

system.res

god_menu.res

SYSTEM_UPDATE.EDAT

877_Step_DlcSurvival051.res

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	50	72	65	73	20	00	00	00	08	02	6E	7C	03	00	00	00	Presn
00000010	70	23	00	00	00	00	00	00	00	00	00	00	00	00	00	00	p#.....
00000020	80	0E	00	00	03	00	00	00	60	0E	00	00	01	00	00	00	€.....`.....
00000030	60	00	00	00	1F	00	00	00	00	00	00	00	00	00	00	00	`.....
00000040	40	04	00	00	47	00	00	00	20	0D	00	00	0A	00	00	00	@...G... ..
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

[Start Point]

- ~ 0x00: Magic/Header: 4 bytes (Pres)
- ~ 0x04: 4 bytes, little endian offset (0x20). jumps to table offset
- ~ 0x08: byte, group table count. (always set to **08**)
- ~ 0x09: some kind of value?. changes on different RES archives.
- ~ 0x0A: 2 bytes checksum?. not sure what to do with it. (but i guess it could be a a MD5 hash due to **libmd5.prx** being used)
- ~ 0x0C: 4 bytes, seems to be stuck at 3, must be a version of that RES archive?
- ~ 0x10: 4 bytes little endian offset, jumps you to where certain chunks of data are stored (related to ToC)
- ~ 0x14->0x1F: lengthy zeroes, 4 bytes each. not really useful mostly. but can contain other data like **0x10** out of nowhere so keep an eye out.

[Table Offsets] ranging: 0x20-0x5F

- ~ 0x20: empty usually on some other .res archive, but it represent as ToC offset (`00 15 00 00` -> 0x1500 for example)
- ~ 0x24: empty usually on some other .res archive, but it represent as count for ToC (`03 00 00 00` -> 3 for example)
 - Table Offsets always uses 16 bytes. though sometimes the other 8 bytes will be an empty table, be creative when reading them.
 - Remember to multiply the Table Counts by 32 bytes (*count*32*) as ToC uses 32 bytes, if 0x1500 has 3 counts of TOC. then multiply that to get 96 bytes (or 60 in hex interpretation). This will accurately pinpoint where the Table of `0x1500` ends. (use addition to check where the table ends, which **0x1500** ends at **0x1560**. basically: 0x1500+60=0x1560)

[Table of Contents] (ToC) [32 bytes in size]

~ **File Offset**: a little endian offset. brings you to a area where a offset chunk is stored.

~ **Size/Compressed Size**: defines the size of that area starting with **File Offset**. **Size** can vary if it's **ZLIB** compressed or not based on it's header (blz2)

~ **Name Offset Table**: a little endian offset. brings you to a area where the name is located. Which is a name table on how i call it.

~ **Name Element**: a name element (that's how i call it). can vary from 1 and 3. (can go above 5 on PSVITA SIDE)

~ **Zeroes**: bunch of zeroes, length of 12 (0xC)

~ **Size (Decompressed)**: Game's expected decompressed size of that **Compressed Size**. Value can match with **Size** if it's not compressed by **ZLIB**.

[Identifying External File Data]

in **0x940**. you see that the 4th byte has **`40`**. let's call this **Address Types**. If you encounter any of these values, it means that the file source is external. here's the following lists for these values:

~ 00 (0x0) = none

~ 30 (0x3) = Nosets (for PSvita, usually exists in folders `data_[value]`)

~ 40 (0x4) = package.rdp

~ 50 (0x5) = data.rdp

~ 60 (0x6) = patch.rdp

If you encounter any of these. You will need to do multiplications.

~ for **0xC**, it's a current/internal file. so it's inside of the **.res** file being read. as for **0xD**. it seem to be a **non-dlc** type of file, as the file already exists in between **data.rdp** or **package.rdp**, and has not been modified inside of **patch.rdp** and to be loaded when needed.

000007C0	10 12 00 C0	01 00 00 00	20 12 00 00	01 00 00 00	...À.....
000007D0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000007E0	40 12 00 C0	0F 00 00 00	80 12 00 00	01 00 00 00	@..À....€.....
000007F0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000800	A0 12 00 C0	01 00 00 00	B0 12 00 00	01 00 00 00	..À....°.....
00000810	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000820	D0 12 00 C0	01 00 00 00	E0 12 00 00	01 00 00 00	Ð..À....à.....
00000830	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000840	00 13 00 C0	03 00 00 00	10 13 00 00	01 00 00 00	...À.....
00000850	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000860	30 13 00 C0	08 00 00 00	50 13 00 00	01 00 00 00	o..À....P.....
00000870	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000880	80 13 00 C0	08 00 00 00	A0 13 00 00	01 00 00 00	€..À.....
00000890	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000008A0	D0 13 00 C0	08 00 00 00	F0 13 00 00	01 00 00 00	Ð..À....ð.....
000008B0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000008C0	00 00 00 00	00 00 00 00	20 14 00 00	01 00 00 00
000008D0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000008E0	40 14 00 C0	A8 00 00 00	F0 14 00 00	01 00 00 00	@..À~...ð.....
000008F0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000900	10 15 00 C0	12 00 00 00	30 15 00 00	01 00 00 00	...À....o.....
00000910	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000920	50 15 00 C0	02 00 00 00	60 15 00 00	01 00 00 00	P..À....`.....
00000930	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000940	F5 78 05 40	E4 7E 19 00	70 15 00 00	03 00 00 00	õx.@ä~..p.....
00000950	00 00 00 00	00 00 00 00	00 00 00 00	E4 7E 19 00ä~..

Obtaining **Absolute Offset** (for external files)

~ Reassign **0x940**'s **File Offset** into Big Endian order, you'll get something like this: "400578F5"/"0x400578F5".

~ Remove or Change "4" to "0" and multiply the new offset by 800:

"400578F5" * "800" = "2BC7A800"/"0x2BC7A800"

~ The **true/absolute offset** is "2BC7A800"

***.rtbl** files have this structures but spreads out of nowhere with lengthy paddings keep an eye out.

[Name Tables] (PSP SIDE) (16 bytes in total. 4 bytes each)

Note: If **Name Element** is less than three (sometimes it could be 1). it will rely only on **Name Offset**. There's only **1**, **3**, and **4** value on the PSP side, so if its **4**, then there's **4** tables despite not having a number **2** value.

~ 1 = **Name Offset**: Little Endian offset, this is an offset that jumps you to the **File Name**.

~ 2 = **Extension Offset**: Little Endian offset. this offset jumps you to the **File Extension**.

~ 3 and 4 = **Path**: an offset that jumps you to the **zeros**. but if its non-zero, then it is a path (folder name). usually can form with `/` or not. 4th table **Path** usually shares the same value is 3rd table but with a +1 offset.

[illegible]

[Name Tables] (PSVITA SIDE) (20 bytes in total, 4 bytes each)

Note: continuation of Name Tables (PSP Side). this is for the value **5**. value **3** is treated different here if value is above **4**.

~ 5 = **Instruction Directory**: an offset that jumps you to a instruction of that file's directory. A exclusion file that is outside of the RDP file

00C800	14 C8 00 00	20 C8 00 00	29 C8 00 00	2A C8 00 00	.È.. È..)È.*È..
00C810	36 C8 00 00	62 67 6D 5F	65 61 62 5F	30 30 31 00	6È..bgm_eab_001.
00C820	6E 75 73 33	62 61 6E 6B	00 00 45 58	43 4C 55 44	nus3bank..EXCLUD
00C830	45 5F 52 44	50 00 50 41	54 48 3D 64	61 74 61 5F	E_RDP.PATH=data_
00C840	38 2F 62 67	6D 5F 65 61	62 5F 30 30	31 2E 6E 75	8/bg_m_eab_001.nu
00C850	73 33 62 61	6E 6B 00 00	00 00 00 00	00 00 00 00	s3bank.....

[ZLIB Compression] *BLZ2 only, i have no clue much on BLZ4*

0x00: Header 4 bytes. "blz2"

0x04: Compressed file size, 2 bytes.

the rest: compressed data.

Padding: adjustment for another file chunk or compressed data

[decompression sample code (python)]

```
def blz_decompress(data, csize, dsize):
```

```
data = BytesIO(data)
```

```
magic = data.read(4)
```

```
if magic != b"blz2":
```

```
raise ValueError("Data is not in BLZ2 format.")
```

```
decom = b""
```

```
if dsize >= 0xFFFF:
```

```
size = int.from_bytes(data.read(2), "little")
```

```
dekor = zlib.decompress(data.read(size), -15)
```

```
while data.tell() < csize:
```

```
size = int.from_bytes(data.read(2), "little")
```

```
decom += zlib.decompress(data.read(size), -15)
```

```
return decomp + ekor
```

```
else:
```

```
size = int.from_bytes(data.read(2), "little")
```

```
decom = zlib.decompress(data.read(size), -15)
```

return decom

62	6C	7A	32	7E	00	2B	2E	2E	4C	11	60	60	60	60	62	blz2~.+..L.````b
90	63	60	07	D2	2C	4C	40	82	81	11	88	FF	FF	67	18	.c`.Ò,L@,..^ÿÿg.
04	40	92	09	C6	82	B8	87	03	08	D1	81	01	D0	B9	09	.@'.E, ,+..Ñ..Ð¹.
40	3C	81	11	55	FC	00	23	6E	73	5F	00	E5	3E	00	F1	@<..Uü.#ns_.å>.ñ
0F	74	35	4C	18	4A	99	98	98	A0	41	02	06	05	4E	D8	.t5L.J™~ ~ A...NØ
8D	C5	25	CE	41	A2	78	83	3D	2E	67	33	62	15	63	64	.Å%ÎAçxf=.g3b.cd
C0	A1	FE	3F	1A	06	82	07	0E	D8	E8	E0	CA	E2	92	D4	À;p?.., ..øèàÊâ'Ô
DC	78	B7	C4	94	D4	78	A7	9C	C4	E4	EC	78	CF	3C	06	Üx·Ã"Ôx\$œÄäixĩ<.
5D	88	0A	00	00	00	00	00	00	00	00	00	00	00	00	00] ^.....