Base Language - Feature #12222

Remove name_map.xml by using annotations

11/19/2026 02:41 AM - Dănuț Filimon

Status:	Review	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Dănuț Filimon	**% 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:**	Conversion
version_resolved:			

Description

Related issues:

Related to Base Language - Feature #6407: name_map.xml improvements	**Test**
Related to Runtime Infrastructure - Bug #12168: Reduce FWD server heap: read ...	**Test**

History

#1 - 11/19/2026 02:45 AM - Dănuț Filimon

The goal is to remove the name_map.xml entirely by replacing the class-mapping nodes with class annotations.

Relevant notes: #6407-322 tested a basic implementation to determine the additional cost at the server startup, #6407-327 a few details about the implementation

#2 - 11/19/2026 02:46 AM - Dănuț Filimon

- Related to Feature #6407: name_map.xml improvements added

#3 - 11/19/2026 02:46 AM - Dănuț Filimon

- Related to Bug #12168: Reduce FWD server heap: read DMO @Property without reflection, share proxy Method objects added

#4 - 11/19/2026 02:47 AM - Dănuț Filimon

- Status changed from New to WIP

- Assignee set to Dănuț Filimon

Created task branch 12222a from trunk/16857.

#5 - 11/19/2026 07:54 AM - Dănuț Filimon

- % Done changed from 0 to 80

Committed 12222a/16858. Removed name_map.xml, added class level LegacyProgram annotation.

Changes:

- removed name_map.xml along with methods, rules, commets/documentation
- removed path-mapping
- Added LegacyProgram annotation which contains pname, ooname, publishedEvents, withServices

- Added AnnotationScanner as a generic scanner instead of a "LegacyProgram" scanner only. Compared to #6407-322 where all classes were checked (including .ui and .dmo), I added a way to exclude specific packages from the search which brought the time under 1 second for SourceNameMapper.
- removed internal p2j name_map.xml

Currently doing /code-review

#6 - 11/19/2026 08:26 AM - Dănuț Filimon

- Status changed from WIP to Review

- reviewer Constantin Asofiei added

Committed **12222a/16859**. Removed the exclude packages, we can have classes with LegacyProgram annotations in ui/dmo. Improved error handling.

Constantin, please review.

#7 - 11/19/2026 08:27 AM - Dănuț Filimon

- % Done changed from 80 to 100

#8 - 11/20/2026 03:16 AM - Constantin Asofiei

Danut, please rebase, there will be conflicts with #12168

#9 - 11/20/2026 04:35 AM - Dănuț Filimon

Constantin Asofiei wrote:

> Danut, please rebase, there will be conflicts with #12168

12168a was merged to **trunk/16850** and my branch is based on **trunk/16857**. I rebased 12222a to latest trunk/16864, the branch is now at revision 16866.

#10 - 11/20/2026 06:09 AM - Constantin Asofiei

ChUI must be updated manually - even the hand-written REST services I think. Please look at the review bellow and do some preemptive testing.

Major

1. A deleted or moved program can stay runnable

ConversionData.cleanupArtifacts deletes the .java file and the conversion artifacts, not the compiled .class, and its call to NameMappingWorker.deleteClassMappings was removed. If the project build compiles into a class directory it does not clean, the stale class keeps its annotation and SourceNameMapper.scanLegacyPrograms registers it at startup. RUN foo.p then runs the deleted code where it used to fail with 293. A program moved to another directory behaves the same way: the old class still answers the old name.

- Whether stale classes survive depends on each project's build. Nothing in the FWD repository cleans them.
- This was not reproduced by test: the test project deletes its classes before every build.

Suggested fix: remove the compiled class in cleanupArtifacts, or require a clean build after a deletion.

Minor, reproduced by test

3. Annotation string values are not escaped

mapping_definitions.rules emits publishedEvents (and pname) as raw Java string literals. An event name with a double quote or a backslash produces Java that does not compile, which fails the whole application build. Before this change the value went through the XML writer, which escaped it.

Reproducer (pp/evx.p):

```
PROCEDURE fire:
   PUBLISH "say ""hi""".
   PUBLISH "back\slash".
   PUBLISH 'single"dq'.
   RETURN "fired".
END PROCEDURE.
```

Generated Java and compiler errors:

```
@LegacyProgram(pname = "pp/evx.p", publishedEvents = "single"dq,back\slash,say ""hi""")

Evx.java:13: error: ')' expected
Evx.java:13: error: illegal character: '\'
Evx.java:13: error: illegal text block open delimiter sequence, missing line terminator
```

OpenEdge 11.6 accepts all three names. A subscriber receives all three events and PUBLISHED-EVENTS returns:

```
single"dq,back\slash,say "hi"
```

4. A program whose class is named LegacyProgram breaks its package

The annotation is emitted by its simple name and relies on the on-demand import of com.goldencode.p2j.util. A source that converts to a class named LegacyProgram shadows that import for its whole package.

Reproducer: add legacy-program.p (any content) to a directory that holds other programs, then convert and build. Every converted class of that package fails:

```
TnNames.java:19: error: incompatible types: LegacyProgram cannot be converted to Annotation
```

Suggested fix: emit the qualified annotation name, or reserve the class name in the conflicts set of naming.rules.

5. Extension-less RUN picks a different program than before

When two programs share a base name (zz.p and zz.w), an extension-less RUN takes the first registered one (progs.get(0) in SourceNameMapper.convertName). Registration order changed from sorted by legacy name to jar entry order, which follows the Java class name.

Reproducer:

1. Convert ord/zz.w alone. It gets class Zz.
2. Add ord/zz.p and convert it in a later incremental run. It gets class Zz_1.
3. Run RUN VALUE("ord/zz").

Result on this branch: zz.w runs, and the server logs:

```
Found first match ord/zz.w for program ord/zz, and additional ord/zz.p
```

The old registry was written from a map sorted by legacy name, which lists ord/zz.p first. When both files are converted in the same run the .p still

wins (8 of 8 pairs tested).

Minor, from code reading

6. classNameExists looks in the wrong folder for most output-root values

The check for classes left by earlier runs uses Configuration.getParameterAsPath("output-root"), which keeps at most one directory. The Java files are written to the raw output-root (brew.xml).

output-root	Folder checked	Correct
./src	./src/	yes (confirmed by test)
src	./	no
./src/main/java	./src/	no
build/gen/src	./gen/	no
/abs/out/src	./abs/	no

Effect: in an incremental run that leaves out a program named action, Action.X is emitted where BlockManager.Action.X is needed, and the generated Java does not compile. Suggested fix: use Configuration.getParameter("output-root"), as the writers do.

7. Nothing is logged when no application program is registered

The WebSpeed package is always scanned, so the registry is never empty and the server starts silently when the application jar is missing from the class path, when pkgroot does not match, or when the jar was converted before this change. The first sign is error 293, or a misleading "can't load" for the entry point. The old code logged SEVERE "No mapping data exists".

Suggested fix: log SEVERE or WARNING when nothing under pkgroot was registered, and log the number of registered programs.

8. A Java REST service can be registered twice

SourceNameMapper.registerRestService has no duplicate check: a class with withServices and no pname bypasses the j2pMap guard that protects ordinary programs. Directory roots of the scan are not de-duplicated. With a development class path that holds both the classes directory and the jar built from it, or the same directory twice, the class is added to servicePrograms twice. RestHandler.registerJavaService then throws "More than one REST service mapped to", which fails web service startup.

CA - what happens if a jar (or converted class) is added twice in the classpath? It can happen by mistake; only the first occurence must be registered.

9. Java REST services declared in a separate name_map.xml are dropped silently

A Java REST service declared through a rest-service element of a name_map.xml shipped in its own jar worked before this change, because every name_map.xml on the class path was read. After the upgrade it is no longer registered. The new declaration, LegacyProgram(withServices = true) without a pname, is described only in the javadoc of LegacyProgram.withServices. There is no migration note or runtime warning.

10. The startup scan reads every class under pkgroot

AnnotationScanner.scanJar reads and inflates every top-level class under pkgroot before the byte pre-filter can reject it. That includes the DMO and frame interfaces under dmo and ui, almost none of which carry the annotation. The scan runs under the SourceNameMapper class lock. Startup cost now grows with the application's bytecode instead of the size of name_map.xml.

Scanning ui and dmo is needed for correctness, because a legacy directory may have one of those names. An index resource written at conversion time would avoid the full read.

Style

- AnnotationValues.Collector: the blank lines before visit, visitEnum, visitAnnotation and visitArray carry trailing spaces.
- LegacyProgram: the blank lines between the members carry trailing spaces.

#11 - 11/20/2026 06:42 AM - Dănuț Filimon

- % Done changed from 100 to 90

- Status changed from Review to WIP

Currently looking into #12222-10.

#12 - 11/20/2026 06:46 AM - Constantin Asofiei

Dănuț Filimon wrote:

> Currently looking into #12222-10.

Thanks. This part:

> 10. The startup scan reads every class under pkgroot

I think it can be ignored.

#14 - 11/20/2026 06:55 AM - Constantin Asofiei

Before putting back in review, please rebase. There are some changes in the JIT support from #11721 which will rely on name_map.xml - let me fix those.

#15 - 11/20/2026 07:09 AM - Dănuț Filimon

Constantin Asofiei wrote:

> Before putting back in review, please rebase. There are some changes in the JIT support from #11721 which will rely on name_map.xml - let me fix those.

Rebased to trunk/16865

#16 - 11/20/2026 11:32 AM - Constantin Asofiei

Danut, in rev 16868 I've fixed the #11271 issues:

- RepositoryProgramProvider finds the programs of its folder or jar from their LegacyProgram annotations (read from the class files, no class loading) instead of a name_map.xml; the name-map configuration node is removed. The index is rebuilt when the repository changes (a replaced jar, a changed folder, a previously unreadable file which became readable), without blocking the lookups, and it is deterministic (classes

indexed in path order).
- A program whose class is missing from the repository is now unknown (STOP 293, as a missing .r); a known program with an unreadable or corrupt class is still reported as an ERROR.
- AnnotationScanner skips and logs unreadable folders, truncated class files and a class whose name doesn't match its path, instead of failing the whole scan.
- PUBLISHED-EVENTS of a just-in-time program is read from its LegacyProgram annotation.
- Documented the provider rules: a program class has one name and one mode under every RUN name; legacy services of a just-in-time program are not registered.

#17 - 11/23/2026 05:56 AM - Dănuț Filimon

- % Done changed from 90 to 100

- Status changed from WIP to Review

About the #12222-10 review:

1. A deleted or moved program can stay runnable: This is not something we should fix in the code, because every time we run incremental conversion we also have to **rebuild the jar**. Modifying the jar from the code is not something that needs to be addressed, the process is conversion -> jar -> run the application, same applies to incremental conversion.
2. Annotation string values are not escaped: Valid, I added a method in NameMappingWorker that calls StringEscapeUtils.escapeJava().
3. A program whose class is named LegacyProgram breaks its package: This is true and not limited to LegacyProgram (LegacySignature, LegacySignatures, LegacyParameter, LegacySerializable, LegacyWidgetPool).
4. Extension-less RUN picks a different program than before: Fixed, ordered entries.
5. classNameExists looks in the wrong folder for most output-root values: Removed getParameterAsPath() call.
6. Nothing is logged when no application program is registered: Logging added
7. A Java REST service can be registered twice: Only register the first occurence
8. Java REST services declared in a separate name_map.xml are dropped silently: added a warning that mentions that name_map.xml is no longer read.

Committed **12222a/16869**. Constantin, please review.

#18 - 11/23/2026 06:02 AM - Constantin Asofiei

Dănuț Filimon wrote:

> About the #12222-10 review:
>
> 1. A deleted or moved program can stay runnable: This is not something we should fix in the code, because every time we run incremental conversion we also have to **rebuild the jar**. Modifying the jar from the code is not something that needs to be addressed, the process is conversion -> jar -> run the application, same applies to incremental conversion.

Yes, but the point is: the .class file exists in the build/classes folder even if the 4GL program was deleted via incremental conversion. Current trunk has name_map.xml as registry of such programs; now, we need at least to make it mandatory for incremental conversion to say Ensure the .class files for these files have been deleted or otherwise delete the build/ folder before javac.

#19 - 11/23/2026 06:19 AM - Dănuț Filimon

Constantin Asofiei wrote:

> Dănuț Filimon wrote:
>
> > About the #12222-10 review:
> >
> > 1. A deleted or moved program can stay runnable: This is not something we should fix in the code, because every time we run incremental conversion we also have to **rebuild the jar**. Modifying the jar from the code is not something that needs to be addressed, the process is conversion -> jar -> run the application, same applies to incremental conversion.
>
> Yes, but the point is: the .class file exists in the build/classes folder even if the 4GL program was deleted via incremental conversion. Current trunk has name_map.xml as registry of such programs; now, we need at least to make it mandatory for incremental conversion to say Ensure the .class files for these files have been deleted or otherwise delete the build/ folder before javac.

Committed **12222a**/**16870** to log a warning when programs are deleted.

#20 - 11/23/2026 08:11 AM - Constantin Asofiei

Danut, why not use progressToJavaString ? Is the event name properly processed (i.e. tilde for escape chars and octal values)?