Conversion Tools - Feature #11650

Serialize AST or any intermediate conversion file into more lightweight form.

07/23/2026 12:38 AM - Teodor Gorghe

Status:	Review	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Teodor Gorghe	**% 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:**	
version_resolved:			

Description

Related issues:

Related to TRPL - Feature #1755: implement database-backed filtering/walking ...	**New**

History

#1 - 07/23/2026 01:00 AM - Teodor Gorghe

I have done some profiling and experiments for hotel_gui conversion, and I have a summary about what is the time spent on conversion.

Rank	Method / logic	% main thread	What it is
1	AST XML deserialize (loadTree -> xerces parse + readAst)	**18.6%**	Re-parsing each .ast from XML at the start of every pass
2	Compiled-expression execution (rule condition/action bodies)	11.7%	The actual rule logic - Expression.execute, Rule.apply, CommonAstSupport
3	AST XML serialize (saveTree -> DOM build + serialize)	9.9%	Re-writing each .ast to XML at the end of every pass
4	Named-function dispatch (execLib/evalLib + name->function resolve)	9.8%	Resolving TRPL functions by string name on every call
5	AST deep-copy (duplicate/graft)	9.0%	Per-ruleset whole-tree clone
6	Resolver node registration (registerAst/setAsts)	8.8%	Re-registering every node into the resolver each pass
7	Walker traversal + listener callbacks	6.2%	Tree walk + ascent/descent/next-child notifications
8	Directory listing / filesystem walk	4.3%	File.listFiles - almost entirely driven by rank 11
9	Keyword dictionary lookup	2.2%	KeywordDictionary.processKeyword
10	Parsing / lexing (front-end scan)	1.4%	-
11	Case-insensitive path scan (matchPathCaseInsensitively)	0.8%	Include-path resolution; triggers the rank 8 listings
-	Symbol/var resolution, expr-compile, schema, H2, reflection	<1% each	-
-	Uncategorized (string-intern, ThreadLocal, zip/classload, native I/O)	15.9%	JVM/plumbing spread thin across all of the above

Let's suppose that the conversion baseline time is **3:30 minutes**
I have optimized item 4, 7, 8, 11, and the conversion time has been reduced to **2:47 minutes**.
Serializing/Deserializing **AST** into a binary form instead of XML provided additional improvement just to **2:07 minutes**.

I'll put the changes here as it is right now here, and I will do refine them until we reach to a good and safe state.

Another conversion improvement opportunity: implement **AST** hand-off through conversion stages, **which might optimize 5 and 6**.

Conversion measurements were done with trunk rev 16665.

#3 - 07/23/2026 01:20 AM - Teodor Gorghe

Created task branch **11650a** and committed revision **16669**:

- Initial binary AST work.

#4 - 07/23/2026 03:57 AM - Constantin Asofiei

Teodor, for AST serialization, we still plain .ast files for debugging work. Please default to binary and via a p2j.cfg.xml flag, it can be selected either file or binary. Thanks.

#5 - 07/23/2026 06:09 AM - Greg Shah

I think we will need a tool to convert the binary form into the text/XML form.

#6 - 07/24/2026 05:14 PM - Teodor Gorghe

I have done making a tool which translates binary ast to an xml form.

I started doing conversion testing on two projects. On the first project, the conversion time has been reduced from 3 hours to 1.58 hours. For the second project, time spent on conversion has been reduced from 10 hours to 4.19 hours.

I have diffed the generated source code and there is no difference.

#7 - 07/27/2026 12:32 AM - Constantin Asofiei

Teodor, 11650a has more changes than the binary AST feature. Also, please do not use System.getProperty - the flag needs to be configured via p2j.cfg.xml.

#8 - 07/27/2026 01:07 AM - Teodor Gorghe

I know, I have already done this, but these are uncommitted yet. I need to test the tool for converting binary AST to XML AST.

Currently, the binary form is used only for .ast and not for .schema, .jast files. I don't think it will provide substantial performance improvement because they don't get serialized to XML as much as .ast files, but should I keep as it is or I should serialize entirely in binary form?

#9 - 07/27/2026 01:45 AM - Teodor Gorghe

Committed revision **16670** on task branch **11650a**:

- Made CFG_BINARY_AST configurable through p2j.cfg.xml. Added BinaryAstToXml.java.
- BinaryAstToXml.java is a tool which converts binary AST to a human readable AST XML form.

#10 - 07/27/2026 01:57 AM - Constantin Asofiei

Teodor, please put the binary AST changes in a separate branch.

.dict and .p2o are also used at runtime, and I'm not sure if is best to leave a binary version or the XML version behind. .jast may also help, they are read in the brew phase.

Otherwise, what extension does the binary AST form have?

#11 - 07/27/2026 02:02 AM - Teodor Gorghe

Constantin Asofiei wrote:

> Otherwise, what extension does the binary AST form have?

They have the same file extension, .ast, for simplicity, and both forms (XML and binary) has some magic bytes, to check which form it is.

It is checked first for binary form (always starting with FAST). If the magic bytes doesn't match, it tries to parse as XML file.

#12 - 07/27/2026 03:45 AM - Constantin Asofiei

I'm also curious, how did the disk space required for the full conversion change?

#13 - 07/27/2026 03:47 AM - Teodor Gorghe

Required disk space has been decreased significantly, about **75%** for a full conversion on a big project.

#14 - 07/27/2026 08:17 AM - Teodor Gorghe

Moved micro-optimizations like these for evalLib, dir listing, to **11650b**.

11650a now contains only the AST tree serialization in binary form.

#15 - 07/27/2026 01:58 PM - Constantin Asofiei

Teodor, a couple of notes about 11650a:

- the new files use a 80-char line length limit; the standard length is 110. Please adjust comments/etc.
- XmlFilePlugin should be created with a flag telling it to use 'binary serialization' - do not interrogate Configuration in this class
- EXT_AST should not be needed; in binary-ast mode, any AST file (.schema, .dict, etc) should be managed in binary mode (anything managed by XmlFilePlugin is an AST).
- the binary form uses the same extension as the XML text form. So there is no need to preserve info about the file's extension. And the name of the AST implementation class is already written in the header.
- please enhance the tool to receive a source folder and a destination folder, where:
  - it reads all binary ASTs from source
  - it writes the XML form in the destination folder, using the same structure (and extension, etc)
  - be careful not to overwrite the binary form
- the point is - lets get all AST forms used by conversion to binary format. There may be some changes for runtime loading of the .dict/.p2o.

#16 - 07/28/2026 01:38 AM - Constantin Asofiei

I've done a conversion with the #10614 app on cust001 and:

- previously it took 9h21m and ~60GB of disk space
- now it took 5h24m and ~35GB of disk space

Teodor, actually, can you sub-class XmlFilePlugin for the binary-ast mode? Is more clear this way.

#17 - 07/28/2026 01:42 AM - Teodor Gorghe

With some uncommitted changes (with jast/dict/schema in binary form), hotel_gui converts in **2:05** minutes, which doesn't include **11650b** micro-optimizations.

#18 - 07/28/2026 03:53 AM - Teodor Gorghe

- Status changed from New to WIP

- Assignee set to Teodor Gorghe

- % Done changed from 0 to 100

- reviewer Constantin Asofiei added

Done the things requested in #11650-15 and the XmlFilePlugin separation in 11650a/r16672.

#19 - 07/28/2026 03:53 AM - Teodor Gorghe

- Status changed from WIP to Review

#20 - 07/28/2026 05:32 AM - Constantin Asofiei

Teodor, the changes look good. I've tested also Hotel GUI reports and it works.

#21 - 07/28/2026 03:55 PM - Constantin Asofiei

- Status changed from Review to Internal Test

Constantin Asofiei wrote:

> I've done a conversion with the #10614 app on cust001 and:
>
> - previously it took 9h21m and ~60GB of disk space
> - now it took 5h24m and ~35GB of disk space

Ran with 11650a/r16672 and now is ~5h and ~22GB of disk space.

Please move to full conversion of other apps.

#22 - 07/29/2026 12:28 AM - Teodor Gorghe

I ran the conversion for an app, which it failed. Currently investigating the issue.

#23 - 07/29/2026 01:16 AM - Teodor Gorghe

Fixed the issue on 11650a/r16673, the issue was related to string null coercion, wasn't right for jast (java null parameter was converted to nothing instead of null).

#24 - 07/29/2026 10:33 AM - Teodor Gorghe

Conversion + import + smoke testing for a large GUI app has passed.
I am investigating the issue reported by Artur and continue testing ETF + another project.

#25 - 07/30/2026 03:47 AM - Razvan-Nicolae Chichirau

Runtime + conversion testing for ChUI regression tests passed.

#26 - 07/30/2026 03:48 AM - Teodor Gorghe

Thanks Razvan!
Two more projects conversion + import + regression testing has passed.

#27 - 07/30/2026 04:33 AM - Radu Apetrii

These are the results on the two applications I worked with:

- Conversion on the first one completed in 4h24m, compared to the original which is 6h41m, which means an improvement of 34.1%.
  - Memory-wise, the project now takes up 15.7GB instead of 35.6GB.
- Conversion on the second project completed in 6h48m, compared to the original which is 10h6m, which means an improvement of 32.7%.
  - Memory-wise, the project now takes up 24.4GB instead of 56,7GB.

Take the memory comparisons with a grain of salt, as I believe I haven't done them on the exact same revisions. Anyway, I hope I'm not tripping, but overall this looks extremely good.

#28 - 08/05/2026 05:48 AM - Teodor Gorghe

From Dănuț:

Since I am unable to log into redmine and post on the task, the performance results were:

	PG	MariaDB
trunk	22562.71	23476.96
11650a	21357.93	23142.14

#29 - 08/05/2026 06:49 AM - Greg Shah

Teodor Gorghe wrote:

> From Dănuț:

> Since I am unable to log into redmine and post on the task, the performance results were:

	PG	MariaDB
trunk	22562.71	23476.96
11650a	21357.93	23142.14

It is not clear the units of measure or what is being tested here.

#30 - 08/05/2026 06:51 AM - Dănuț Filimon

Greg Shah wrote:

> It is not clear the units of measure or what is being tested here.

Seconds, it compares the latest trunk/16684 with the same trunk + 11650a.

	PG (s)	MariaDB (s)
trunk	22562.71	23476.96
11650a	21357.93	23142.14

#31 - 08/05/2026 06:55 AM - Greg Shah

Is this a database import? It isn't clear what test is being executed. I guess it isn't conversion since it should not differ much between databases.

#32 - 08/05/2026 06:57 AM - Teodor Gorghe

Dănuț also reported the conversion time through email, which has decreased from **18h26m** to **8h31m**.

#33 - 08/05/2026 06:59 AM - Dănuț Filimon

Greg Shah wrote:

> Is this a database import? It isn't clear what test is being executed. I guess it isn't conversion since it should not differ much between databases.

It is the runtime performance testing for 100 runs. Conversion took 8h31m as mentioned above. deploy + jar + prepare took 531 minutes.

#34 - 08/06/2026 02:19 AM - Teodor Gorghe

In summary, conversion and runtime testing for all projects has passed.
Constantin, I think this is ready for merge after the trunk release.

Also, **11650b** changes are still, which are still relevant, providing additional 20% improvement. I think it can deferred after #3211 to avoid conflicts.

#35 - 08/06/2026 09:11 AM - Greg Shah

Should we consider a p2j.cfg.xml flag to force all ASTs into XML mode? Using the utility to convert one at a time is not convenient when we are looking at a lot of ASTs.

We would certainly default everything to binary mode.

#36 - 08/06/2026 09:13 AM - Dănuț Filimon

Teodor Gorghe wrote:

> Committed revision **16670** on task branch **11650a**:
>
> - Made CFG_BINARY_AST configurable through p2j.cfg.xml. Added BinaryAstToXml.java.

I have to say that being able to read the .ast/.jast is a lot more convenient, this convenience helps us to debug and find the problems faster. Even if we have a way to keep the old readable ast/jast files with the p2j.cfg.xml flag, I feel that we are missing an opportunity to have a faster conversion for testing purposes only because we can't read those files.

Maybe a separate tool can be used to translate the binary ast to the old plain file, it might receive a folder/file that will have to go to this process without being required to run a separate conversion with this feature disabled.

#37 - 08/06/2026 09:31 AM - Constantin Asofiei

Danut, Teodor already added a tool to decode a single file or a full folder of AST files. So if you convert 'in binary' you can decode the entire cvt/ folder if you want.

#38 - 08/06/2026 01:44 PM - Teodor Gorghe

This is how to use it:

```
java -cp deploy/lib/p2j.jar com.goldencode.p2j.util.BinaryAstToXml
Usage: BinaryAstToXml <input.ast> [output.xml]
       BinaryAstToXml <source-dir> <destination-dir>
```

Example:

```
java -cp deploy/lib/p2j.jar com.goldencode.p2j.util.BinaryAstToXml cvt cvt_xml
...
Converted 310 binary AST file(s), skipped 122 non-binary file(s), refused 0 destination binary AST(s), 0 failu
re(s).
```

#39 - 08/10/2026 06:17 AM - Alexandru Lungu

> Dănuț also reported the conversion time through email, which has decreased from 18h26m to 8h31m.

Danut, please make sure you did not convert the POC with Octavian -numThreads multi-threaded conversion. I say this because on the machine where the POC is installed I delivered tests for Octavian's branch; hopefully you did not include that in the conversion testing. Just make sure it is not a false positive here.

#40 - 08/10/2026 06:20 AM - Alexandru Lungu

- Related to Feature #1755: implement database-backed filtering/walking optimization added

#41 - 08/10/2026 06:24 AM - Dănuț Filimon

Alexandru Lungu wrote:

> Dănuț also reported the conversion time through email, which has decreased from 18h26m to 8h31m.

> Danut, please make sure you did not convert the POC with Octavian -numThreads multi-threaded conversion. I say this because on the machine where the POC is installed I delivered tests for Octavian's branch; hopefully you did not include that in the conversion testing. Just make sure it is not a false positive here.

This option was not used.

#42 - 08/10/2026 06:28 AM - Alexandru Lungu

I related this to #1755, because in that task there is an idea to store the AST directly in a DB to leverage query planner and executor. This idea will conflict with #1755 as both suggest changing the representation of the AST. Of course, we can keep both on-disk and in-DB, but it may be redundant (?).

I don't want to mitigate the work being done already in #11650, but I wonder if we can have the best of both words here: representation of AST as fast as this task shows and the TRPL resolution as fast as suggested in #1755.

If in #1755 we choose to use an H2 database, then the serialization/deserialization will be done by H2, which I doubt is using a verbose XML anyway. As for debugging purposes, maybe using the H2 console or some H2 queries would be enough anyway.

#43 - 08/10/2026 06:55 AM - Teodor Gorghe

11650a is just a small change, which is centralized to XmlFilePlugin, which implements AST file save as binary AST, instead of in a XML format.

The save-load cycles for each file, for each conversion stage, still takes a huge amount of time, but at least it was reduced to a smaller fraction.

#1755 is the way to speedup even further, as proved in #11650, but the work in there might eliminate the need to serialize ASTs in the filesystem. Also for DB representation, a relational database might be slow because it needs a predefined structure and the insertion/projection speed for a tree-structure data is slower than compared to a non-relational database such as neo4j.

#44 - 08/10/2026 07:09 AM - Greg Shah

My plan with #1755 was that there would be no filesystem storage.

#45 - 08/11/2026 05:55 AM - Constantin Asofiei

- Status changed from Internal Test to Merge Pending

Teodor, please merge 11650a now.

#46 - 08/11/2026 06:04 AM - Teodor Gorghe

- Status changed from Merge Pending to Review

Branch 11650a was merged into trunk as rev. 16698 and archived.

Main point of this task is done, but I have set this to **review** because there are some changes in **11650b**, which were split apart from **11650a**.