Conversion Tools - Feature #11679

JIT conversion (a.k.a. RUN from Source)

07/31/2026 09:29 AM - 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

Related issues:

Related to Conversion Tools - Feature #11528: implement COMPILE stmt	**New**
Related to Conversion Tools - Feature #11733: Use of existing RuntimeJastInte...	**New**

History

#1 - 07/31/2026 09:35 AM - Greg Shah

Once #11528 (COMPILE stmt) is available, we can implement the ability to run a 4GL project from source code.

1. Dynamic Code Generation
   - This is something we've previously excluded from project scopes but we have seen many applications with this pattern.
   - It is easy to imagine 4GL code that writes other 4GL code to a file and then RUNs that code (implicit COMPILE in the RUN stmt) or uses COMPILE explicitly.
2. Running From Source
   - a.k.a. Just In Time (JIT) conversion
   - Old school 4GL environments sometimes ran from a source implementation, even in production. (terrible idea but it did happen)
   - For development purposes, it can be pretty useful and is still in use for that reason.
   - It would make a POC using FWD much easier. You would just have to specify an entry point to start with and we would have to compile and run it on the fly.

The core idea here is that any invocation which does not exist as already-compiled code, should be compiled on the fly and executed. This tends to be a recursive thing as it tries to load/run other code (OO or procedures), it would continue compiling on the fly and executing that code as needed.

Once the COMPILE support is available, the key thing here is to instrument the various places where code invocations occur. This must be done for both explicit cases (e.g. RUN statements, DYNAMIC-FUNCTION calls, OO invocations) and implicit cases (e.g. the startup procedure for a session).

An additional requirement will be to ensure that location of the new code is properly managed and that all artifacts like name mapping are updated both in the runtime environment AND in some way that can be cached so that it persists as if it was part of a conversion run.

#2 - 07/31/2026 09:36 AM - Greg Shah

- Related to Feature #11528: implement COMPILE stmt added

#3 - 07/31/2026 09:55 AM - Constantin Asofiei

There are only 2 APIs in FWD which load legacy converted code:

- for .cls - ControlFlowOps.initializeLegacyClass - loads static fields and other static initialization
- for external programs - ControlFLowOps$ExternalProgramResolver - this is what resolves an external program
- everything else goes through these (activate procedures, etc). DYNAMIC-FUNCTION relies on an existing program, so nothing special about it.

Above is done relying on SourceNameMapper to determine what needs to be ran, so we will need these two to work in tandem (i.e. determine a file exists in 4GL and not converted, run conversion, and after that pass it to FWD). COMPILE statement will just give us the .java/.class artifacts, and load them in SourceNameMapper. How do we get the Java compiled .class file? Rely on JDK tools to compile them? Do we generate the bytecode ourself? Also, we need to consider that there will be more than one .java generated for a program/class - considering DMOs, frames, menus, etc.

Possible issues:

- name_map.xml can be saved in directory.xml IMO and not on disk.
- if a file is for a .p or .cls as one in name_map.xml from the full conversion - how to unload the Java class (especially if you decide to unload a .cls in the middle of the hierarchy).
- for .cls - we need to see what happens if a super-class gets modified and the sub-classes are already in r-coded (already loaded or not); if the failure is at runtime and what is exactly.
- we need to decide if what was run in a full conversion can be unloaded at all.
- in stateless clustering, things get complicated, as we will need to keep all this in sync across nodes (but this is a discussion for later).

#4 - 07/31/2026 11:02 AM - Greg Shah

> How do we get the Java compiled .class file? Rely on JDK tools to compile them?

Definitely, we want to use the JDK. We will generate source code in our conversion and I don't want to replace the JDK compiler.

> name_map.xml can be saved in directory.xml IMO and not on disk

In #6407, I intended to essentially eliminate the name map from our runtime requirements. I think something remains but most is just moving to annotations. We need to finish that work and that will make this artifact mostly irrelevant.

#6 - 08/12/2026 07:58 AM - Ovidiu Maxiniuc

- Related to Feature #11733: Use of existing RuntimeJastInterpreter for dynamic execution of 4GL code added