Conversion Tools - Bug #12029

Incremental conversion aborts with "Prototype AST doesn't match the implementation of TEMP-TABLE" when a SHARED TEMP-TABLE view becomes the authoritative temp-table peer

10/20/2026 05:06 AM - Octavian Adrian Gavril

Status:	Test	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Octavian Adrian Gavril	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	Constantin Asofiei
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	Incremental Conversion
version_resolved:	16815		

Description

History

#1 - 10/20/2026 05:12 AM - Octavian Adrian Gavril

- topics Incremental Conversion added

Here is a testcase verified on **trunk/16803** in a minimal project. Three programs sharing one definition, so all three produce the same AstKey:

abl/ttappsrv.i:

```
DEFINE {1} SHARED TEMP-TABLE tt-appsrv
   FIELD partition AS CHARACTER
   FIELD host      AS CHARACTER
   INDEX pi-part IS PRIMARY UNIQUE partition.
```

abl/ttaview.p (the view):

```
{ttappsrv.i}

FOR EACH tt-appsrv NO-LOCK:
END.
```

abl/ttdef1.p (owns the definition):

```
{ttappsrv.i "NEW GLOBAL"}

CREATE tt-appsrv.
ASSIGN tt-appsrv.partition = "one".
```

abl/ttdef2.p (second definer, identical definition):

```
{ttappsrv.i "NEW GLOBAL"}
```

```
CREATE tt-appsrv.
ASSIGN tt-appsrv.partition = "two".
```

The file name and single-threaded mode matter. The view must be converted before the definers, which is why it is named ttaview.p and not ttview.p. The phase processes the file set in order. With ttview.p the definer registers first, the view is pulled in as a dependency on every incremental run, and the defect does not appear.

Steps:

1. Full conversion.
2. Touch abl/ttdef1.p only (any content change) and run an incremental conversion.

Result on trunk: the incremental run reports ./abl/ttaview.p - unchanged and ./abl/ttdef1.p - added for conversion, then aborts with:

```
Prototype AST [TtAppsrv_1_1] should match the interface [null].
```

#2 - 10/20/2026 05:14 AM - Octavian Adrian Gavril

This was noticed while running an incremental conversion on the main embedded file of a large customer GUI application.

#4 - 10/20/2026 06:19 AM - Octavian Adrian Gavril

- Assignee set to Octavian Adrian Gavril

- Status changed from New to WIP

The root cause is that rules/schema/p2o.xml keeps two db-backed maps keyed by the temp-table's AstKey: tmpTabNames (implementation class name) and tmpTabNodes (peer AST). The check at p2o.xml:940 compares the name from the first against the tt_implementation annotation on the peer from the second.

Only a definition that owns the implementation gets that annotation. p2o.xml:1015-1021 guards it with !tt_view. A view therefore stores a peer with tt_implementation = null, yet the first program to register the temp-table was marked authoritative unconditionally:

```
<rule>storeTTPeer
   <action>peer.putAnnotation('authoritative', true)</action>
   <action>tmpTabNodes.put(tmpTabKey, peer)</action>
```

H2Map's restore applies the authoritative rows last, overriding everything. So when a view is converted before the NEW SHARED definition that owns the temp-table, the view's incomplete peer wins every later restore; the definer, arriving second, only force-logs its peer as non-authoritative and can never take the slot back.

It surfaces only incrementally: while the definer's own authoritative row exists it has the higher position and wins, but once that program is re-converted ConversionData.clean() purges its rows, the view's null peer becomes the only authoritative one, and the name survives in another program's logged read.

Created task branch **12029a**. I committed a fix that addressed rules/schema/p2o.xml in **rev16810**: the authoritative annotation is now guarded with

!tt_view, and the definer's peer is annotated authoritative in the "already registered" branch before forceMapValue. So the definition owning the implementation holds the slot even when it registers second.

The patch changes what is written, not what is stored. An affected project needs one full conversion to clear the inherited rows.

This is closely related to what is documented in #11720-47.

Paula used a patch with the changes and confirmed that the issue is fixed. The testcase documented in #12029-1 is fixed as well with **12029a**/**16810**.

#5 - 10/20/2026 06:19 AM - Octavian Adrian Gavril

- % Done changed from 0 to 100

- Status changed from WIP to Review

- reviewer Constantin Asofiei added

Constantin, please review.

#6 - 10/20/2026 12:43 PM - Ovidiu Maxiniuc

Octavian,

In 11903a/r16803 I also fixed an 'authoritative' issue: the attribute I was interested (code-page in my case but other as well) was taken from the latest processed file, which was the view (when sorted alphabetically). This was incorrect and it is a regression from the original implementation of attribute selection for the 'model' temp-table.

I will test to see if this is the same issue. At any rate, my fix was not targetting a conversion failure, but an incorrectly generated DMO interface (wrong attributes), so they probably differ.

#8 - 10/21/2026 03:12 AM - Octavian Adrian Gavril

Ovidiu Maxiniuc wrote:

> Octavian,
>
> In 11903a/r16803 I also fixed an 'authoritative' issue: the attribute I was interested (code-page in my case but other as well) was taken from the latest processed file, which was the view (when sorted alphabetically). This was incorrect and it is a regression from the original implementation of attribute selection for the 'model' temp-table.
>
> I will test to see if this is the same issue. At any rate, my fix was not targetting a conversion failure, but an incorrectly generated DMO interface (wrong attributes), so they probably differ.

I checked out branch **11903a**, and it looks to me like they are two different mechanisms in different rule sets. In **12029a**, an authoritative annotation is used to determine which is the authoritative row last in the H2Map, whereas in **11903a**, a decision is made about which file generates the DMO interface. Therefore, I don't think these are related.

#9 - 10/23/2026 07:14 AM - Constantin Asofiei

- Status changed from Review to Internal Test

Octavian, please look at the #11720 tests and test incremental conversion (and expand the tests).

#10 - 10/23/2026 07:16 AM - Octavian Adrian Gavril

Constantin Asofiei wrote:

> Octavian, please look at the #11720 tests and test incremental conversion (and expand the tests).

Sure, I'm starting the testing plan right away.

#11 - 10/23/2026 10:54 AM - Octavian Adrian Gavril

I ran the following incremental conversion tests, on **12029a**/**16810** and on its trunk baseline **16809**, each with 1 and 4 threads. Every test starts with a full conversion and then runs one incremental conversion per change.

The shared temp-table tests are these.

- #12029-1 testcase: touch ttdef1.p, ttdef2.p, both definers, the view, all three, as a chain and separately.
- #12029-1 testcase with the view sorting after the definers: the same touches.
- #12029-1 testcase with NEW SHARED instead of NEW GLOBAL SHARED: the same touches.
- #11720-42 testcase: touch the definer, then the view, also with the view sorting first.
- #11720-46 case with two identical SHARED views: touch each program, also with both views sorting before the definer.
- Change the definition in the include: add a field, remove it, add an index.
- A view becomes a NEW SHARED definer and goes back to SHARED.
- Add a new view and a new definer.
- Delete the definers, then restore them.
- Delete the view, then restore it.

The #11720-21 tests are these.

- Temp-table in two procedures: delete the definition from one, delete it from both, change one definition, change both definitions, and all of these as a chain.
- Temp-table in three procedures: change one procedure, delete one procedure, and a chain which changes one, deletes the other two and restores them.
- The same for a .cls hierarchy (abstract parent with a protected temp-table, a middle class with its own temp-table, a leaf class which implements an interface, an unrelated class with an identical temp-table): touch the parent, change the parent's temp-table, change the middle class's temp-table, delete the middle class's temp-table, delete the leaf class, and a chain of all these.
- #11720-25 testcase: remove tt1 from Test1.cls, then touch Test3.cls.

Every test in the note was OK with **12029a** changes.

I continue running the **ChUI regression application**, **ETF**, an **OO application**, the tests from the **testcases project** which use shared or class temp-tables, and the **large customer GUI application** where the defect was found, with full and incremental conversions, on **12029a**/**16810** and on **16809**.

#12 - 10/26/2026 10:09 AM - Octavian Adrian Gavril

I also tested **12029a/16810** against its parent **16809** on the following projects.

The full conversions are these.

- The **ChUI regression application**.
- **ETF**.
- An **OO application**.
- The **large customer GUI application** where the defect was found.
- Three folders of the **testcases project**: tests/table/shared, tests/persistence/temp_table/undo_state and tests/oo/progress/lang/class/invoke/tt_ds_hdl.

The incremental conversions are these.

- The **large customer GUI application**: touch its main embedded file, then a program with a NEW SHARED temp-table, then a program with its SHARED view.
- The **ChUI regression application** and **ETF**: several single-file touches, one of them in a shared include.
- The **OO application**: touch a program with a NEW SHARED temp-table, then a program with its SHARED view.
- The **testcases project** folders: touch a NEW SHARED definer, a second definer of the same temp-table and a SHARED view, and for the classes add a field to a temp-table.

The undo_state folder reproduces the defect with **16809**, which aborts with Prototype AST [Ttsharedundo_1_1] should match the interface [null]. **12029a** converts it. No regression was found.

Constantin, is there anything left for this task, or can we go ahead and merge **12029a**?

#13 - 10/26/2026 10:50 AM - Constantin Asofiei

- Status changed from Internal Test to Merge Pending

Octavian, it can be merged now.

#14 - 10/26/2026 10:58 AM - Octavian Adrian Gavril

- Status changed from Merge Pending to Test

- version_resolved set to 16815

Branch **12029a** was merged into trunk as rev. **16815** and archived.