Database - Feature #12195

migrating a single-byte OpenEdge database to a UTF-8 converted database

11/16/2026 03:54 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	
version_resolved:			

Description

History

#1 - 11/16/2026 03:54 PM - Greg Shah

Problem

A project whose OpenEdge database is single-byte (for example ISO8859-1) may choose to create the converted PostgreSQL database with UTF-8 encoding. The basic path exists: the import reads each .d file in the codepage of its cpstream footer (or the configured default) and writes Java strings, so the character data arrives in a UTF-8 database correctly converted.

What is missing is the handling of large objects and mixed content, and a defined, tested procedure:

- **CLOB columns may already declare their own codepage.** A CLOB field declared CLOB-CODEPAGE "UTF-8" holds UTF-8 bytes even in a single-byte database, and must not be converted again. Other CLOB fields inherit the database codepage.
- **Content can be in the wrong encoding for its declaration.** Applications commonly store text received from outside (XML, JSON, email bodies) in a CLOB declared with the database codepage, but containing UTF-8 bytes. Converting that content from the declared codepage corrupts it ("Ã©" for "é").
- **BLOBs must never be transcoded.**
- **Values that depend on the session codepage.** Data that was encrypted or hashed after a conversion that used CPINTERNAL (for example ENCRYPT of a CHARACTER value converted to a MEMPTR) decrypts to different bytes when CPINTERNAL changes. Such values need to be decrypted with the old codepage and encrypted again with the new one during migration; which fields hold them is known only to the application.

Work

1. Document and test the procedure for importing a single-byte database into a UTF-8 PostgreSQL database: database creation and locale, cpstream handling for .d files without a footer, the import settings, and checks afterwards.
2. Large objects: read each CLOB file in the codepage its field declares (CLOB-CODEPAGE, otherwise the database codepage), and copy BLOB files byte for byte. Verify this in the import and fix where it does not hold.
3. Mixed content detection: add an import option that checks CLOB (and optionally CHARACTER) values declared in a single-byte codepage for byte sequences that are valid, non-ASCII UTF-8. Values that are UTF-8 are stored as decoded UTF-8 instead of being converted again. Write a report listing table, field, record key and the decision taken, so the project can review it. Make the check configurable per table and field (on, off, or report only).
4. Provide a hook in the import, or a documented post-import step, for a project to re-encrypt or re-hash values that depend on the session codepage, given the old and new codepages.
5. Collation: the converted database and temp-tables need a collation that is compatible with the original for sorting and comparison, which is #6441. Sessions that run with CPINTERNAL UTF-8 are #6431. This task depends on both and does not repeat them.
6. Write up the expected behaviour differences when moving from a single-byte codepage to UTF-8 (sort order, comparisons, LENGTH in bytes versus characters, fixed-width export formats), so a project can review its code.

Testcases

- Import of a single-byte test database with accented characters in CHARACTER fields, CLOB fields with no codepage, CLOB fields declared UTF-8, and BLOB fields, into a UTF-8 database; verify every value against the OpenEdge source.
- A CLOB declared in the single-byte codepage that holds UTF-8 text: with the detection option on, off and in report mode.
- .d files with and without the cpstream footer.
- A value encrypted with ENCRYPT after a CPINTERNAL-dependent conversion, re-encrypted through the hook, decrypted correctly in a UTF-8

session.