Dell iDRAC8 Enterprise License — PowerEdge T130

Reverse-engineering, failed license avenues, the XXE/TOCTOU work, and the final working method for a PowerEdge T130 iDRAC8 (2.86.86.86).

- Overview & Final Result
- Environment & Access
- License Format & Error Codes
- Failed Avenues (Forgery, Donor, XXE)
- Winning Method (Leaked License + Tag)
- Revert, Cleanup & Command Reference

Overview & Final Result

Overview & Final Result

Goal: get iDRAC8 **Enterprise** working on a Dell PowerEdge T130, ideally perpetual, without buying a license.

Result: achieved with a **genuine Dell-signed iDRAC8 Enterprise perpetual license** (bound to service tag `XXXXXXX`) plus a **service-tag change** on the T130. The XML-signature / XXE route was explored in depth but was not needed.

Final state

```
racadm license view


iDRAC.Embedded.1
    Status               = The license binding ID does not match the device unique identifier.
    Device               = iDRAC.Embedded.1
    Device Description   = iDRAC8
    Unique Identifier    = XXXXXXX
        License #1
            Status               = The license binding ID does not match the device unique
identifier.
            License Description  = iDRAC8 Express License
            License Type         = PERPETUAL
            Entitlement ID       = FD00000000000000
            License Bound        = XXXXXXX
        License #2
            Status               = OK
            License Description  = iDRAC8 Enterprise License
            License Type         = PERPETUAL
            Entitlement ID       = DLF0000000000000
            License Bound        = XXXXXXX
            Expiration           = Not Applicable
```

- `racadm remoteimage -s` no longer reports "*A required license is missing or expired. Required License -- Remote File Share.*"
- `racadm getsvctag` returns `XXXXXXX`.

The device-level warning is caused by the **old Express license**, which is bound to the original tag `XXXXXXX` and is now invalid. It is harmless; the Enterprise license is `OK`.

Short version of the path

1. Found a leaked, Dell-signed iDRAC8 Enterprise perpetual license bound to `XXXXXXX` (Entitlement `DLF0000000000000`) in a public blog post.
2. Importing it under the original tag gave `LIC008` (binding ID mismatch) — signature, product class and term were all accepted.
3. Changed the T130 service tag to `XXXXXXX` via the 13G PWRD_EN jumper + F2 *Service Tag Settings*.
4. Imported the license through the iDRAC web UI: `licenseAction=1`, `status=ok`.

Artifacts

File	Purpose
`idrac8/licenses/iDRAC8_Enterprise_REDACTED.xml`	the working license
`idrac8/licenses/iDRAC7_Enterprise_9QYZF42.xml`	iDRAC7 donor (fails `LIC010`)
`idrac8/licenses/iDRAC8_Enterprise_Extended_Eval_License.xml`	240-day eval (fails `LIC009`)
`idrac8/RESULT.md`	short result + revert notes
`idrac8/xxe/`	payload builders, DTD server, SSH/import helpers

Redactions

This book is public, so the following were replaced:

- **Credentials** — the account name and password are replaced with `operator` / `REDACTED`.
- **Infrastructure** — internal addresses use the RFC 5737 documentation range (`192.0.2.0/24`), the MAC uses an RFC 7042 documentation address, and the host/domain is `idrac-01.example.com`.
- **Identities** — service tags -> `XXXXXXX`; entitlement IDs zeroed. The public iDRAC7 donor tag `9QYZF42` is left as-is (it is published upstream).

- `tracelog.txt` was removed (it contained login/session telemetry).

Because the service tag and entitlement ID live inside the signed `LicenseData`, the attached license XMLs no longer pass digest verification. The unredacted originals are kept locally.

Environment & Access

Environment & Access

Hardware / firmware

Item	Value
Server	Dell PowerEdge T130 (13G)
BIOS	2.0.8
iDRAC	iDRAC8, firmware **2.86.86.86**
iDRAC IP	`192.0.2.10` /24, gateway `192.0.2.1` (DHCP)
iDRAC MAC	`00:00:5E:00:53:01` (shared LOM)
Original service tag	`XXXXXXX`
New service tag	`XXXXXXX`
DNS name	`idrac-01.example.com`
Host power state	Off during most of the work

Access used

- **SSH as** `operator` (administrator) → the iDRAC SMASH shell `/admin1->`, then `racadm`. Paramiko helper: `idrac8/xxe/idrac_ssh.py`.
- **Web UI session** (used to import licenses):
  - `POST /data/login` with `user`/`password` → `forwardUrl` containing `ST1`/`ST2`.
  - Every request needs header `ST2: <token>` (and `ST1` in the query/form).
  - License import: `POST /lm/lmupload.lmp?set=licenseForce:0,licenseImport:1,deviceOID:1,licenseFileName:` with multipart field `licInput_licenseOptions_0` and form field `ST1`.
- **Redfish**: `POST /redfish/v1/SessionService/Sessions` returns an `X-Auth-Token`. No license-import action is exposed on 2.86.

- **Session limit**: the iDRAC allows a handful of concurrent web/Redfish sessions. `racadm closessn -a` clears them when logins start failing with "*The maximum number of user sessions has been reached*".

PC-side helpers

File	Purpose
`idrac8/xxe/idrac_ssh.py`	drive the `/admin1->` CLI over SSH (paramiko)
`idrac8/xxe/serve.py`	DTD / generic HTTP server on port 80, logs to `requests.log`
`idrac8/xxe/smtp_probe.py`	minimal SMTP listener used as a TCP-egress probe
`idrac8/xxe/build_duration.py`	build the UTF-16 TOCTOU license
`idrac8/xxe/check_duration.py`	local digest / Duration assertions

License Format & Error Codes

License Format & Error Codes

License XML

Dell licenses are XML DSig documents:

```
<lns:LicenseClass xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
                  xmlns:lns="http://www.dell.com/2011/12G/licensing">
  <lns:LicenseData>
    <lns:Schema lns:Vendor="Dell" lns:ID="iDRAC" .../>
    <lns:TransferableLicense>true</lns:TransferableLicense>
    <lns:EntitlementID>...</lns:EntitlementID>
    <lns:DeviceClass lns:ID="iDRAC"/>
    <lns:ProductDescription>...</lns:ProductDescription>
    <lns:LicenseTerm><lns:Perpetual/></lns:LicenseTerm>
    <lns:Restrictions><lns:ServiceTag>XXXXXXX</lns:ServiceTag></lns:Restrictions>
    <lns:DeviceInfo .../>
    <lns:Feature lns:ID="253" lns:Description="... iDRAC8 Enterprise" lns:Enabled="true"/>
  </lns:LicenseData>
  <dsig:Signature>
    <dsig:SignedInfo>
      <dsig:CanonicalizationMethod Algorithm=".../xml-exc-c14n#"/>
      <dsig:SignatureMethod Algorithm=".../xmldsig#rsa-sha1"/>
      <dsig:Reference URI="">
        <dsig:Transforms>
```

```
          <dsig:Transform Algorithm=".../xmldsig#enveloped-signature"/>
        </dsig:Transforms>
        <dsig:DigestMethod Algorithm=".../xmldsig#sha1"/>
        <dsig:DigestValue>...</dsig:DigestValue>
      </dsig:Reference>
    </dsig:SignedInfo>
    <dsig:SignatureValue>...</dsig:SignatureValue>
    <dsig:KeyInfo><dsig:X509Data>...</dsig:X509Data></dsig:KeyInfo>
  </dsig:Signature>
</lns:LicenseClass>
```

- `Reference URI=""` → the digest covers the **whole document minus the Signature** (enveloped transform). Adding/altering any element changes the digest.
- Signed by a key that chains to Dell's embedded CA. Self-signed keys are rejected (`LIC017`).

Error codes observed

Code	Meaning	Seen when
`LIC008`	binding ID ≠ device unique identifier	genuine license for a different service tag
`LIC009`	upgrade unsuccessful	240-day eval on a device that already has a license
`LIC010`	not for the specified device	iDRAC7 donor on an iDRAC8
`LIC011`	non-evaluation license cannot be replaced by an evaluation license	eval vs Express replace
`LIC012`	license file does not exist	`licenseFileName` pointing at a missing file
`LIC017`	corrupted / bad signature / parse failure	forged signature, unresolved entity

Verifier internals (libImpop 2.86)

- `lmVerifySignature` (`0xe5c0`) rejects any document containing the ASCII bytes `<!DOCTYPE` **at any offset**. The scan loop at `0xe5f4..0xe61e` steps by 1 (`mov r1,r8` at `0xe60c` overwrites the `+10`), so it is not positional.
- `lmImport2Device` (`0x8760`), `lmGetLDSRFromLicenseFile` (`0xe940`).

- The license service parses with libxml2 **2.9.2** (`xmlParseMemory`), and `xmlsec` does the DSig check. `xmlLoadExtDtdDefaultValue` is imported by no firmware module.
- In libxml2 2.9.2, `xmlParsePEReference` has **no options gate** for external parameter entities (the `XML_PARSE_NOENT`-style gate was added in 2.11), so `xmlNewEntityInputStream → xmlLoadExternalEntity` is reached.

`LIC008` is the only real gate for a leaked license

The genuine `XXXXXXX` license verifies under `XXXXXXX`; only the service-tag comparison fails. That is what motivated the tag change.

Failed Avenues (Forgery, Donor, XXE)

Failed Avenues

1. Self-signed / forged license

Replaced the `SignatureValue`/cert with a self-signed key and recomputed the digest. Result: `LIC017`. `xmlsec` only accepts a key chaining to Dell's CA, so the signature cannot be forged without Dell's private key.

2. iDRAC7 donor

The genuine iDRAC7 Enterprise license (`9QYZF42`, Entitlement `DE00000009660549`) is signed and perpetual, but the iDRAC8 rejects it with `LIC010` ("not for the specified device") — the product/device-class check fires before any tag binding.

3. 240-day eval

`iDRAC8_Enterprise_Extended_Eval_License.xml` (Entitlement `DE436f7669643230`, `<lns:Evaluation lns:Duration="P240D"/>`) imports as `LIC009` ("The license upgrade was unsuccessful"), with or without `licenseForce`, even after deleting the expired eval. Dell's KB also states a trial license can only be imported once per product.

4. XXE / parse-vs-verify TOCTOU

Design: the verifier and the struct parser parse the license separately. Serve the verifier `<!ENTITY x "P240D">` (digest still matches) and the parser `<!ENTITY x "P3650D">`, by making the DTD content

change between the two fetches.

UTF-16 DOCTYPE bypass: the `<!DOCTYPE` byte scan is defeated by encoding the whole file as UTF-16LE with a BOM (`FF FE`). libxml2 auto-detects it; the canonical form (and therefore the digest) is unchanged.

Payload shape:

```
<?xml version="1.0" encoding="UTF-16"?>
<!--Copyright ...-->
<!DOCTYPE lns:LicenseClass [
<!ENTITY % d SYSTEM "http://192.0.2.20/license.dtd">
%d;
]>
<lns:LicenseClass ...>
  ... <lns:Evaluation lns:Duration="&x;"/> ...
```

Controls that worked on hardware:

- `donor_utf16.xml` → `LIC010` (UTF-16 accepted).
- `donor_utf16_doctype.xml` → `LIC010` (DOCTYPE hidden from the scan).
- `donor_utf16_pe_internal.xml` → `LIC010` (internal parameter entity + attribute entity expansion works).

What failed:

- External HTTP parameter entity (`forged_duration.xml`) → `LIC017`, **zero** requests in the DTD server log.
- External `file://` entities (several paths, including files placed via the firmware upload to `/mnt/scratchpad/idrac_fw.bin`) → `LIC017`.
- External **general** entity in an attribute → `LIC017`.

Port discovery: the iDRAC HTTP clients **ignore the port in the URL and always connect to port 80**. With `serve.py` on 8000/8001/2525, `racadm update` never connected; with the server on **port 80**, it produced:

```
CONNECT 192.0.2.10:57376
GET 192.0.2.10 '/firmimg.d7' (non-dtd)
```

Even with the DTD on port 80, the license service never fetched it, so its external-entity load is disabled or fails before `connect()`.

Conclusion: the XXE/TOCTOU is technically sound but blocked by the license service's entity loader. The leaked-license route made it unnecessary.

Winning Method (Leaked License + Tag)

Winning Method

Source

Public blog post found via search:

> **“□□□□ iDRAC 7 / 8 □ Enterprise License”** — Ajax's Blog, 2025-07-14
> `https://aajax.top/2025/07/14/GettingiDRACEnterprise/`

It contains:

- the iDRAC7 license for `9QYZF42`;
- a base64-encoded **iDRAC8 Enterprise perpetual** license for `XXXXXXX` (Entitlement `DLF0000000000000`);
- the 13G service-tag procedure.

The license was decoded from the page and saved as `idrac8/licenses/iDRAC8_Enterprise_REDACTED.xml`. The SHA-1 digest over the enveloped-signature C14N form matches, confirming it is a genuine signed file.

Why the tag must change

Importing under `XXXXXXX` returned:

```
LIC008: The license binding ID does not match the device unique identifier.
```

Everything else (signature, iDRAC8 product class, perpetual term, feature 253) was accepted. The tag is read-only in software — every `ServiceTag` / `ChassisServiceTag` attribute is `RO="true"` or the group is unsupported on a T130.

Service-tag procedure (13G / T130)

1. Power off, unplug.
2. Find **PWRD_EN** (6-pin block near NVRAM_CLR). Note the original position.
3. Move **PWRD_EN to pins 4–6**, close up, plug in, power on.
4. **F2 → System Setup → Service Tag Settings** (or System Information / Miscellaneous Settings).
5. Delete `XXXXXXX`, enter `XXXXXXX`, save, boot.
6. Power off, restore the jumper, power on.

Verify:

```
racadm getsysinfo
...
Service Tag             = XXXXXXX
```

The iDRAC picks up the new tag after the host boots. `racadm getsvctag` may lag briefly.

Import

Web UI: **Licenses → Import**, or the API equivalent:

```
POST /lm/lmupload.lmp?set=licenseForce:0,licenseImport:1,deviceOID:1,licenseFileName:
  header ST2: <token>
  form   ST1: <token>
  file   licInput_licenseOptions_0 = iDRAC8_Enterprise_REDACTED.xml
```

Response:

```
<root><licenseAction>1</licenseAction><status>ok</status></root>
```

Verification:

```
License #2
  Status        = OK
  Description   = iDRAC8 Enterprise License
  License Type  = PERPETUAL
  Entitlement   = DLF0000000000000
  License Bound = XXXXXXX
  Expiration    = Not Applicable
```

`racadm remoteimage -s` no longer reports the missing Remote File Share license.

Trade-offs

- The Express license (bound to `XXXXXXX`) becomes invalid and produces the device-level `LIC008` warning. Delete transaction 2 for a clean status.
- The server's identity (warranty/support lookups) is now `XXXXXXX`.
- Reverting the tag to `XXXXXXX` restores Express and invalidates Enterprise; re-import the XML after setting the tag back to `XXXXXXX`.

Revert, Cleanup & Command Reference

Revert, Cleanup & Command Reference

Revert the service tag

Repeat the PWRD_EN procedure and set the tag back to `XXXXXXX`. Express becomes valid again; Enterprise fails `LIC008`. The Enterprise XML can be re-imported after setting the tag to `XXXXXXX` again.

Clean up the stale Express license (optional)

```
racadm license view
racadm license delete -t 2
```

Transaction 2 is the old Express license. Deleting it removes the device-level `LIC008` warning. Keep it if you may revert the tag.

Key commands

```
racadm getsysinfo
racadm getsvctag
```

```
racadm license view
racadm license delete -t <id>
racadm remoteimage -s
racadm closessn -a
```

SSH (iDRAC CLI is `clpd`, prompt `/admin1->`):

```
ssh -c aes128-ctr -o MACs=hmac-sha2-256 -o Compression=no operator@192.0.2.10
```

Web/API login and import are wrapped in `idrac8/xxe/` helpers; the DTD server must listen on **port 80** if the XXE route is retried.

Repository map

```
idrac8/
  RESULT.md                     final result + revert
  VULNS.md                      avenue analysis
  licenses/
    iDRAC8_Enterprise_REDACTED.xml   working license
    iDRAC7_Enterprise_9QYZF42.xml  donor (LIC010)
    iDRAC8_Enterprise_Extended_Eval_License.xml  eval (LIC009)
  xxe/
    README.md
    build_duration.py / check_duration.py
    serve.py / smtp_probe.py / idrac_ssh.py
    forged_duration*.xml, donor_utf16*.xml
    dtd_dur_verifier.dtd, dtd_dur_parser.dtd
```

Timeline

1. Forged license → `LIC017`.
2. iDRAC7 donor → `LIC010`.
3. UTF-16 DOCTYPE bypass → accepted (`LIC010` with the donor).
4. Internal parameter entity → works; external HTTP/file entity → `LIC017`.
5. Found port-80 behaviour of the iDRAC HTTP client.
6. 240-day eval → `LIC009`.
7. Found leaked iDRAC8 license (`XXXXXXX`) → `LIC008`.
8. Service-tag change to `XXXXXXX` → import → **Enterprise perpetual** `OK`.