Conversion Tools - Bug #12009

OO class level marking calculated from methods

10/15/2026 07:55 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 - 10/15/2026 07:56 PM - Greg Shah

Derive the support level of a built-in OO class from its members

Project: Base Language
Related: #11892 (gap marking review, where the rule was set), #9488 (built-in OO classes)

Problem

Every built-in OO class in com.goldencode.p2j.oo carries a class-level LegacyResourceSupport annotation, and most of its members carry their own. Both are maintained by hand, and they drift apart. The #11892 review found the drift in both directions:

- Four classes were marked Full at the class level while about 46 of their members still read STUB from the original hand-written skeletons, although those members had real implementations.
- 957 of the 4,412 members carry no marking at all. They silently inherit the class level, so a member is reported at whatever level its class happens to be marked.

Constantin set the rule in #11892-17:

- A method that has an implementation, or delegates to one, is Full. A method carrying a TODO is Partial.
- **A class takes the lowest conversion level and the lowest runtime level of its annotated methods, and this should be calculated rather than written by hand.**
- A LegacyResource class with no markings at all, on the class or on its members, defaults to conversion Partial and runtime Stub, so that a developer has to review it.

The member half of that rule was applied by hand to OpenEdge.Net.URI and OpenEdge.Core.String in branch 11747a. This task implements the class half.

Where class levels are read today

- pattern/CommonAstSupport.java: getGapMarking / getGapMarkingEx (near lines 600-760) read the class annotation for a class reference. For a member with no marking they fall back to the class annotation. When neither exists, the fallback is CVT_LVL_FULL | RT_LVL_STUB.
- uast/SupportLevelDocumentationGenerator.java (near lines 298 and 349) reads the class and member annotations to produce the support level documentation.

Work

1. Write one shared helper that computes the effective level of a built-in class:
   - the conversion level is the minimum over all members of the class that carry a marking, and the runtime level is the minimum likewise, with the two sides taken independently;
   - a member that carries no marking does not take part in the minimum, which keeps the calculation from being circular;
   - a class with no marked members and no class annotation gets CVT_LVL_PARTIAL | RT_LVL_STUB, replacing the CVT_LVL_FULL | RT_LVL_STUB fallback.

2. Decide how inherited members count. The natural answer is that a class's level covers the members it declares plus any it inherits from a built-in superclass. Settle this explicitly, and record the decision in the helper's javadoc.
3. Decide what to do with the hand-written class annotation. There are two options:
    - remove it wherever it now just repeats the computed value;
    - keep it as an explicit override, but have a check report every class whose annotation disagrees with the computed value.
      The second option is safer for the first run, because it produces the full list of disagreements for review.
4. Generate the gap detail for a class whose computed level is below Full by naming the members that hold it down, for example: "GetConstructor, GetEvent and SetPropertyValue are stubs". A member's own gap detail stays on the member.
5. Use the helper in both CommonAstSupport and SupportLevelDocumentationGenerator, so that the gap analysis and the published support levels agree.
6. Run the check over all of com.goldencode.p2j.oo and produce the list of classes whose level changes. Review that list with the class owners before the change is committed, because some class rows will reappear in customer gap documents and others will disappear.

Notes

- A Partial level on a TODO is a blunt instrument. Some TODOs, such as the one in Memptr._Debug's exception handler, do not affect behaviour. The review step above is where such cases are caught, and the member annotation, not the class annotation, is where they are corrected.
- #9488 will replace or confirm many of these classes against the ADE. The helper means a corrected member annotation is enough, and nobody has to remember to update the class as well.