



# **CORTEX Interaction Portal**

## Merging Guide

WE ARE  
**CORTEX**

# Contents

1 Introduction .....	4
1.1 AppGyver Overview .....	4
1.2 Problem Statement .....	4
2 Checklist .....	6
3 Steps to Merge Pages .....	7
3.1 Import New Version .....	7
3.2 App Level Steps .....	7
3.3 Page Level Steps .....	9

## About this Document

This document provides a guide to merging newly built pages with further updates of the CORTEX Interaction Portal solution.

## Audience

This document is intended for developers who intend to update their version of the CORTEX Interaction Portal solution without deleting pages they have built for use alongside it.

## Related Material

Document	Version
CORTEX Interaction Portal Deployment Guide	v3.0
CORTEX Interaction Portal Developer Guide	V3.0
CORTEX Interaction Portal User Guide	V3.0


## Abbreviations used in this Document

<b>UI</b>	User Interface
<b>UX</b>	User Experience

# 1 Introduction

## 1.1 AppGyver Overview

The CORTEX Interaction Portal is built using SAP AppGyver: a low-code web (and mobile) application building tool which offers pre-built *components* and the ability to create your own component templates. Also included is integrated logic to issue API requests, navigate pages, show data, and many more options.

 *AppGyver is offered in several different pricing tiers. At the time of writing the Community Edition includes all the core functionality required.*

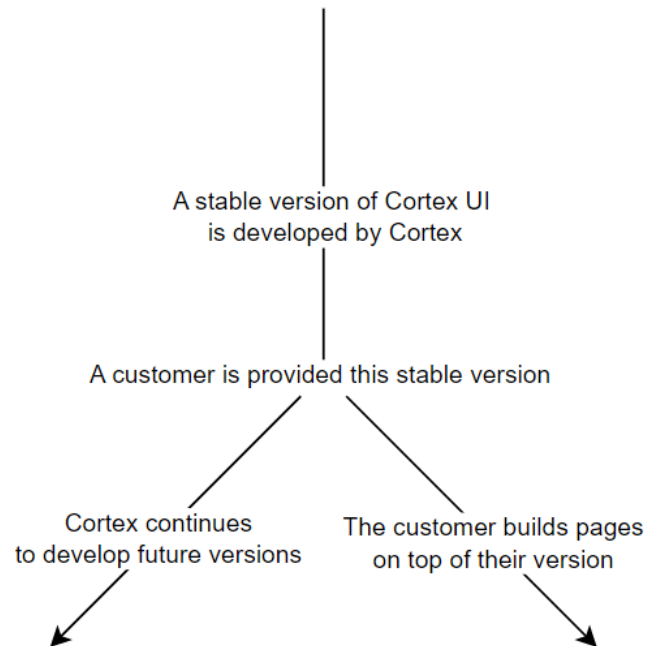
## 1.2 Problem Statement

Appgyver contains the ability to export an app containing many pages as a package file, to be imported into another developer account. However, there is (at the time of writing) no ability to export individual pages and merge them into another app.

Pages to use alongside processes automated with CORTEX are built in the same app as the CORTEX Interaction Portal pages, so this presents a problem.

Consider the following situation that may arise:

1. A customer downloads and imports their own version of the CORTEX Interaction Portal web app into their own AppGyver development account.
2. They then build pages to use as Service Requests or as manual tasks raised by an otherwise fully automated process.
3. Meanwhile, CORTEX adds new features and bug-fixes to the CORTEX Interaction Portal solution.
4. The customer downloads and imports this new version of the CORTEX Interaction Portal into a new web app in their AppGyver development account.
5. The customer then wishes to use their pages alongside this newer version of the CORTEX Interaction Portal solution, but they cannot export them individually and add them to the newly imported app.



The conclusion of this is that any pages created between the import of the original version of the CORTEX Interaction Portal pages, and the later updated version, will have to be created manually in the later version. Here, steps to do this will be outlined.

## 2 Checklist

At a high level, the following steps must be undertaken:

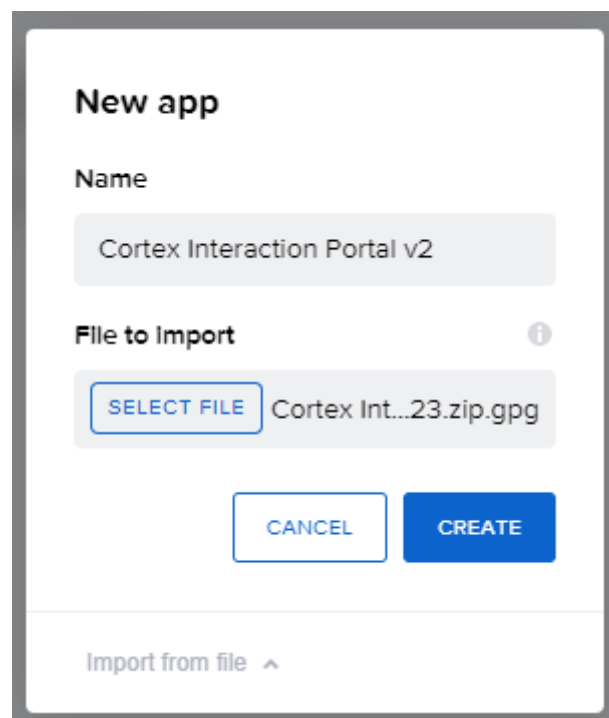
1. Import new version of CORTEX Interaction Portal as a new app.
2. In the new app:
  - a. Re-create pages that were created in the old app as blank pages.
  - b. Re-create any App Variables that are in the old app but are not in the new app.
3. For each of the pages:
  - a. Copy components of these pages from the old app.
  - b. Re-create required page variables, page parameters and data variables that are in the old app.
  - c. Copy Page Mounted logic on each page from the old app.
  - d. Go through the logic flows and check expressions reference the correct blocks.
  - e. Re-create custom styles.
  - f. Re-install custom components and logic blocks from the marketplace.

## 3 Steps to Merge Pages

### 3.1 Import New Version

To import the new version, complete the following steps:

1. Obtain the latest version of CORTEX Interaction Portal as a package
2. Navigate to Appgyver at <https://platform.appgyver.com/>.
3. Log in using the credentials of the account to which the Appgyver app should be imported.
4. Select the 'Create New' button.
5. In the dialog which appears, select 'Import from file'.
6. Select the package.
7. Give the app a relevant name (such as 'CORTEX Interaction Portal v2') and click 'Create'.



The screenshot shows a 'New app' dialog box. At the top, it says 'New app'. Below that is a 'Name' field with the text 'Cortex Interaction Portal v2'. Underneath is a 'File to Import' section with an information icon. It contains a 'SELECT FILE' button and a file name 'Cortex Int...23.zip.gpg'. At the bottom of the dialog are 'CANCEL' and 'CREATE' buttons. Below the dialog, there is a link 'Import from file' with an upward arrow.

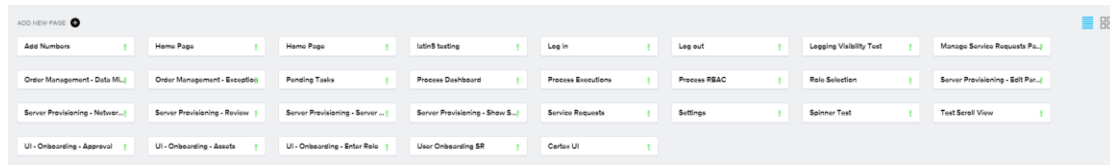
### 3.2 App Level Steps

In the new version of the app, complete the below.

## Re-create Pages

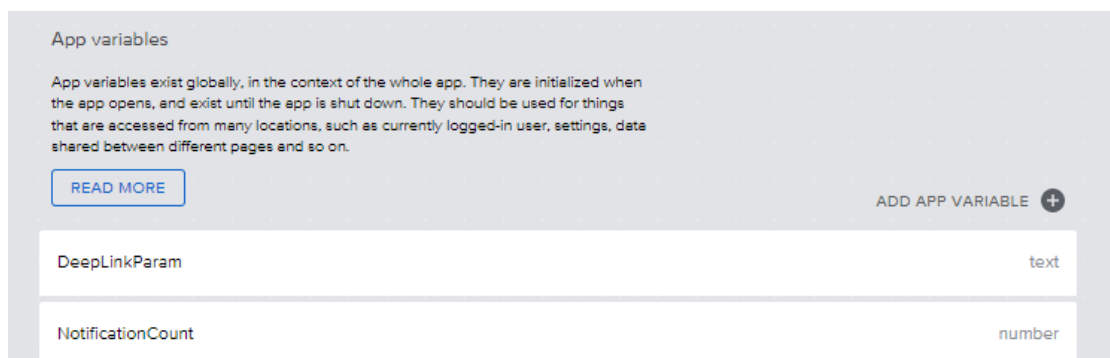
Before starting, it is necessary to identify precisely which pages from the old version of the CORTEX Interaction Portal app should be re-created in the new one.

For each page in the old version of the app, create a page in the new version with the same name, and delete all its contents.



## Re-create App Variables

For each App Variable in the old version of the app that is not in the new version, create a new one in the new version with the same name and properties.



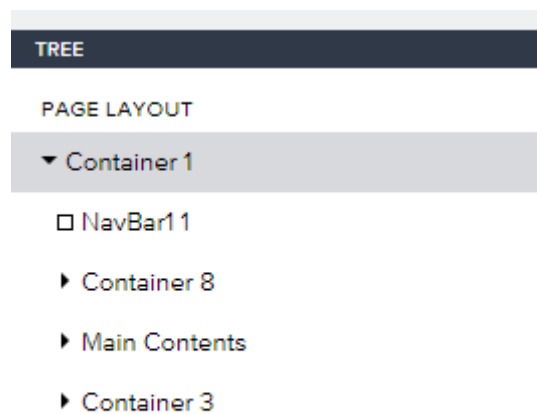


## 3.3 Page Level Steps

In each newly created page, complete the below.

### Copy Components

To recreate the page, page components may be copied from the old version of the page to the blank new page, so they do not need to be created from scratch.



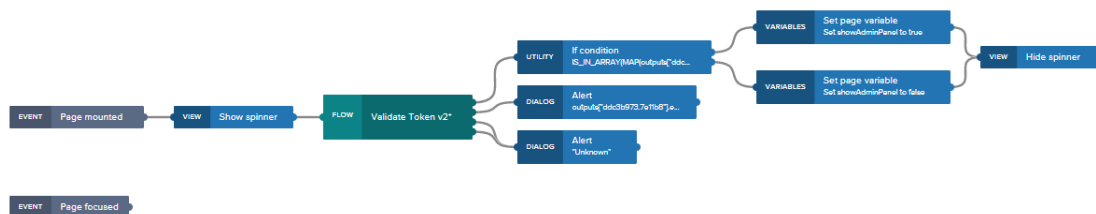
### Re-create Page Variables

For each Page Variable in the old version of the page that is not in the new version, create one in the new version with the same name and properties.

Page variables	
<p>Page variables exist in the context of the current page. They are initialized when the page opens, and removed from app state when the page is closed. They should be used for things that exist in the context of the current page, such as form data, loading state of the current page, selected filters and so on.</p> <p><a href="#">READ MORE</a></p> <p>ADD PAGE VARIABLE <span>+</span></p>	
▼ AllParameters	object with 1 property
id	text
FormItem	text
ParameterNamesGivenValue	list of texts
▼ ParameterValue	object with 3 properties
Name	text
Value	text
id	text

## Copy Logic

Copying a component with associated logic (e.g. a button that calls a CORTEX Flow) will also copy that logic. Therefore, the only logic that requires copying from the old version of the page to the new one is the 'Page mounted' logic.



## Check Logic

At the time of writing, there is an anomaly that can occur when copying logic between AppGyver pages. This occurs when a property of the copied and pasted logic block is bound to a formula containing a reference to the output of another block.

For example, the 'Page mounted' logic will often contain a standard call to a CORTEX flow to validate the authentication token of the logged in user. The output of this is then passed to another AppGyver block when another flow specific to this page is called. This logic is shown below:



Here, the AppGyver flow to call a CORTEX flow has properties like the following:

The AuthToken formula, as seen above in the properties window for the block, is as follows:

```
outputs["a79c6d9d.040c4"].sessionDetails.authToken
```

However, when opening the formula in the formula editor, it appears as follows:

```
outputs["Validate Token v2"].sessionDetails.authToken
```

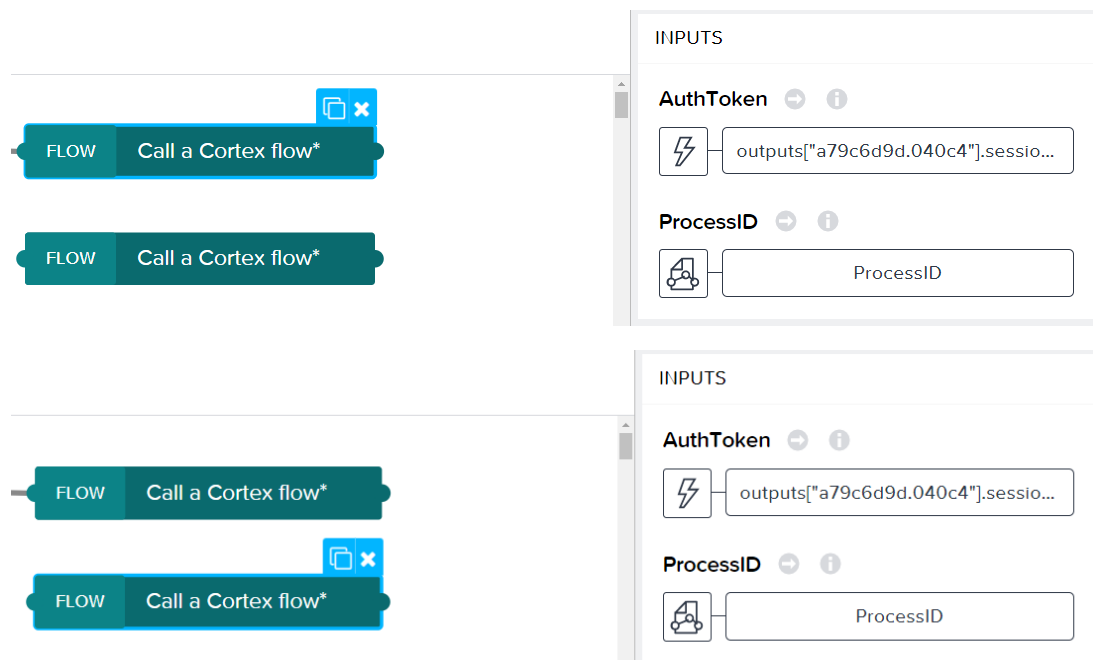
There is a hidden feature of AppGyver where every logic block (or flow) placed by a user has its own unique ID. AppGyver is intelligent enough to record which blocks are available to have their outputs used by the block being configured, shown by selecting 'Output of another node' in the formula editor:



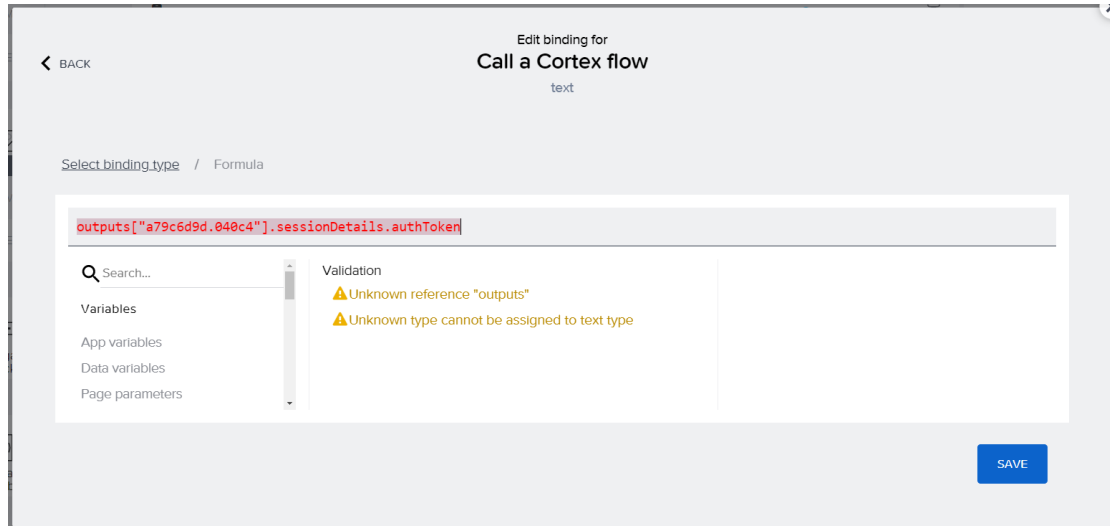
Selecting one of these will ensure that the ID of the block is recorded in the background to refer to it, and the name of it can be used in the formula.

This link can be broken by copying and pasting the block referring to the output of another block.

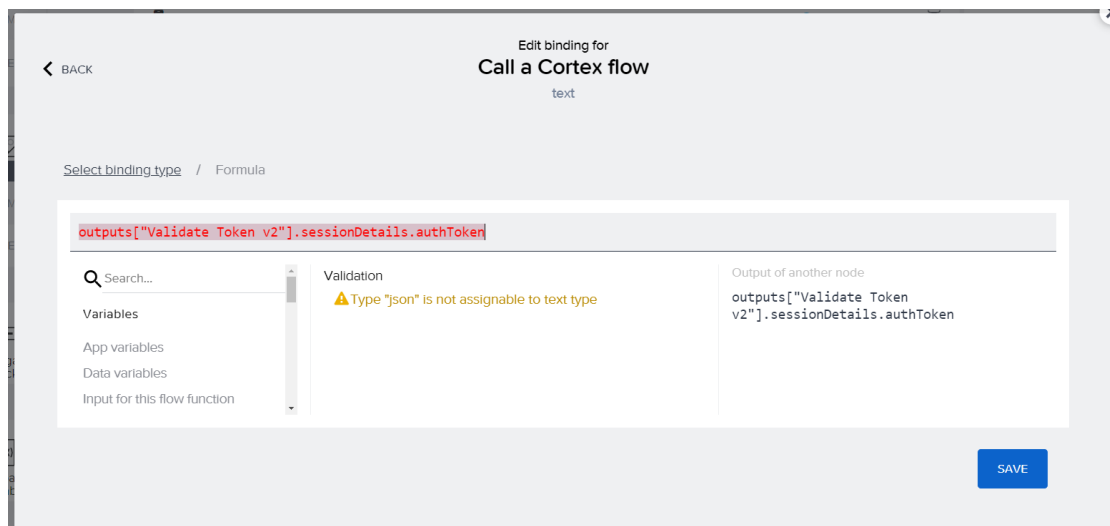
For example, the upper most block below has been copied and pasted underneath it. At first glance, they have the same 'AuthToken' property value.



However, when opening the property windows, the property of the pasted block still contains the ID, as opposed to the name, of the block which is having its output referenced.



It is recommended then after any logic is copied and pasted, that it is checked to ensure this has not occurred. If IDs like this are present within formulae, then they must be replaced with the name of the block, like the below:



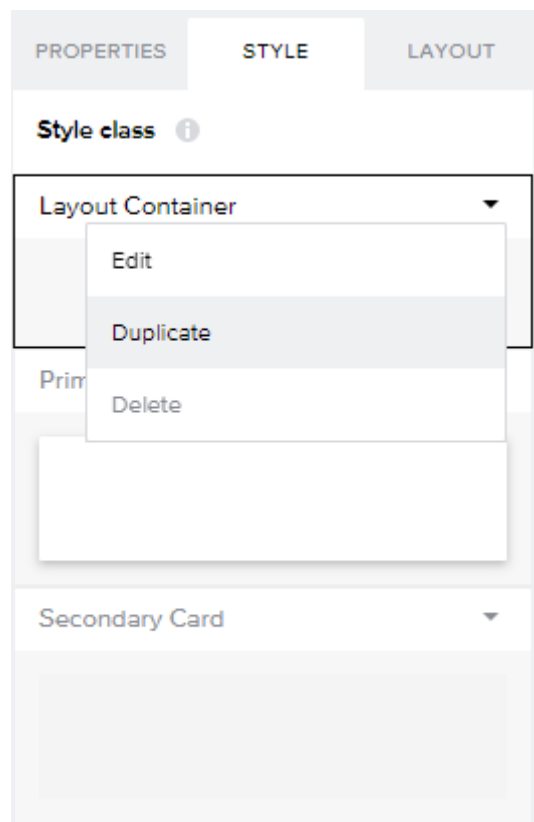
*The warning 'Type "json" is not assignable to text type' may be safely ignored, despite the formula being highlighted in red.*

## Re-create Custom Styles

If any custom styles have been created for a component, they should be re-created in the new app.

New styles may be created by completing the following steps:

1. Select the component in the new page.
2. Select the 'STYLE' tab in the property window on the right.
3. Select the dropdown in the top right of an existing style.



Select 'Duplicate' to copy that style to a new style.

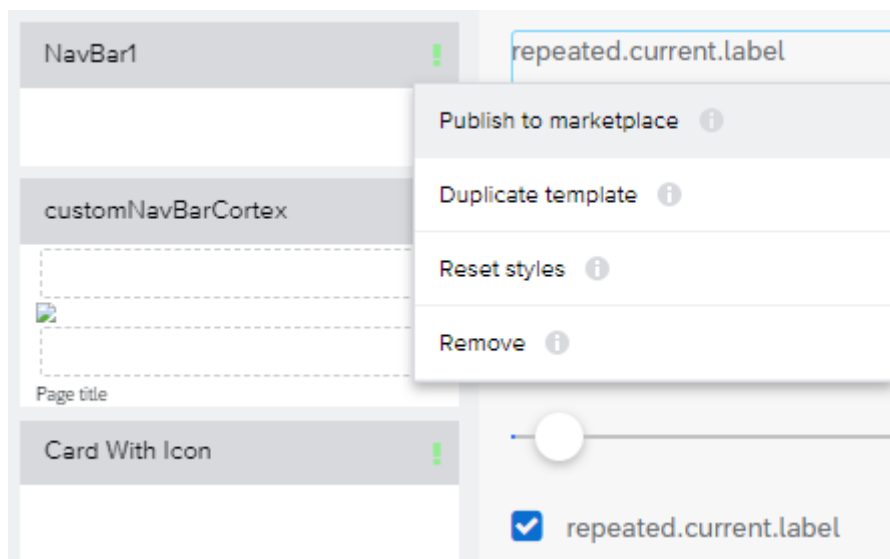
Select the new style and edit it so that it has the same properties as the style that was in the old app.

## Re-install Custom Components

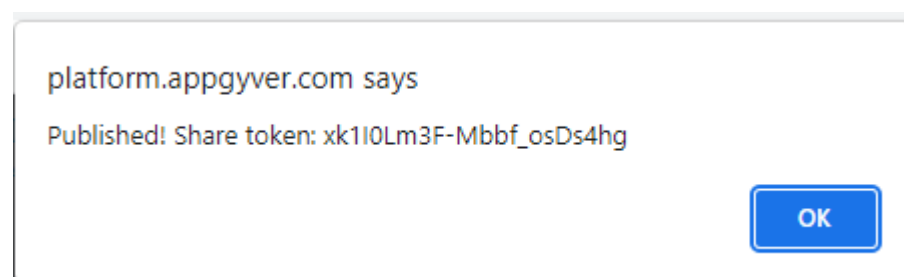
If any custom components have been created in the old app, they should be re-created in the new app. This may be done by publishing each component to the marketplace from the old app, then installing them in the new app.

To do this, complete the following steps:

1. Open the old app to any page.
2. Locate the component to publish in the 'BY ME' palette to the left of the screen.
3. Select the menu symbol in the top right of the component, and select 'Publish to marketplace'.



4. Select 'Publish New' in the window that is shown.
5. Once the publishing has completed, a share token will be displayed to the user, copy this, and paste it into a text editor for later use.



6. Open the new app to any page.
7. Go to the marketplace.
8. Paste the share token into the search bar.
9. Install the component that results.

## Re-install Custom Logic Blocks

When copying logic between apps, if a block that was installed in the old app does not exist in the new app, then an error will be raised to the user, and that block will not be pasted.

These custom blocks should be re-installed from the marketplace and inserted into the logic that was copied from the old app and pasted into the new app.

If any blocks have been created by users in the old app, then they may be published and installed into the new app using the method in Section 3.3.7.