Conversion Tools - Bug #11836

a CR produced by ~r or ~015 inside a define value is stripped or converted to a newline

09/06/2026 01:30 PM - Greg Shah

Status:	Review	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Paula Păstrăguş	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	Greg Shah
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	4GL Preprocessor
version_resolved:			

Description

Related issues:

Related to Conversion Tools - Bug #9894: new line with no carriage present in...	**New**

History

#1 - 09/06/2026 01:31 PM - Greg Shah

Measured on OE 11.6, only a produced 0x0A ends a definition. A produced 0x0D does not — it is carried in the value:

```
&scoped-define x "A~r B"     compiles; 4 character value holding a real CR
&scoped-define x "A~015 B"   compiles; identical
&scoped-define x "A~n B"     compile ERROR 247/198 - the ~n ends the definition
&scoped-define x "A~012 B"   compile ERROR - identical
```

With the entry 039 fix in 11747a, FWD no longer terminates the definition on a ~r. But it cannot represent the CR in the listing:

```
OE                    c = "A<CR> B".
FWD no-keeptildes     c = "A B".        CR stripped
FWD -keeptildes       c = "A\n B".      CR turned into a newline
```

Converted output is still correct — .preproc carries the literal ~r and the lexer resolves it at the point of use — but the listing cannot match OE, so the construct cannot be asserted.

1. Why this is not simply #9894

#9894 is about the terminator written at end of line. This is about a CR **inside a string literal** being destroyed. They share the blanket \r to \n normalization in ClearStream.read() and should be fixed together, but the acceptance criteria differ.

2. What was tried

Emitting the translated 0x0D directly, and extending the CR escape-marker mechanism (which ClearStream inserts when isInString() || isInComment()) to cover inDefine. Neither worked: Preprocessor.stripNL() discards any CR or NL not carrying the marker, and an unquoted define value is not a STRING token so there is nothing to consume the marker. This is the "late signalling" hole described in the ClearStream header comment.

This is not a general inability to emit a bare CR. alternative_coding_quirk_06 emits one correctly — its baseline holds a bare CR and FWD's output holds one in the same position. Nine baselines contain a bare CR and it is tempting to read a shared cause across them; the three pre-existing failures among them fail for unrelated reasons (tab expansion, NUL handling, comment line breaks). Scope this issue to the define case only.

3. Testcases

Six, committed and currently failing: define_tilde_r_basic, define_tilde_r_global, define_tilde_r_continuation, define_tilde_r_midline, define_tilde_r_unquoted, define_octal_015_cr.

#2 - 09/06/2026 01:31 PM - Greg Shah

- Related to Bug #9894: new line with no carriage present in FWD conversion on windows added

#4 - 09/09/2026 05:24 AM - Paula Păstrăguș

- Status changed from New to WIP

I'm looking into this one.

#5 - 09/10/2026 05:36 AM - Paula Păstrăguș

- Status changed from WIP to Review

- % Done changed from 0 to 100

- Assignee set to Paula Păstrăguș

- reviewer Greg Shah added

The fix was committed as rev **16749** / 11836a.

This commit resolves the issue where Carriage Returns (0x0D) produced by escapes (~r, ~015) inside preprocessor defines were being destroyed. A provenance-tracking approach was implemented (similar to the recent Tab fix in #11837) to protect these produced CRs from line-end normalizers.

- ClearStream.java (Protecting the CR): Removed the blanket 0x0D to 0x0A rewrite. Introduced a producedCRs counter that increments when an escape is resolved. The newline normalizer at the top of read() now checks this counter and allows a produced \r to pass through untouched as a raw character.
- braces.g: Because a macro expansion loses its character history when pushed back into the stream (un-read), a call to markProducedCRs(str) was added. This explicitly re-marks all CRs inside the expansion text so they survive their second pass through ClearStream.
- Preprocessor.java (Relaxing stripNL): Narrowed the marker detection pattern. stripNL() no longer treats CR CR as an escape marker (only CR NL). This ensures that adjacent produced CRs (like ~r~r) are treated as valid characters and survive into the .cache.

Greg, please review.

#6 - 09/10/2026 05:42 AM - Paula Păstrăguș

Testing was completed on branch 11747a with the TAB fix in place plus the changes from this branch.

- With the CR fix: **123** PASSED, **18** FAILED
- Without the CR fix: **117** PASSED, **24** FAILED

Now, this test suite is passing: tilde_escape_in_define_test_set.