.startswith(stringEndArchive64) and reloff != offset:
282	+	# Since we already have seen the Zip64 EOCD Locator, it's
283	+	# possible we got here because there is prepended data.
284	+	# Assume no 'zip64 extensible data'
285	+	fpin.seek(offset)
286	+	extrasz = 0
287	+	data = fpin.read(sizeEndCentDir64)
288	+	if len(data) != sizeEndCentDir64:
289	+	raise OSError("Unknown I/O error")
290	+	if not data.startswith(stringEndArchive64):
291	+	raise BadZipFile("Zip64 end of central directory record not found")
292	+	
278	293	sig, sz, create_version, read_version, disk_num, disk_dir, \
279	294	dircount, dircount2, dirsiz, diroffset = \
280	295	struct.unpack(structEndArchive64, data)
281	-	if sig != stringEndArchive64:
282	-	return endrec

```

296 +     if (diroffset + dirsiz != reloff or
297 +         sz + 12 != sizeEndCentDir64 + extrasz):
298 +         raise BadZipFile("Corrupt zip64 end of central directory record")

283 299
284 300     # Update the original endrec using data from the ZIP64 record
285 301     endrec[_ECD_SIGNATURE] = sig
@@ -289,6 +305,7 @@ def _EndRecData64(fpin, offset, endrec):
289 305     endrec[_ECD_ENTRIES_TOTAL] = dircount2
290 306     endrec[_ECD_SIZE] = dirsiz
291 307     endrec[_ECD_OFFSET] = diroffset
308 +     endrec[_ECD_LOCATION] = offset - extrasz
292 309     return endrec
293 310
294 311
@@ -322,7 +339,7 @@ def _EndRecData(fpin):
322 339     endrec.append(filesiz - sizeEndCentDir)
323 340
324 341     # Try to read the "Zip64 end of central directory" structure
325 -     return _EndRecData64(fpin, -sizeEndCentDir, endrec)
342 +     return _EndRecData64(fpin, filesiz - sizeEndCentDir, endrec)
326 343
327 344     # Either this is not a ZIP file, or it is a ZIP file with an archive
328 345     # comment. Search the end of the file for the "end of central directory"
@@ -346,8 +363,7 @@ def _EndRecData(fpin):
346 363     endrec.append(maxCommentStart + start)
347 364
348 365     # Try to read the "Zip64 end of central directory" structure
349 -     return _EndRecData64(fpin, maxCommentStart + start - filesiz,
350 -                          endrec)
366 +     return _EndRecData64(fpin, maxCommentStart + start, endrec)
351 367
352 368     # Unable to find a valid end of central directory structure
353 369     return None
@@ -1458,9 +1474,6 @@ def _RealGetContents(self):
1458 1474
1459 1475     # "concat" is zero, unless zip was concatenated to another file
1460 1476     concat = endrec[_ECD_LOCATION] - siz_cd - offset_cd
1461 -     if endrec[_ECD_SIGNATURE] == stringEndArchive64:

```

```

1462 - # If Zip64 extension structures are present, account for them
1463 - concat -= (sizeEndCentDir64 + sizeEndCentDir64Locator)
1464 1477
1465 1478     if self.debug > 2:
1466 1479         inferred = concat + offset_cd
    ↓
    ↑
@@ -2082,7 +2095,7 @@ def _write_end_record(self):
    " would require ZIP64 extensions")
2083 2096     zip64endrec = struct.pack(
2084 2097         structEndArchive64, stringEndArchive64,
2085 - 44, 45, 45, 0, 0, centDirCount, centDirCount,
2098 + sizeEndCentDir64 - 12, 45, 45, 0, 0, centDirCount,
    centDirCount,
2086 2099         centDirSize, centDirOffset)
2087 2100     self.fp.write(zip64endrec)
2088 2101
    ↓

```

...5-10-07-19-31-34.gh-issue-139700.vNHU10.rst

```

... @@ -0,0 +1,3 @@
1 + Check consistency of the zip64 end of central directory record. Support
2 + records with "zip64 extensible data" if there are no bytes prepended to the
3 + ZIP file.

```

Comments 0