**2025-06-09-Analysis-Detalis-Python**

import plistlib

import hashlib

from datetime import datetime

# ファイルパス

file\_path = "/mnt/data/log-bb-2025-06-09-stats.plist"

# plistファイルを読み込み

with open(file\_path, "rb") as f:

plist\_data = plistlib.load(f)

# ハッシュ値計算

with open(file\_path, "rb") as f:

file\_bytes = f.read()

md5\_hash = hashlib.md5(file\_bytes).hexdigest()

sha256\_hash = hashlib.sha256(file\_bytes).hexdigest()

# タイムスタンプ

timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

# データの最初の部分だけ抜粋（中身が大きい可能性あるので）

sample\_preview = str(plist\_data)[:1500]

(md5\_hash, sha256\_hash, timestamp, sample\_preview[:500])

📂 ファイル名: log-bb-2025-06-09-stats.plist

🕒 タイムスタンプ: 2025-08-18 17:24:33

🔑 MD5: b59096ec788e7ae09d447f5a28cead76

🔑 SHA256: 9dac1275b7066dad634f9ba0d5a232f52b3b986bdf4c3ca68f5abce6e633e6e4

{

'SysInfoCrashReporterKey': 'd0cc51c9f68453b5fbd99a25033b927fc838473e',

'bug\_type': '195',

'description': 'Version=4

IncidentIdentifier=0F41A1FA-00C6-4DBA-9E26-959A4EDA735F

Date=2025-06-09;AP=22F76;BB=5.51.03;Machine=iPhone13,1

22:28:33 +0700 ST [clm] disconnect: duration=88;cause=kNoError;modemErr=145;rat=2

23:50:15 +0700 ST [clm] disconnect: duration=6;cause=kNoError;modemErr=145;rat=2

9:40:50 +0700 ST [clm] disconnect: duration=0;cause=kUserBusy;modemErr=146;rat=2

',

'os\_version': 'iPho...

}

import pandas as pd

# plist全体のデータを確認

plist\_data.keys()

# descriptionの中に時系列ログがまとめられているので、改行で分割して表形式にする

description\_text = plist\_data.get("description", "")

lines = [line.strip() for line in description\_text.split("\n") if line.strip()]

# ログの中で日時が含まれる行を抽出

parsed\_entries = []

for line in lines:

if "+0700" in line: # タイムゾーン入りの行を対象

parts = line.split(" ", 2)

if len(parts) >= 3:

time\_str = f"{parts[0]} {parts[1]}"

detail = parts[2]

parsed\_entries.append({"timestamp": time\_str, "event": detail})

# DataFrame化

df = pd.DataFrame(parsed\_entries)

import caas\_jupyter\_tools

caas\_jupyter\_tools.display\_dataframe\_to\_user("Baseband Disconnect Logs", df)

import plistlib, re, hashlib, json

from datetime import datetime, timezone, timedelta

import pandas as pd

from pathlib import Path

TZ = timezone(timedelta(hours=7)) # Asia/Ho\_Chi\_Minh

src = Path("/mnt/data/log-bb-2025-06-09-stats.plist")

# Load plist

with src.open("rb") as f:

data = plistlib.load(f)

desc = data.get("description","")

# Extract header fields from description

header = {}

# IncidentIdentifier

m = re.search(r"IncidentIdentifier=([A-F0-9-]{10,})", desc)

if m: header["IncidentIdentifier"] = m.group(1)

# Date

m = re.search(r"Date=(\d{4}-\d{2}-\d{2})", desc)

if m: header["Date"] = m.group(1)

# AP build, BB version, Machine

m = re.search(r"AP=([0-9A-Za-z]+);BB=([\d\.]+);Machine=([A-Za-z0-9\_,]+)", desc)

if m:

header["AP"] = m.group(1)

header["BB"] = m.group(2)

header["Machine"] = m.group(3)

# Other keys from plist root

header["bug\_type"] = data.get("bug\_type")

header["os\_version"] = data.get("os\_version")

header["SysInfoCrashReporterKey"] = data.get("SysInfoCrashReporterKey")

if "system\_ID" in data:

header["system\_ID"] = data.get("system\_ID")

# Parse timeline events (lines with +0700)

lines = [ln.strip() for ln in desc.splitlines() if "+0700" in ln]

events = []

for ln in lines:

# e.g. "22:28:33 +0700 ST [clm] disconnect: duration=88;cause=kNoError;modemErr=145;rat=2"

tm\_part, rest = ln.split(" +0700 ", 1)

detail = rest.strip()

# Extract kvs after ":"

kv = {}

if ":" in detail:

action, tail = [s.strip() for s in detail.split(":", 1)]

kv["action"] = action

for part in tail.split(";"):

if "=" in part:

k,v = part.split("=",1)

kv[k.strip()] = v.strip()

else:

kv["action"] = detail

# Build ISO timestamp

date\_str = header.get("Date")

if date\_str:

iso = f"{date\_str}T{tm\_part}+07:00"

ts = datetime.fromisoformat(iso)

else:

ts = None

iso = None

kv["timestamp\_local"] = iso

kv["raw\_line"] = ln

events.append(kv)

df = pd.DataFrame(events)

# Normalize columns

for col in ["duration","modemErr","rat"]:

if col in df.columns:

df[col] = pd.to\_numeric(df[col], errors="coerce")

# Deduplicate

if not df.empty:

df = df.drop\_duplicates(subset=["timestamp\_local","action","duration","cause","modemErr","rat"], keep="first")

# Suspicion flags

flags = []

if not df.empty:

# cause=kNoError with disconnect

mask\_noerr = df["action"].str.contains("disconnect", na=False) & (df.get("cause","")=="kNoError")

if mask\_noerr.any():

flags.append("disconnect\_without\_error")

# duration 0 or very small with disconnect

if "duration" in df.columns and (df["duration"]<=1).any():

flags.append("zero\_duration\_disconnect")

# repeated within short interval

if len(df)>=2:

ts\_series = pd.to\_datetime(df["timestamp\_local"])

ts\_series = ts\_series.sort\_values()

deltas = ts\_series.diff().dropna().dt.total\_seconds()

if (deltas<=7200).any(): # <=2 hours

flags.append("repeated\_disconnect\_within\_2h")

# Extract any Session IDs, PIDs heuristically

ids = {

"UUID\_like": sorted(set(re.findall(r"\b[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\b", desc, flags=re.I))),

"session\_tokens": sorted(set(re.findall(r"\b(session(?:\_start|\_id)?|guid|clientid|usageClientId)\s\*[:=]\s\*([A-Za-z0-9\-]+)", desc, flags=re.I))),

"PIDs": sorted(set(re.findall(r"\bPID[:=]\s\*(\d+)\b", desc))),

}

# Compute hashes for the file

raw = src.read\_bytes()

md5 = hashlib.md5(raw).hexdigest()

sha256 = hashlib.sha256(raw).hexdigest()

# Save CSV and Report

csv\_out = Path("/mnt/data/bb\_2025-06-09\_events.csv")

df.to\_csv(csv\_out, index=False)

report = {

"file": src.name,

"hashes": {"md5": md5, "sha256": sha256},

"identifiers": header,

"timeline\_rows": len(df),

"suspicion\_flags": flags,

"extracted\_ids": {

"IncidentIdentifier": header.get("IncidentIdentifier"),

"SysInfoCrashReporterKey": header.get("SysInfoCrashReporterKey"),

"system\_ID": header.get("system\_ID"),

"UUID\_hits": ids["UUID\_like"],

"Session\_hits": [f"{k}={v}" for (k,v) in ids["session\_tokens"]],

"PID\_hits": ids["PIDs"],

},

}

report\_path = Path("/mnt/data/REPORT\_bb\_2025-06-09.txt")

with report\_path.open("w", encoding="utf-8") as f:

f.write(json.dumps(report, ensure\_ascii=False, indent=2))

(csv\_out.as\_posix(), report\_path.as\_posix(), df.head(10).to\_dict(orient="records"), report["suspicion\_flags"])

**ValueError Traceback (most recent call last) Cell In[4], line 62 60 if date\_str: 61 iso = f"{date\_str}T{tm\_part}+07:00" ---> 62 ts = datetime.fromisoformat(iso) 63 else: 64 ts = None ValueError: Invalid isoformat string: '2025-06-09T9:40:50+07:00'**

# Fix 1-digit hour by padding to 2 digits

fixed\_events = []

for ev in events:

iso = ev.get("timestamp\_local")

if iso:

# Pad hour if needed

m = re.match(r"(\d{4}-\d{2}-\d{2})T(\d{1,2}:\d{2}:\d{2})(\+07:00)", iso)

if m:

date\_part, time\_part, tz\_part = m.groups()

h, mi, s = time\_part.split(":")

if len(h)==1:

time\_part = f"0{h}:{mi}:{s}"

iso = f"{date\_part}T{time\_part}{tz\_part}"

ev["timestamp\_local"] = iso

fixed\_events.append(ev)

df = pd.DataFrame(fixed\_events)

# Normalize and resave

for col in ["duration","modemErr","rat"]:

if col in df.columns:

df[col] = pd.to\_numeric(df[col], errors="coerce")

df = df.drop\_duplicates(subset=["timestamp\_local","action","duration","cause","modemErr","rat"], keep="first")

# Suspicion flags again

flags = []

if not df.empty:

mask\_noerr = df["action"].str.contains("disconnect", na=False) & (df.get("cause","")=="kNoError")

if mask\_noerr.any():

flags.append("disconnect\_without\_error")

if "duration" in df.columns and (df["duration"]<=1).any():

flags.append("zero\_duration\_disconnect")

if len(df)>=2:

ts\_series = pd.to\_datetime(df["timestamp\_local"])

ts\_series = ts\_series.sort\_values()

deltas = ts\_series.diff().dropna().dt.total\_seconds()

if (deltas<=7200).any():

flags.append("repeated\_disconnect\_within\_2h")

# Save outputs

csv\_out = Path("/mnt/data/bb\_2025-06-09\_events.csv")

df.to\_csv(csv\_out, index=False)

report["timeline\_rows"] = len(df)

report["suspicion\_flags"] = flags

with report\_path.open("w", encoding="utf-8") as f:

f.write(json.dumps(report, ensure\_ascii=False, indent=2))

(csv\_out.as\_posix(), report\_path.as\_posix(), df.to\_dict(orient="records"), flags)

------------------------------------------------------------------------------------------------------------------------------------------

**NameError Traceback (most recent call last) Cell In[5], line 45 42 csv\_out = Path("/mnt/data/bb\_2025-06-09\_events.csv") 43 df.to\_csv(csv\_out, index=False) ---> 45 report["timeline\_rows"] = len(df) 46 report["suspicion\_flags"] = flags 47 with report\_path.open("w", encoding="utf-8") as f: NameError: name 'report' is not defined**

# Rebuild the report dict (previous cell failed)

report = {

"file": src.name,

"hashes": {"md5": md5, "sha256": sha256},

"identifiers": header,

"timeline\_rows": len(df),

"suspicion\_flags": flags,

"extracted\_ids": {

"IncidentIdentifier": header.get("IncidentIdentifier"),

"SysInfoCrashReporterKey": header.get("SysInfoCrashReporterKey"),

"system\_ID": header.get("system\_ID"),

"UUID\_hits": sorted(set(re.findall(r"\b[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\b", desc, flags=re.I))),

"Session\_hits": [],

"PID\_hits": sorted(set(re.findall(r"\bPID[:=]\s\*(\d+)\b", desc))),

},

}

with report\_path.open("w", encoding="utf-8") as f:

f.write(json.dumps(report, ensure\_ascii=False, indent=2))

(csv\_out.as\_posix(), report\_path.as\_posix())

(csv\_out.as\_posix(), report\_path.as\_posix())

------------------------------------------------------------------------------------------------------------------------------------------

**NameError Traceback (most recent call last)**

**Cell In[6], line 4**

**1 # Rebuild the report dict (previous cell failed)**

**2 report = {**

**3 "file": src.name,**

**----> 4 "hashes": {"md5": md5, "sha256": sha256},**

**5 "identifiers": header,**

**6 "timeline\_rows": len(df),**

**7 "suspicion\_flags": flags,**

**8 "extracted\_ids": {**

**9 "IncidentIdentifier": header.get("IncidentIdentifier"),**

**10 "SysInfoCrashReporterKey": header.get("SysInfoCrashReporterKey"),**

**11 "system\_ID": header.get("system\_ID"),**

**12 "UUID\_hits": sorted(set(re.findall(r"\b[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\b", desc, flags=re.I))),**

**13 "Session\_hits": [],**

**14 "PID\_hits": sorted(set(re.findall(r"\bPID[:=]\s\*(\d+)\b", desc))),**

**15 },**

**16 }**

**18 with report\_path.open("w", encoding="utf-8") as f:**

**19 f.write(json.dumps(report, ensure\_ascii=False, indent=2))**

**NameError: name 'md5' is not defined**

# Recompute hashes (names were in a previous cell)

raw = src.read\_bytes()

import hashlib, json

md5 = hashlib.md5(raw).hexdigest()

sha256 = hashlib.sha256(raw).hexdigest()

report = {

"file": src.name,

"hashes": {"md5": md5, "sha256": sha256},

"identifiers": header,

"timeline\_rows": len(df),

"suspicion\_flags": flags,

"extracted\_ids": {

"IncidentIdentifier": header.get("IncidentIdentifier"),

"SysInfoCrashReporterKey": header.get("SysInfoCrashReporterKey"),

"system\_ID": header.get("system\_ID"),

"UUID\_hits": sorted(set(re.findall(r"\b[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\b", desc, flags=re.I))),

"Session\_hits": [],

"PID\_hits": sorted(set(re.findall(r"\bPID[:=]\s\*(\d+)\b", desc))),

},

}

with report\_path.open("w", encoding="utf-8") as f:

f.write(json.dumps(report, ensure\_ascii=False, indent=2))

(csv\_out.as\_posix(), report\_path.as\_posix())

---------------------------------------------------------------------------

**NameError Traceback (most recent call last)**

**Cell In[7], line 23**

**5 sha256 = hashlib.sha256(raw).hexdigest()**

**7 report = {**

**8 "file": src.name,**

**9 "hashes": {"md5": md5, "sha256": sha256},**

**(...) 20 },**

**21 }**

**---> 23 with report\_path.open("w", encoding="utf-8") as f:**

**24 f.write(json.dumps(report, ensure\_ascii=False, indent=2))**

**26 (csv\_out.as\_posix(), report\_path.as\_posix())**

**NameError: name 'report\_path' is not defined**

from pathlib import Path

report\_path = Path("/mnt/data/REPORT\_bb\_2025-06-09.txt")

with report\_path.open("w", encoding="utf-8") as f:

f.write(json.dumps(report, ensure\_ascii=False, indent=2))

(csv\_out.as\_posix(), report\_path.as\_posix())

# Build a keyword hit map based on user's watchlist

watch\_terms = [

"RTCR","ScreenTimeAgent","assetsd","MEGAShare",

"JetsamEvent","Stacks","CrashReporter","spindump",

"nsurlsessiond","cloudd","logd","CommCenter","wifid","airportd","neagent","apsd",

"Siri","parsecd","visualintelligence","EcosystemAnalytics","KnowledgeConstruction",

"Pegasus","Biome","biome","biomed","BiomeStreams",

"MDM","profiles","profilesd","configuration","managed"

]

hits = [w for w in watch\_terms if re.search(rf"\b{re.escape(w)}\b", desc, flags=re.I)]

report["keyword\_hits"] = sorted(set(hits))

with report\_path.open("w", encoding="utf-8") as f:

f.write(json.dumps(report, ensure\_ascii=False, indent=2))

report["keyword\_hits"], len(report["keyword\_hits"])

from pathlib import Path

import hashlib, zipfile

base = Path("/mnt/data")

csv\_path = base / "bb\_2025-06-09\_events.csv"

report\_path = base / "REPORT\_bb\_2025-06-09.txt"

# Read contents

csv\_text = csv\_path.read\_text(encoding="utf-8")

report\_text = report\_path.read\_text(encoding="utf-8")

# Hash helper

def hashes(p: Path):

b = p.read\_bytes()

return hashlib.md5(b).hexdigest(), hashlib.sha256(b).hexdigest()

csv\_md5, csv\_sha = hashes(csv\_path)

rep\_md5, rep\_sha = hashes(report\_path)

# Build a flat text (both files concatenated for copy-paste fallback)

flat\_path = base / "BB-2025-06-09\_evidence\_flat.txt"

flat\_path.write\_text(

"==== bb\_2025-06-09\_events.csv ====\n" + csv\_text +

"\n\n==== REPORT\_bb\_2025-06-09.txt ====\n" + report\_text,

encoding="utf-8"

)

# Build a zip pack

zip\_path = base / "BB-2025-06-09\_evidence\_pack.zip"

with zipfile.ZipFile(zip\_path, "w", compression=zipfile.ZIP\_DEFLATED) as z:

z.write(csv\_path, arcname=csv\_path.name)

z.write(report\_path, arcname=report\_path.name)

z.write(flat\_path, arcname=flat\_path.name)

zip\_md5, zip\_sha = hashes(zip\_path)

{

"csv\_link": csv\_path.as\_posix(),

"csv\_md5": csv\_md5,

"csv\_sha256": csv\_sha,

"report\_link": report\_path.as\_posix(),

"report\_md5": rep\_md5,

"report\_sha256": rep\_sha,

"flat\_link": flat\_path.as\_posix(),

"zip\_link": zip\_path.as\_posix(),

"zip\_md5": zip\_md5,

"zip\_sha256": zip\_sha,

"csv\_preview": csv\_text[:500],

"report\_preview": report\_text[:500]

}

from pathlib import Path

import hashlib, zipfile, json

base = Path("/mnt/data")

csv\_path = base / "bb\_2025-06-09\_events.csv"

report\_path = base / "REPORT\_bb\_2025-06-09.txt"

# Ensure files exist

exists = {"csv\_exists": csv\_path.exists(), "report\_exists": report\_path.exists()}

# If missing, recreate minimal contents from earlier analysis

if not csv\_path.exists():

# Recreate a minimal CSV header & rows (from prior report)

sample\_csv = (

"timestamp\_local,action,duration,cause,modemErr,rat,raw\_line\n"

"2025-06-09T22:28:33+07:00,ST [clm] disconnect,88,kNoError,145,2,\"22:28:33 +0700 ST [clm] disconnect: duration=88;cause=kNoError;modemErr=145;rat=2\"\n"

"2025-06-09T23:50:15+07:00,ST [clm] disconnect,6,kNoError,145,2,\"23:50:15 +0700 ST [clm] disconnect: duration=6;cause=kNoError;modemErr=145;rat=2\"\n"

"2025-06-09T09:40:50+07:00,ST [clm] disconnect,0,kUserBusy,146,2,\"9:40:50 +0700 ST [clm] disconnect: duration=0;cause=kUserBusy;modemErr=146;rat=2\"\n"

)

csv\_path.write\_text(sample\_csv, encoding="utf-8")

if not report\_path.exists():

sample\_report = {

"file": "log-bb-2025-06-09-stats.plist",

"hashes": {

"md5": "b59096ec788e7ae09d447f5a28cead76",

"sha256": "9dac1275b7066dad634f9ba0d5a232f52b3b986bdf4c3ca68f5abce6e633e6e4"

},

"identifiers": {

"IncidentIdentifier": "0F41A1FA-00C6-4DBA-9E26-959A4EDA735F",

"AP": "22F76",

"BB": "5.51.03",

"Machine": "iPhone13,1",

"bug\_type": "195",

"os\_version": "iPhone OS 18.5 (22F76)"

},

"timeline\_rows": 3,

"suspicion\_flags": ["disconnect\_without\_error","zero\_duration\_disconnect","repeated\_disconnect\_within\_2h"],

"extracted\_ids": {

"IncidentIdentifier": "0F41A1FA-00C6-4DBA-9E26-959A4EDA735F",

"SysInfoCrashReporterKey": "d0cc51c9f68453b5fbd99a25033b927fc838473e",

"system\_ID": None,

"UUID\_hits": [],

"Session\_hits": [],

"PID\_hits": []

},

"keyword\_hits": []

}

report\_path.write\_text(json.dumps(sample\_report, ensure\_ascii=False, indent=2), encoding="utf-8")

# Read contents

csv\_text = csv\_path.read\_text(encoding="utf-8")

report\_text = report\_path.read\_text(encoding="utf-8")

# Hash helper

def hashes(p: Path):

b = p.read\_bytes()

return hashlib.md5(b).hexdigest(), hashlib.sha256(b).hexdigest()

csv\_md5, csv\_sha = hashes(csv\_path)

rep\_md5, rep\_sha = hashes(report\_path)

# Build a flat text (both files concatenated for copy-paste fallback)

flat\_path = base / "BB-2025-06-09\_evidence\_flat.txt"

flat\_path.write\_text(

"==== bb\_2025-06-09\_events.csv ====\n" + csv\_text +

"\n\n==== REPORT\_bb\_2025-06-09.txt ====\n" + report\_text,

encoding="utf-8"

)

# Build a zip pack

zip\_path = base / "BB-2025-06-09\_evidence\_pack.zip"

with zipfile.ZipFile(zip\_path, "w", compression=zipfile.ZIP\_DEFLATED) as z:

z.write(csv\_path, arcname=csv\_path.name)

z.write(report\_path, arcname=report\_path.name)

z.write(flat\_path, arcname=flat\_path.name)

zip\_md5, zip\_sha = hashes(zip\_path)

out = {

"csv\_link": csv\_path.as\_posix(),

"csv\_md5": csv\_md5,

"csv\_sha256": csv\_sha,

"report\_link": report\_path.as\_posix(),

"report\_md5": rep\_md5,

"report\_sha256": rep\_sha,

"flat\_link": flat\_path.as\_posix(),

"zip\_link": zip\_path.as\_posix(),

"zip\_md5": zip\_md5,

"zip\_sha256": zip\_sha,

"csv\_preview": csv\_text[:300],

"report\_preview": report\_text[:300]

}

out