UCPACT Web Application Starter Guide

Riverside Research

Version 1.0

August 2025

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and Naval Information Warfare Center Pacific (NIWC Pacific) under N66001-22-C-4020 and HR001122C0185.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA and NIWC Pacific.

Revision Sheet

Release No.	Date	Revision Description
1.0	08/08/2025	Initial Release

UCPACT Web Application Starter Guide

TABLE OF CONTENTS

Distribution A: Approved for Public Release

LIST OF FIGURES

1 Introduction

This guide describes the UCPACT Web application. This guide will go through obtaining the web application

2 Preliminaries

2.1 Obtaining the UCPACT Web Application

To get the UCPACT Web Application download the source code from https://github.com/riversideresearch/ucpact. It contains a readMe with these instructions as well. We recommend using Ubuntu as that has been proven to work the best but there are alternatives depending on your system.

2.1.1 Requirements for Various OS's

- Ubuntu (as host or guest OS on a VM)
  - Latest version of Docker Engine Community Edition (Docker CE), which does not run natively on Windows machines
  - Docker Compose version 2.x
- Windows (host OS)
  - Latest version of Windows Subsystem for Linux (be sure to install Ubuntu v22.04 or v24.04 as the distro)
  - Only available on a Windows 11 or 10 (build 1903 or later) host OS
  - If using Windows 11 or 10 (build 2004 or later), use the PowerShell command `wsl --install` as described in https://learn.microsoft.com/en-us/windows/wsl/install
  - Otherwise (or to troubleshoot), follow these instructions: https://learn.microsoft.com/en-us/windows/wsl/install-manual
- MacOS (host OS)
  - Docker Desktop (latest version)
  - Requires one of the 3 latest (currently supported) versions of MacOS
  - Installation instructions: https://docs.docker.com/desktop/setup/install/mac-install/
- Any OS that supports the following browsers (may be a separate machine if backend is hosted on a network)
  - Firefox or Chrome (latest versions) to configure Keycloak via its admin UI, and to access the UC-PACT client UI
  - For Virtual Machines, it's preferred to run the browser within the guest (Ubuntu) OS, not the host
  - For WSL, you should be able to access everything from either browser running on your Windows 11/10 host OS
  - For MacOS, simply open either of the 2 browsers that you have previously installed on your Apple computer

2.2 Running the UCPACT Web Application

After downloading the code base and installing the applications required for your chosen Operating System we can get into running the Web application itself. There are several ways to run the Web Application that vary in the amount of set up the user is required to do. Below we detail the various ways that can be used to run the web application. We recommend using the Single-user mode script for personal use as it does not require setting up and using key cloak.

2.2.1 Single-user Mode

Single-user mode is designed to be used locally on individual (not shared) machines. All authentication is disabled internally, with protections in place to prevent any remote client users from deactivating Keycloak authentication if a server is launched in the (default) multi-user mode.

Although this mode only supports a single user on the same local machine as the application's backend, that user may still open multiple models in different tabs, windows, or browsers, while still preserving the existing read-only protections.

The primary benefit of single-user mode is a much quicker startup (especially if the frontend has already been built), since a Docker "spin-up" doesn't need to initialize Keycloak authentication, and the user-creation and login steps are completely eliminated.

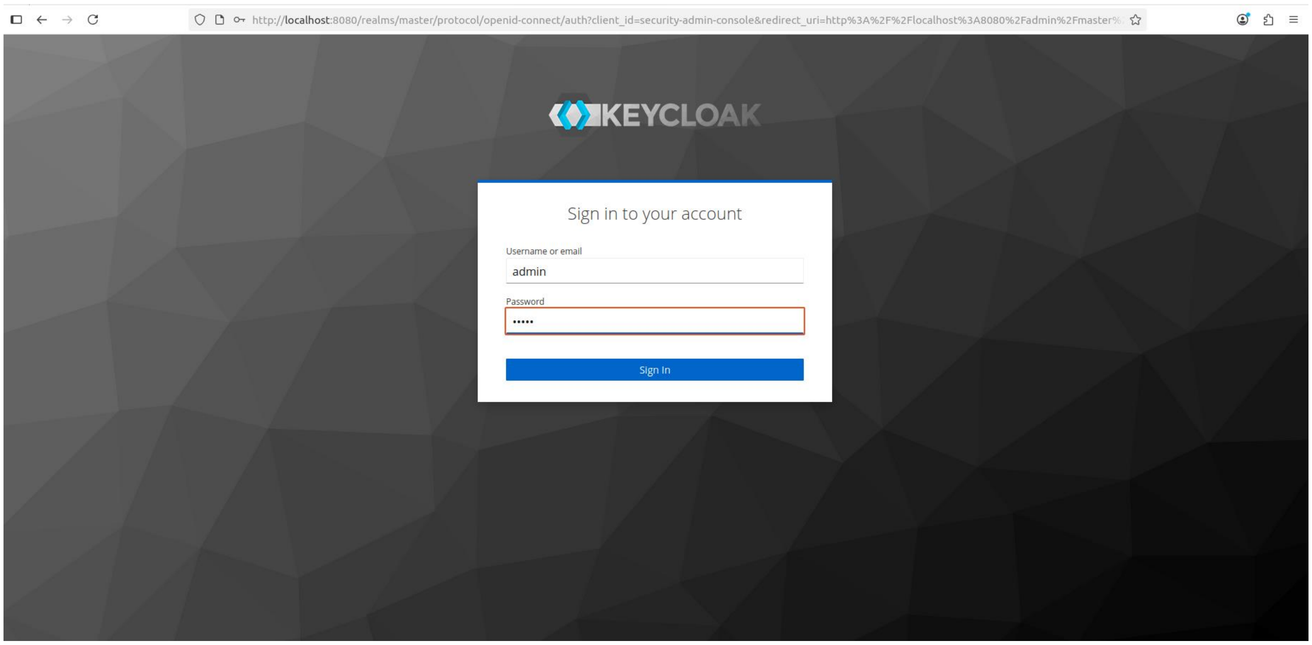
To launch UC-PACT in single-user mode there are two ways to go about it.

- Running with the docker commands
    1. From the top-level repo folder run `*cd single_user_config*`
    2. Then run `*docker compose --profile default build*` followed by `*docker compose --profile default up*`
        - If you plan to do feature development switching out the *'default'* profile for the *'dev'* profile is recommended. The *'dev'* profile allows for not rebuilding the frontend when you change the codebase.
    3. After launching with either the `default` or `dev` profiles, navigate to http://localhost/. However, depending on your system set up you may need to use http://localhost:3000/.
- Running with the Launch Script
    1. simply run `*./launch_single_user.sh*` in the top-level repo folder
    2. Type in the profile you want to run from the list
    3. After launching with either the `default` or `dev` profiles, navigate to http://localhost/. However, depending on your system set up you may need to use http://localhost:3000/.

2.2.2 Multi-user Mode

The Web Application has a setting for being able to handle multiple users for a single instance. This would allow a team of UCPACT developers to collaborate on models that are being developed. This process does require authenticating to the Single Sign on Solution. We utilize the Key Cloak program to handle our authentication needs and will take you through the steps of setting up the development Key Cloak realm. We **do not** recommend you use this realm when you deploy and should set up a production Key Cloak instance or another Single Sign on Solution to use with our application. To run the multi-user version of the application two ways can be used. We recommend using the launch script but it can also be done with docker commands.

- Running with the docker commands
  1. From the top-level repo folder run `*docker compose --profile default build*` followed by `*docker compose --profile default up*`
     - If you plan to do feature development switching out the *'default'* profile for the *'dev'* profile is recommended. The *'dev'* profile allows for not rebuilding the frontend when you change the codebase.
  2. Now Follow the KeyCloak setup instructions below
- Running with the Launch Script
  1. simply run `*./launch.sh*` in the top-level repo folder and type in the profile you want to run when instructed
     - If you choose the default or dev profile continue to step 2
  2. Now Follow the KeyCloak setup instructions below
- KeyCloak Setup
  1. After spinning up with either the `*default*` or `*dev*` profiles, proceed to the `Keycloak` admin interface at http://localhost:8080/admin.
  2. Login with the credentials `username="admin"` and `password="admin"



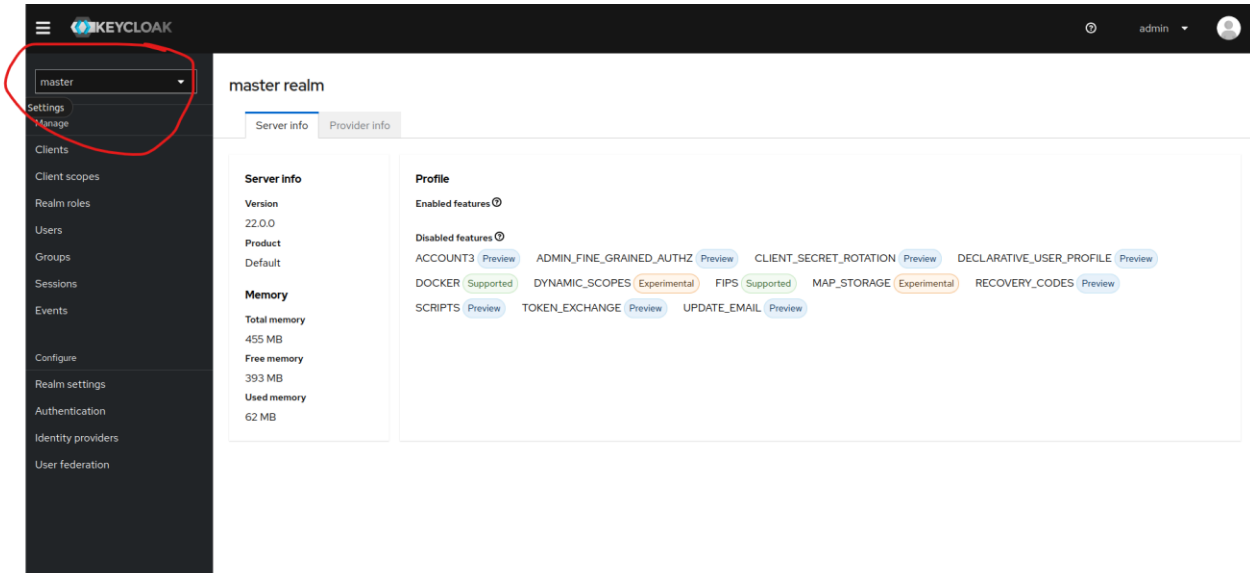
http://localhost:8080/realms/master/protocol/openid-connect/auth?client_id=security-admin-console&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fadmin%2Fmaster%

Username or email: admin

Password: •••••

Figure 1: KeyCloak Admin Login Page

  3. You will need to switch the realm in the upper left corner from '*master*' to '*UCPACT-Realm'*



master realm

Server info

Version
22.0.0

Product
Default

Memory

Total memory
455 MB

Free memory
393 MB

Used memory
62 MB

Profile

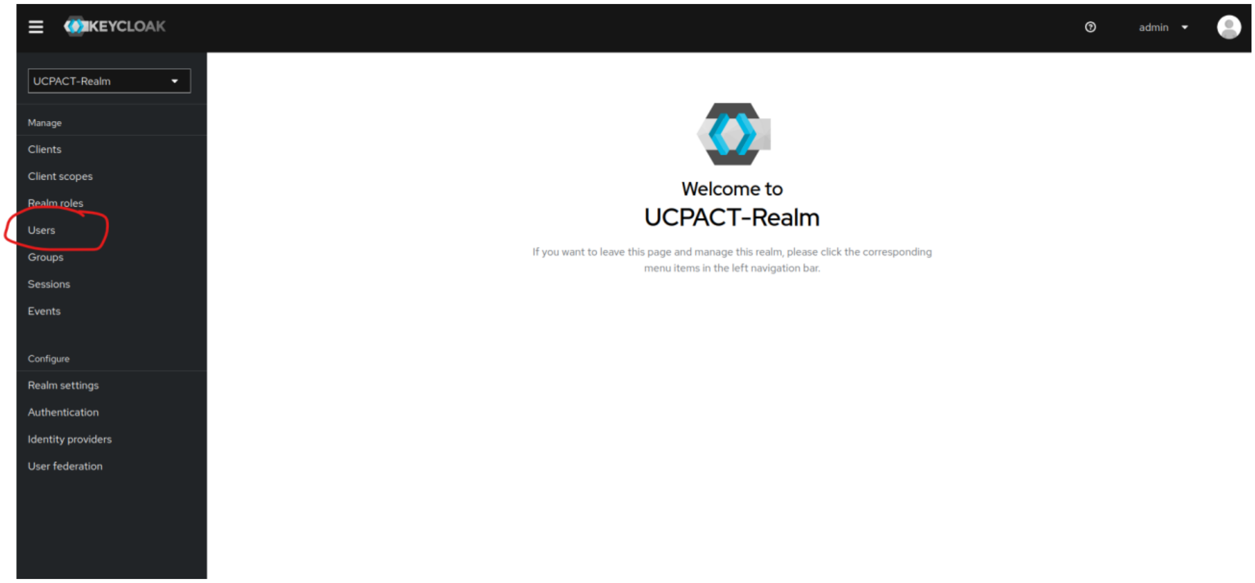
Enabled features

Disabled features

ACCOUNT3 Preview ADMIN_FINE_GRAINED_AUTHZ Preview CLIENT_SECRET_ROTATION Preview DECLARATIVE_USER_PROFILE Preview
DOCKER Supported DYNAMIC_SCOPES Experimental FIPS Supported MAP_STORAGE Experimental RECOVERY_CODES Preview
SCRIPTS Preview TOKEN_EXCHANGE Preview UPDATE_EMAIL Preview

Figure 2: KeyCloak page highlighting where the dropdown menu is to change Realms

4. Next in the navigation window on the left select ‘*Users’*



Welcome to
UCPACT-Realm

If you want to leave this page and manage this realm, please click the corresponding menu items in the left navigation bar.

Figure 3: Page showing where to select the Users on the navigation menu

5. Then on the page click ‘*Add User’*

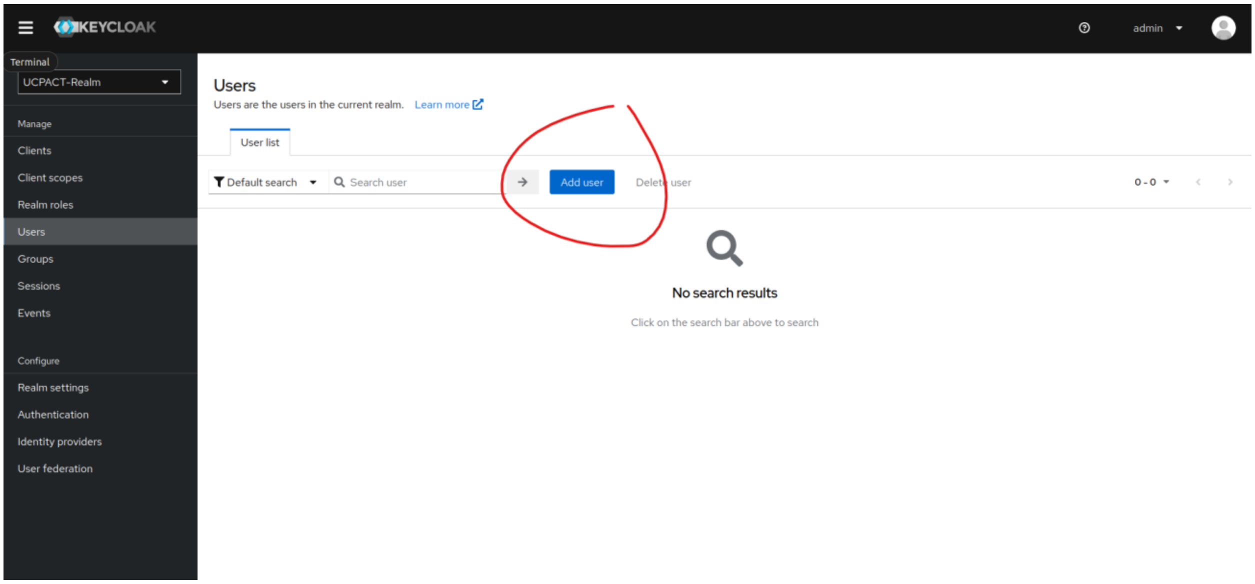
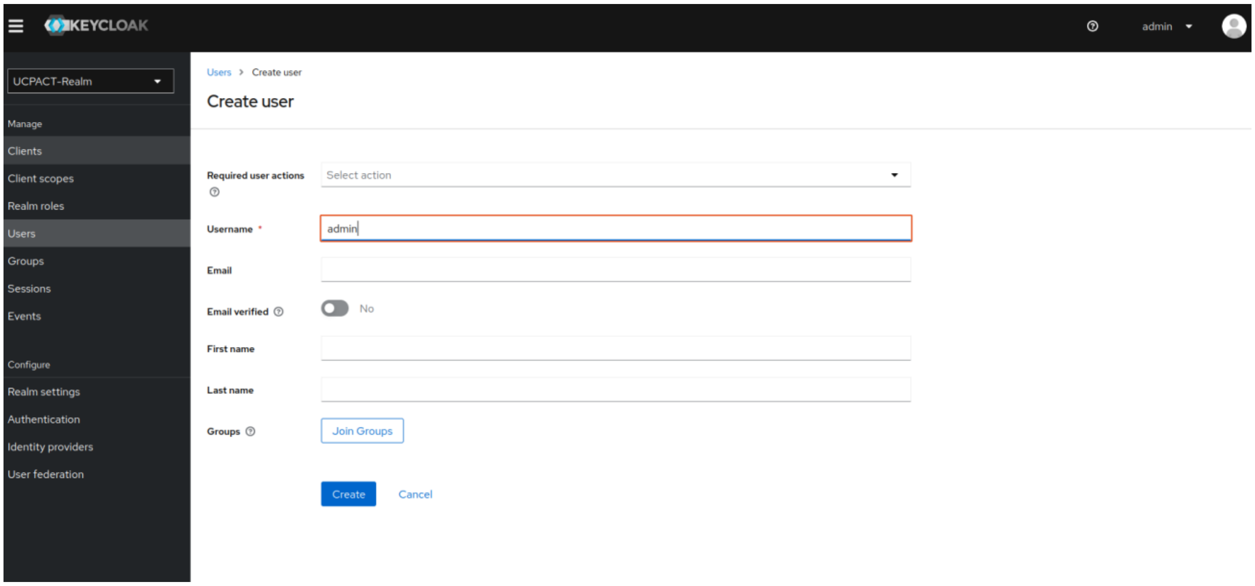


Figure 4: Shows where the Add User button is on the Users tab

6. Fill out the information you need on the page (only the username is required) and click ‘*Create’* when you are done.

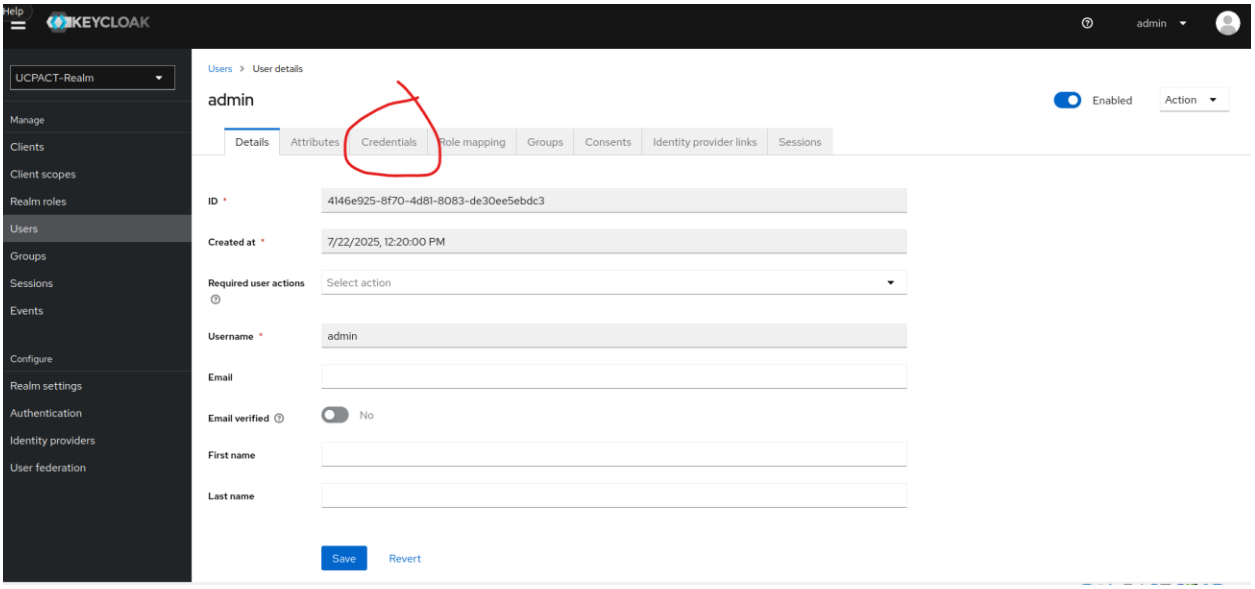


Username * admin

Email verified No

Figure 5: A view of the Create User page with the username filled out

7. After clicking create you should then be navigated automatically to that user’s details. In there we will be clicking on ‘*Credentials’*



ID * 4146e925-8f70-4d81-8083-de30ee5ebdc3

Created at * 7/22/2025, 12:20:00 PM

Required user actions: Select action

Username * admin

Email

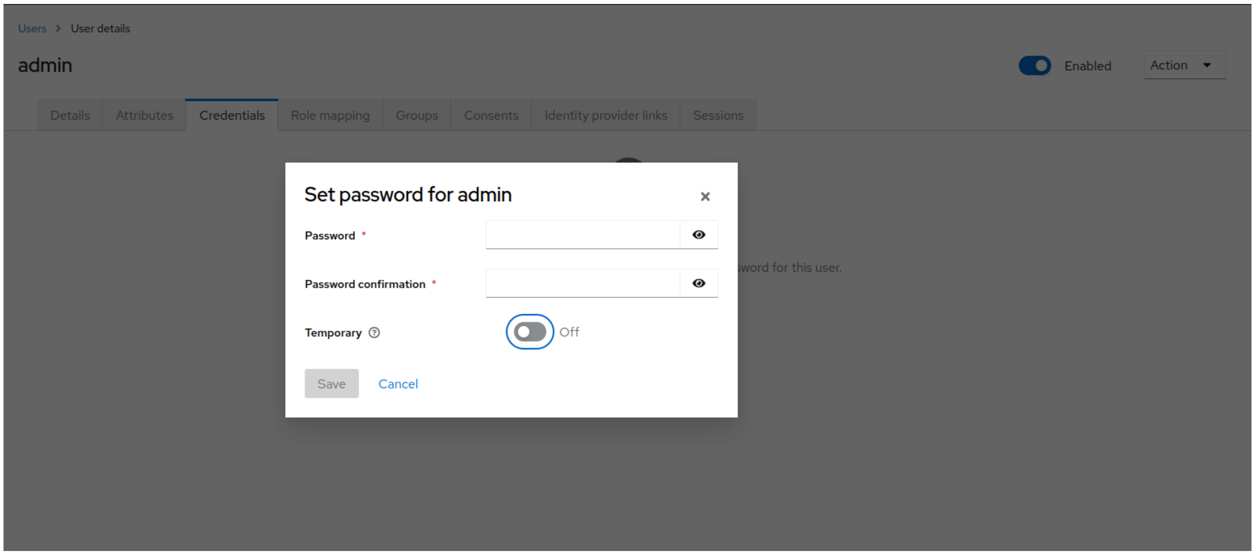
Email verified: No

First name

Last name

Figure 6: Highlights the Credentials tab so that the user's password can be set

8. We will now click on Set password for this user. You can make it temporary or permanent.



Set password for admin

Password *

Password confirmation *

Temporary: Off

Figure 7: Shows the modal after clicking Set Password on the page

9. After setting up KeyCloak, navigate to http://localhost/. However, depending on your system set up you may need to use http://localhost:3000/.
10. You should see the screen below when you navigate to the web application and you log in with the set-up username and password

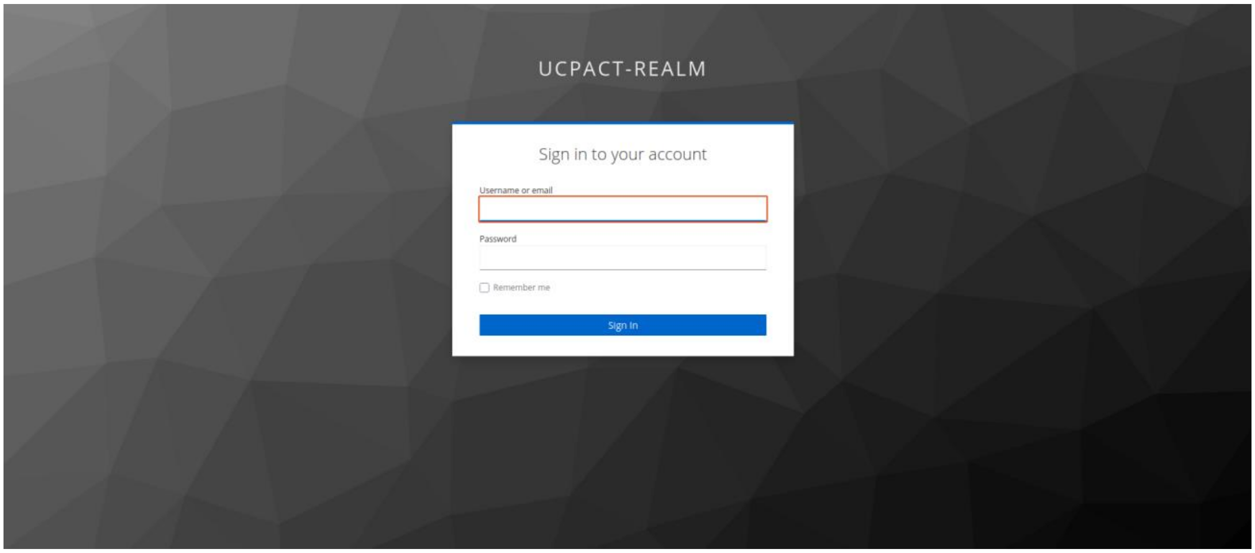
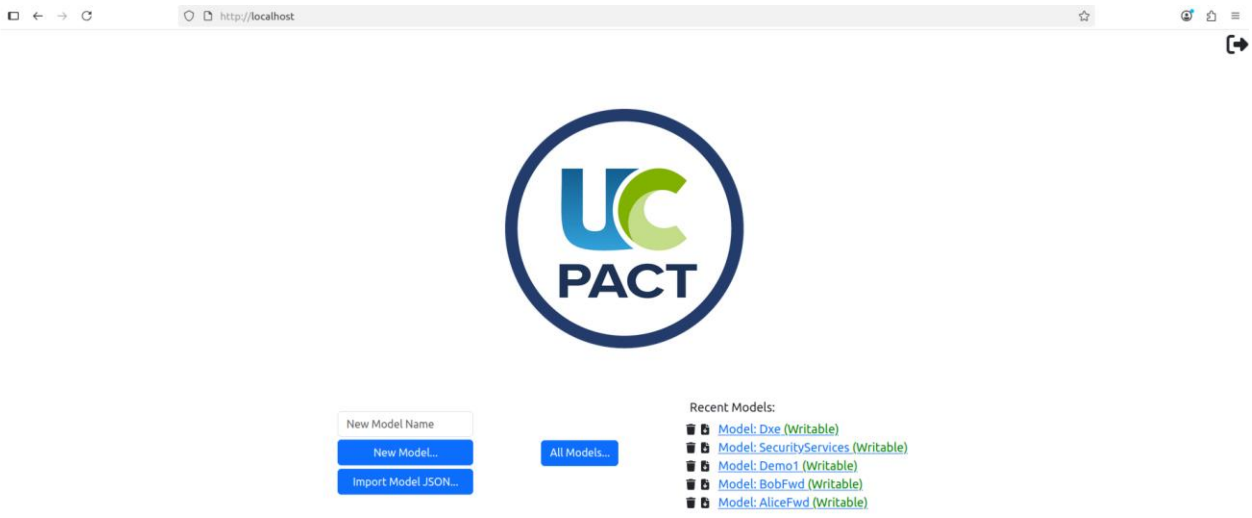


Figure 8: Login page for the UCPACT web application

On a successful login you should see this screen with the recent models being blank or having models that you put in the top-level 'models' directory.



Recent Models:
- Model: Dxe (Writable)
- Model: SecurityServices (Writable)
- Model: Demo1 (Writable)
- Model: BobFwd (Writable)
- Model: AliceFwd (Writable)

Figure 9: Home Page of the UCPACT Web Application

3 Home Page

Now that we have set up the application to be running, we are taken to the home page of the application shown in Figure 9. This is where we can create, load, import, delete, and export web application models. Throughout this section we will be going through all the functionality that the home page contains starting with creating a new model.

3.1 New Model Creation

For creating a new model, the steps are quite simple. You will need to type in a name into the text box shown in Figure 10. After you have chosen a name, you will then click on the "*New Model…"* button which will then start setting up the web application. The model will not be created if the name is already in use or if the name does not follow the naming conventions.

Naming Convention Requirements for a Model name.

- Name does not start with UC_
- Name does not have a double "_"
- Names are alphanumeric with the addition of _
- The name does not end with an _
- The first character must be a capitalized letter [A-Z]
- The File's name is not used by another file in the storage directory

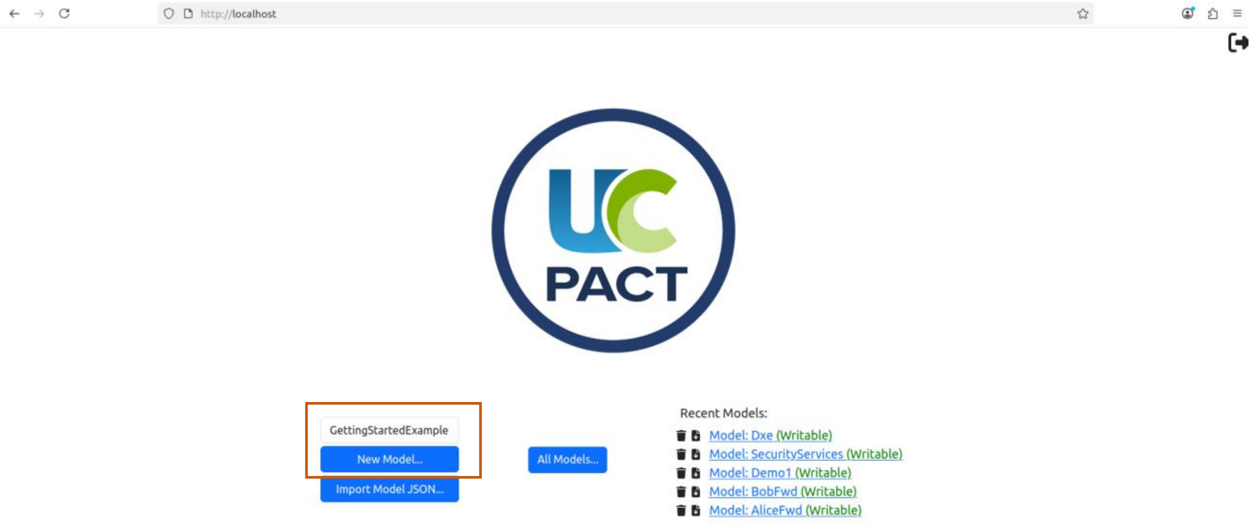


Figure 10: How to Create a New Model

After we create the model, you will be pulled to the Real World tab of the model you just created. This is the main page where you will be creating your UC model image. In Figure 11 you can see the name of the model you are working on in the URL of the page. It is good to check here to ensure you are working on the right model. The URL will not change as you click between tabs unless you go back to the home page of the web application.

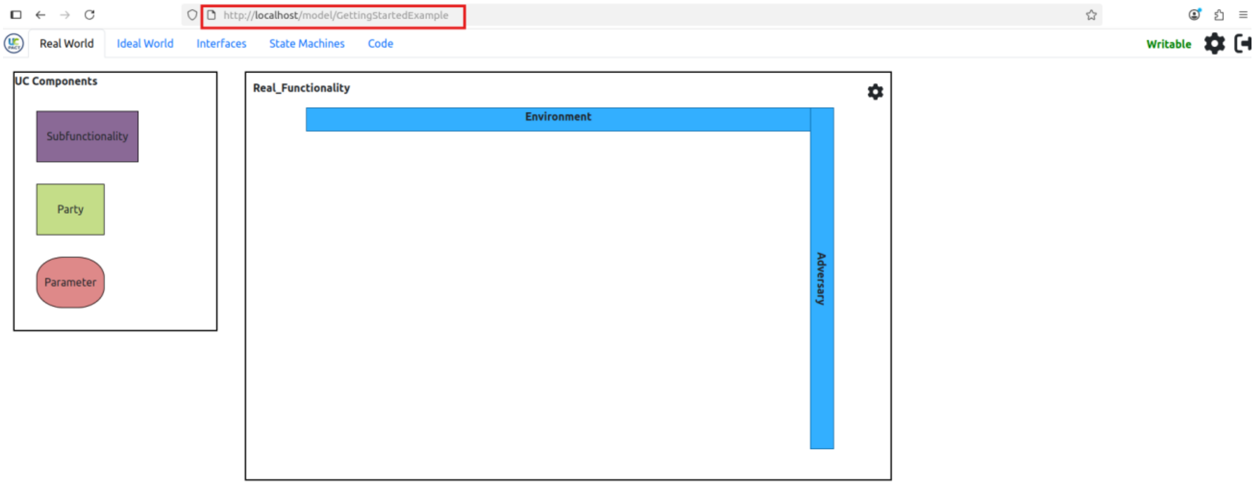


Figure 11: Shows the URL with the model's name in the URL path

3.2 Import a previously created model

It is also possible to import a model json file that someone else has created. To do this you will click on the *'Import Model JSON'* shown in Figure 12. When you click the button, it will open up the modal shown in Figure 13.

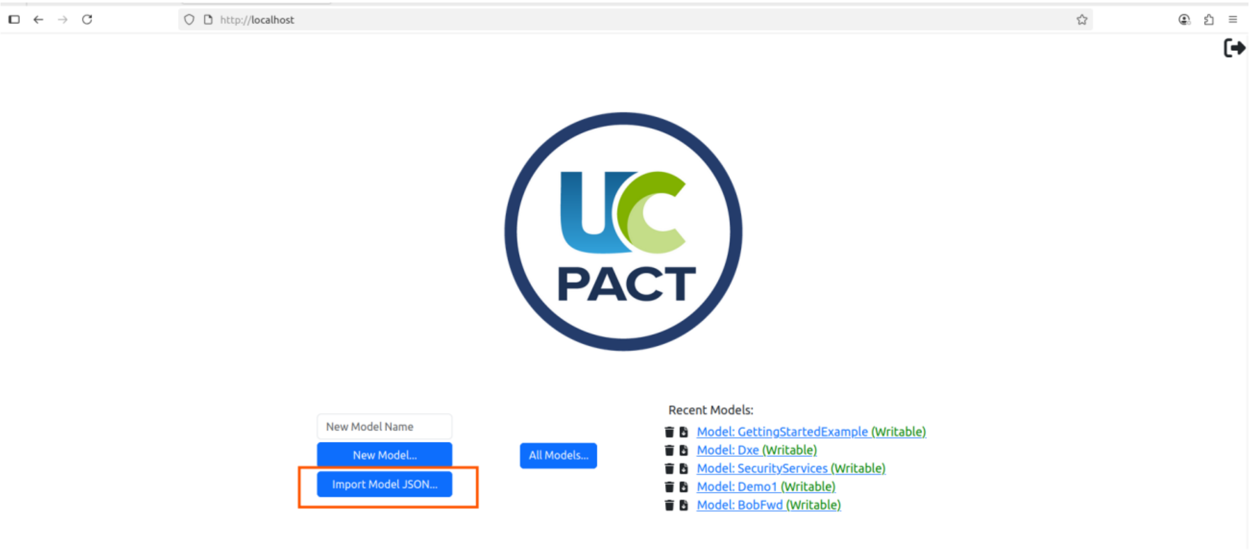
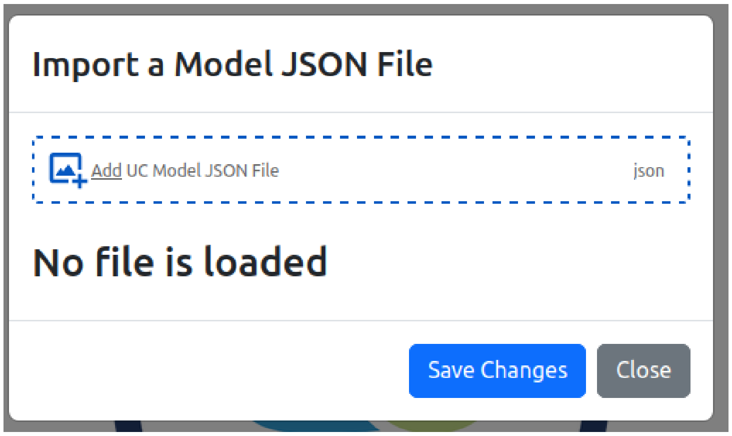


Figure 12: Highlighting the Import Model JSON button

Once the modal is open you then will be able to import the desired model file through clicking on the Add UC Model JSON file and searching your file system for the model file. This can also be accomplished by dragging the desired UC JSON Model over the outlined box. Once that is complete you will see the “No File is Loaded” message change to your file is currently loaded. After you see that message then you will click *‘Save Changes’* to complete the import process. To ensure that model names are unique in this case we attached a date and time of upload. This name can be changed once you open the file in the application which we show in section 4.3 of this document.

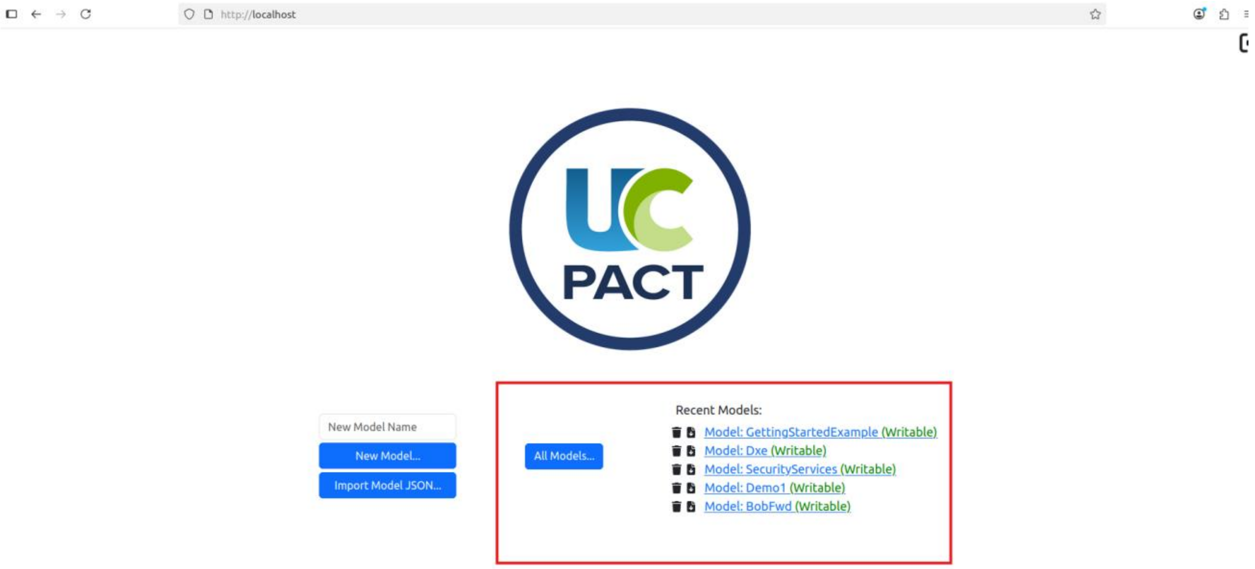


Import a Model JSON File

No file is loaded

Figure 13: Import Model JSON Modal

3.3 Loading a model on the system



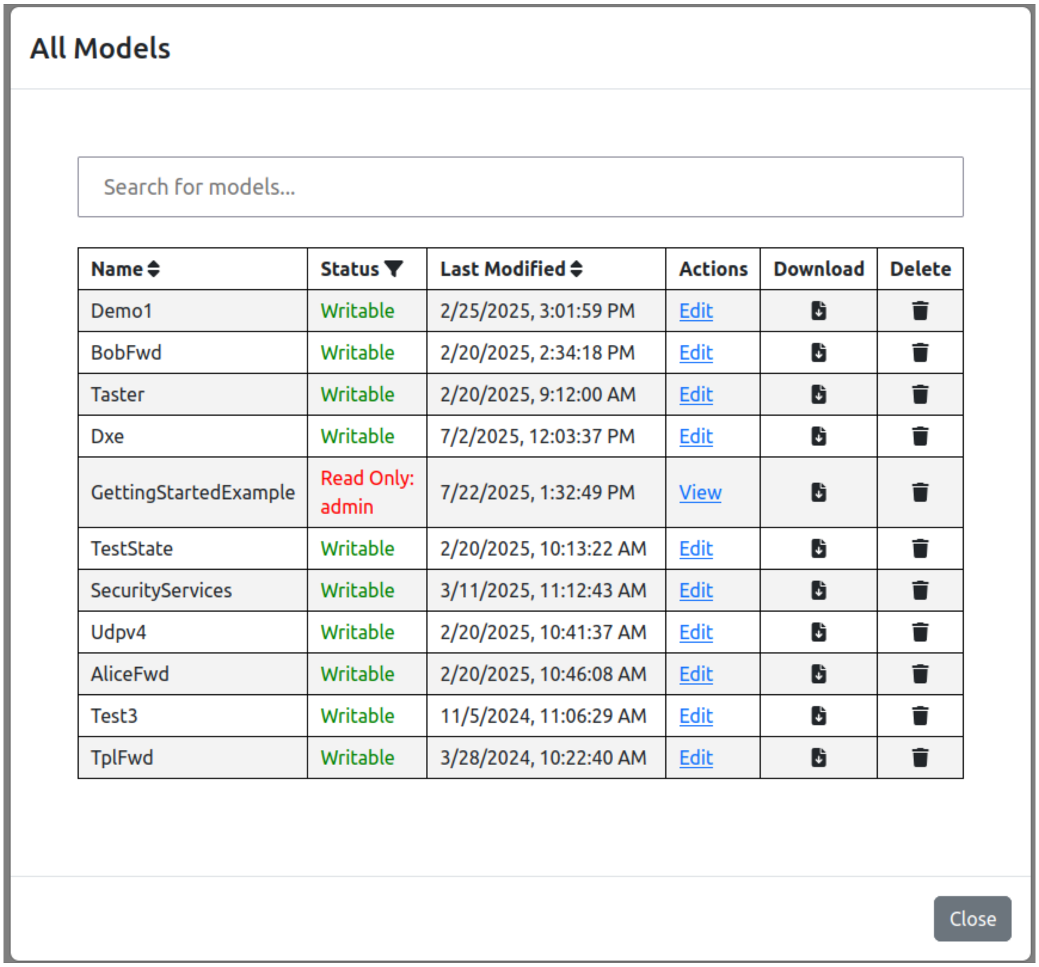
Recent Models:

- Model: GettingStartedExample (Writable)
- Model: Dxe (Writable)
- Model: SecurityServices (Writable)
- Model: Demo1 (Writable)
- Model: BobFwd (Writable)

Figure 14: Highlight of the Recent Models list and All Models button

After you start creating and importing your own models you will then have models to load from the Recent Models area and the All Models are shown in Figure 14. The Recent Models list shows the five most recent models loaded in the system. If you click on the name of the model in Blue from the picture it will open the model in the application. To the right of the name is text that will either say writable in green or read-only in red this is set based off of if someone is already accessing the file in write mode which we will go into further detail in section 4.2. Next to the left you can see two icons. The farthest left icon allows you to delete a model from the system when pressed. The icon next to it on the right will allow you to download the JSON version of the models.

As for the All Models button it has very similar functionality to the Recent Models list but contains all the models that are available to open. You can search for a model by name or sort the list of models by Name or the Last Modified tag. To open these models you will have to click on the actions which will either be "Edit" or "View" depending on whether it is writable. You also have the buttons to download or delete the models in the modal as well.



Name ⬍	Status ▼	Last Modified ⬍	Actions	Download	Delete
Demo1	Writable	2/25/2025, 3:01:59 PM	Edit	🗎	🗑
BobFwd	Writable	2/20/2025, 2:34:18 PM	Edit	🗎	🗑
Taster	Writable	2/20/2025, 9:12:00 AM	Edit	🗎	🗑
Dxe	Writable	7/2/2025, 12:03:37 PM	Edit	🗎	🗑
GettingStartedExample	Read Only: admin	7/22/2025, 1:32:49 PM	View	🗎	🗑
TestState	Writable	2/20/2025, 10:13:22 AM	Edit	🗎	🗑
SecurityServices	Writable	3/11/2025, 11:12:43 AM	Edit	🗎	🗑
Udpv4	Writable	2/20/2025, 10:41:37 AM	Edit	🗎	🗑
AliceFwd	Writable	2/20/2025, 10:46:08 AM	Edit	🗎	🗑
Test3	Writable	11/5/2024, 11:06:29 AM	Edit	🗎	🗑
TplFwd	Writable	3/28/2024, 10:22:40 AM	Edit	🗎	🗑

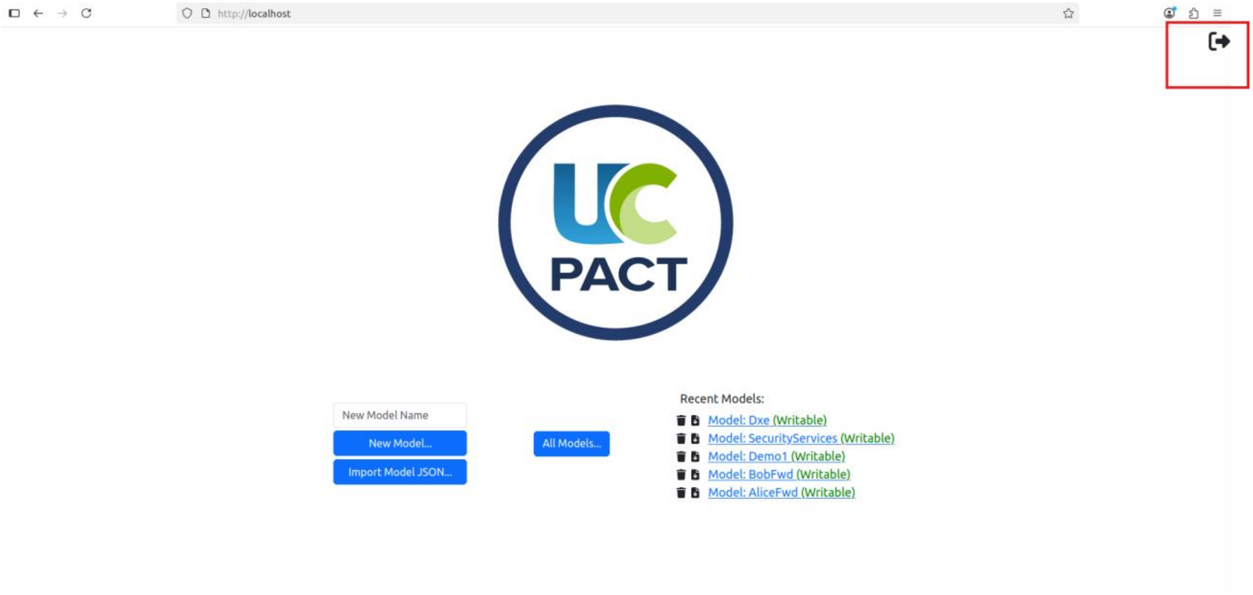
Figure 15: All Models Modal

4 Common Features

There are several features that are common throughout the usage of this web application. This section is focused on describing these features which are mostly for Multi-user mode users. These features include how to log out of KeyCloak, Read and Write Permissions for Models, and how to change the name and return a model to be writable again.

4.1 Log Out of KeyCloak (Multi-user mode only)

For Multi-user mode only if you are done using the system or want to logout it is quite easy. In the top right of any page you will see a bracket with an arrow, see Figure 16 for the image, clicking this will log you out of KeyCloak and send you back to the KeyCloak login page.



Recent Models:

- Model: Dxe (Writable)
- Model: SecurityServices (Writable)
- Model: Demo1 (Writable)
- Model: BobFwd (Writable)
- Model: AliceFwd (Writable)

Figure 16: Highlighting the Logout Button

4.2 Model Read/Write Permissions

Read and Write Permissions for files are only relevant when the application is running in the Multi-user mode. This is implemented to prevent conflicts and model corruption when multiple users have a model file open. In Figure 15Figure 14, the column for status shows whether a model is in a writable state. This status displays Read-Only if that model is open by another person who is making changes to the models. It will be in Writable if the model is not open. While the model is in Read-Only mode you will still be able to view it but none of your changes will be saved if you make any. An indicator for the model being open in Read-Only mode shown in the top right of Figure 17 or that it is Writable is shown in Figure 18.

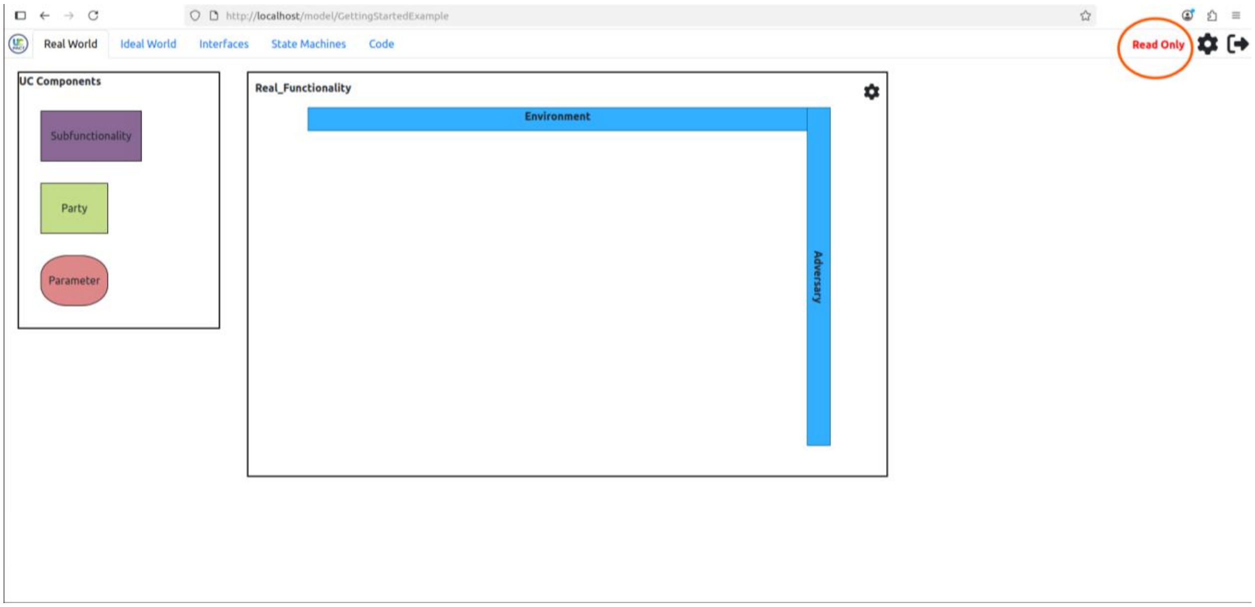


Figure 17: Model open in Read Only Mode

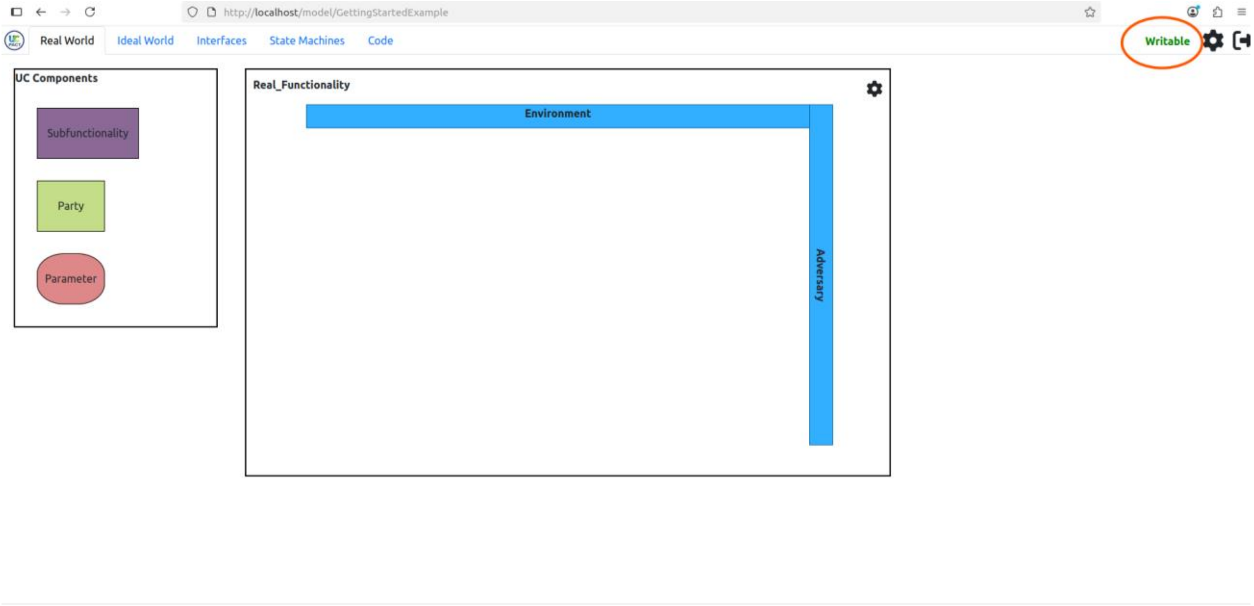


Figure 18: Model Open in Writable mode

4.3 Changing the model's name and returning the model

To change the model's name and return a model that you have opened in the web application, you will need to click on the gear at the top right of the web page shown in Figure 19. That will open the modal shown in Figure 20, which has a text box with the model's name and a button to return the model to be writable to others and then close the model. When you decide to change the name for the name to stick you will have to press save changes otherwise any changes you make will not be saved.

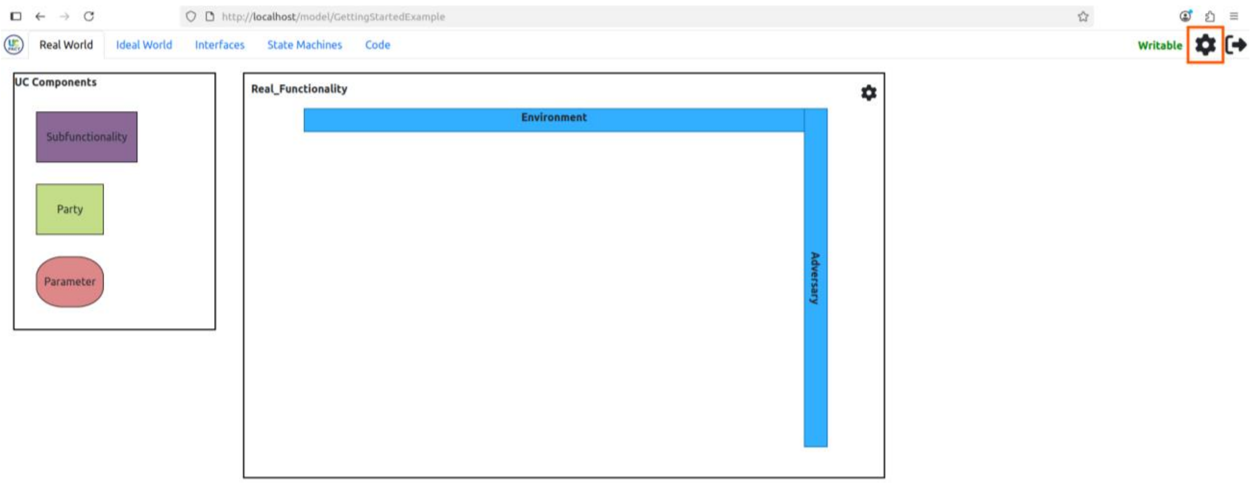
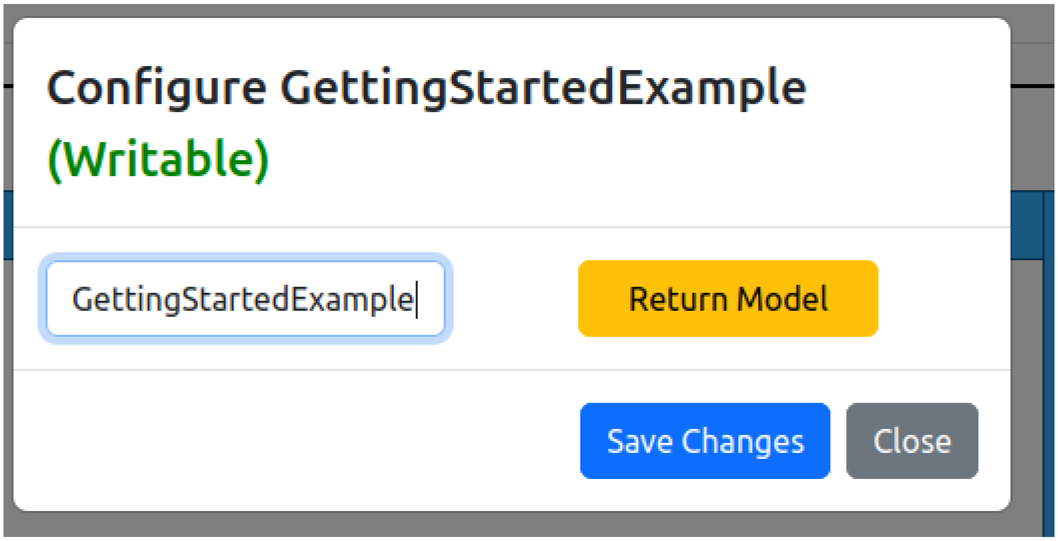


Figure 19: Gear to access Model's name and button to return the model



Configure GettingStartedExample (Writable)

GettingStartedExample

Figure 20: Modal that contains the Return Model button and text box to change the Model's name

5 Real Functionality Tab

The Real Functionality Tab is where we draw the main view for the UC model. It has five entities that are shown in Figure 21, which are Parties. SubFunctionalities, Parameter Interfaces, the Environment, and the Adversary. These pieces come together to form a Real World system that can then be formally checked in the UC Framework. Our Web Application lets you layout these pieces in the system and see which pieces are talking between each other. In this section we will be diving deeper into the UC components and how they are used and when you might use them. We will also talk about the

Real_Functionality options menus and what the Environment and Adversary are in the Real Functionality section.

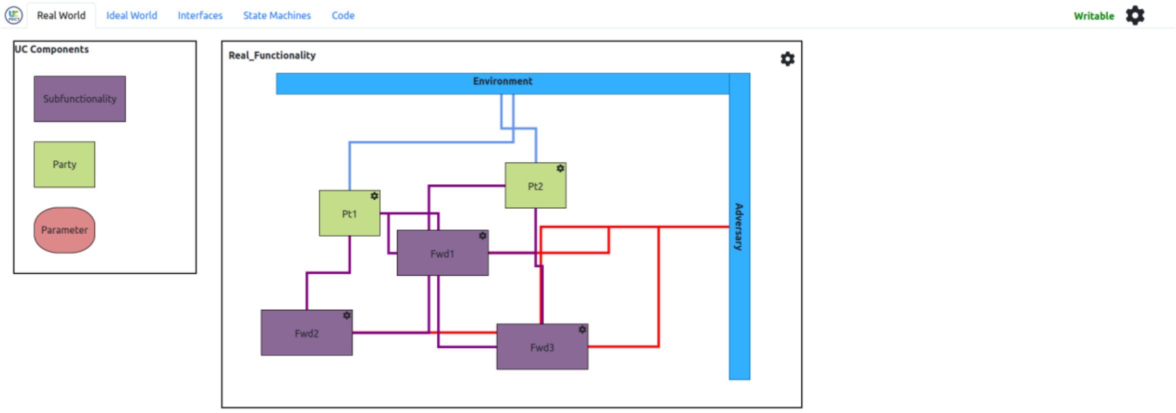


Figure 21:Real Functionality Page

5.1 UC Components

This section goes over each of the three UC Components that are used to make the UC Model picture. The components in question are shown in Figure 22 where this appears on the left side of the Real Functionality tab. Each of these components can be dragged over into the Real Functionality box which makes a new instance of them which can then be edited for its purpose in the model.

There are some distinguishers between the three components and we will give a brief description of those differences. Parties are what contain the messages and interactions that are important and used by the system being modeled. A subfunctionality takes the ideal portion of another model, its simplest version, and brings that functionality into the current model that is being developed. Subfunctionalities are typically intended for things that you don't plan on ever building a real world model for. Finally, we have Parameter Interfaces, or called "Parameter" in Figure 22, bring in the Composite Direct Interface of another model which for the DSL is another ideal functionality from a different UC model. The Parameter Interface can also be interpreted using either their Real World or Ideal World functionality when using the UC Interpreter tool. This brings in the functionality of that model but does not use the simplified model code like a subfunctionality does. The important things to note are that Parties are for functions specific to the system you are modeling. Subfunctionalities are for bringing in ideal models of functions that are needed for your model but are not the focus of your model. Parameter Interfaces are for binging in functionalities from other models that are not the focus of the current model but play a role in the current model. They can be interpreted in the UC Interpreter tool as their Ideal or Real World.

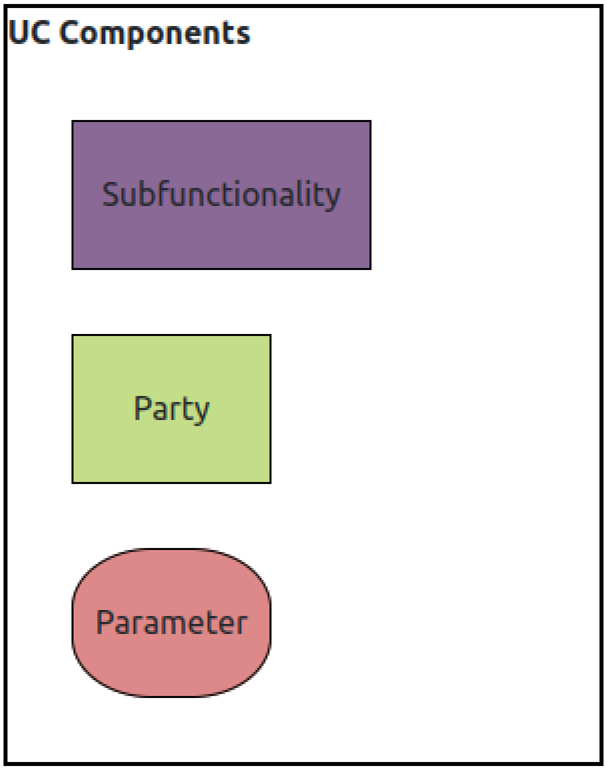
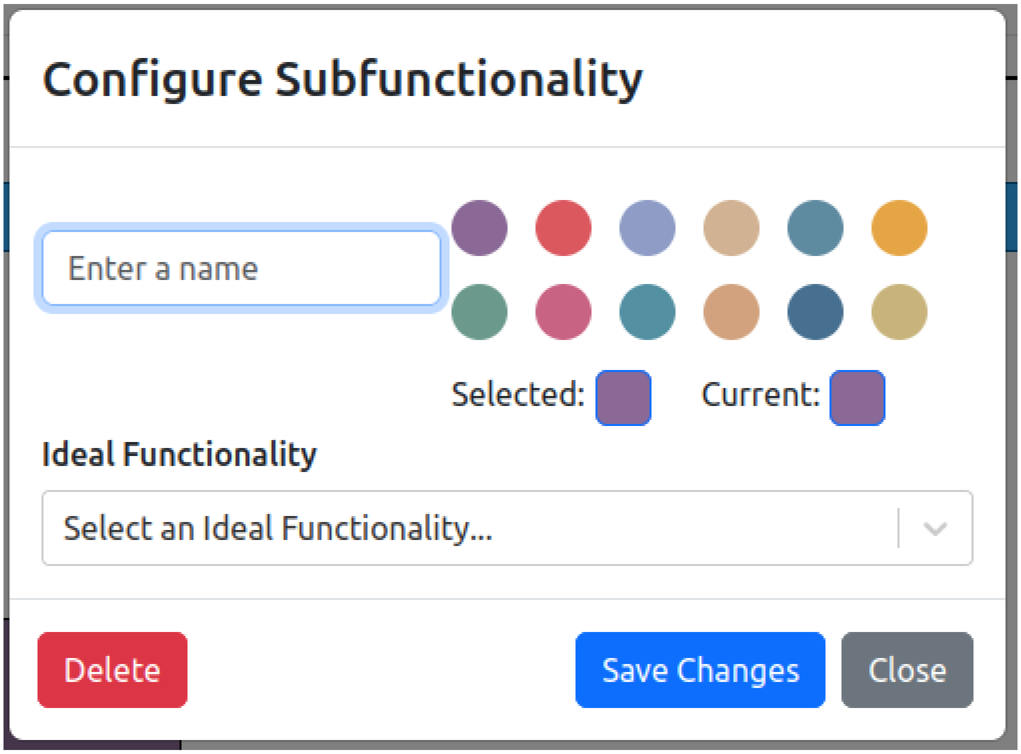


Figure 22: UC Components box with Subfunctionality, Party, and Parameter drag objects. Where Parameter is short for a Parameter Interface.

5.1.1 Subfunctionalities

To add a subfunctionality to your model you will need to click and drag the purple rectangle from the UC Component box to the Real_Functionality box. Once this is dropped in the Real_Functionality it will open the modal shown in Figure 23 which has a default title of “Configure Subfunctionality”. In here you will be able to enter a name into the text box on the left of the modal. On the right we have 12 colors that you can use to change the coloring of the Subfunctionality box as desired. It will default to the purple color you initially dropped but as you click between the different circles you will see the box labeled “Selected” will update to the current color you have clicked through. Finally, for a subfunctionality we have a selection of ideal functionality. This dropdown menu goes through other models that are in the model folder and will pull in their ideal functionality data to be used for the current model. After all this is selected you can save your changes by hitting save changes. To go back to this screen after you have saved your changes or closed that modal you can click on the gear icon on the component to access this modal. This is also how you delete the component from the model as well by clicking the red Delete button on the modal.


Configure Subfunctionality

Enter a name

Selected: Current:

Ideal Functionality

Select an Ideal Functionality...

Figure 23: Configuring Modal for a SubFunctionality

5.1.2 Parties

To add a party to your model you will need to click and drag the green rectangle from the UC Component box to the Real_Functionality box. Once this is dropped in the Real_Functionality it will open the modal shown in Figure 24 which has a default title of "Configure Party". In here you will be able to enter a name into the text box on the left of the modal. On the right we have 12 colors that you can use to change the coloring of the Party box as desired. It will default to the green color of the box you initially dropped but as you click between the different circles you will see the box labeled "Selected" will update to show the current color you have clicked through. For a party we have two dropdown menus, one for a basic direct interface and one for a basic adversarial interface. These are interfaces that are found in the Composite interface you set up and put into the Real Functionality which we will talk about further in section 5.2. After all of this is selected you can save the changes you have made by hitting save changes. To go back to this screen after you have saved your changes or closed that modal you can click on the gear icon on the component to access this modal. This is also how you delete the component from the model as well by clicking the red Delete button on the modal.

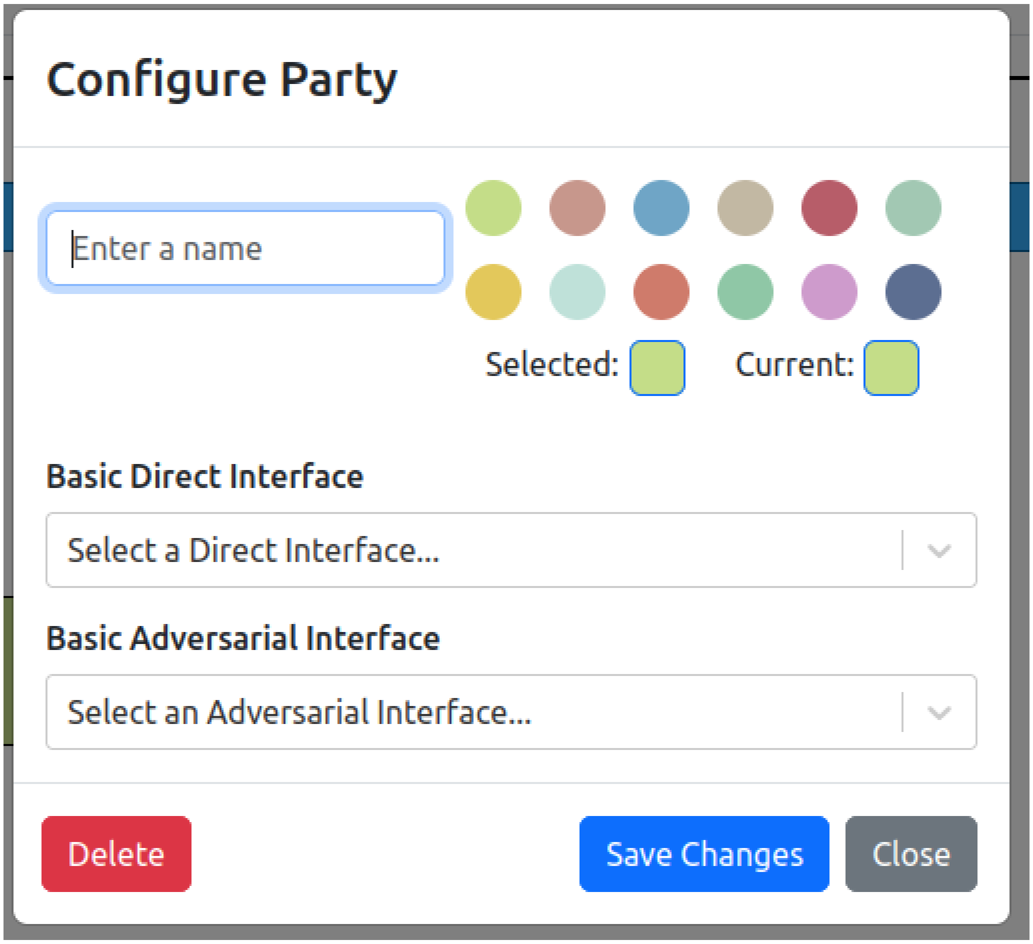
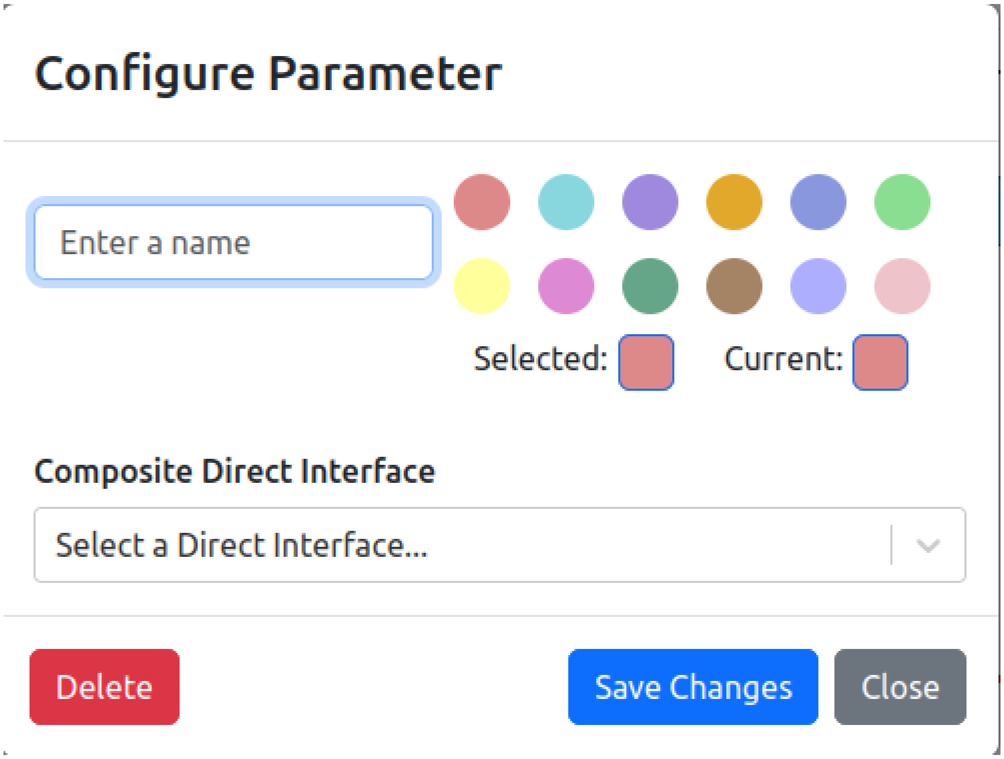

Configure Party
Enter a name
Selected:
Current:
Basic Direct Interface
Select a Direct Interface...
Basic Adversarial Interface
Select an Adversarial Interface...
Delete
Save Changes
Close

Figure 24: Configuration Modal for a Party

5.1.3 Parameter Interfaces

To add a Parameter Interface to your model you will need to click and drag the red curved rectangle from the UC Component box to the Real_Functionality box. Once this is dropped in the Real_Functionality it will open the modal shown in Figure 25Figure 24 which has a default title of "Configure Parameter". In here you will be able to enter a name into the text box on the left of the modal. On the right we have 12 colors that you can use to change the coloring of the Parameter Interface box as desired. It will default to the red color of the box you initially dropped but as you click between the different circles you will see the box labeled "Selected" will update to show the current color you have clicked through. For a parameter interface we have a single dropdown menu that will take a Composite Direct Interface from a different model. After all of this is selected you can save the changes you have made by hitting save changes. To go back to this screen after you have saved your changes or closed that modal you can click on the gear icon on the component to access this modal. This is also how you delete the component from the model as well by clicking the red Delete button on the modal.



Configure Parameter

Composite Direct Interface

Figure 25: Configuration Modal of a Parameter Interface

5.2 Real Functionality

For the Real Functionality Box we have 3 features that we will go through. First is the Environment bar that is shown as the horizontal blue bar in Figure 26 represents the environment of the model that the system will be given initial instructions from and send information out to when the system has information to send. The other blue bar that is vertical is the adversary. The adversary is a piece of the UC Model that acts as the entity that is trying to retrieve information from the system and is trying to break the model with its set of activities that it is allowed to perform. We give the adversary capabilities through the adversarial interfaces we create as to what the adversary has access to as we perform the functionality of the system. For these two boxes as we fill out information in the various UC Components lines will be drawn between each of the boxes and the respective UC Component to mark that they are connected and messages are flowing between those entities.

We also have the gear icon in the top right of the box, which does more general set up for the Real Functionality. If you click on the icon you will see the modal pop up from Figure 27. We have three parts where you can change the name of the Real_Functionality box as you would like by replacing the name in the textbox. You can also add the Composite interfaces for the Real Functionality which contain the basic interfaces that are used to populate the various parties' interfaces. The Composite interfaces, both Direct and Adversarial, are selectable from the Composite Interfaces that are available from the

Interfaces tab. Finally after you have selected the right interfaces to use you will then need to click save changes for the changes to take effect.

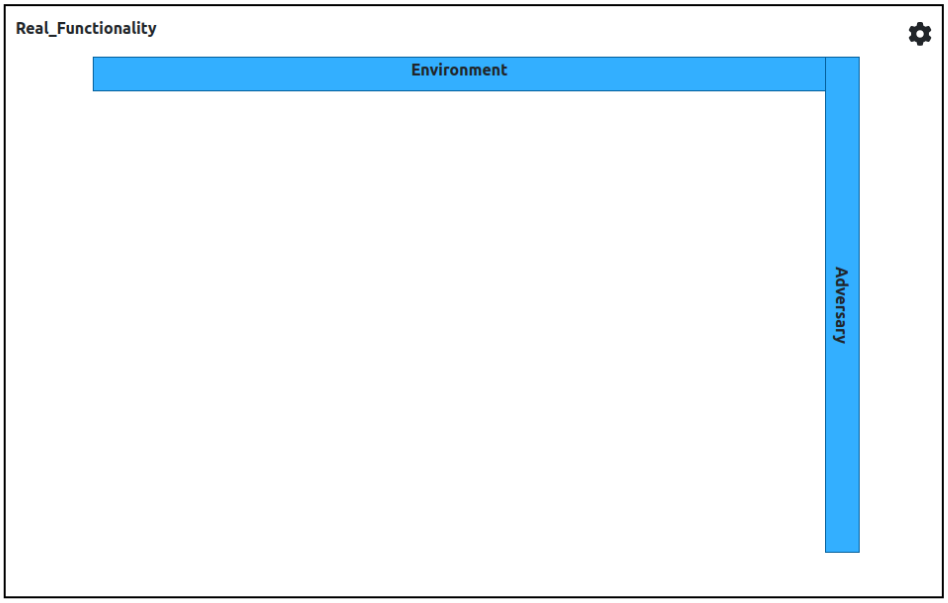
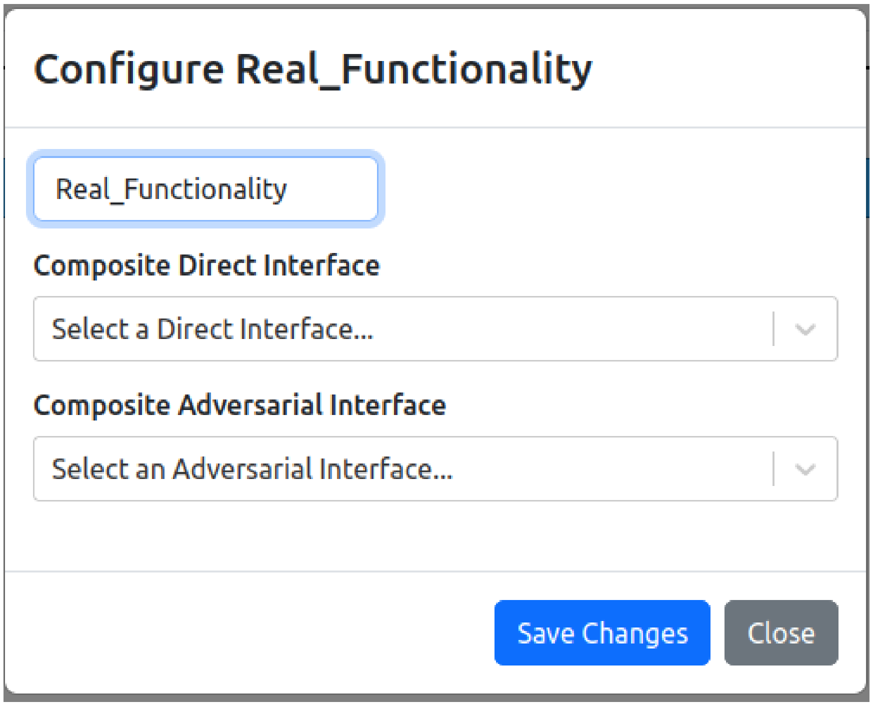


Figure 26: The Real Functionality Box


Configure Real_Functionality

Real_Functionality

Composite Direct Interface

Select a Direct Interface...

Composite Adversarial Interface

Select an Adversarial Interface...

Figure 27: Real Functionality Configuration Modal

6 Ideal Functionality Tab

The Ideal Functionality tab has four components of concern that we will talk about. First is the Environment bar that is shown as the horizontal blue bar in Figure 28 represents the environment of the model that the system will be given initial instructions from and send information out to when the system has information to send. In the Ideal World only the Ideal Functionality, listed as IF in Figure 28, interacts with the environment. The other blue bar that is vertical is the adversary. The adversary is a piece of the UC Model that acts as the entity that is trying to retrieve information from the system and is trying to break the model with its set of activities that it is allowed to perform. We give the adversary capabilities through the adversarial interfaces we create as to what the adversary has access to as we perform the functionality of the system. In the Ideal World case only, the simulator, labeled as Sim in Figure 28, interacts with the adversary. In the next two sections we will dive deeper into the Ideal Functionality and the Simulator

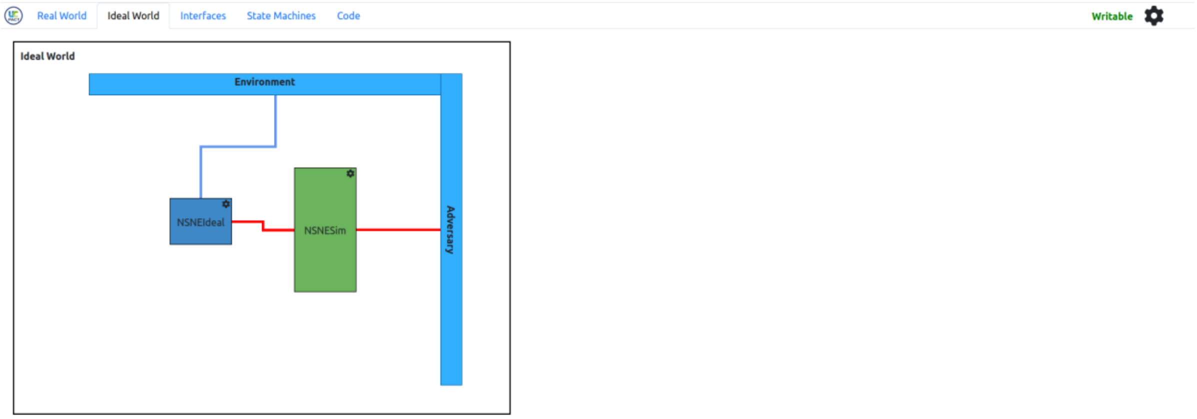
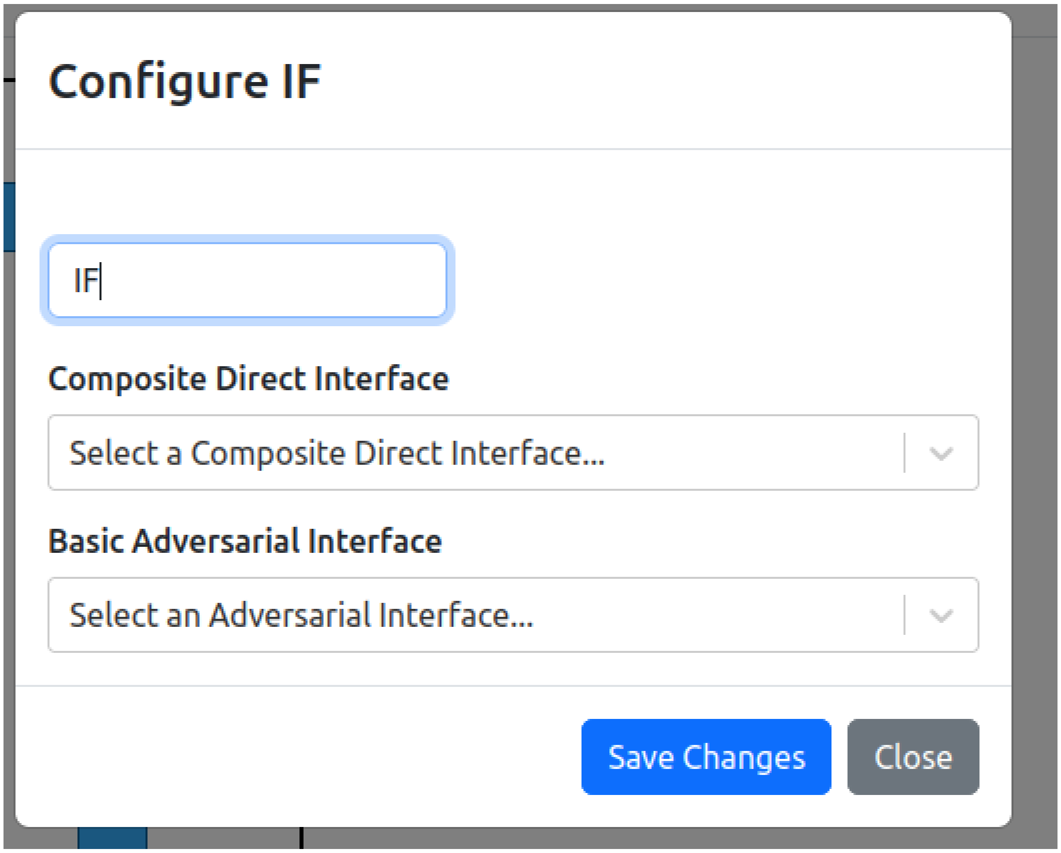


Figure 28: Ideal Functionality Page

6.1 Ideal Functionality

The Ideal Functionality is the simplest form of the process that the system being modeled is to still perform the functions of that system. In Figure 29, we can see how we can configure the Ideal functionality by clicking on the gear icon on the IF box in Figure 28. First, we have a name we can change, if we would like, using the textbox. Next, the Ideal functionality has a dropdown menu for a Direct Composite Interface. It typically utilizes the same Composite Direct Interface as that of the Real Functionality as the calls the system makes should be the same from the environment to the Ideal Functionality. The goal of the Ideal World is to mimic the functionality of the Real World and be as simple as possible. The ideal functionality also has a dropdown menu for a Basic adversarial interface. This interface is separate from the Composite Adversarial Interface that we specified in the Real world. This interface is to drive the commands between the Ideal Functionality and the Simulator which lets the

Simulator simulate certain aspects about the Real Functionality that the Ideal functionality does not do itself. To save the changes that we have made we need to hit the save changes button.



Configure IF

IF

Composite Direct Interface

Select a Composite Direct Interface...

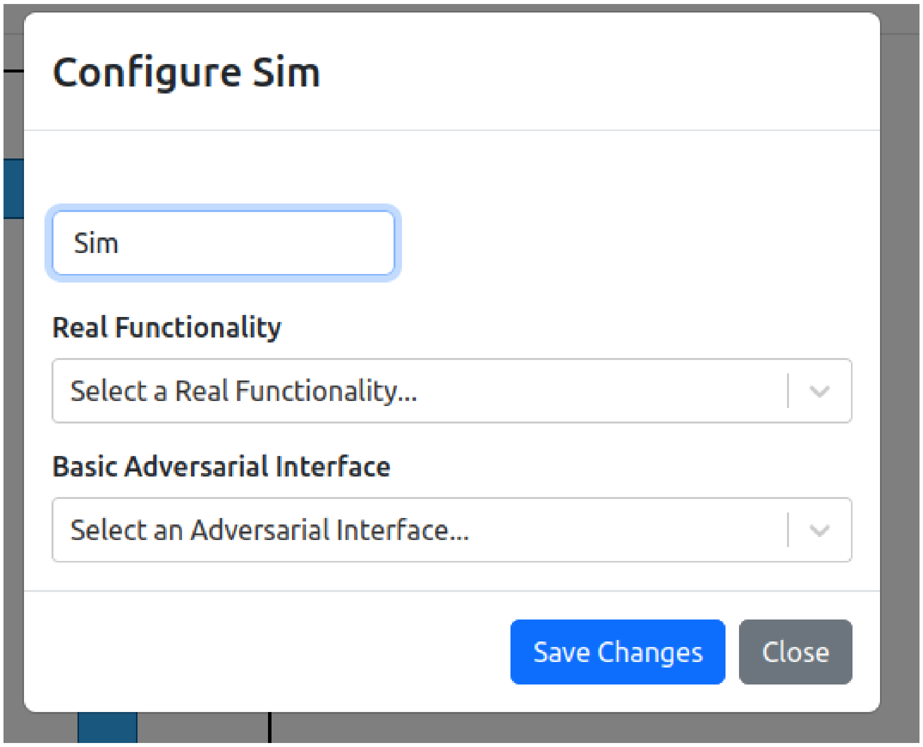
Basic Adversarial Interface

Select an Adversarial Interface...

Figure 29: Ideal Functionality Configuration

6.2 Simulator

The Simulator is a function in the Ideal World that simulates all the extra pieces of the Real World that the ideal functionality does not handle. This is essential to ensure that the Adversary and Environment cannot tell the difference between the Real and Ideal worlds. To do this we can configure the simulator by clicking on the gear icon of the Sim box in Figure 28 and then that will open the Simulator Configuration box shown in Figure 30. There are three pieces that you can modify for the simulator. First is its name by typing in a new name in the shown textbox. Second, we have a dropdown menu to select the Real Functionality that the simulator is simulating. This is typically the Real World we have defined in the Real World tab. This is important because this contains the interfaces needed by the simulator to talk appropriately to the Adversary if these do not align, we will have distinct differences between our two worlds. Finally, we have another basic adversarial interface that talks between the Simulator and the Ideal world to pass information between the two components. To save the changes that we have made we need to hit the save changes button.



Configure Sim

Sim

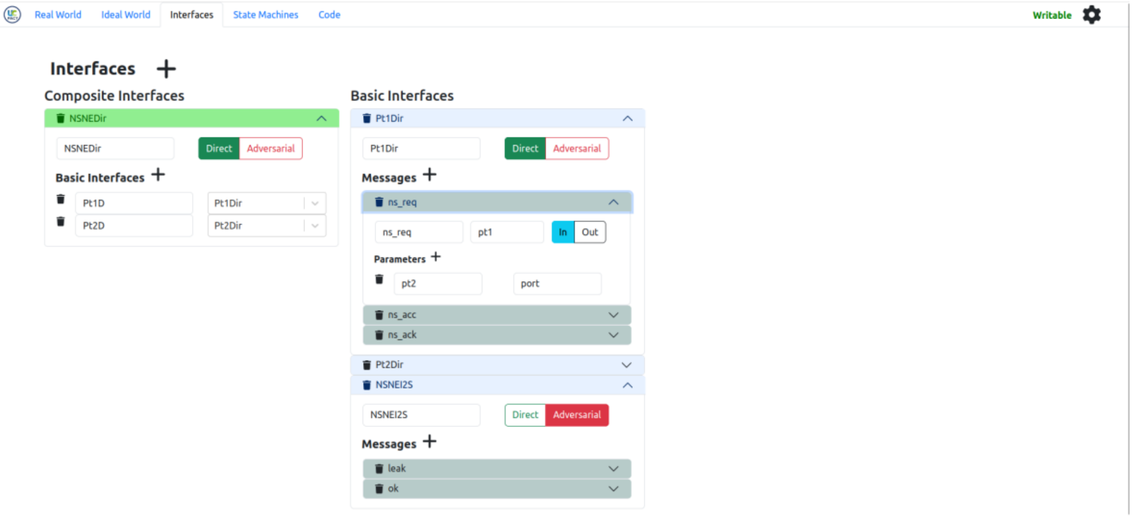
Real Functionality: Select a Real Functionality...

Basic Adversarial Interface: Select an Adversarial Interface...

Figure 30: Simulator Configuration

7 Interfaces Tab

The interfaces tab is the section where all the data and messages get defined for the model.



NSNEDir

NSNEDir [✓ Direct] Adversarial

Basic Interfaces +

Pt1D Pt1Dir

Pt2D Pt2Dir

Pt1Dir

Pt1Dir [✓ Direct] Adversarial

Messages +

ns_req

ns_req pt1 [✓ In] Out

Parameters +

pt2 port

ns_acc

ns_ack

Pt2Dir

NSNEI2S

NSNEI2S Direct [✓ Adversarial]

Messages +

leak

ok

Figure 31: Display of the Interfaces Tab in a New Model

7.1 Adding Interfaces

This section goes into how to add either a Basic or Composite interface. All you will need to do is click the large plus sign shown in Figure 32 which will open a dropdown menu, and you will select either the Basic Interface option or the Composite Interface option. The next two sections will go into what each interface is their structure in the web application and where they will be used throughout the rest of the application and modeling process.

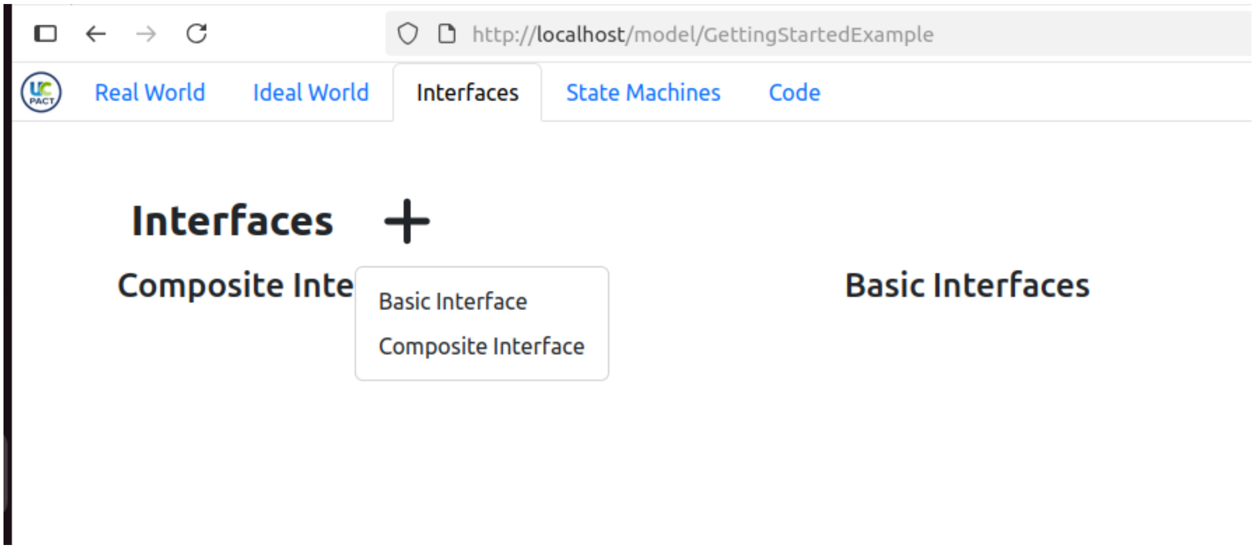


Figure 32: Adding a Basic Interface or Composite Interface

7.2 Composite Interfaces

Composite Interfaces are where we define all the Basic interfaces that are to be used by our parties in the Real functionality. We have two types of Composite interfaces to store our basic interfaces. They are split between "Direct" and "Adversarial" where Direct interfaces supply the functionality of the parties and the Adversarial Interfaces supply the capabilities of our adversary in the model. This is information that the adversary could receive from the system or protocol. Composite interfaces use the Basic Interfaces from the model to assign a different name than the name the Basic interface has.

Below in Figure 33 we can see the different settings for the Composite interface as well as adding basic interfaces to the Composite interfaces. First, we have our name currently it is not filled in, so it has the placeholder of Interface Name. Names for Composite interfaces must be unique between them and Basic Interfaces. The button to the right of the name allows you to switch the type of Composite interface between Direct and Adversarial. Direct is the default value when you first create an interface. Below those two items we have adding a basic interface. To do this we click on the plus sign that is next to the Basic Interfaces text, and it will create a line for a new basic interface to be added to the composite. On the far left is for deleting that line from the Composite interface. To the right of that is for naming the Basic interface that will be used in the Composite. Finally on the right is a dropdown menu that contains all the currently available Basic interfaces. This dropdown menu will only show interfaces that have the same type as the Composite interface. You can have multiple instances of the same basic interface in a composite interface the instance just has to have unique names.

In the next section we will go into what is in a Basic Interface.

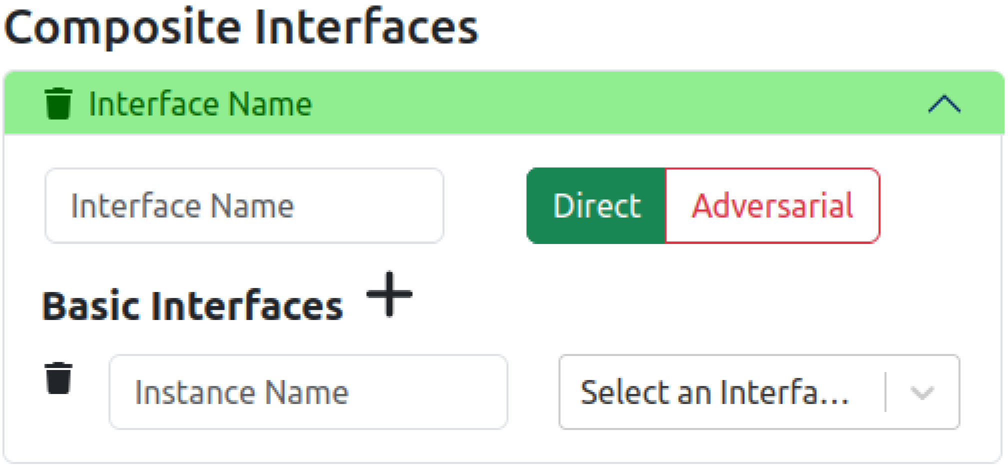


Figure 33: Composite Interface Dropdown Menu

7.3 Basic Interfaces

Basic Interfaces contain all the information for how a party or functionality will operate.

Below in Figure 34 shows the layout of an empty basic interface that has one message with a parameter. For basic interfaces they have the same information at the top level which includes a name textbox. A basic interface must have unique names between other basic interfaces and the composite Interfaces. To the right of the textbox there is the button to switch the interfaces type between “Direct” and “Adversarial”. Direct is the default setting for a new basic interface.

After those settings we then get into the messages of a basic interface. These messages can be added by clicking on the plus sign next to the “Messages” label. In a message we have on the left a Message Name box that contains the name of the message. Message names must be lower case. After the message name on the left we then have a port name that will be filled in. After the port name we have a toggle that determines whether the message is coming into the interface or leaving the interface. Finally, we have the parameters which contain the other data that is being passed along within the message. Each parameter has a name and a type that will need to be filled in. Finally for deleting parameters, messages and the interface itself you would click the trash can icon next to the name of the parameter, message and basic interface as appropriate.

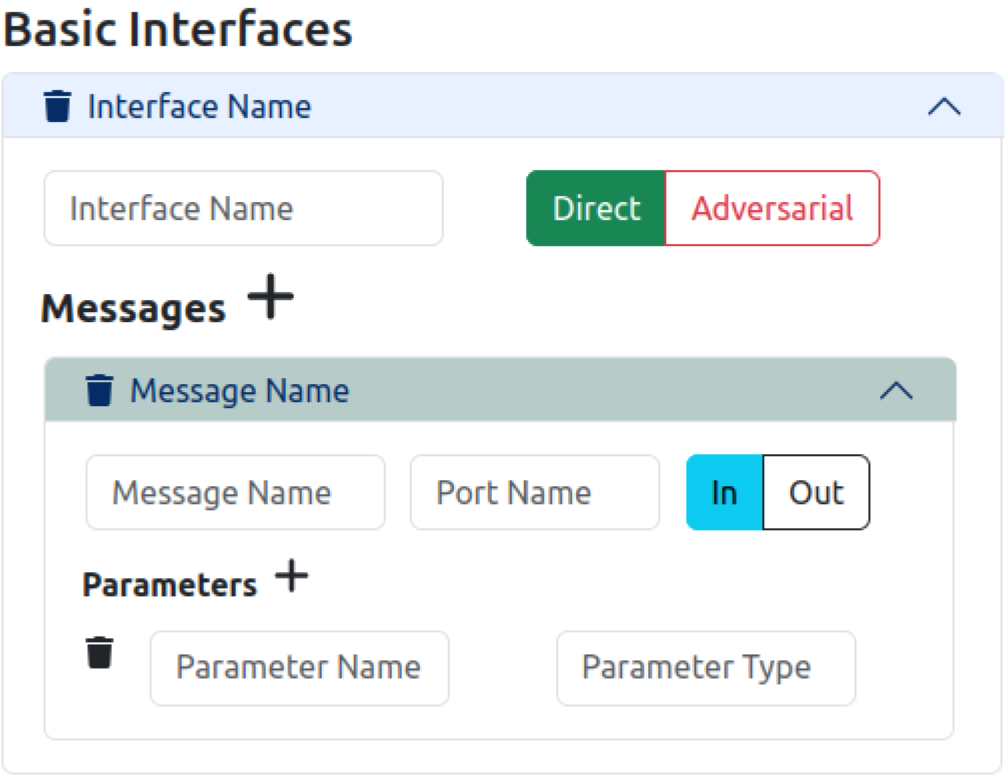


Figure 34: Basic Interface Menu with Messages

8 State Machine Tab

The State Machine tab contains how the parties, ideal functionality, and simulator transition between their different states in the model. Figure 35 shows a new Party1 state machine that has yet to be filled in. All tabs initially look the same with our draggable state on the left with an initial state already filled into the state machine. To swap between the different state machines you just need to click on the tab with the name of the state machine you are looking for. In this example we are currently on the Party1 tab and can transition to the IF or Sim tab. First, we will go into setting up a state then we will talk about how to create transitions.

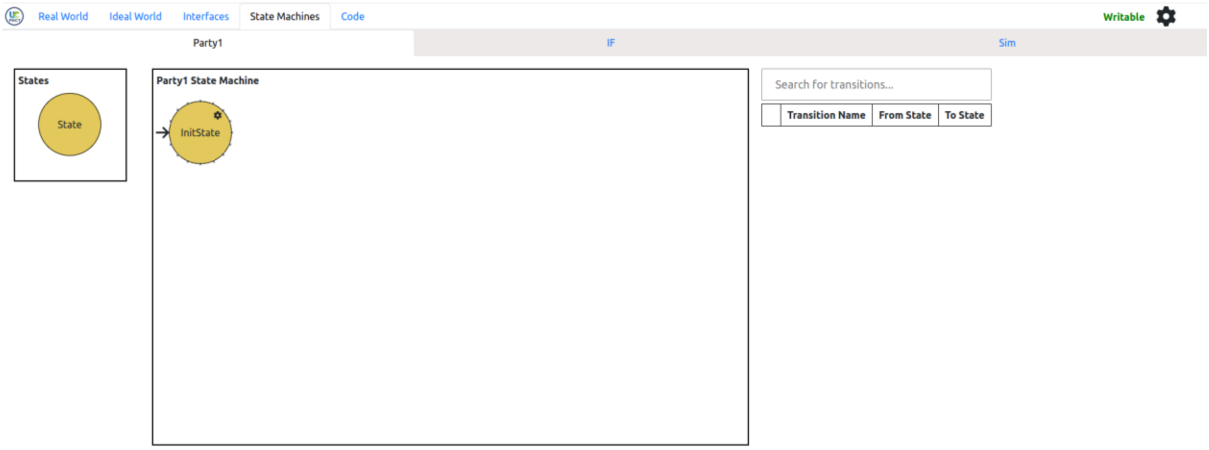
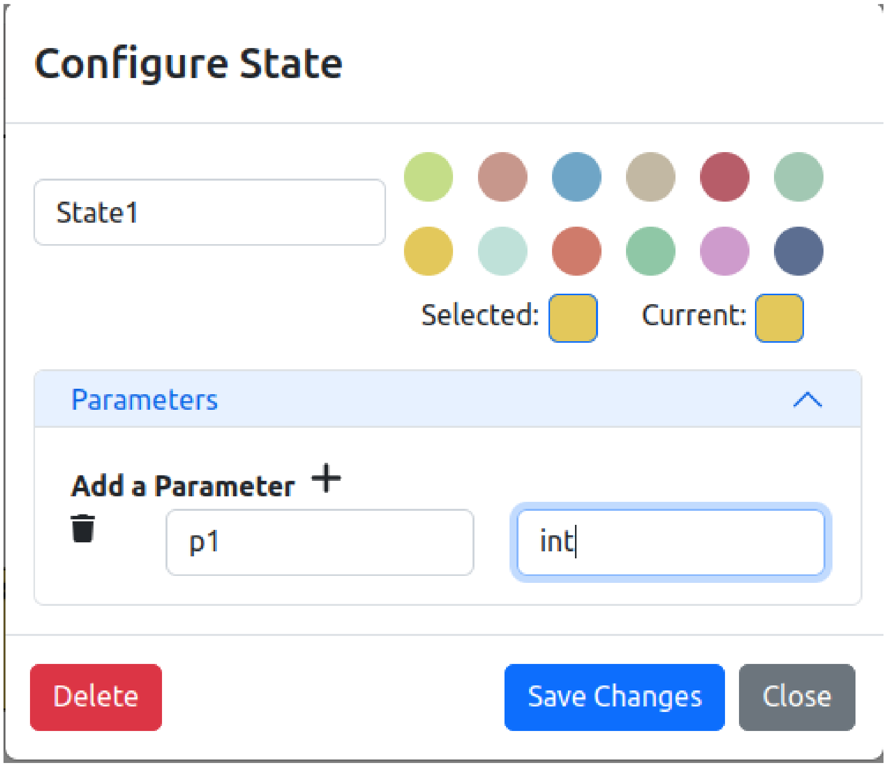


Figure 35: State Machine Landing Page

8.1 States

To create a new state in the desired state machine we need to drag the State circle from the States box in Figure 35 and drop it in the State machine box. This will then open the configuration menu that is shown in Figure 36. In here we can then update the name. Currently we have the name set as State1 in our example. To the right of the textbox, we can change the color of the State itself through the colored circles. A preview of the color will be shown in the box labeled "Selected". Finally, we can set up what parameters are defined in the state. Parameters have a name and a type that are associated with them. In Figure 36 we named the parameter p1 and set its type as an int. For states the parameters could be information that the state needs to make decisions in how it will next proceed. After we have set these up, we can then save our changes by clicking the save changes button. If we need to delete the state, we will need to come back into this modal through the gear icon located on the state and then hitting the red delete button. States cannot be deleted if they have transitions on them so make sure to remove the transitions before attempting to delete a state.



Configure State

State1

Selected: Current:

Parameters

Add a Parameter +

p1

int

Figure 36: State Configuration Modal

8.2 Transitions

Transitions contain all the information needed to transition between two states. The only state that cannot be transitioned to is the initial state. To create a transition, you need to draw an arrow from one state to another. To do this we need to click one of the dots at the edges of a state and drag it to another state. In Figure 38, we can see that this was done between the initial state and State1. This will the open the modal shown in Figure 37. The modal contains a variety of information that needs to be filled out. First, you can name the transition if you would like. This makes transitions easier to identify in the table list, shown on the right in Figure 38. If you do not add a name, the default name will be the name of the two messages.

After the name textbox we have four dropdown menus. Starting at the top left is the state you are transitioning from, in this case it is the Initial State. Then to the right of that dropdown menu is the message that came into that state to cause the transition to happen in this case being CID.BD1.bm2. Then below that message is the State we are transitioning to which is State1. Finally, to the left of the state being transitioned to, we have the message being sent to that state which in this case is CID.BD1.bm1.

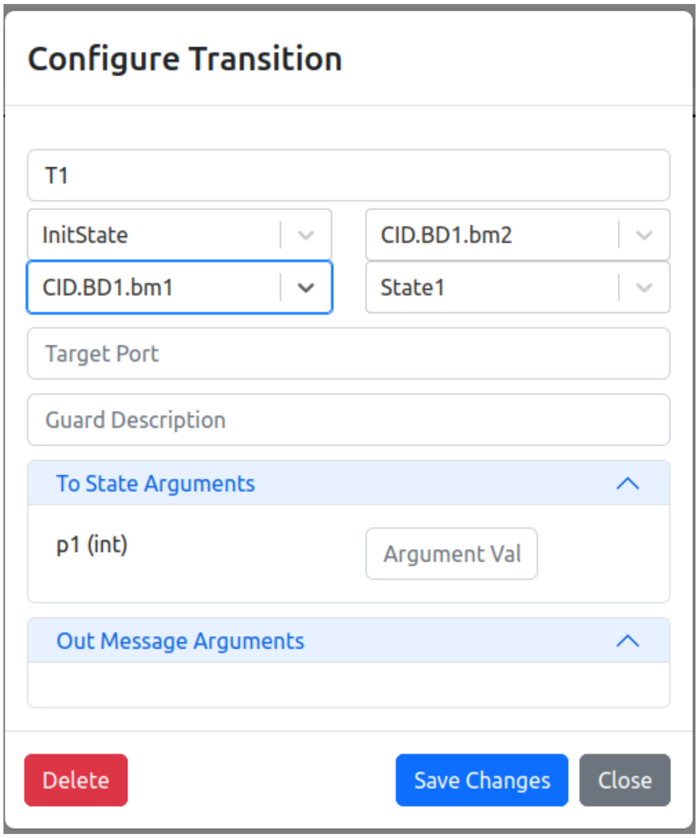
After the dropdown menus we have what is the target port. This is a value to identify what entity this is being sent to. This would be filled out with the value in the message or can be changed by the user if they have a preferred name for the port.

Below the target port we have what we call a Guard Statement. The Guard Statement in simple terms are the conditions required to transfer to the state using the messages. This can be as short or as long as you would like and is there as an option to use but is not required. These will need to be defined in the UC Model but in the web application they are used as placeholders for what those values will need to be.

Below that, are the arguments that the state you are transitioning to has and allows you to set what those values should be going into that state. In this example we have State1 having and argument of p1 which has a type of "int", so we would be expected to fill in an integer value for that argument.

Finally, we have the out message arguments. These would be filled in similarly to the ToState Arguments, but our out message does not have any parameters, so it is empty in this example. After we have finished setting up the transition, we can save the information by selecting the save changes button.

To open back up a transition configuration modal after you have initially created it, you can either click on the arrow's body to open back up the modal, or you can click on the row in the transition table that is on the right in Figure 38. You can also search that table for a transition by its name by typing into the search box above the table. You can see a fully filled out state machine in Figure 39.



Configure Transition

T1

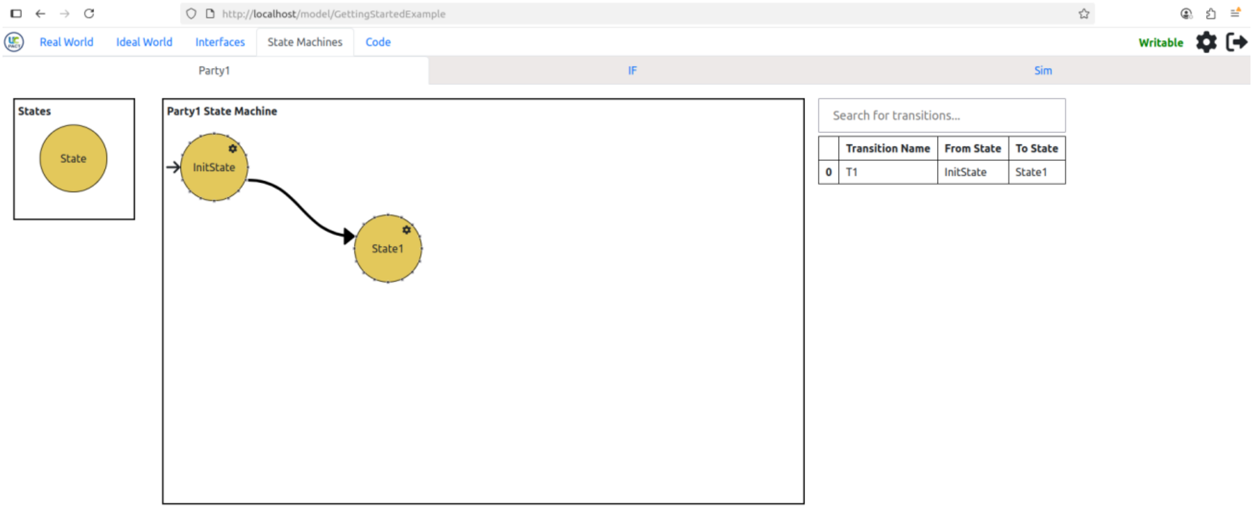
InitState

CID.BD1.bm2

CID.BD1.bm1

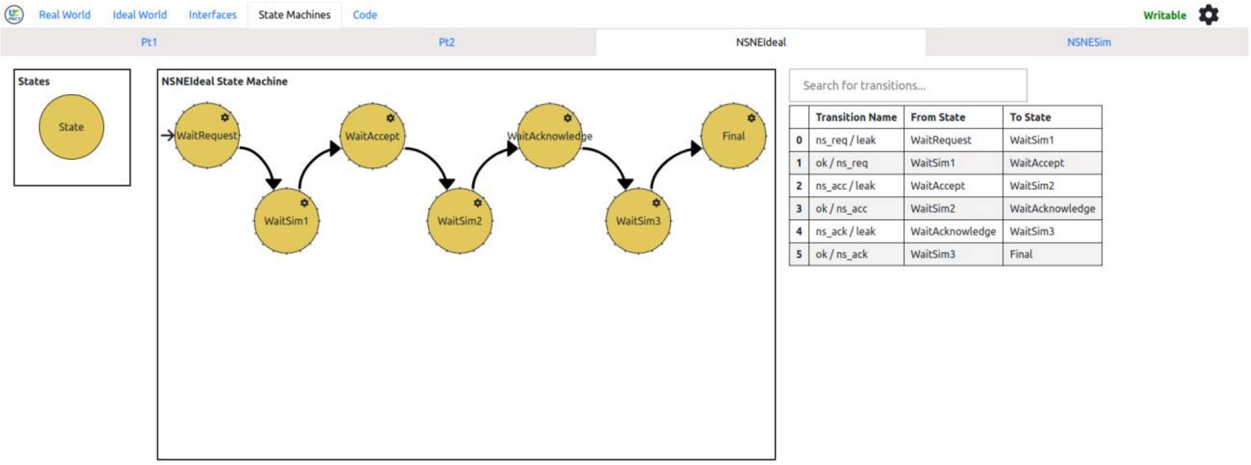
State1

Figure 37: Transition Configuration Modal



	Transition Name	From State	To State
0	T1	InitState	State1

Figure 38: Showing a State Machine with a Transition



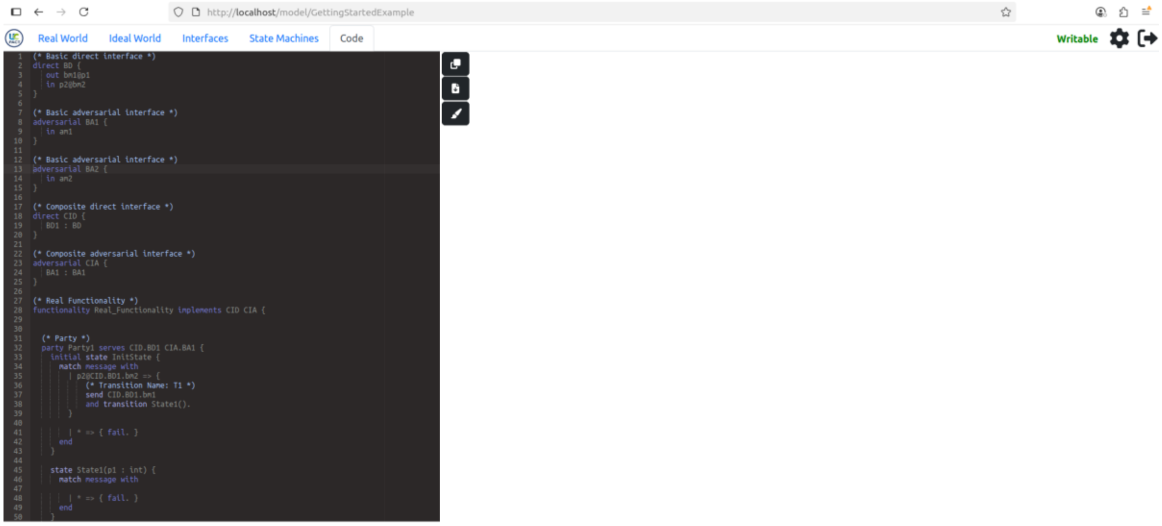
	Transition Name	From State	To State
0	ns_req / leak	WaitRequest	WaitSim1
1	ok / ns_req	WaitSim1	WaitAccept
2	ns_acc / leak	WaitAccept	WaitSim2
3	ok / ns_acc	WaitSim2	WaitAcknowledge
4	ns_ack / leak	WaitAcknowledge	WaitSim3
5	ok / ns_ack	WaitSim3	Final

Figure 39: Fully defined State Machine for a Model

9 UC DSL Code Generation

After you have completed working on your model you can come to the last tab called Code to retrieve your UC DSL code to then work on it outside of the web application. In Figure 40, we can see an example of the code that is generated throughout the process of working on a model in the web application. The code editor here is just a read-only view of the code that has been created by all the work done throughout the other tabs. The code here is also just helper code that will need to be further refined manually to add in details that we did not go into, which involves things like setting up the parameter types and including your EasyCrypt code into the model.

Now as for the other functionality on the page, next to the code editor, at the top right of the editor, are three buttons that each have a function associated with them. The top button will copy all that code to a clipboard for you to paste into a file of your choice. The middle button will download your code into a .uc file with the name of that file being the name you chose for the model. Finally, the bottom button allows you to switch between a light and dark mode for the editor.



```
(* Basic direct interface *)
direct BD {
  out bm1@p1
  in p2@bm2
}

(* Basic adversarial interface *)
adversarial BA1 {
  in am1
}

(* Basic adversarial interface *)
adversarial BA2 {
  in am2
}

(* Composite direct interface *)
direct CID {
  BD1 : BD
}

(* Composite adversarial interface *)
adversarial CIA {
  BA1 : BA1
}

(* Real Functionality *)
functionality Real_Functionality implements CID CIA {


  (* Party *)
  party Party1 serves CID.BD1 CIA.BA1 {
    initial state InitState {
      match message with
        | p2@CID.BD1.bm2 => {
          (* Transition Name: T1 *)
          send CID.BD1.bm1
          and transition State1().
        }

        | * => { fail. }
      end
    }

    state State1(p1 : int) {
      match message with

        | * => { fail. }
      end
    }
```

Figure 40: UCDSL Code Generation Tab