IFS Projects for RDi

License

This program and the accompanying materials are made available under the terms of the Common Public License v1.0 which accompanies this distribution, and is available at http://www.eclipse.org/legal/cpl-v10.html

Overview

The IFS Projects for RDi plug-in is used for managing RPGLE projects stored in the IFS:

- Subsystem *IFS Projects* in the *Remote Systems* view.
- Local IFS projects: projects in the RDi workspace, e.g. Git working copies, which are copied to the IFS and built there.
- Building projects and files with TOBi (The Object Builder for i).
- Building projects and files with gmake.
- SSH terminal view.
- Checking the syntax of RPGLE and SQLRPGLE files, with the errors shown in the *Error List* view of RDi.
- Running the build commands in the SSH terminal or in a job of the RSE connection, with the errors shown in the *Error List* view.
- Exporting the source files of a project into source members of a library, e.g. for a change management system.
- Migrating the members of source files into a new IFS project, e.g. to move a utility from source members to TOBi.

Requirements

- RDi 9.6, 9.8 or 9.9.
- The SSH server of the IBM i (`STRTCPSVR SERVER(*SSHD)`) for the terminal and for builds in the terminal.
- TOBi or GNU Make, installed from the open source packages of the IBM i.
- The feature of your build tool, if the members, that are migrated, have its headers: *IFS Projects STRPREPRC Headers* for `STRPREPRC`, *IFS Projects BUILD Headers* for the `BUILD` command of Scott Klement, *IFS Projects COMPILE Headers* for the `COMPILE` command of Alan Campin or *IFS Projects CPLCAS Headers* for the `CPLCAS` command of Denis Robitaille.

Menu Options

The menu options are in the context menus of the *Remote Systems* view:

- *Open SSH Terminal* on an IBM i connection, *Open SSH Terminal Here* on an IFS directory, file or filter.
- *Open New SSH Terminal*, *Open New SSH Terminal Here* and *Open New SSH Terminal in Project Directory* open another terminal instead of reusing the terminal of the same directory.
- *New > IFS Project...* on the subsystem *IFS Projects* and on IFS directories. The directory must not contain `$` or `#`, because make interprets them.
- *Build* on a project or a file of a project.
- *Check Syntax* on an RPGLE or SQLRPGLE file of a project.
- *Open SSH Terminal in Project Directory* on a project.
- *Export to Library...* on a project copies the source files into members of a library.
- *Migrate to IFS Project...* on a library or on source files copies the members into a new IFS project.
- *Properties > IFS Project* on a project edits `iproj.json`.
- *Properties > IFS Project > Environment* edits the environment variables of the project.

Local IFS projects are created with the *File* menu. Their menu options are in the context menu of the project or the file, e.g. in the *Project Explorer*:

- *File > New > Project... > IFS Projects > Local IFS Project* creates a local IFS project.
- *File > Import... > IFS Projects > Existing Local IFS Project* imports a directory, e.g. a Git working copy.
- *Configure as IFS Project...* on a project with `iproj.json` makes it a local IFS project.
- *Build on IBM i* on a project or a file, *Check Syntax* on an RPGLE or SQLRPGLE file.
- *Upload Changes to IFS* and *Upload All to IFS* on a project.
- *Export to Library...* on a project copies the files of the workspace into members of a library.
- *Properties > IFS Project* edits the connection, the deploy directory and `iproj.json`.

Preferences

The preference page *IFS Projects* contains the settings of the builds and the defaults of the IFS projects. Its sub-pages *SSH Terminal*, *Migration* and *Export to Library* contain the settings of the SSH connection and of the terminals, the build tools whose headers are read, and the defaults of the export to a library.

Run build and compile commands selects, where *Build* runs: in the SSH terminal of the project (default) or in a job of the RSE connection.

The preset sets the default build and compile command for gmake or TOBi. Each control has a tool tip with a short description.

The default environment variables apply to all projects. A variable of a project overrides the default of the same name.

Restore terminals after a restart on the sub-page *SSH Terminal* keeps the terminals, when RDi is closed. After the restart, they are shown disconnected.

The sub-page *Migration* lists the build tools, whose headers are read, when source members are migrated. All of them are used by default; switch one off, if its headers are obsolete, e.g. because the members are no longer built with it. The list is empty, if no feature of a build tool is installed.

The sub-page *Export to Library* sets the CCSID and the record length of new source files for the first export of a project. The CCSID is also the default for source files with CCSID 65535 of the migration.

A more detailed description can be found on the IFS Projects project page on GitHub.

SSH keys are configured on page *General* > *Network Connections* > *SSH2*. JSch supports RSA and ECDSA keys in PEM format, but no Ed25519 keys.

SSH Terminal

All terminals are shown in the *SSH Terminal* view: the active terminal on the left, the list of all terminals on the right. A double-click on a disconnected terminal in the list connects it again. The context menu of the list connects, disconnects, renames and closes the selected terminals.

A terminal of the same connection and directory is reused. *Open New SSH Terminal* or the *New Terminal* button opens another one. If the view has been closed, open it with *Window* > *Show View* > *Other...* > *IFS Projects* > *SSH Terminal*.

Keys of the terminal:

- Ctrl+C copies the selected text. Without a selection, it stops the running command.
- Ctrl+V pastes the text of the clipboard.
- Shift+Page Up and Shift+Page Down scroll by one page, Ctrl+Shift+Home and Ctrl+Shift+End to the beginning or the end.
- Up, Down and Ctrl+R (command history) need bash as login shell. bash becomes your login shell with the SQL statement `CALL QSYS2.SET_PASE_SHELL_INFO('*CURRENT', '/QOpenSys/pkgs/bin/bash')`.

Syntax Check

Check Syntax compiles an RPGLE or SQLRPGLE stream file of an IFS project with option *EVENTF and shows the errors in the *Error List* view of RDi. A double-click on an error opens the source file or the copy book at the line of the error. For a local IFS project, the changes are copied to the IFS first.

The file is compiled in a job of its own, which is set up like the build jobs of TOBi:

- The current directory is the project directory.
- The commands of `setIBMiEnvCmd` are executed and the libraries of `curlib`, `preUsrlibl` and `postUsrlibl` are added to the library list. Variables, such as `&ENVCMD`, are replaced by the environment variables of the project.
- The directories of `includePath` are passed as include directories.

RPGLE files are compiled with CRTBNDRPG or, for NOMAIN modules, with CRTRPGMOD and option *NOGEN, so no object is created. SQLRPGLE files are compiled with CRTSQLRPGI into a module in QTEMP, because *NOGEN would skip the RPG compiler.

If the file has unsaved changes, you are asked whether to save them before the syntax check.

LPEX Editor

In the LPEX editor, the syntax check is available as LPEX user action. The plug-in does not change the LPEX profile. Add the user action on the LPEX preference pages:

1. *LPEX Editor* > *User Actions*: name `ifsCheckSyntax`, class `de.tools400.ifsprojects.lpex.CheckSyntaxAction`.
2. *LPEX Editor* > *User Key Actions*: assign a key to action `ifsCheckSyntax`.

The action can also be run on the LPEX command line:

```
action ifsCheckSyntax
```

Build in the SSH Terminal

By default, *Build* runs in the SSH terminal of the project, which shows the output. Afterwards the errors of the compiled objects are shown in the *Error List* view, as described in Build in a Job of the RSE Connection.

To learn when the build has ended, the plug-in runs the command in a subshell. The subshell writes marker files into the directory of temporary files of the plug-in (`<home>/.ifsprojects/tmp`), also when the build is interrupted with Ctrl+C or the terminal is closed. Therefore the command line in the terminal is longer than the build command. While the plug-in waits for the end of the build, a job is shown in the *Progress* view. Canceling it only stops waiting; interrupt the build with Ctrl+C in the terminal.

While a build is running, the next build of the project opens a new terminal. A build command with a comment (`#`) or with `&` or `|` at the end runs unchanged, without *Error List*.

Build in a Job of the RSE Connection

If *Run build and compile commands* is set to *In a job of the RSE connection*, *Build* runs in a job of the RSE connection, without SSH login. The output is shown line by line in the *Commands Log* view. Afterwards the errors of the compiled objects are shown in the *Error List* view. The SSH terminal remains available for interactive work.

The objects must be compiled with option *EVENTF. TOBi writes the events files into directory .evfevent of the project. For gmake builds, the events file EVFEVENT of the library specified as objlib (or curlib) in `iproj.json` is read.

Builds of the same project run one after the other. *Cancel* in the *Progress* view stops the build, including running compiles.

Local IFS Projects

A local IFS project is a project in the RDi workspace, e.g. a Git working copy with `iproj.json`. Before each build, the plug-in copies the changed files to a directory in the IFS, the deploy directory, and builds the project there. Files are only copied from the PC to the IFS, never back. The default deploy directory is `<home>/.ifsprojects/builds/<project>`.

Not copied are the files listed in `.deployignore` or, if it does not exist, in `.gitignore` of the project directory, and `.git`, `.project`, `.settings` and the output folders of TOBi.

If a file in the deploy directory has been changed by someone else, e.g. with Code for IBM i, you are asked whether it may be overwritten. Other files of the deploy directory, e.g. the output of TOBi, are never deleted.

The files are copied unchanged. A new stream file gets the CCSID of the encoding of the local file, e.g. 1208 for UTF-8. A double-click on an error in the *Error List* view opens the local file.

Export to Library

Export to Library... in the context menu of an IFS project or a local IFS project copies the source files of the project into source members of a library, e.g. for a change management system that works with members. The files of the project are not changed. Local IFS projects are exported from the workspace; unsaved changes are saved first, if you agree.

The stream file `name.type` becomes the member `NAME` of the source type `TYPE`, e.g. `hello.pgm.rpgle` becomes `HELLO` of type `RPGLE`, the reverse of `makei cvtsrcpf` of TOBi. Copy books have the extension of their source type followed by `inc` and get that type, e.g. `*.rpgleinc`, `*.sqlrpgleinc` and `*.clleinc`; `*.clinc` of Code for IBM i becomes CLLE. Only the file types, that TOBi knows, are exported. The same files as by the upload of local IFS projects are excluded.

The source file is selected on the first page of the export:

- *Name of the directory, otherwise by source type* (default): `qrpglesrc/hello.pgm.rpgle` goes to `QRPGLESRC`.
- *By source type*: the default source file of the IBM command, that compiles the source, e.g. `QRPGLESRC` for RPGLE and `QCLSRC` for CLLE.
- *One source file*: all files into one source file.

New source files get the CCSID of the first page and at least its record length; a source file with longer lines gets a longer record length. The CCSID must be an EBCDIC CCSID with one byte per character. If the library does not exist, you are asked whether the export shall create it.

Include directives are converted, so that the members compile from the source files: `/copy 'inc/consts.rpgleinc'` becomes `/copy inc,consts` and `INCLUDE SRCSTMF('vars.clinc')` becomes `INCLUDE SRCMBR(VARS) SRCFILE(QCLSRC)`. The copy books are searched like the compilers do, e.g. in the directories of `includePath`. CL `INCLUDE` commands on several lines are not converted.

`EXEC SQL INCLUDE 'inc/hostvars.sqlrpgleinc'` becomes `EXEC SQL INCLUDE HOSTVARS`, also if the statement spans several lines or stands in fixed format between `C/EXEC SQL` and `C/END-EXEC`. The statement names only a member; its source file is the parameter `INCFILE` of the compile command, by default the source file of the compiled member. If the copy

book goes into another source file, the preview warns that `INCFILE` has to be specified.

The member text is taken from the comment `%TEXT`, that `makei cvtsrcpf --text` writes into the first lines of a file.

The second page shows the files with their members and their status. Files with errors are not exported, e.g. because a line is longer than the record length of an existing source file. Warnings, e.g. characters that do not exist in the CCSID of the source file or include directives that are not converted, do not prevent the export.

A double-click on a file shows the source, that would be written into the member. The member does not exist yet, so it cannot be opened in an editor: the tab *Result* shows the source in the viewer of the text editors, with line numbers and the search of the editors (button *Find...* or Ctrl+F). Replacing is not possible, the source is only shown. The tab *Comparison* shows the file of the project and the member side by side, so that the converted lines are easy to find; the search works there, too, and searches the side that has the focus.

Finish starts the export in the background. Before an existing member is replaced, you are asked. The commands are shown in the *Commands Log* view. The next export of the project starts with the same settings.

Migrate to IFS Project

Migrate to IFS Project... in the context menu of a library or of source files in the *Remote Systems* view copies the members into a new IFS project, the reverse of the export. The members are not changed.

The first page lists the source files of the library or the selected source files. All are selected; *Select All* and *Deselect All* are to the right of the list. Options:

- *Member text as comment %TEXT* inserts the member text at the beginning of the file, like `makei cvtsrcpf --text`.
- *Keep the RPG directives of the members for the member build* keeps the `/copy` and the `EXEC SQL INCLUDE` of the member in a block `/if defined(MBR_BUILD)`, so that the members still compile with `DEFINE(MBR_BUILD)`.
- *CCSID of source files with CCSID 65535* is the CCSID, with which their members are read.

The member `NAME` of type `TYPE` in the source file `QXXXSRC` becomes `qxxxsrc/name.type`, like `makei cvtsrcpf --tolower`. A copy book, i.e. a member included by other members, gets the extension `typeinc`, e.g. `qcpysrc/consts.rpgleinc`. The files are written in UTF-8.

Include directives are converted to the stream files: `/copy QCPYSRC,CONSTS` becomes `/copy 'qcpysrc/consts.rpgleinc'`, relative

to the project directory, and `INCLUDE SRCMBR(VARS) SRCFILE(QCPYSRC)` becomes `INCLUDE SRCSTMF('../qcpysrc/vars.clleinc')`, relative to the directory of the source. Members of other libraries or of source files, that are not migrated, and CL `INCLUDE` commands on several lines are not converted.

`EXEC SQL INCLUDE HOSTVARS` becomes `exec sql include 'qcpysrc/hostvars.sqlrpgleinc'`, also over several lines and in fixed format. The statement names only a member, whose source file is the parameter `INCFILE` of the compiler. Because `INCFILE` names one source file, all statements of a member refer to the same one: the migration takes it from the header of the member, e.g. of `STRPREPRC`, or, if it has none, from the source file, that contains the members of all its statements. If it cannot be determined, the statements are not converted and the preview shows it.

If the build tool of the new project is TOBi, the migration also writes `.ibmi.json` with the CCSID of the source file and `Rules.mk` into each directory, and `Rules.mk` with the directories into the project directory. The `Rules.mk` files of the directories are drafts with one rule per member: check them and set the parameters of the installer or of the `STRPREPRC` headers. A source with an `EXEC SQL INCLUDE` of a stream file gets `DBGVIEW := *LIST`, because the default `*SOURCE` of TOBi fails for it.

The headers of the build tools are read by features of their own, so that only the build tools, that are used, are installed: *IFS Projects STRPREPRC Headers* reads the `STRPREPRC` headers, *IFS Projects BUILD Headers* those of the `BUILD` command of Scott Klement, *IFS Projects COMPILE Headers* those of the `COMPILE` command of Alan Campin and *IFS Projects CPLCAS Headers* those of the `CPLCAS` command of Denis Robitaille. Without them, the migration takes the source file of the `EXEC SQL INCLUDE` statements from the members alone and the rules take the object type of a member from its source type.

The second page shows the members with their stream files, their status and their problems. Members without source type are not migrated.

A double-click on a member shows the source, that would be written into the stream file, and compares it with the member, as described in Export to Library.

The third page is the page of *New > IFS Project...*. *Finish* creates the project and writes the files in the background. Before an existing file is replaced, you are asked.

Change Log 1.0.0.r (17.10.2026)

Plug-in: 9.6.0.13+

- Initial version

IFS Projects On The Internet

Project page on GitHub	https://github.com/tools-400/ifs-projects/
Homepage	https://tools-400.github.io/ifs-projects/
RDi 9.6+ Update Site	https://tools-400.github.io/ifs-projects/update-site/eclipse/rdi8.0/
RDi 9.6 Beta Update Site	https://tools-400.github.io/ifs-projects/beta-version/update-site/eclipse/rdi8.0/