Cortex Interaction Portal

Deployment Guide

Contents

Versions

Date	Version	Notes
13/02/2023	0.1	Internal release
10/04/2023	1.0	First release

Preface

About this manual

This manual provides a general guide to the deployment of the Cortex Interaction Portal, including the steps to configure and deploy alongside a Cortex Innovation instance.

Audience

This document is intended for developers who intend to install the Cortex Interaction Portal alongside a Cortex Innovation instance.

Related Material

Document	Version
Cortex Interaction Portal Developer Guide	1.0
Cortex Interaction Portal Merging Guide	1.0
Cortex Interaction Portal User Guide	1.0

Abbreviations used in this Document.

UI   User Interface
UX   User Experience
IIS   Internet Information Services
RBAC   Role Based Access Control

Referenced Files

File	Description
Cortex.Interaction.Portal.zip.gpg	AppGyver Solution package (to import into the AppGyver development environment)
Cortex.Interaction.Portal.Flows.studiopkg	Cortex Innovation flows: User Access Management (for AppGyver integration and user management)
Cortex.Interaction.Portal.postman_collection.json	Postman collection for testing Cortex Innovation flows

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 High-Level Architecture

1.2.1 Single Node

The simplest possible architecture for a Cortex Interaction Portal is to install it alongside Cortex Gateway and a single Cortex node on one server. It may be accessed by a user alongside Cortex Gateway and interacts with the flows published to the Cortex node via its API Gateway Service.
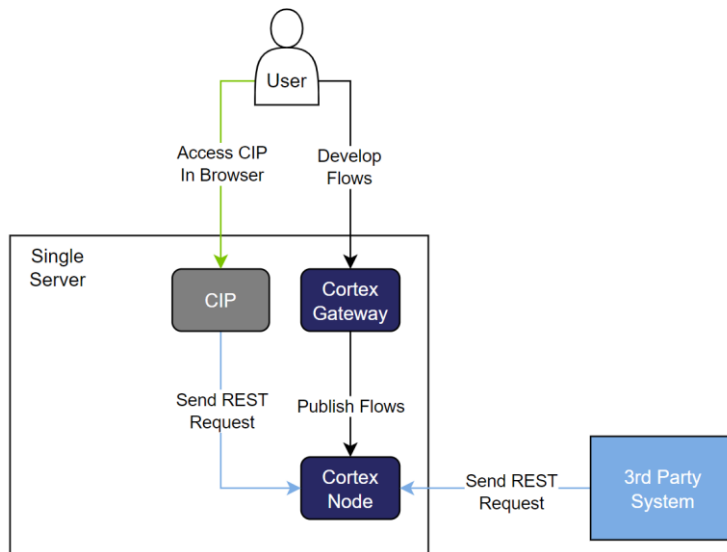
While this may serve for a simple proof-of-concept or development environment, often Cortex Gateway is installed on its own dedicated web server. The Cortex Interaction Portal may be installed alongside it here.
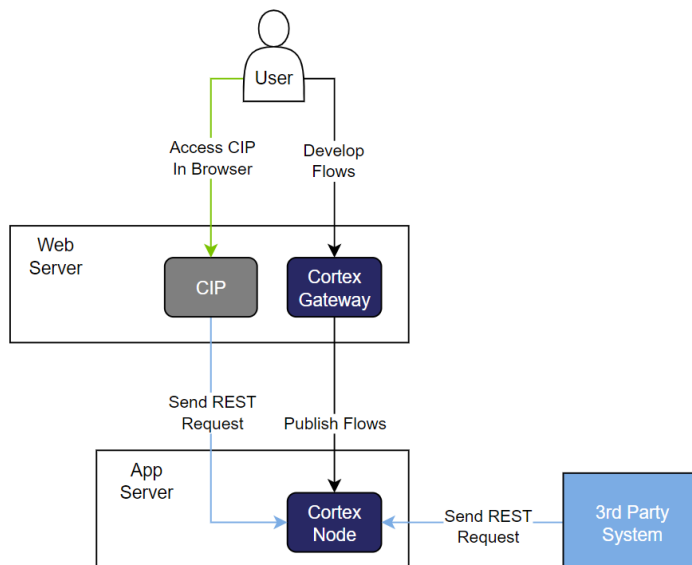
Alternatively, it may be installed alongside the Cortex Node on the Application Server.
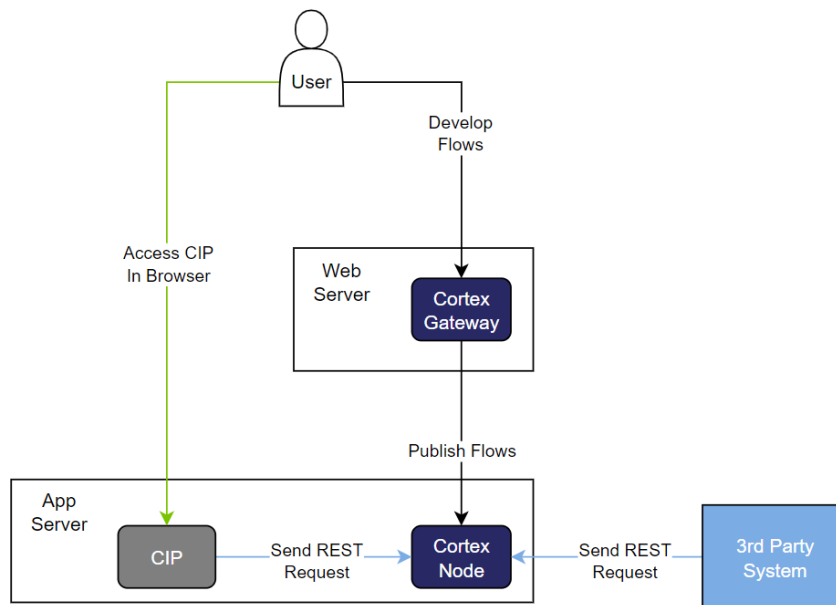
An Interaction Portal may be installed on its own dedicated server as well.
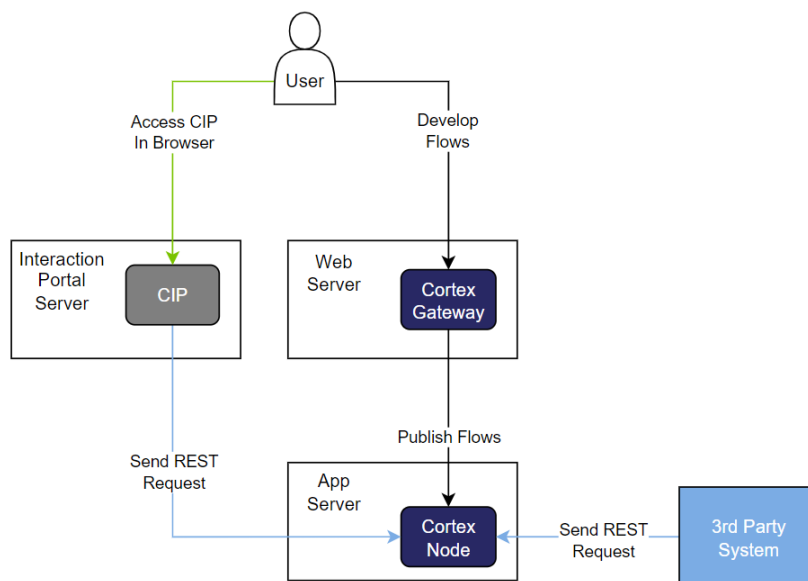
1.2.2 Multi Node

The below is the default set-up for a three node Cortex environment without an Interaction Portal.

> 🖺 *For simplicity, the web server hosting Cortex Gateway has been omitted.*

During a standard Cortex installation including a Load Balancer server, the GoBetween service is installed there in order to route each inbound REST requests to one of the three nodes.
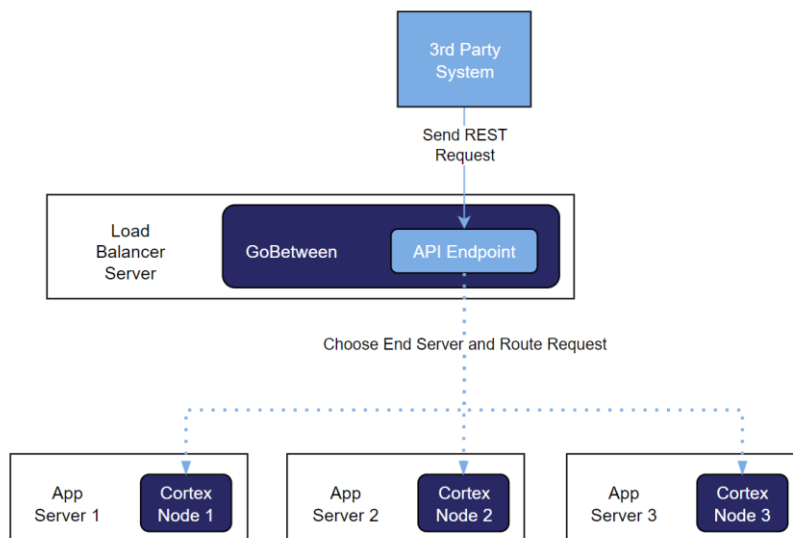
Similarly, if a Cortex Interaction Portal web application is deployed to each of the three application servers, then it is possible to configure the load balancer to also route a user's session to one of them, as below:
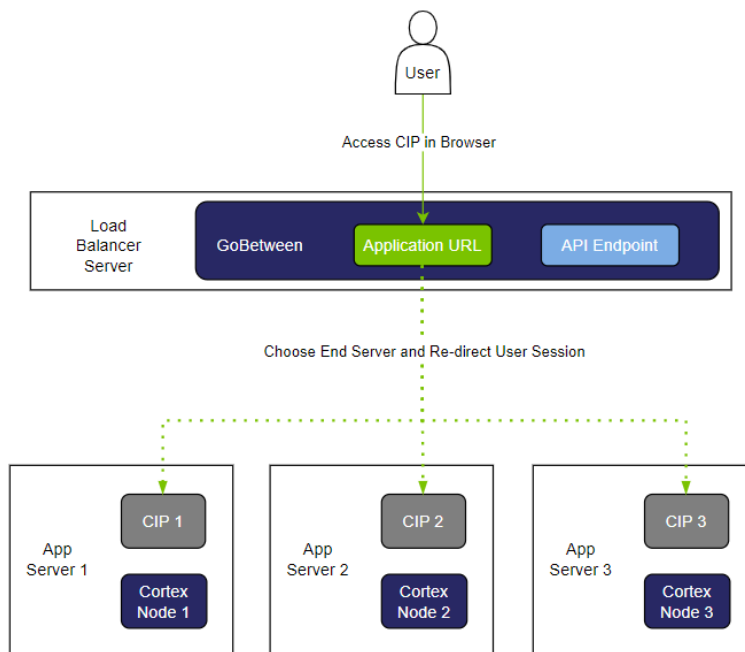
Combining these diagrams, we may see the full interactions between the Cortex Interaction Portals, the Cortex Nodes, and the Load Balancer.
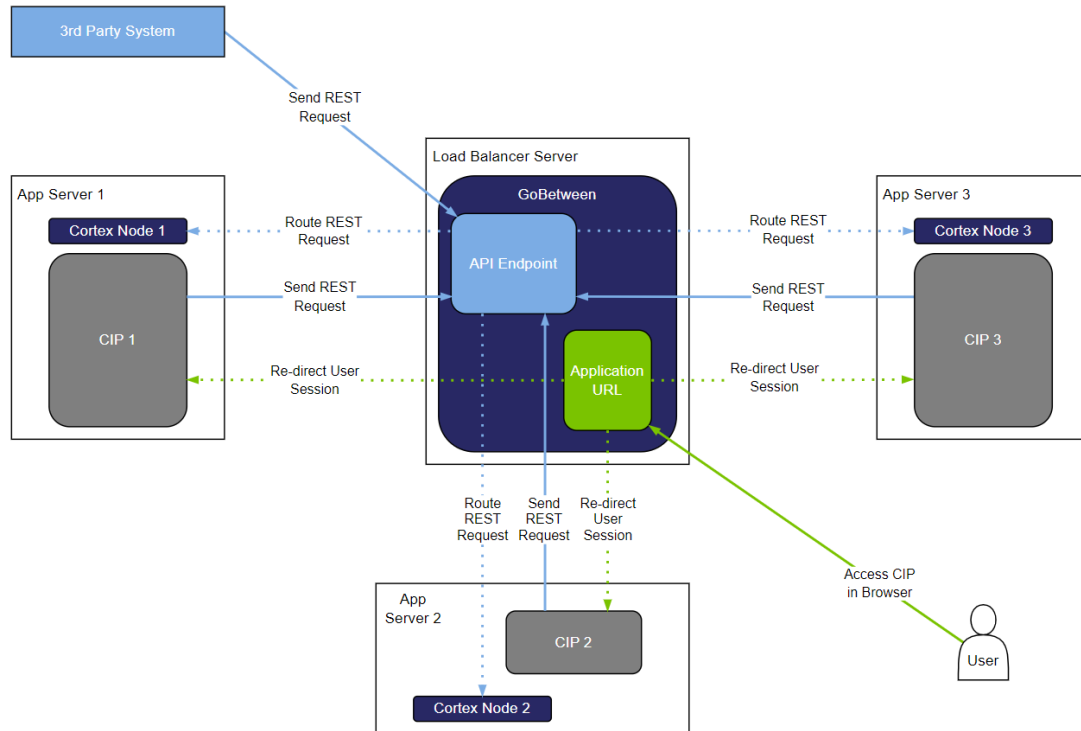
Note that in a multi-node setup, the end users would access the Web Application via the Load Balancer URL which would then route to an available node accordingly.

In terms of the functional aspect of the AppGyver Web Application, the Cortex Interaction Portal will be sending requests to each Cortex node's API Gateway via the Load Balancer, and performing actions based on the response. This includes the below core tasks, as well as any bespoke actions:

1. Authenticate User
2. Validate User Session (auth token)
3. Get / Set System Settings
4. Get Service Requests
5. Manage Service Requests and Groups (CRUD)
6. Get Dashboard Data - Process Counters
7. Get Process Execution Data
8. Get Process Task Input Data
9. Set Process Task Response Data

2   Setup and Deployment

Detailed steps for deploying the Cortex Interaction Portal to a host server will be outlined in this section.

> ⚐   *For a higher-level checklist, see Appendix B*

2.1   Deployment Package Preparation

In order to install the Cortex Interaction Portal, the relevant installation package must be acquired and made ready.
1. Download Cortex Innovation 2023.3 - Interaction Portal.zip
2. Extract the .zip to a suitable folder.

2.2   AppGyver Account Setup

The online AppGyver platform serves as a development environment for AppGyver apps and importing the pre-built Cortex Interaction Portal app will allow us to make small changes to it so that it works as expected when deployed.

> ⚐   *This platform is also where further pages and functionality may be developed.*

To do this, complete the following steps:
1. Navigate to AppGyver online platform at **https://platform.appgyver.com/**.
2. If an account is available, login to it before proceeding to Section 2.3.
3. If no account is available, then complete the following steps to create one.
    a. Select 'Create a new account'.
    b. Enter your details and accept the terms of service and privacy policy.
    c. Select 'Get your free account'.
    d. An email will be sent to the address listed as part of your details. Follow the link included to activate your account.
    e. Once the account is activated, sign into it.
    f. Complete the set-up steps.
    g. If the MyApps page is not immediately displayed, select the AppGyver logo in the top right corner.

2.3   App Variables Setup

Once logged into an existing AppGyver platform account, complete the following steps to configure the application:

1. Select the 'Create New' button under 'My Apps'.
2. In the dialog which appears, select 'Import from file'.
3. Select Cortex.Interaction.Portal.zip.gpg, provided in the package downloaded in Section 2.1.
4. Give the app a relevant name (such as Cortex Interaction Portal) and click 'Create'.

Once the app has been imported, the Config Object must be set up correctly. The elements of this are global parameters that govern how the app behaves.

This single configuration is required, then when the app is built and deployed their values will be included in the deployment, providing all the necessary information to interact seamlessly with Cortex flows.

To set up the App Variables, complete the following steps:
1. Open the app to any page.
2. Select the 'View-Variables' toggle in the top right of the screen to switch to the variables view.

VIEW ⬤——— VARIABLES

3. App Variables should be shown by default, but page variables and parameters may be viewed using the panel on the left.
4. Select 'App Variables'

APP VARIABLES

PAGE VARIABLES

PAGE PARAMETERS
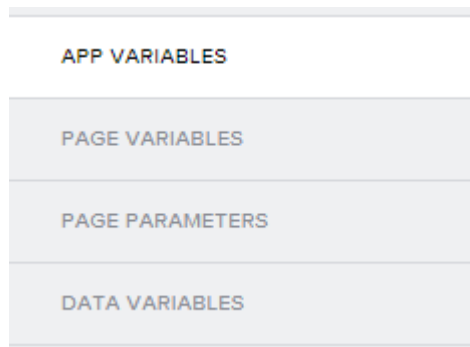
DATA VARIABLES

5. Selecting the 'ConfigVariables' object will allow the default value of each element to be configured in the panel on the right.

ConfigVariables
App variable

Varlable name ⓘ

ConfigVariables

Varlable value type ⚙

Object ▼

Add new property

Type property name ➕

Initial value

CortexUrl

https://<server>.<domain>.com

UamTenant

default

CortexPort

8722

CortexTenant

default

FlowAuthBase64

QmFzaWNbdXRoVXNlcjpBREE5ODgz

6. The following elements should be configured:

Variable Name	Variable Usage	Default Value Description	Default Value Example
cortexUrl	URL to communicate with the Cortex API endpoint.	Base Cortex App URL.	https://cortexApp1.domain.com
uamTenant	Cortex Tenant containing the published User Access Management flows.	For most configurations this will be "default".	default
cortexPort	Port to communicate with the Cortex API endpoint.	For single node installations where REST requests are sent directly to a Cortex Node's API Gateway, this is 8722. For multi-node installations where REST requests are sent to a Load Balancer, this is 443.	8722
cortexTenant	Cortex Tenant	For most configurations this will be "default".	default
FlowAuthBase64	Base64 encoded username and password to communicate with the Cortex API endpoint.	This is generated in the form of username:password Note that there are tools online to encode text in Base64. You can also test in Postman using Basic Authentication, and check the auto-generated headers.	dXNlcm5hbWU6cGFzc3dvcmQ=
UamEnvironment	Cortex Environment containing the published User Access Management flows.	For most configurations this will be "default"	default
UamPackageName	Name of the package containing the published User Access Management flows.	When creating and publishing the package of flows, if the name is changed, this variable will need to be updated.	UserAccessManagement
MobileBreakpoint	Used to determine the size of the screen before the app switches from a side bar to a hamburger menu.	For most configurations this will be 769.	769

| cortexEnvironment	Cortex Environment	For most configurations this will be "default".	default
UamPackageVersion	The version of the package containing the published User Access Management flows.	This should be left blank.	

7. Save the app.

2.4 Theme Variables

This section is optional and will only be necessary if a custom theme for the Interaction Portal is required.

Throughout the core pages of the Cortex Interaction Portal, the styles used reference colours and text sizes that are stored as theme variables. For example, the 'Primary Font' variable is used below:
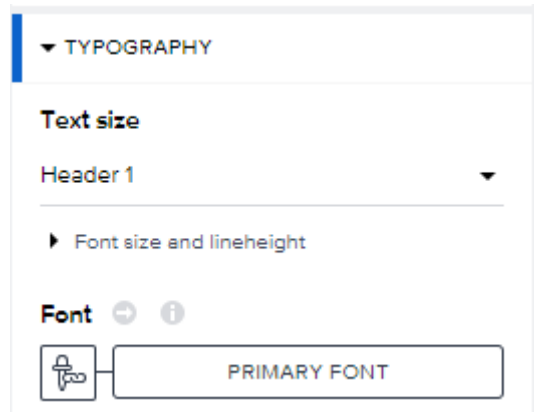
These variables control the look and feel of the Interaction Portal and are set to We Are CORTEX colours by default. To change the theme of the application, complete the following steps:

1. In the AppGyver online platform, with the application imported ready for editing as in Sections 2.2 and 2.3, select 'Themes' from the bar at the top.
2. Select 'Theme Variables' from the panel to the left.
3. From the menu that is displayed, many different parameters that control the look and feel of the Interaction Portal can be configured.
4. The colour scheme of the core pages is stored under 'Color Variables' > 'Cortex'. Update these where necessary to quickly change the colour scheme of the Interaction Portal.
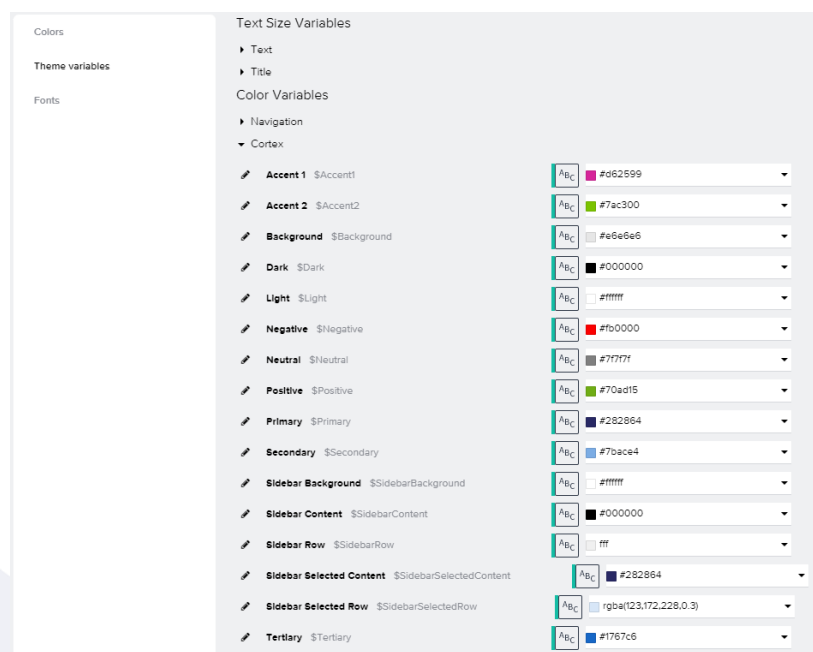
2.5 Import Cortex Flows

The flows must be imported into the Cortex Innovation environment(s).

1. Log into the Cortex Gateway that was installed with the Cortex application server.
2. In Cortex Gateway, click 'Settings', then 'Studio Import'.
3. Select Cortex.Interaction.Portal.Flows.studiopkg, provided in the package downloaded in Section 2.1, and import it.

The import hierarchy should not need to change unless the flows need to be stored in a different location.

4. Set up the access to these flows using Studio Authorisation

Once the flows are imported, they should be available from the 'Flows' charms menu.

2.6 UAM-Get-Config Flow Configuration

1. Log into the Cortex Gateway that was installed with the Cortex application server.
2. In the Flows charms menu, navigate to User Access Management > UAM-Get-Config
3. You may also search for the flow directly.
4. This flow contains one Set Variable block which must be edited accordingly:

5. Update the following parameters only, leaving any that are not listed as they are already configured.

Parameter	Details	Example
PowerShellDetails.Username	Username of an account to use PowerShell, e.g., Service Account	"ctx_ServiceAccount"
PowerShellDetails.Password	Corresponding encrypted password	"#_124211015226168!130 105247000243225146179 242013178~14613515910 0034!21421612819102523 8010012072111212#"
PowerShellDetails.Domain	Domain of the account	"myDomain"
PowerShellDetails.Host	Hostname of the machine	"cortexApp1.myDomain.com"
PowerShellDetails.Port	PowerShell Port	5985
PowerShellDetails.SSL	Boolean indicating whether SSL should be used	False
RecursiveAccessControl	Boolean indicating whether child user groups should inherit access control granted to parents. e.g. User 1 is in 'Group A', Group A is a member of 'CIP Users', CIP Users has access to the system. If this is true, the CIP would allow User 1 access. Default is False	True
OptionalAdConfig.DomainController	The name of the domain controller server. If empty, the domain that the node is attached to will be used.	"server-name"
OptionalAdConfig.BaseAdGroupSearch	The base path within the domain from which users can be selected. If empty, the entire domain will be used.	"CN=Builtin,DC=CortexUsers,DC=com"

6. The flow should then be saved and committed.
7. A new version of the UserAccessManagement package should be created and published, so that these changes are taken into account going forward.

2.7   Publish Cortex Flows

The flows then need to be added to a package and published to the server.

1. In Cortex Gateway, click 'Settings', then 'Packages'.
2. Select 'Add Package Definition' and enter a Package Name
3. The default name is 'UserAccessManagement', however any value can be entered.
4. If changing from default, the UamPackageName element of the ConfigVariables App Variable in AppGyver will need to be updated accordingly.
5. From the Flows panel below, select all the imported flows, along with any additional flows you might need with it.
6. The core flows will be the User Access Management group.
7. Click Save, then when saved, click Publish.

2.8   Website Deployment

The deployment script included in the package obtained in Section 2.1 will perform the following required actions:

1. Create the necessary program data for the application.
2. Create and populate the application settings configuration file, which governs the user groups in Active Directory that will be granted user and admin roles for the application.
3. Create and populate the role-based access control file, which governs a more granular level of authorisation, allowing user groups in Active Directory access to specific tasks raised by a process.
4. Set up the website in IIS ready for the application to be deployed to it.

To run the script, complete the following steps.

1. Copy the package obtained in Section 2.1 to a server where the CIP is to be installed.
2. If it has not been unzipped, then unzip it to a suitable location.
3. Open Windows PowerShell ISE as an administrator
4. In the PowerShell window, run the following to navigate to the directory where the file was unzipped:

```
cd <file path>\Cortex.Interaction.Portal
```

For example:

```
PS C:\Windows\system32> cd C:\'CIP Install Files'\Cortex.Interaction.Portal
PS C:\CIP Install Files\Cortex.Interaction.Portal>
```
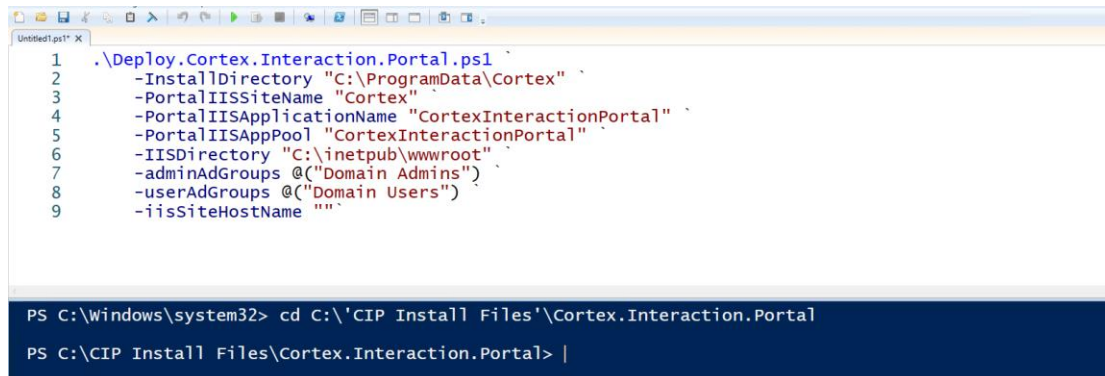
5. In the script window, paste the following:

```
.\Deploy.Cortex.Interaction.Portal.ps1 `
-InstallDirectory "C:\ProgramData\Cortex" `
-PortalIISSiteName "Cortex" `
-PortalIISApplicationName "CortexInteractionPortal" `
-PortalIISAppPool "CortexInteractionPortal" `
-IISDirectory "C:\inetpub\wwwroot" `
-adminAdGroups @("Domain Admins") `
-userAdGroups @("Domain Users") `
-iisSiteHostName ""
```

6. Update any parameters as necessary.

Parameter Name	Description	Example
InstallDirectory	The directory to where the application will be installed	"C:\ProgramData\Cortex"
PortalIISSiteName	The name of the web site to which the application will be deployed	"Cortex"
PortalIISApplicationName	The name of the application when it is deployed	"CortexInteractionPortal"
PortalIISAppPool	The name of the application pool under which the application will run	"CortexInteractionPortal"
IISDirectory	The IIS root folder	"C:\inetpub\wwwroot"
adminAdGroups	A comma separated list of the user groups in Active Directory to give admin roles	@("Domain Admins", "Local Admins")
userAdGroups	A comma separated list of the user groups in Active Directory to give user roles	@("Domain Users", "Local Users")
iisSiteHostName	The hostname to bind to the website	""

🏳 *In practise, only the adminAdGroups and userAdGroups should require configuring, with all other parameters remaining as default.*

7. The PowerShell ISE window should look as follows:

8. Run the script using the play button above the script window.

9. Once complete, check that the folders were created correctly.
   a. Navigate to C:\ProgramData\Cortex\User Access Management
   b. Check that the folders and files are present as below:

> This PC > Windows (C:) > ProgramData > Cortex > User Access Management

Name	Date modified	Type	Size
Archived Service Requests	11/24/2022 11:07 ...	File folder	
Manual Intervention	11/22/2022 3:03 PM	File folder	
Service Requests	11/30/2022 10:13 ...	File folder	
Sessions	11/30/2022 11:04 ...	File folder	
config.json	11/25/2022 1:51 PM	JSON File	1 KB
rbac.json	11/25/2022 1:51 PM	JSON File	1 KB

2.9 Web Application Deployment

1. Login to AppGyver and go to the App.
2. Select a page (for example Home Page) and select Launch from the toolbar at the top.
3. In the left-hand panel, select 'Distribute', then select 'Open Build Service'.
4. For the first build operation, you will need to configure the build:
   a. Select 'Configure' under the 'Web App' option.
   b. For the Hosted Domain, enter any value – this will not be used as we will be deploying directly to the server.
   c. Select 'Save & Next'.
   d. Select ZIP as the Build Scheme.
   e. Select 'Save & Next'.
   f. Upload a favicon to act as a logo for the portal if desired, for example, the Cortex Logo or your company logo.
   g. Select 'Save & Next'.
   h. Nothing is required in the 'Permissions' sections.
   i. Select 'Save & Next'.
5. Click 'Build' under the 'Web App' option.
   a. Ensure ZIP is selected.
   b. Select the latest Client runtime version.
   c. Select 'No' for whether the app should be deployed to the cloud.
   d. Enter a version number, for example 0.0.1
   e. Click Build

Once the operation has completed after several minutes, an email will be sent to the email address for this AppGyver account. This will contain a link from which the .zip file may be downloaded. This can then be extracted and placed into an IIS Site on the App Server(s) that was created by the script executed in Section 2.5.

1. Unzip the file that was downloaded.
2. Navigate to the folder where the web application was installed in File Explorer
   a. This will be of the form:
   ```
   <IISDirectory>\<PortalIISSiteName>\<PortalIISApplicationName>
   ```
   e.g. **C:\inetpub\wwwroot\Cortex\CortexInteractionPortal**

3. Copy all the unzipped files across
   a. If the copy operation fails, you may need to first copy them to a directory local to your user (such as a folder in Downloads)

Once this has been completed, the Web Application is accessible from your server directly, for example: **https://www.myserver.myDomain.com/CortexInteractionPortal.**

To change the displayed Website Name at the top of the browser when using the web app, see **Appendix A: Renaming Web App Title.**

2.10 CORS Configuration

CORS, or Cross-Origin Resources Sharing, is a security protocol in websites which prevents requests from an unknown origin. To ensure the Web application can call Cortex correctly, It must be listed as an allowed origin.

An origin is identified based on the scheme (e.g., HTTPS), domain, and port. The Cortex API Gateway should share the same scheme and domain, but with a different port – this means that by default, the requests will not be permitted.

> *Note that the CORS settings are only available with Cortex version 2023.3 and newer.*

1. Navigate to C:\ProgramData\SF\<Customer>.<Node>\Fabric\work\ImageCache\Store\Cortex.Innovation. Core\Cortex.ApiGatewayPkg.Code.<Version>

> *<Customer> and <Node> will depend on how the node was configured in Service Fabric during the installation of Cortex.*

2. Open the appsettings.json file.
3. Under the Cors section, edit it as follows:

Parameter	Details	Example
Enabled	Sets CORS as enabled within the rules specified below.	true

AllowedOrigins	Array of Strings containing all the allowed origins. Note that the origin of the app server(s) hosting the Web App is required, and the origin of the AppGyver Preview App site is optional.	[
		  "https://*.<domain>.com",
		  "https://*.<appgyver>.com"
		]

AllowCredentials	Whether or not credentials are permitted in the request	true
AllowWildcardSubdomains	Whether or not wildcards (*) are permitted for subdomains. Based on the example in AllowedOrigins, this should be true	true

Once this is done, navigate to the Service Fabric Explorer page and restart any Service Fabric nodes for the changes to take effect.

1. Navigate to the Service Fabric Explorer page.
   a. For example, https://cortexApp1.myDomain.com:9080/Explorer/

2. Expand Nodes in the left-hand panel, and in turn, select the node and restart it. If you need to continue operations on the server, they should be restarted 1 at a time.
   a. Click the node.
   b. In the right-hand panel select the 'Actions' dropdown and click Restart
   c. This can take 5-10 minutes per-node to complete.

2.11 Housekeeping Tasks

Included as part of the flow package are the following flows that should be scheduled using guides available at **docs.wearecortex.com**

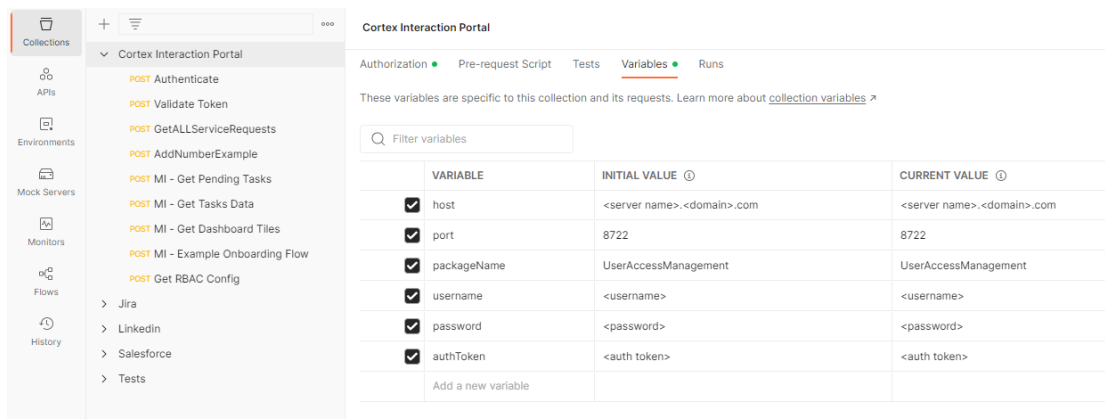| Flow Name	Description	Schedule
UAM-Session-Cleanup	A flow to delete Session json files that are older than 24 hours.	Daily
MI-Process-Cleanup	A flow to delete Process files that are older than 1 week.	Daily

3   Testing

3.1   Testing User Access Management Flows

Once the configuration is complete, the flows can be tested from Cortex Gateway and from Postman.

1. In Cortex Gateway, open the User Access Management > Authentication > UAM-Authenticate-User flow.

2. In the right-hand panel, enter a username and password in the input variables.
    a. Username should be in the format "domain\\username"

3. Set a breakpoint on the final 'End Flow' block and run the flow.
    a. If any exceptions occur, these should be diagnosed and resolved.
    b. If it goes through successfully, check the output variables (accessLevel and authToken) and ensure a file has been written to C:\ProgramData\Cortex\User Access Management\Sessions

Once this flow has been validated, the published flows can be tested using the postman collection **Cortex.Interaction.Portal.postman_collection.json**, provided as part of the package obtained in Section 2.1.

1. Import the collection to Postman, which if required can be downloaded and installed from **https://www.postman.com/**.

2. Update the Collection variables.

Variable	Description	Example Value
host	The host for communication with the relevant Cortex API Gateway	https://cortexApp1.myDomain.com
port	The port for communication with the relevant Cortex API Gateway	8722
packageName	The package containing the user access management flows.	UserAccessManagement
username	Username to use to retrieve an authentication token. The same as the username entered in the login screen for the portal.	
password	Password to use to retrieve an authentication token. The same as the password entered in the login screen for the portal.	
authToken	Authentication token returned from the 'Authenticate' request	ABPzaWNBmahYVXNlcjpIIHJyMXNcbg

3. Select save to persist the variable values.

4. Test the 'Authenticate' request. This is what is called when entering a username and password to access the Interaction Portal.

   a. Select the request and examine the 'Body' tab.

   b. Ensure that the domain, username, and password variables are referenced in this body.
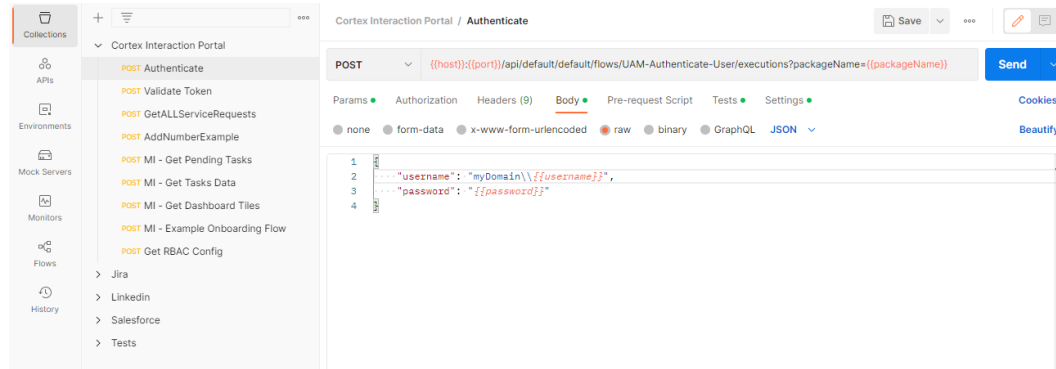
   c. Select 'Send'

   d. Examine the response. If successful it will take the following structure:

```
{
    "accessLevel": "<e.g. Admin>",
    "userProperties": {
        "name": "<username>",
        "accountName": "<username>",
        "upn": "<username>@myDomain.com",
        "roles": [
            <user groups to which the user belongs>"
        ]
    },
    "status": {
        "Status": "OK"
    },
    "authToken": "84f01914-6j74-4055-gf38-e6de9ns9ba41"
}
```

   e. The authToken returned at the bottom of the response may be used for further requests.

3.2 Testing the Web Application

> *If the Cortex Application server does not have inbound internet connectivity, you will not be able to test from the AppGyver preview mode and this step can be skipped.*

> *If you have not configured CORS to enable the AppGyver Preview website, this will not work.*

To test the app, complete the following steps:
1. Login to AppGyver and go to the app.
2. Select a page (for example Home Page) and select Launch from the toolbar at the top.
3. In the left-hand panel, select Preview, then select Open App Preview Portal
   a. Once the page loads, you should open Developer Tools to diagnose any potential issues with the requests to Cortex.
4. You can then attempt to login to the Web Application using domain credentials and validate the request from the Network tab in Developer Tools.

Even if CORS is configured correctly as in Section 2.6, an 'Is CORS enabled' error message may be encountered still while attempting to call a Cortex flow. To resolve this, see Appendix C.

3.3 Testing UI-Driven Process (Service Requests)

> *Note that the example Service Request is a very simple example used to validate that the Service Requests are working correctly and show the general flow of data within the UI.*

1. Navigate to the UI hosted on the server and open Developer Tools to diagnose any potential issues.
   a. For example: **https://www.cortexApp1.myDomain.com/CortexInteractionPortal**
2. Login using domain credentials with the relevant permissions.
3. Click Service Requests – by default there should be 1 Service Request configured.
4. Select 'Add Numbers' in the left-hand panel, and then 'Start' in the right-hand panel.
5. In the page that opens, enter 2 numbers, and click the button.
6. Ensure that a response is received from Cortex and shown in the page.

> *For a more complex example and more rigorous test, start the example 'Server Provisioning' Service Request and follow it through to completion.*

3.4 Testing Process-Driven UI (Process Dashboard)

- *For this process, the role-based access control will need to be set up – you can check rbac.json if required*

- *Note that it is possible to assign different levels of permission, for example user access for the 1st and 2nd UI and manager access for the 3rd (approval) UI.*

1. From Cortex Gateway or Postman, start 'Example-Onboarding-Flow'.
    a. In Cortex Gateway, the flow can be found under User Access Management/Example Flows.
    b. In Postman, the execution will be synchronous. Therefore, no response will be received until the flow completes its execution, which will require manual interventions using the Process Dashboard.

2. Ensure there are no exceptions after it starts running
3. Navigate to the UI hosted on the server and open Developer Tools to diagnose any potential issues.
4. Login using domain credentials with the relevant permissions.
5. Click 'Process Dashboard'
    a. Data should be received from Cortex and used to render the table view.

6. Select either the 'Onboarding Example' Process filter or the 'Pending' Status filter
    a. A list of relevant tasks should be displayed, most likely with 1 item.
    b. Providing the RBAC configuration is set up correctly, you should also be able to see this execution on the top-level Dashboard page.

7. Click the 'i' button to view a list of logs for this process execution.
    a. Close this once it appears.

8. Click the 'Open Task' button in the Action column.
    a. You should be taken to a UI where you can select the persons department – complete this.

9. Refresh the task list – after a short time it should be showing a new task again.
10. Click Open Task again – you should be prompted to assign the person some assets.
    a. Complete this UI by selecting some assets and clicking the button.

11. Refresh again – it should now be pending manager approval.
    a. Complete this, either approving or denying the request

4 Appendix A: Renaming Web App Title

By default, the Web Application title is set to 'AppGyver'. This can be updated by modifying content in some of the post-build files. The below script can automate this action by providing the directory of the unzipped files and the required website name.

> 🏳 *This can be done either before or after copying the files to the IIS directory on the server.*

> 🏳 *For a multi-node environment, it is suggested to perform these steps before the copy to save running it multiple times.*

```
$websiteName = "Cortex Interaction Portal"
$websiteDirectory = "<website directory>"

# Replace <Title> in HTML Files
$htmlPages = Get-ChildItem -Path "$websiteDirectory\*.html" -Recurse |
ForEach-Object {
    # Read the file and use replace()
    (Get-Content
$_).Replace("<title>AppGyver</title>","<title>$websiteName</title>") | Set-
Content $_
}

# Replace title JSON in JavaScript File
$javaScriptFile = Get-ChildItem -Path
"$websiteDirectory\_next\static\chunks\pages\*app*.js" | ForEach-Object {
    # Read the file and use replace()
    (Get-Content
$_).Replace("(""title"",null,""AppGyver"")","(""title"",null,""$websiteName
"")") | Set-Content $_
}
```

5 Appendix B: Quick Start Deployment

5.1 AppGyver Setup

Obtain the deployment package, then In AppGyver, import the solution files. Open the solution and navigate to the variables, setting the relevant values in the below variables:

1. DeepLinkParam
2. cortexEnvironment
3. cortexTenant
4. flowAuthBase64
5. cortexPort
6. cortexUrl
7. uamEnvironment
8. uamTenant
9. uamFlow
10. uamPackageName

5.2 Cortex Setup

1. In Cortex Gateway, import the studiopkg file from the deployment package and ensure the Studio Authorisation is set correctly. These should all be contained within the User Access Management group.
2. Go to the UAM-Get-Config flow and set up the Set Variable flow with the relevant config.
   a. Ensure the PowerShellDetails section is filled in correctly so the flows can run PowerShell scripts.
   b. Save and commit this flow when done.
3. Create a package with all these flows and ensure it is published. The Package Name must match the uamPackageName variable.

5.3 Files Setup

1. Copy the deployment package to the relevant server and unzip it.
2. In PowerShell ISE, change directory to the unzipped deployment package and run the following script after configuring the parameters.

```
.\Deploy.Cortex.Interaction.Portal.ps1 `
-InstallDirectory "C:\ProgramData\Cortex" `
-PortalIISSiteName "Cortex" `
-PortalIISApplicationName "CortexInteractionPortal" `
-PortalIISAppPool "CortexInteractionPortal" `
-IISDirectory "C:\inetpub\wwwroot" `
-adminAdGroups @("Domain Admins") `
-userAdGroups @("Domain Users") `
-iisSiteHostName ""
```

3. You should now see 2 JSON files in the 'C:\ProgramData\Cortex\User Access Management' directory, named config.json and rbac.json.
4. In IIS a new site should have been created for the Interaction Portal if one did not already exist.

5.4 Deploying the Web App

In AppGyver, select Launch > Distribute and then click 'Open Build Service'. In the window that opens, select 'Configure' under the Web App column.

1. In the hostname field, enter any hostname – we are not hosting using the AppGyver cloud so it will not be used.
2. For Bundle Settings, select ZIP.
3. Upload a favicon if desired, for example, the Cortex logo.
4. For Permissions, nothing needs to be done.

Click Build, then ensure the following configuration is applied:

1. File Type = ZIP
2. Client Runtime Version = Latest
3. Should the app be deployed to the cloud = No
4. Version = any semantic version, e.g., 0.0.1

Once the solution is built, unzip the files into the directory for the IIS site on the server that was created in Section 5.4. It should now be accessible from a browser, although some CORS configuration may need to be applied the Service Fabric config for the requests to Cortex to succeed.

To change the Web Application Title, see **Appendix A: Renaming Web App Title**.

6   Appendix C: Issues with Self Signed Certificates

Even if CORS is configured correctly as in Section 2.6, an 'Is CORS enabled' error message may be encountered still while attempting to call a Cortex flow if a Self-Signed Certificate is used.

To resolve this, complete the following steps:

1. Go to **https://<server>.<domain>.com:8722**
2. An error dialog should be displayed, click 'Advanced' then 'Continue Anyway'.
3. Wait for page to navigate to a 'failed to load' page.
4. This ensures that your browser will ignore the self-signed cert in the Flow API Gateway endpoint, so the browser can run flows.