Conversion Tools - Feature #6202

allow .df files to exist in a path that is not directly in data/

03/23/2022 08:38 AM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Ovidiu Maxiniuc	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Conversion Tools - Feature #4105: improve flexibility of schema lo...	**Closed**
Related to Conversion Tools - Feature #5586: move all conversion artifacts/ou...	**New**
Related to Database - Feature #4722: data import should be able to run with o...	**Test**
Related to Conversion Tools - Feature #6203: create cvtpath and move conversi...	**Closed**

History

#1 - 03/23/2022 08:38 AM - Greg Shah

- Related to Feature #4105: improve flexibility of schema loading added

#2 - 03/23/2022 08:38 AM - Greg Shah

- Related to Feature #5586: move all conversion artifacts/outputs into the cvtdb or into a dedicated directory sub-tree that is separate from the original 4GL code/schema inputs added

#3 - 03/23/2022 08:58 AM - Greg Shah

- Related to Feature #4722: data import should be able to run with only the converted application jar file (and FWD) added

#4 - 03/23/2022 09:03 AM - Greg Shah

In #4105-52, Roger attempted to organize .df files in subdirectories of data/. This worked for the front end stage of conversion, but then failed downstream when creating the .p2o files. His idea makes a great deal of sense. One objective in this task is to allow that approach to work. Something like this should be possible:

```
data/secondary.df
data/merged/primary.df
data/multi/primary.df
data/multi/primary_part1.df
data/multi/primary_part2.df
```

As noted in #5586, we have two additional requirements:

- It should be possible to specify any path to a .df. There should be no requirement for existing under the data/ subdirectory. This will allow configurations to point into the structure of an existing 4GL project, rather than forcing our project structure on them.
- The intermediate artifacts created during conversion should not be placed alongside the .df or even in a data/namespace/ directory. These artifacts should be created in directory path that is:
  - The path specified in cfg/p2j.cfg.xml for the (new) global parameter cvtpath.
  - By default (no configuration needed), cvtpath will be set to cvt/.
  - As noted in #5586, the relative pathing of the artifacts will match the path and filename of the original source file.
    - data/something.df will result in <cvtpath>/data/something.* (.dict, .schema...).
    - abl/some/path/to/awesome-db.df will result in <cvtpath>/abl/some/path/to/awesome-db.* (.dict, .schema...).

In #4722, it is noted that the database import has dependencies upon the data/ directory:

- Dump files must exist in data/dump/<database_name>/, but we really want to be able to pass a path in to the import driver command line.

- The import process also depends on data/namespace/ to find .p2o files, which should be read from the application jar without any further configuration. To run imort, we already need to have the converted DMOs and the jar file already has the .p2o files, so this is just about being smarter with our processing.

This task is intended to resolve all 5 of these requirements, though the last 2 could be worked in #4722 if needed.

Eric: Do you have any comments or concerns?

#6 - 03/23/2022 09:05 AM - Greg Shah

- Related to Feature #6203: create cvtpath and move conversion database into that directory added

#7 - 03/24/2022 01:52 PM - Eric Faulhaber

Greg Shah wrote:

> Eric: Do you have any comments or concerns?

We have to consider runtime use of these resources (i.e., dynamic database), and FWD Analytics.

We use these resources at runtime for dynamically converted temp-tables and queries. Runtime conversion needs at least the P2O files and IIRC, the DICT and SCHEMA files as well, in some cases. There must be assumptions in the runtime code about their location, relative to the jar root. If we make this location variable, we'll need to update the appropriate logic in the runtime conversion code.

FWD Analytics currently expects to find many conversion artifacts at their current locations. This includes all the database-related ones mentioned above, as well as the 4GL source code cache files and ASTs. It also looks for the original (i.e., un-preprocessed) source code (external procedures and include files) at their original locations in the file system, but as I understand this task, that location will not change. If we move the conversion artifacts to another location in the file system, the changes to the reporting code should be relatively minor. If we move them into a database, the changes will be significantly more intrusive.

Also, while storing the artifacts in a database is cleaner, fetching them from the database instead of from the file system is likely to further slow the performance of the reporting web app, which already can be painfully slow for large projects. The performance of the search feature in particular, which loads and walks every AST in the project, is likely to be impacted pretty severely, in the absence of major refactoring of this feature.

OTOH, moving the resources into a database opens up some interesting possibilities, in terms of versioning the artifacts and retrieving the correct version as needed (rather than the current approach, which overwrites the existing ASTs as the reporting or conversion execute different TRPL rules). Done properly, this would allow us to use the same project for conversion and reporting. However, this is a more involved change than what seems to be the goal of this task.

#8 - 03/24/2022 02:01 PM - Greg Shah

We aren't moving to a database in this task. Even in #5586, the database is only likely to hold non-artifacts. Some day, in a galaxy, far far away, we will consider moving to a database for everything. See #1755, #1759, #1760, #1761, #1762, #1763 and #1764.

#9 - 03/24/2022 08:57 PM - Ovidiu Maxiniuc

In #4105 I added flexibility for loading the .df file from a configured place, but there were some issues with the output of the schema artifacts.

I have further investigated and fixed FWD so now the output of .dict, .schema and .p2o is fully configurable (they are kept in same location, though). The normal full conversion (f2+m0+cb profile) now runs successfully with very customised schema location: there is no data folder and the *main* and __*meta* databases do not even share the same folder in project's root.

I also removed the data/namespace from the callgraph target. The location of the .dict, .schema and .p2o is computed using the xmlFile attribute configured in cfg/p2j.cfg.xml.

I am now trying to fix the data_analyzer (optionally invoked using f2+m1+cb conversion profile). However, this requires that the data files (.d) to be found at a very specific place - you guessed, data/dump/<database_name>/. The solution I am implementing is to add a new attribute dataFolder for the each namespace in p2j.cfg.xml which will specify the dump folder. This will allow me to compute automatically the required paths. But I have the following problem: where will the *.stats.xml be stored? Now they are created in data/namespace/<dbname>/. My initial though was to put them beside their .d files in dataFolder but this is wrong for multiple reasons (dirty, R/O location to name just two).

So my questions are:

- should I add yet another attribute in the namespace node for .stats.xml files?
- alternatively I can just use the path of artifacts (xmlFile) and get them together with .dict, .schema and .p2o.
- is this new dataFolder attribute too static? It helps the data_analyzer to find its input and can/will be also used by import utility, assuming the analysed data is also imported, but this means will NOT be passed as command line argument to import. In this case, importing from other location requires changes in cfg/p2j.cfg.xml. (or remapping the respective folder at OS level).

Please let me know your opinion.

#10 - 03/25/2022 06:00 AM - Greg Shah

> I am now trying to fix the data_analyzer (optionally invoked using f2+m1+cb conversion profile).

I think we have not used that in 17 years. We should just remove it.

Eric?

#11 - 03/25/2022 01:14 PM - Eric Faulhaber

Greg Shah wrote:

> I am now trying to fix the data_analyzer (optionally invoked using f2+m1+cb conversion profile).

I think we have not used that in 17 years. We should just remove it.

Eric?

I stopped using this once I realized that we can use the PostgreSQL effectively unlimited length text type for 4GL character data, without performance penalty, and that we had a similar type in H2. We may need something like this if we need to add support for a new database which does not support such a data type. That being said, I never liked this approach, because it requires scanning actual data to determine schema length settings, and we have no way of knowing whether a certain data set is fully representative of all possible data sets.

Anyway, I think we should keep it around in case we find a use for it later, but add a note that it is currently not in use and is not being maintained. I would not invest any time in making it work now.

#12 - 03/25/2022 11:16 PM - Ovidiu Maxiniuc

- Status changed from New to WIP

- % Done changed from 0 to 90

As noted above I added a dataFolder attribute in namespace. This specifies the location of .d data files. Together with importFile which specifies the .df schema definition and xmlFile which specifies the location of the generated .dict file for current database we can create any tree structure with only constraint that the .schema and .p2o will be generated in the same place as .dict.

For the moment, I kept the default values to be data, date/namespace and data/dump/<dbname>/ so that the projects which are not aware of these changes can run unchanged.

Committed revision 13695.

#13 - 03/29/2022 07:09 AM - Greg Shah

Overall, I'm good with the approach. When #6203 (cvtpath) is available, you should remove xmlFile and use cvtpath instead.

Eric: Please review.

#14 - 03/29/2022 09:49 AM - Ovidiu Maxiniuc

Greg,

I worked with latest POC (#6088, which Roger is maintaining) when adding these changes. I have chosen it because it is rather small so fast to reconvert/import and because it is just a POC in incipient stage. There are not so many developers involved hence affected.

I would like to commit my changes in that project which will demo the new architecture. Maybe with the database-related artefacts moved to cvt/, with a bit of adjustments. The cfg/p2j.cfg.xml, build.xml, and build-db.xml will be affected and probably the install wiki a bit. Is this OK?

#15 - 03/29/2022 10:02 AM - Roger Borrello

I would be fine with those updates to that POC project.

#16 - 04/01/2022 09:33 PM - Ovidiu Maxiniuc

I have the changes ready to commit but there are two issues.

- I only tested on PoC and hotel_gui projects. That is, insufficient testing.
- The configuration and build files must be (carefully) updated. I need to do it myself for each active project or write a wiki on how this must be done.

#17 - 04/02/2022 09:57 AM - Greg Shah

> I have the changes ready to commit but

Are these changes to FWD or are they the project changes referenced in #6202-14? If they are FWD changes, please post a patch here and describe the ideas.

> I need to do it myself for each active project or write a wiki on how this must be done.

Please start with the wiki page and we will discuss who should do further changes.

#18 - 04/03/2022 10:38 AM - Greg Shah

Please shift to use the Configuration.getConversionFolder() instead of the xmlFile approach. By default it should write the output to <cvtpath>data/. I don't see a reason that we need the extra /namespace/ dir there.

#19 - 04/04/2022 11:23 AM - Ovidiu Maxiniuc

Greg Shah wrote:

> Please shift to use the Configuration.getConversionFolder() instead of the xmlFile approach. By default it should write the output to <cvtpath>data/. I don't see a reason that we need the extra /namespace/ dir there.

I have already removed the old xmlFile (this was declared in the DTD - this is the first reason why the old project will fail with unknown tag) and created a parallel implementation of the accessor to cvt conversion parameter. I have some conflicts and I am deciding how to fix them. The problem is that, unlike the rest of the conversion, the result of processing which is stored relative to cvt, will need runtime access. This means that when the data location is requested from configuration, there are two cases: write - which is done at conversion time (absolute path is OK) and read - which is used in both conversion and runtime. In the latter case, the full path is fine for conversion, but at runtime the path must be relative to jar's root. That is because both the import and server will run with the application's jar only, not with full cvt folder.

#20 - 04/04/2022 11:32 AM - Constantin Asofiei

Ovidiu, are you planning to jar the entire cvt/ folder? For runtime, I would use a pre-defined/static 'cvtpath' (i.e. cvt), and not rely on the one from p2j.cfg.xml. And, at build time, when adding to jar various resources, you can always pick-and-place the i.e. some/path/to/cvt/namespace/ folder and copy it to build/classes/cvt/namespace.

#21 - 04/04/2022 11:40 AM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> Ovidiu, are you planning to jar the entire cvt/ folder?

Of course, not. The build.xml will pick the exact files to put in jar. The other temporary artifacts will remain there (the .cache, .ast, jast, etc).

> For runtime, I would use a pre-defined/static 'cvtpath' (i.e. cvt), and not rely on the one from p2j.cfg.xml.

Actually cvtpath is a parameter in p2j.cfg.xml and it must be obeyed, in both conversion and other occasions (like the import).

> And, at build time, when adding to jar various resources, you can always pick-and-place the i.e. some/path/to/cvt/namespace/ folder and copy it to build/classes/cvt/namespace.

This is true, but I have a set of methods in SchemaConfig which automatically compute the paths for the artifacts (instead of using the old xmlFile parameter). It uses Configuration.getConversionFolder() at conversion but, at runtime, the returned path should be relative.

#22 - 04/04/2022 11:42 AM - Constantin Asofiei

Ovidiu Maxiniuc wrote:

> It uses Configuration.getConversionFolder() at conversion but, at runtime, the returned path should be relative.

At runtime, getConversionFolder can just return 'cvt', so we have a standard path in the jar, where the resources are placed. Is just the build.xml needs to place these artifacts in the cvt/ folder and not obey the p2j.cfg.xml setting (it can't read it anyway).

For conversion/import/etc, one can use any absolute path, maybe on another disk, and that is OK.

#23 - 04/04/2022 11:50 AM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> Ovidiu Maxiniuc wrote:
>
> > It uses Configuration.getConversionFolder() at conversion but, at runtime, the returned path should be relative.
>
> At runtime, getConversionFolder can just return 'cvt', so we have a standard path in the jar, where the resources are placed. Is just the build.xml needs to place these artifacts in the cvt/ folder and not obey the p2j.cfg.xml setting (it can't read it anyway).

According to #5586, there will not be a cvt in jar. The directory structure for all artifacts must be preserved in the cvtpath and in final jar.

> For conversion/import/etc, one can use any absolute path, maybe on another disk, and that is OK.

The import will be run with app's jar only, not full conversion environment. The import utility must take this information from jar. The database dump location is a command-line variable dataPath.

#24 - 04/04/2022 11:53 AM - Constantin Asofiei

Ovidiu Maxiniuc wrote:

> Constantin Asofiei wrote:
>
> > Ovidiu Maxiniuc wrote:
> >
> > > It uses Configuration.getConversionFolder() at conversion but, at runtime, the returned path should be relative.
> >
> > At runtime, getConversionFolder can just return 'cvt', so we have a standard path in the jar, where the resources are placed. Is just the build.xml needs to place these artifacts in the cvt/ folder and not obey the p2j.cfg.xml setting (it can't read it anyway).
>
> According to #5586, there will not be a cvt in jar. The directory structure for all artifacts must be preserved in the cvtpath and in final jar.

I'm confused - how will the artifacts be kept in the jar - will they be relative to the jars root, like some.jar:/data/namespace/p2j_test.schema and on disk is cvt/data/namespace/p2j_test.schema?

#25 - 04/04/2022 11:56 AM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> I'm confused - how will the artifacts be kept in the jar - will they be relative to the jars root, like some.jar:/data/namespace/p2j_test.schema and on disk is cvt/data/namespace/p2j_test.schema?

Exactly. If the original database import file was data/namespace/p2j_test.df.

#26 - 04/04/2022 11:59 AM - Constantin Asofiei

Ovidiu Maxiniuc wrote:

> Constantin Asofiei wrote:
>
> > I'm confused - how will the artifacts be kept in the jar - will they be relative to the jars root, like some.jar:/data/namespace/p2j_test.schema and on disk is cvt/data/namespace/p2j_test.schema?
>
> Exactly. If the original database import file was data/namespace/p2j_test.df.

OK, then the runtime doesn't need 'cvtpath' at all, as the root folder is the project root (so getConversionFolder should return '/') . For import, then... you may want to consider this also a 'runtime usage' case.

#27 - 04/04/2022 12:02 PM - Roger Borrello

Just as an FYI, in my customer project, I utilized the "deploy.dist" target to include everything needed to perform the database import:

```
   <target name="deploy.dist" description="
Create an archive with the application jars and other files required for deployment on other systems.">
      <mkdir dir="${dist.home}/"/>
      <property name="deploy.dist_destfile" value="
${dist.home}/${appname}_deploy_${code_rev}_${appname_code}_${appname_rev}_${p2j_branch}-${p2j_rev}_${appname_d
ate}.zip" />
      <zip destfile="${deploy.dist_destfile}" >
         <zipfileset dir="${basedir}" excludes="**/*.sh **/*.zip **/*.so"
              includes="build/lib/**
                        ddl/**
                        data/*.df
                        data/dump/**
                        data/namespace/**
                        cfg/*.xml
                        deploy/client/**
                        deploy/server/**
                        *.xml
                        *.properties" />
         <zipfileset dir="${basedir}/."
              filemode="755"
              includes="deploy/client/*.sh
                        deploy/server/*.sh
                        install_spawner.sh" />
      </zip>
      <chmod file="${deploy.dist_destfile}" perm="755"/>
   </target>
```

#28 - 04/04/2022 12:03 PM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> OK, then the runtime doesn't need 'cvtpath' at all, as the root folder is the project root (so getConversionFolder should return '/') . For import, then... you may want to consider this also a 'runtime usage' case.

Yes. I am working on this. This was a secondary reason I din not commit on Friday night. Because insufficient testing it would have probably block all developers.

#29 - 04/04/2022 12:05 PM - Greg Shah

> According to #5586, there will not be a cvt in jar. The directory structure for all artifacts must be preserved in the cvtpath and in final jar.

I did not intend to specify this for runtime usage. Actually, I don't see a good reason to provide flexibility for the jar. I'm OK having the conversion artifacts inside cvt/ in the jar.

#30 - 04/04/2022 12:14 PM - Ovidiu Maxiniuc

Greg Shah wrote:

> > According to #5586, there will not be a cvt in jar. The directory structure for all artifacts must be preserved in the cvtpath and in final jar.
>
> I did not intend to specify this for runtime usage. Actually, I don't see a good reason to provide flexibility for the jar. I'm OK having the conversion artifacts inside cvt/ in the jar.

I think this intermediary token does not matter as much. The problem is that, at runtime (and including the import in order to allow #4722 to work) the path to the (database) artifacts is computed differently (relative to jar's root) instead of the absolute path usable at conversion time. But for the sake of symmetry/transparency I would go without it.

#31 - 04/06/2022 07:04 PM - Ovidiu Maxiniuc

With r13746 we have the abilities to:

- pick the .df from any subdirectory under the project's root. The path is specified using importFile of namespace. If not specified, the data/<dbName> is used. This path will define the relative path to data.
- the result of schema processing is stored in cvtpath, in a subdirectory which will match the path and the original .df file. For example, if we have importFile="my_dbs/fwd.df", the following files will be created as result of schema conversion: cvt/my_dbs/fwd.dict, cvt/my_dbs/fwd.schema, and cvt/my_dbs/fwd.p2o (assuming default cvtpath). Each namespace/database can be configured independently.
- when building the project, build.xml should make sure the cvt/my_dbs/fwd.schema, and cvt/my_dbs/fwd.p2o will be copied/jarred so that their path is preserved. That is, they can be found as app.jar!/my_dbs/fwd.schema and app.jar!/my_dbs/fwd.p2o respectively. It is recommended that the FWD configuration file to be also present as app.jar!/cfg/p2j.cfg.xml;
- to specify the location of the .p2o file to be loaded (the second-to-last parameter of import statement), the data relative path is required;
- when doing the import, define dataPath as the location where the database dump .d files are read from. If this variable is not set, then the default location is data/dump/<dbName>/. In this case data/dump/fwd/ is used.

Here is an example of import command:

Command	Comments

`java -Xmx1g \\`
`  -Djava.locale.providers=SPI,JRE -Djava.ext.dirs=deploy/lib/spi \\`
`  dbname="fwd" \\`
`  targetDb="h2" \\`
`  url=... uid=... pw=... \\`
`  dataPath="data/dump/fwd" \\`
`  schema/import \\`
`  my_dbs \\`
`  fwd.p2o`
	SPI info needed by h2 dialect
	the database name
	using H2 dialect
	info for database connection and authentication
	the location of .d files
	TRPL script name
	relative database path
	the .p2o file to be processed

Note: the my_dbs/fwd.p2o will be read from the application's jar which must be added to classpath (not present above) but the data/dump/fwd is a relative directory to current folder. An absolute path can also be specified.

#32 - 04/07/2022 06:53 AM - Greg Shah

All of this is very good. A couple of items:

- The dataPath parameter for import could be optional. I like to have the option to override the calculated one, but why not allow a calculated default based on the app.jar!cfg/p2j.cfg.xml?
- It seems like we should always be able to calculate the .p2o file to load from the jar since we already have the dbname and the app.jar!cfg/p2j.cfg.xml. Can't the command line parameter be eliminated?

#33 - 04/07/2022 11:48 AM - Ovidiu Maxiniuc

Greg Shah wrote:

- The dataPath parameter for import could be optional. I like to have the option to override the calculated one, but why not allow a calculated default based on the app.jar!cfg/p2j.cfg.xml?

It is optional. If not present data/dump/<dbName>/ is used. Note that this defines the location where the .d files are, to allow the import for custom locations. This is not the path of the schema artifacts we generate. It does not make sense to bundle the database content inside the jar, right?

- It seems like we should always be able to calculate the .p2o file to load from the jar since we already have the dbname and the app.jar!cfg/p2j.cfg.xml. Can't the command line parameter be eliminated?

Yes, theoretically we can. It is equivalent of <dbname>.p2o. The problem is the PatternEngine syntax, which requires as last parameter the "list of absolute and/or relative file names of persisted AST files to process".

#34 - 04/07/2022 01:42 PM - Greg Shah

> Note that this defines the location where the .d files are, to allow the import for custom locations. This is not the path of the schema artifacts we generate. It does not make sense to bundle the database content inside the jar, right?

Understood and agree.

> The problem is the PatternEngine syntax, which requires as last parameter the "list of absolute and/or relative file names of persisted AST files to process".

Yes, I forgot that we are directly running the PatternEngine. We really should have a custom data import command line program. That can wait for #4723.

#35 - 04/07/2022 07:05 PM - Ovidiu Maxiniuc

This note was added because of the #5034-1965. It was caused because the conversion got confused about the location of the .hints files.

Firstly, I understand the $cvtpath will store the temporary artifacts and can be removed at any time if a project clean-up is desired. This means we keep the .hints files beside their original files. These are not to be deleted, they are required to do the conversion from the scratch.

FWD used to locate the .hints at the same location of the file processed. In case of schema, this location was data/namespace. However, with new changes, the original .df remains in a source folder, while the .schema / .p2o were generated in $cvtpath so FWD was looking for hint thin this last location. As noted above, I think the schema .hints should be stored with the .df file, not with .schema. More than that, to keep consistency with the code hints, we should rename them to fwd.df.hints instead of fwd.schema.hints, to reflect the original file they serve.

The same problem would have arise when the code artifacts will be moved to $cvtpath. That is, FWD would try to load the hints from cvt/some-path/code.p.hints instead of abl/some-path/code.p.hints. There is a bit of difference which can easily confuse the conversion.

Committed revision 13753.

#36 - 04/08/2022 07:45 AM - Greg Shah

> I think the schema .hints should be stored with the .df file, not with .schema.

Correct.

> More than that, to keep consistency with the code hints, we should rename them to fwd.df.hints instead of fwd.schema.hints, to reflect the original file they serve.

Agreed, that makes more sense.

#37 - 06/06/2022 11:14 AM - Greg Shah

What is left to do on this task?

#38 - 06/06/2022 12:24 PM - Ovidiu Maxiniuc

- % Done changed from 90 to 100

I think the task is done, everything was implemented, including notes 35 and 36.

#39 - 06/06/2022 01:32 PM - Greg Shah

- Status changed from WIP to Closed