UTF-8 in Cluster Networks

General guideline and DXSpider-specific adaptation

1. Purpose

This document defines a general UTF-8 usage policy for any type of cluster node (DXSpider or others), and describes:

- Why UTF-8 must be considered the base interoperability format.
- How to verify the actual configuration of a system.
- How to correct incorrect configurations on common systems.
- Which aspects are specific to DXSpider.
- How this policy applies to Docker containers.

The approach is operational and reproducible, not theoretical.

2. General context: heterogeneous cluster nodes

In any distributed cluster network (radio, messaging, telemetry, etc.), the following coexist:

- Modern systems with native UTF-8 support
- Legacy systems configured for ISO-8859-1 / CP1252
- Older software assuming ASCII
- Modern software assuming Unicode

Data is exchanged as raw bytes, not as 'Unicode text'. Without an explicit encoding policy:

- character corruption (mojibake) appears
- errors propagate across the network
- the origin of problems cannot be reliably identified

3. Why UTF-8 is the recommended base format

3.1 UTF-8 as the current universal standard

UTF-8 is the default format in:

- Modern Linux distributions (Debian, Ubuntu, Fedora, etc.)
- Docker containers
- Modern programming languages (Perl, Python, Go, Rust, Java)
- Terminals and diagnostic tools

3.2 ASCII compatibility

All valid ASCII is valid UTF-8.

This guarantees that:

- nodes sending ASCII-only data continue to work unchanged
- UTF-8 does not break backward compatibility

3.3 Limitations of legacy encodings

ISO-8859-1 and CP1252 are not self-describing, reuse byte values with different meanings, and do not allow reliable validation. As a result, errors cannot be reliably detected or isolated.

4. General operational principles (valid for any node)

Principle 1 — Tolerant input

The node accepts any incoming byte stream. It must not crash or discard data due to encoding.

Principle 2 — Explicit validation

Data is explicitly validated as UTF-8. Non-UTF-8 input is flagged as such.

Principle 3 — Normalisation for presentation

Legacy-to-UTF-8 conversion is acceptable only after the problem has been recorded.

Principle 4 — Strict output

The node must always emit UTF-8. Output must never be downgraded to legacy encodings.

5. Operational procedure: checking the current state

5.1 Checking the active locale

Run on the node:

```
locale
locale charmap
echo "LANG=$LANG"
echo "LC_ALL=$LC_ALL"
echo "LC_CTYPE=$LC_CTYPE"
```

Correct state:

- `locale charmap` reports UTF-8
- LANG and/or LC_ALL set to C.UTF-8 or xx_XX.UTF-8

5.2 Quick functional test

```
printf 'UTF8 test: ñ Ç á é í ó ú €\n'
```

Characters must be displayed correctly.

6. Correcting configuration by distribution

6.1 Debian / Ubuntu / Raspbian

List available locales:

```
locale -a
```

Enable UTF-8 if missing:

sudo dpkg-reconfigure locales

Select at least:

- C.UTF-8
- or en_GB.UTF-8, en_US.UTF-8, etc.

Apply recommended configuration:

```
sudo update-locale LANG=C.UTF-8 LC_ALL=C.UTF-8 LC_CTYPE=C.UTF-8
```

Restart the session or service.

6.2 Fedora / RHEL / Rocky / Alma

Fedora uses UTF-8 by default. Verify with:

```
localectl status
```

Force configuration if required:

```
sudo localectl set-locale LANG=C.UTF-8
```

7. DXSpider-specific adaptation

7.1 DXSpider considerations

DXSpider is written in Perl, receives network data as raw bytes, and processes historical logs. Input must therefore be treated as raw data, while output must always be UTF-8.

7.2 DXSpider runtime environment

Apply to the service user (sysop or equivalent):

```
export LANG=C.UTF-8
export LC_ALL=C.UTF-8
export LC_CTYPE=C.UTF-8
```

Preferably set in the startup script or systemd service unit.

7.3 Auxiliary Perl scripts

Recommended practices:

```
use strict;
use warnings;
use utf8;  # for source code literals

binmode(STDIN,  ':raw');
binmode(STDOUT, ':encoding(UTF-8)');
binmode(STDERR, ':encoding(UTF-8)');
```

8. Docker containers

8.1 Principle

Containers do not have terminals; they only emit byte streams.

If the container emits UTF-8 and the host uses UTF-8, output is displayed correctly.

8.2 Minimum recommendation (Dockerfile)

Always include:

```
ENV LANG=C.UTF-8
ENV LC_ALL=C.UTF-8
ENV LC_CTYPE=C.UTF-8
```

8.3 Note on Alpine

Alpine does not generate regional locales. C.UTF-8 is the correct and sufficient choice.

9. Recommended network policy (summary)

- Reception: tolerant
- Validation: strict UTF-8
- Normalisation: permitted
- Emission: UTF-8 mandatory

10. Optional tool: host UTF-8 normalisation script

To facilitate normalisation of nodes that are not correctly configured for UTF-8, an optional Perl script for Linux hosts (not containers) is provided:

- Verifies effective encoding using `locale charmap`.
- If not UTF-8, attempts to enable the equivalent `*.UTF-8` locale for the configured language.
- Ensures availability of `C.UTF-8` (may appear as `C.utf8` on Debian).
- Preserves the existing language; the goal is UTF-8 correctness, not language change.

10.1 Location

```
https://github.com/EA3CV/dxspider_info/blob/main/ensure_utf8_locale.pl
```

10.2 Download

Using curl:

```
curl -fsSL -o ensure_utf8_locale.pl \
https://raw.githubusercontent.com/EA3CV/dxspider_info/main/ensure_utf8_l
ocale.pl
```

Using wget:

```
wget -O ensure_utf8_locale.pl \
https://raw.githubusercontent.com/EA3CV/dxspider_info/main/ensure_utf8_l
ocale.pl
```

10.3 Execution permissions

```
chmod +x ensure_utf8_locale.pl
```

10.4 How to run it and as which user

Verification mode (no changes): may be run as a normal user.

```
./ensure_utf8_locale.pl --check
```

Apply changes (if required): must be run as root or via sudo.

```
sudo ./ensure_utf8_locale.pl --apply
```

Operational notes:
- If the system is already correctly configured, the script reports status and exits.
- If `C.UTF-8` does not appear in `locale -a`, Debian may list it as `C.utf8`.
- The script is intended for hosts only, not containers.

11. Conclusion

Consistent use of UTF-8 does not break compatibility, enables reliable diagnostics, prevents error propagation, and prepares the network for the future.

UTF-8 should be considered the internal and output format of any modern node, with controlled tolerance on input to maintain interoperability with legacy nodes.

DXSpider fits naturally into this model when correctly configured.

Kin EA3CV

20251227