

SDK Upgrade Playbook

Steps to upgrade your SDK to retain compatibility with Pega Infinity



Date	Author	Overview
10 th March 2025	Ian Ovenden	Initial document generation

Table of Contents

SUMMARY	4
UPGRADE STEPS.....	5
PROCESS FLOW	11
SUPPORT	12

Summary

This document outlines the steps to take to upgrade a connected SDK project and ensure compatibility with Pega Infinity.

The number of changes required will vary from upgrade to upgrade and be dependent on the customers upgrade frequency.

A degree of manual intervention is required based on the extent of customization. Components which have been overwritten need to be checked for fundamental changes which need to be integrated.

Upgrade steps

1. Take a copy of any repo specific configuration files e.g., **sdk-config.json** and **package.json**.
2. Create a new upgrade branch from the desired branch. Could be “main” but could also be the latest production branch. Name it so the purpose is clear:

```
git branch sdk-upgrade-24.2.0
```

3. Checkout the branch.

```
git checkout sdk-upgrade-24.2.0
```

i Use `git checkout -b sdk-upgrade-24.2.0` command to create *and* checkout in one.

4. If the repo to be upgraded was cloned and not forked, add the original repo as a remote:

```
git remote add upstream https://github.com/pegasystems/react-sdk.git
```

5. Fetch the latest changes from the upstream repo:

```
git fetch upstream
```

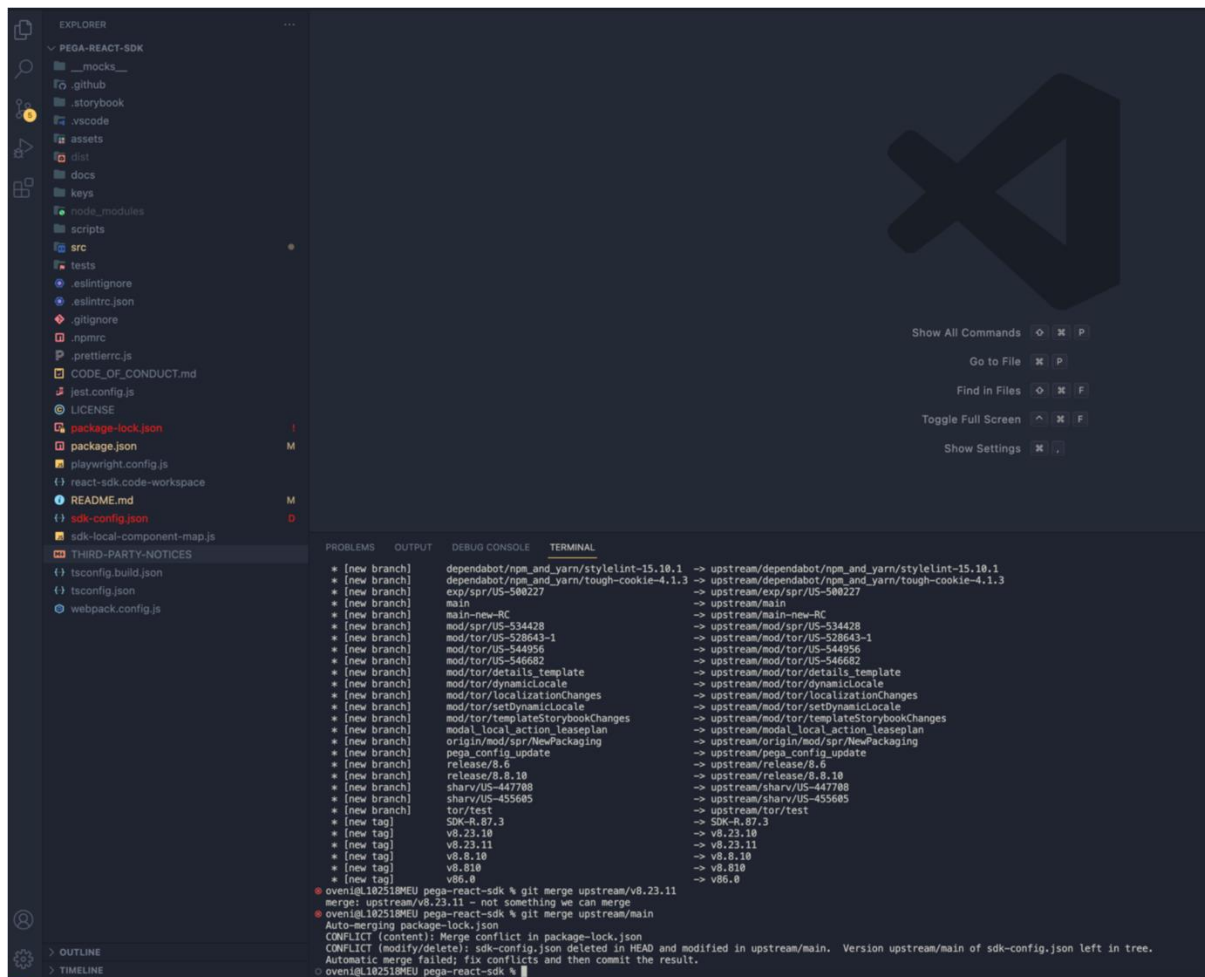
6. Merge changes from upstream branch into local branch:

```
git merge upstream/main --allow-unrelated-histories
```

i "main" is the branch from upstream we want merge. It might be preferable to merge a release branch

i The `--allow-unrelated-histories` flag enables git to merge histories that are unrelated, useful if there is a detached fork from Pegasystems.

7. Resolve any merge conflicts. For example...



In this example, the **sdk-config.json** values backed up in step 1 were merged into the **sdk-config** file.

⚠ It's important that this file isn't added to the git repo - it should remain in local only but need to ensure any changes to properties are accounted for.

With the **package-lock.json** file, the changes were ignored as step 9 will regenerate this based on the changes to the **package.json** file.

There could be a variety of changes required as part of this stage, dependent on the changes made since creation / last upgrade. Some examples include:

- Changes to linting rules
- Added/upgraded/removed dependency packages
- Linting/TypeScript errors. Whilst every effort is made to ensure the PCore / PConnect Typescript definition library is kept up to date, there have been instances where the type definitions do not meet the documentation, causing errors to be thrown.

It's also worth checked the [release notes](#) for the latest update.

8. Manually review and update changes to overridden components. [The Out of the Box SDK components](#) are brought into the core SDK as a dependency library.

The components might be overridden in the project, meaning any of the latest Pega changes may be missed.

The **sdk-local-component-map.js** file in the root directory will provide information on what files have been overwritten as part of the project. These should be checked for added/updated functionality and changes integrated into the component override files.

```
1: sdk-local-component-map.js > [e] localSdkComponentMap
2: // Statically load all "local" components that aren't yet in the npm package
3:
4: import FieldCheckbox from './src/components/override-sdk/field/Checkbox/';
5: import FieldTextInput from './src/components/override-sdk/field/TextInput/';
6: import FieldDate from './src/components/override-sdk/field/Date/';
7: import CheckAnswers from './src/components/custom-sdk/template/CheckAnswers/';
8: import ChangeLink from './src/components/custom-sdk/field/ChangeLink/';
9: import FieldRadioButtons from './src/components/override-sdk/field/RadioButtons/';
10: import TemplateDefaultForm from './src/components/override-sdk/template/DefaultForm/';
11: import InfraAssignmentCard from './src/components/override-sdk/infra/AssignmentCard/';
12: import InfraActionButtons from './src/components/override-sdk/infra/ActionButtons/';
13: import InfraAssignment from './src/components/override-sdk/infra/Assignment/';
14: import InfraFlowContainer from './src/components/override-sdk/infra/FlowContainer/';
15: import FieldPhone from './src/components/override-sdk/field/Phone/';
16: import InfraView from './src/components/override-sdk/infra/View/';
17: import TemplateCaseView from './src/components/override-sdk/template/CaseView/';
18: import FieldDropdown from './src/components/override-sdk/field/Dropdown/';
19: import TemplateFieldGroupTemplate from './src/components/override-sdk/template/FieldGroupTemplate/';
20: import TemplateDetails from './src/components/override-sdk/template/Details/';
21: import InfraViewContainer from './src/components/override-sdk/infra/ViewContainer/';
22: import MimicASentence from './src/components/custom-sdk/template/MimicASentence/';
23: import HmrcOdxGdsInfoPanel from './src/components/custom-sdk/template/HMRC_ODX_GDSInfoPanel/';
24: import HmrcOdxGdsSummaryCard from './src/components/custom-sdk/template/HMRC_ODX_GDSSummaryCard/';
25: import HmrcOdxGdsButton from './src/components/custom-sdk/field/HMRC_ODX_GDSButton/';
26: import HmrcOdxComplexQuestionExplanation from './src/components/custom-sdk/field/HMRC_ODX_ComplexQuestionExplanation/';
27: import HmrcOdxGdsTaskList from './src/components/custom-sdk/widget/HMRC_ODX_GDSTaskList/';
28: import HmrcOdxGdsTaskListTemplate from './src/components/custom-sdk/template/HMRC_ODX_GDSTaskListTemplate/';
29: import AutoComplete from './src/components/override-sdk/field/AutoComplete/';
30: import HmrcOdxGdsTextPresentation from './src/components/custom-sdk/field/HMRC_ODX_GDSTextPresentation/';
31: import HmrcOdxSectionBased from './src/components/custom-sdk/template/HMRC_ODX_sectionBased/';
32: import RichText from './src/components/override-sdk/field/RichText/';
33: import HmrcOdxGdsCheckAnswersPage from './src/components/custom-sdk/template/HMRC_ODX_GDSCheckAnswersPage/';
34: import SimpleTableManual from './src/components/override-sdk/template/SimpleTableManual/';
35: /*import end - DO NOT REMOVE*/
36:
37: // localSdkComponentMap is the JSON object where we'll store the components that are
38: // found locally. If not found here, we'll look in the Pega-provided component map
39:
40: const localSdkComponentMap = {
41:   Checkbox: FieldCheckbox,
42:   TextInput: FieldTextInput,
43:   Date: FieldDate,
44:   HMRC_ODX_CheckAnswers: CheckAnswers,
45:   HMRC_ODX_ChangeLink: ChangeLink,
46:   HMRC_ODX_MimicASentence: MimicASentence,
47:   RadioButtons: FieldRadioButtons,
48:   DefaultForm: TemplateDefaultForm,
49:   Assignment: InfraAssignment,
50:   AssignmentCard: InfraAssignmentCard,
51:   ActionButtons: InfraActionButtons,
52:   Assignment: InfraAssignment,
53:   FlowContainer: InfraFlowContainer,
54:   HMRC_ODX_PhoneNumber: FieldPhone,
55:   View: InfraView,
56:   CaseView: TemplateCaseView,
```

As you can see in the above image, the number of overwritten files can grow over time. They should be checked for fundamental changes in functionality.

⚠ This is a key step and can be missed.

9. Clean and re-install node modules and refresh npm packages by running the following command:

```
npm run build:dev:ci
```

10. Once any conflicts have been resolved, the changes can be committed to the branch:

```
git commit -m "Merge remote-tracking branch 'upstream/main' into sdk-upgrade-24.2.10"
```

11. Test. Test. And test some more.

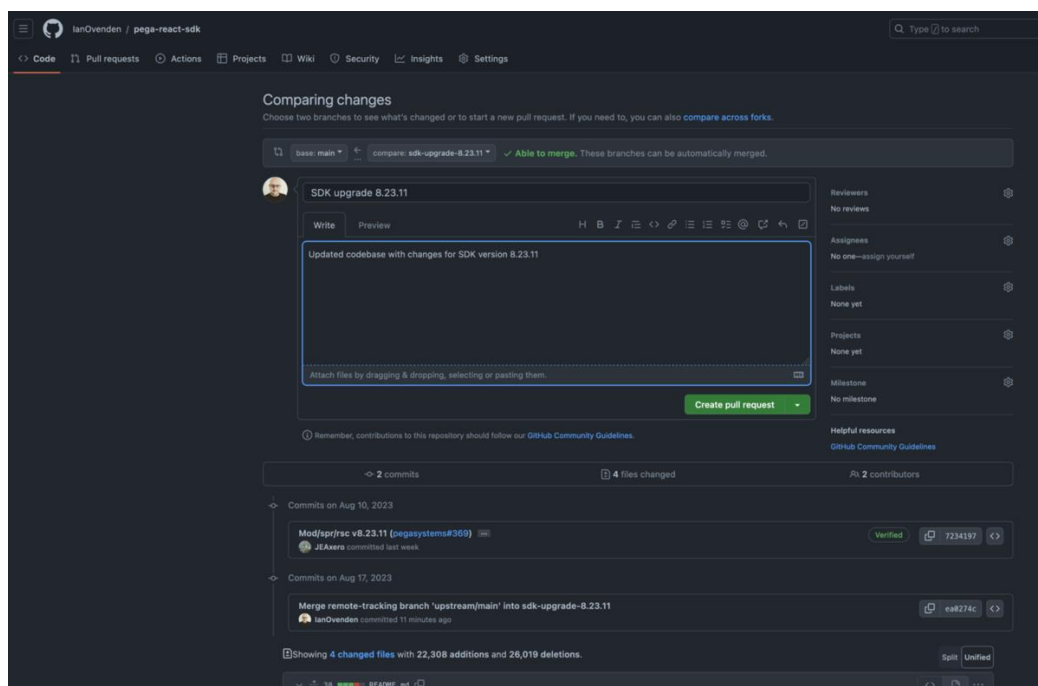
Thoroughly test the code locally.

⚠ Ensure the project is valid by pointing to a version of Pega Infinity that matches that of the SDK – in our example here, 24.2.10

12. Once testing is complete and no further issues found, push the branch and any changes to the repo:

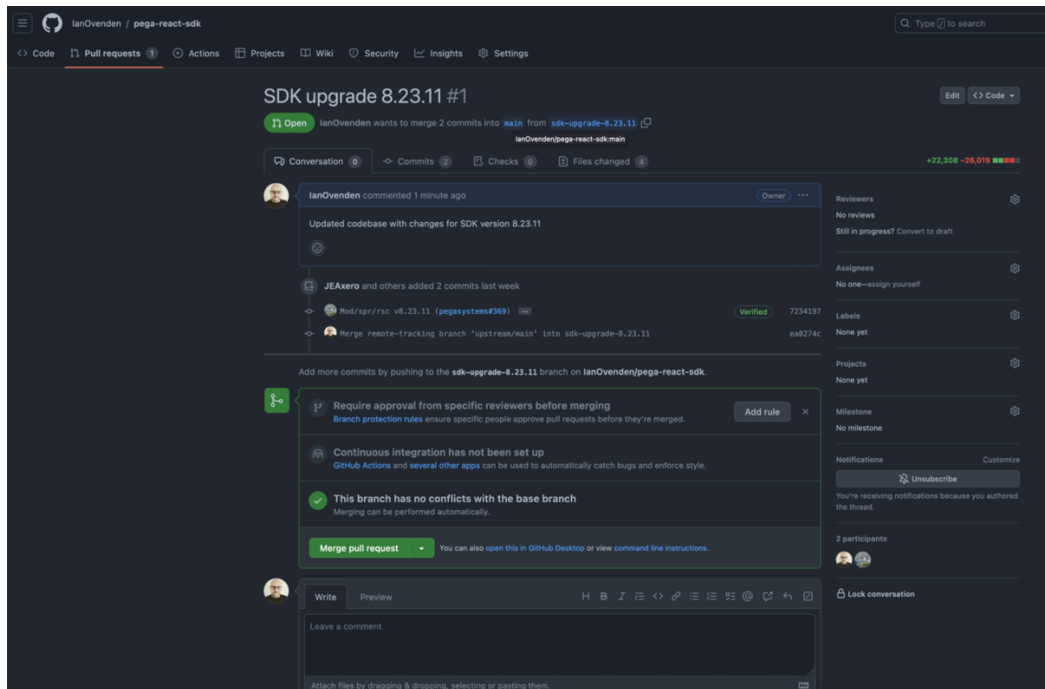
```
git push origin sdk-upgrade-24.2.10
```

13. Create a pull request for merging these changes into main

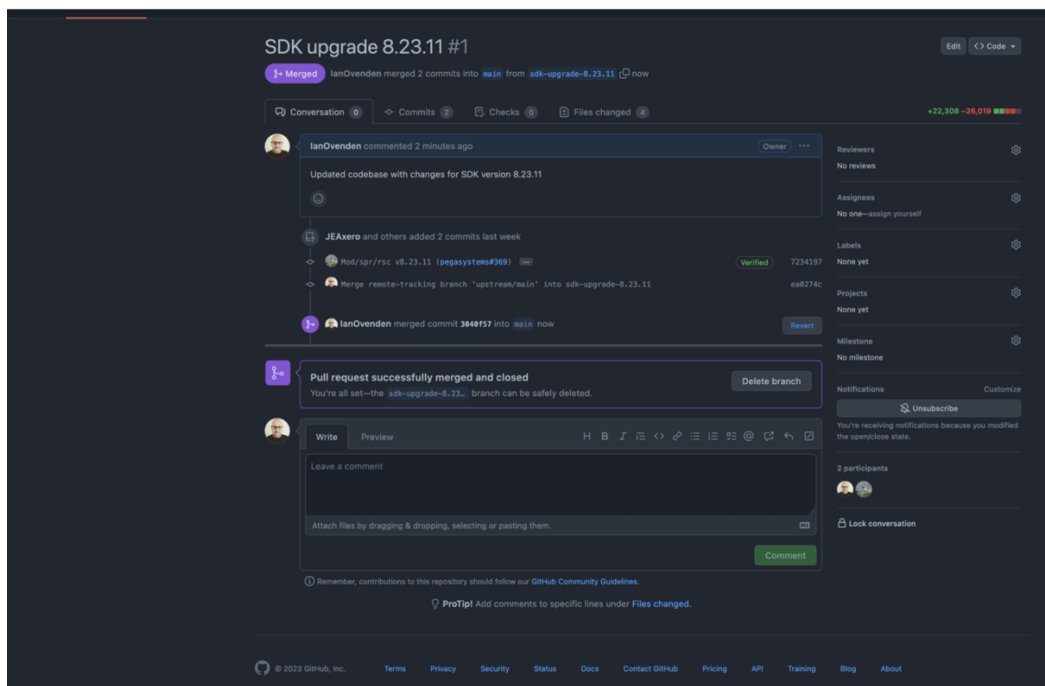


⚠ When raising the pull request, ensure the base is your forked repo and not the Pegasystems upstream

14. After review (if required) merge the changes either yourself or via your React Lead / release manager.



15. Once successfully merged, the upgrade branch can be safely deleted



16. Ensure the list of remote branches is refreshed to reflect the deletion of the branch

```
git fetch --prune
```

```
oveni@L102518MEU pega-react-sdk % git fetch --prune
From https://github.com/IanOvenden/pega-react-sdk
- [deleted]          (none)      -> origin/sdk-upgrade-8.23.11
oveni@L102518MEU pega-react-sdk %
```

17. Delete the branch from your local repo

```
oveni@L102518MEU pega-react-sdk % git branch
* main
  sdk-upgrade-8.23.11
oveni@L102518MEU pega-react-sdk % git branch -d sdk-upgrade-8.23.11
Deleted branch sdk-upgrade-8.23.11 (was ea0274c).
oveni@L102518MEU pega-react-sdk % git branch
* main
oveni@L102518MEU pega-react-sdk %
```

18. Add a tag to ensure we can reference the upgrade snapshot

```
git tag -a sdk-upgrade-8.11.23 -m "Merged latest Pega core SDK changes"
```

- It is possible that the tag will already exist as part of the SDK merge. Verify tags with "git tag"

Process flow

In terms of process, the upgrade should be done just prior to the start or at the start of a new sprint.

Ideally, regression testing should happen straight away.

Sprint work should be done against the new version to prevent potentially complex merge conflicts.

Support

[Pega SDK upgrade documentation](#)

[Overview summary of SDK releases](#)