Tox21 Enricher
Setup Guide

(Last Updated 11/01/2021)

http://hurlab.med.und.edu/Tox21Enricher

Junguk Hur, Ph.D.

Department of Biomedical Sciences

University of North Dakota, School of Medicine and Health Sciences

Grand Forks, North Dakota 58202, USA

Email: InformaticsTools@gmail.com

http://hurlab.med.und.edu

Table of Contents

I. Introduction to Tox21 Enricher

Humans are exposed to tens of thousands of chemicals that are used in daily life, some at levels that may pose a health risk. There is limited toxicological information for many of these chemicals, which makes risk assessment difficult or impossible. The United States Toxicology Testing in the 21st Century (Tox21) program was established to develop more efficient and human-relevant toxicity assessment methods. The Tox21 program is currently screening a set of over 10,000 chemicals, the Tox21 10K library, using quantitative high-throughput screening (qHTS) of assays that measure effects on toxicity pathways. To date, more than 70 assays have yielded >12 million concentration-response curves by Tox21 researchers. To efficiently apply these data for identifying potential hazardous compounds and for informing further toxicological testing, the United States National Toxicology Program (NTP) has developed several web applications (Tox21 Toolbox: **http://ntp.niehs.nih.gov/tbox/**), including tools for data visualization (Tox21 Curve Browser) and exploration (Tox21 Activity Profiler).

One critical usage of this dataset is to perform chemical-relational analysis based on the patterns of activity across the Tox21 assays and then to use nearest neighbor-based prediction to infer the toxicological properties of untested chemicals via their association with tested chemicals. One approach to inferring the specific properties is to perform chemical annotation enrichment of chemical neighborhoods.

Here, we present Tox21 Enricher, a web-based chemical annotation enrichment tool for Tox21 assay data built using the R Shiny framework. Tox21 Enricher identifies significantly over-represented chemical biological annotations among sets of chemicals (neighborhoods), which facilitates the identification of the toxicological properties and mechanisms in the chemical sets.

II. Setup

II.1. Getting a Copy of Tox21 Enricher

The Tox21 Enricher code for both the server-side and client-side applications may be found at https://github.com/hurlab/tox21enricher. Assuming you have cloned the repository into a folder called "/home/user/tox21enricher/" on your machine, server-side code can be found in `/home/user/tox21enricher/` and client-side code can be found in `/home/user/tox21enricher/tox21enricher/`.

II.2. Server Application Setup

There are three main components to the Tox21 Enricher server-side utilities: the Plumber API, the request queue, and the database. All installation guides assume you are using a *nix environment, preferably Ubuntu 20.04 LTE.

II.2.a. Server Configuration

Assuming you have cloned the code to /home/user/tox21enricher/, the server-side code's configuration file can be found at /home/user/tox21enricher/config.yml. This configuration file has two namespaces, `tox21enricher` and `tox21enricher-queue`, for the Plumber API and the queue, respectively.

The importance of each of the variables in this configuration file is explained below:
- **driver** – The database driver. This should always be set to `"Postgres"`.
- **server** – The location of the Tox21 Enricher database in the form `<host_ip>:<port>/<database_name>`. For example, if you are hosting the database on the same machine at the default Postgres port of 5432 with a database name of "tox21enricher", this should be set to `"127.0.0.1:5432/tox21enricher"`.
- **host** – The name or IP address of the machine where the database is hosted. If you are hosting the database on the same machine, this should be set to `"127.0.0.1"` or `"localhost"`.
- **uid** – The username of the Postgres user that owns the Tox21 Enricher database.
- **pwd** – The password of the Postgres user that owns the Tox21 Enricher database.
- **port** – The port that the database is listening on. If you set up PostgreSQL to use the default port, this should be set to `"5432"`.
- **database** – The name of the Tox21 Enricher database on the Postgres server. This will likely be set to `"tox21enricher"`.
- **appversion** – The current version of the application. This is used to know which version of the Tox21 Enricher manual to serve to users.
- **appdir** – The location of the server-side project files. This should be wherever you cloned the repository to (i.e., `"/home/user/tox21enricher/"`).
- **python** – The location of the Python installation **in the RDKit environment** (i.e., `"/home/user/anaconda3/envs/my-rdkit-env/bin/python3.6"`).

II.2.b. Setting Up the Plumber API

The Plumber API serves as the primary point of contact between the Tox21 Enricher client application and the other components of the Tox21 Enricher server-side utilities. Requests to perform enrichment and to access the application's database are processed through the Plumber API via HTTP requests. The Plumber API should ideally run as a daemon on the same machine as the queue and the database.

To set up the Plumber API:
1. If they do not already exist, create an Input/ and Output/ directory (note the capital letters) in the project's root directory. Assuming you have cloned the code to /home/user/tox21enricher, these directories should be created at /home/user/tox21enricher/Input and /home/user/tox21enricher/Output.
2. Download and install R from the appropriate CRAN mirror: https://cran.r-project.org/mirrors.html.
3. Download and install an OpenJDK Java distribution from https://openjdk.java.net/ or through SDKMAN! (https://sdkman.io/).
4. Install a few necessary packages using `apt-get update && apt-get install`:
   - libbz2-dev
   - libpcre2-dev
   - libpq-dev
   - libz-dev
   - r-base-dev
5. Use `install.packages('package_name',dependencies=TRUE)` to install the necessary R libraries for the API:
   - config
   - future
   - ggplot2
   - httr
   - parallel
   - plumber
   - plyr
   - pool
   - promises
   - reticulate
   - rjson
   - RPostgreSQL
   - stringr
   - tidyverse
   - uuid
   - xlsx
6. Ensure you have properly updated the server's configuration file.
7. Ensure the Tox21 Enricher PostgreSQL server is running and listening on an open port.
8. Open the startPlumberAPI.sh script in a text editor and set `port=9000` to whatever port you want the API to listen on.

9. Ensure the startPlumberAPI.sh script has executable permissions by running `chmod +x /home/user/tox21enricher.startPlumberAPI.sh`.
10. Run the API with the command `/home/user/tox21enricher.startPlumberAPI.sh`.

II.2.c. Setting Up the Queue

The queue is an Rscript that continually observes the database to see if any enrichment requests have been submitted and either performs enrichment analysis or fetches corresponding annotations for the inputted chemicals.

To set up the queue:
1. Download and install R from the appropriate CRAN mirror: https://cran.r-project.org/mirrors.html.
2. Download and install an OpenJDK Java distribution from https://openjdk.java.net/ or through SDKMAN! (https://sdkman.io/).
3. Install a few necessary packages using `apt-get update && apt-get install`:
   - libbz2-dev
   - libpcre2-dev
   - libpq-dev
   - libz-dev
   - r-base-dev
4. Use `install.packages('package_name',dependencies=TRUE)` to install the necessary R libraries for the API:
   - config
   - future
   - ggplot2
   - httr
   - parallel
   - plumber
   - plyr
   - pool
   - promises
   - rjson
   - RPostgreSQL
   - stringr
   - tidyverse
   - uuid
   - xlsx
5. Ensure you have properly updated the server's configuration file.
6. Ensure the Tox21 Enricher PostgreSQL server is running and listening on an open port.
7. Ensure the startQueue.sh script has executable permissions by running `chmod +x /home/user/tox21enricher.startQueue.sh`.
8. Run the queue with the command `/home/user/tox21enricher.startQueue.sh`.

II.2.d. Setting Up the Database

Tox21 Enricher uses a PostgreSQL database to store chemical and annotation data. Specifically, the application uses the PostgreSQL cartridge for RDKit (https://www.rdkit.org/docs/index.html).

To install RDKit and the PostgreSQL database:
1. Download and install RDKit, following the directions at https://www.rdkit.org/docs/Install.html.
2. This will require you to download and install the Conda package manager and Python.
3. Install and initialize the RDKit PostgreSQL cartridge, following the directions at https://www.rdkit.org/docs/Install.html#installing-and-using-postgresql-and-the-rdkit-postgresql-cartridge-from-a-conda-environment.
4. Make sure the PostgreSQL server is running by using `<conda_folder>/envs/my-rdkit-env/bin/pg_ctl -D /<data_folder>/`.
5. Obtain a backup or copy of the Tox21 Enricher database from Dr. Hur.
6. Create a database called "tox21enricher" and either:
   a. Log into it by using `<conda_folder>/envs/my-rdkit-env/bin/psql tox21enricher`. Run the command `\i /<path_to_database_backup>/` from within Postgres to restore the database from the backup.
   b. Use the pg_restore utility to restore the database from backup by running from the command line using `<conda_folder>/envs/my-rdkit-env/bin/pg_restore /<path_to_database_backup>/`.

II.3. Client Application Setup

The Tox21 Enricher client application provides a graphical user interface for performing enrichment analysis and interacting with the data stored in the Tox21 Enricher database. The client application may be run on the same machine as the server-side utilities and exposed to the internet so that other remote clients may use the application via a web browser. Otherwise, the client application may also be run directly on the client machine as a standalone executable and connect remotely to the server utilities running on a remote host. To access the Tox21 Enricher client application via an internet browser, your browser must have JavaScript enabled for the application to load properly.

II.3.a. Client Configuration

Assuming you have cloned the code to /home/user/tox21enricher/, the server-side code's configuration file can be found at /home/user/tox21enricher/tox21enricher/config.yml. This configuration file is notably simpler than the configuration file used for the Tox21 Enricher server. The client configuration file has only one namespace, `tox21enricher-client`.

The importance of the two variables in this configuration file is explained below:
- **host** – The name or IP address of the machine where the Tox21 Enricher Plumber API is hosted. If you are hosting the API on the same machine, this should be set to `"127.0.0.1"` or `"localhost"`.
- **port** – The port that the Tox21 Enricher Plumber API is listening on. By default, this is set to `"9000"`.

II.3.b. Setting Up the Client Application (Standalone)

The standalone client application will be available in a future update.

II.3.c. Setting Up the Client Application (Web Application)

The Tox21 Enricher client application is a Shiny (https://www.rstudio.com/products/shiny/) application and thus can be hosted on a machine and accessed from a remote client via an internet browser.
To set up the client application as a web application:
1. Download and install R from the appropriate CRAN mirror: https://cran.r-project.org/mirrors.html.
2. Download and install an OpenJDK Java distribution from https://openjdk.java.net/ or through SDKMAN! (https://sdkman.io/).
3. Install a few necessary packages using `apt-get update && apt-get install`:
    - libbz2-dev
    - libgdal-dev
    - libpcre2-dev
    - libpq-dev
    - libudunits2-dev
    - libz-dev
    - r-base-dev

4. Use `install.packages('package_name',dependencies=TRUE)` to install the necessary R libraries for the API:
   - bslib
   - catmaply
   - CePa
   - config
   - dplyr
   - DT
   - future
   - ggVennDiagram
   - httr
   - igraph
   - igraphdata
   - parallel
   - plotly
   - pool
   - promises
   - rclipboard
   - rjson
   - RPostgreSQL
   - shiny
   - shinyBS
   - shinycssloaders
   - shinydashboard
   - shinyjs
   - shinythemes
   - stringr
   - uuid
   - visNetwork
   - xlsx
5. Ensure you have properly updated the client application's configuration file.
6. Launch the application according to the **Running in a Separate Process** section in the instructions here: https://shiny.rstudio.com/articles/running.html.
7. Open the port that the client application runs on. By default, this is a random port, but you can configure the application to run on the same port every time by specifying a specific port for the `port` argument of the `shiny::runApp()` function as described in the documentation here: https://shiny.rstudio.com/reference/shiny/1.0.1/runApp.html.

III. Deployment

The server components of Tox21 Enricher can be deployed using Docker. A Docker container is the recommended tool for hosting a publicly available Plumber API, as described here: https://www.rplumber.io/articles/hosting.html.

III.1. Configuring and Building the Docker Images

Before you can host the Tox21 Enricher server code, you must create two necessary Docker images: **tox21enricher/plumber** and **tox21enricher/database**. These images should be configured to suit your needs before you attempt to build them:

1. Install Docker following the directions on the official website: https://docs.docker.com/get-docker/.
2. Navigate to the project root folder for Tox21 Enricher (wherever you cloned the repository to). Assuming you have cloned the repository into a folder called "/home/user/tox21enricher/" on your machine, this should be `/home/user/tox21enricher/`.
3. Make sure the file `<project_root>/config.yml` is set up to match the login, host, and port information of the Tox21 Enricher database in its Docker container.
   a. The host name should be whatever the name of the database service is as specified in the docker-compose.yml file (by default this is "db").
4. Configure the **tox21enricher/plumber** image:
   a. Make sure there is a **Dockerfile**, just called "Dockerfile", in this folder. By default, this should already be in the folder if you have cloned the project directly from the GitHub repository.
   b. Open the Dockerfile in any text editor and follow the directions in the file to configure the necessary values.
      i. You will have to come up with a username and password for the user inside the container that you want to run the API process.
      ii. Make sure the file `<project_root>/docs/Tox21Enricher_Manual_vX.X.pdf` matches the correct version number of the manual you want to display to users.
      iii. Make sure the file `<project_root>/docs/Tox21Enricher_Setup_vX.X.pdf` matches the correct version number of the manual you want to display to users.
   c. Make sure the following files exist at the correct locations:
      i. <project_root>/requirements-src.R
      ii. <project_root>/config.yml
      iii. <project_root>/plumber.R
      iv. <project_root>/HClusterLibrary/ **(this is a directory)**
      v. <project_root>/calcReactiveGroups.py
      vi. <project_root>/docs/<user_manual>
      vii. <project_root>/docs/<setup_guide>
   d. From within the project root folder, run the command `docker build -t tox21enricher/plumber .`. Note the period at the end of the command, which signifies that Docker should use the current directory as a context.

5. Configure the **tox21enricher/database** image:
   a. From the project root folder, navigate to the `tox21enricher-database/` directory.
   b. Make sure there is a **Dockerfile**, just called "Dockerfile", in this folder. By default, this should already be in the folder if you have cloned the project directly from the GitHub repository.
   c. Open the Dockerfile in any text editor and follow the directions in the file to configure the necessary values. You will have to come up with a username and password for the user inside the container that you want to run the API process.
   d. Make sure the following files exist at the correct locations:
      i. <project_root>/tox21enricher-database/tox21enricher-database.sql **(This should be a backup of the Tox21 Enricher database, specifically named like this and ideally created with the *pg_dump* utility.)**
      ii. <project_root>/tox21enricher-database/entrypoint.sh
   e. From within the project root folder, run the command `docker build -t tox21enricher/database .`. Note the period at the end of the command, which signifies that Docker should use the current directory as a context.
6. Navigate to the **HClusterLibrary** folder. By default, this should be at `<project_root>/HClusterLibrary/`. Make sure the files **clusterLinux64** and **hclimage-o.jar** have run permissions by running the commands `RUN chmod +x <project_root>/HClusterLibrary/clusterLinux64` and `RUN chmod +x <project_root>/HClusterLibrary/hclimage-o.jar`.
7. Install Docker Compose following the directions on the official website: https://docs.docker.com/compose/install/.
8. Navigate back to the project root folder and make sure the file **docker-compose.yml** is in that folder. By default, this should already be in the folder if you have cloned the project directly from the GitHub repository.
9. Open **docker-compose.yml** in any text editor and make sure you correctly configure the paths for the shared volumes.
10. If testing, you can uncomment lines 29 and 45 in docker-compose.yml that state `network_mode: host` and comment out lines 25-28 and 41-44. This will allow the services to run attached to the host's network which may make it easier to test the Docker setup.
11. The Docker containers can now be started through Docker Compose by using the command `docker-compose up`. This will start the Tox21 Enricher Plumber API, queue, and database.

Note: You may need superuser permissions to run `docker-compose up`.

III.2. Setting Up the Load Balancer

By default, Tox21 Enricher as deployed using Docker Compose does not enable its load balancing system. However, load balancing using **HAProxy** is implemented and can easily be enabled by changing a few lines in the **docker-compose.yml** configuration file. Using the load balancer is the recommended approach for deploying Tox21 Enricher. Not using the load balancer is only recommended for testing purposes.

1. Navigate to the project root folder for Tox21 Enricher (wherever you cloned the repository to). Assuming you have cloned the repository into a folder called "/home/user/tox21enricher/" on your machine, this should be `/home/user/tox21enricher/`.
2. Make sure the file **docker-compose.yml** is in that folder.
3. Open docker-compose.yml in any text editor and:
   a. Un-comment lines 9 – 19 to define the HAProxy service.
   b. Edit line 42 (port mapping for the tox21enricher service) and change the port value left of the colon to a port range formatted like N-M, where N is the first port in the range and M is N + (`<number_of_API_instances_desired> - 1)`. For example, if you want 3 instances of the API running concurrently, you could change the port value to `7998-8000`.
   c. Comment out lines 43 – 44 as we no longer want to expose the ports that the API instance(s) will be listening on.
   d. Comment out lines 29 and 45 that state `network_mode: host` as we do not want the services to run on the host network.
4. The Docker containers can now be started through Docker Compose by using the command `docker-compose up --scale tox21enricher=`N. Where N is the number of concurrently running instances of the Tox21 Enricher API.

III.3. Testing Connectivity

To connect an instance of the Tox21 Enricher client application to the API running on a remote server, follow these steps:

1. Navigate to the project root folder for Tox21 Enricher (wherever you cloned the repository to). Assuming you have cloned the repository into a folder called "/home/user/tox21enricher/" on your machine, this should be `/home/user/tox21enricher/`.
2. Navigate to the client application's folder at `<project_root>/tox21enricher/` and assure the configuration file, **config.yml**, exists.
3. Open config.yml and navigate to the **tox21enricher-client** namespace, under which you should see the **host** and **port** variables.
4. Set **host** to the address of the remote server hosting the Tox21 Enricher API.
5. Set **port** according to the situation:
    a. If you are running Docker Compose on the host network:
        i. Set port to the port the Tox21 Enricher will run on. By default, this is 8000 when deploying through the tox21enricher/plumber Docker container.
    b. If you are NOT running Docker Compose on the host network AND if you are NOT using the HAProxy load balancer:
        i. Set port to the port on the left of the colon on line 42 of docker-compose.yml (the mapped port for the **tox21enricher** service).
    c. If you are NOT running Docker Compose on the host network AND if you ARE using the HAProxy load balancer:
        i. Set port to the port on the left of the colon on line 17 of docker-compose.yml (the mapped port for the **lb** service).
6. Make sure your firewall allows connections on the specified port.
7. Run the client application. If the following success message appears in the sidebar, then the client application has successfully connected to the remote server.

> Connection with Tox21 Enricher
> server successfully established.

8. Alternatively, you may use the cURL command `curl <server_address>:<port>/ping` to ping the Tox21 Enricher API. If your machine can connect to the remote server, this will return a response of `[true]`. This is the same API endpoint that is called by the client application when it is checking if it can connect to the Tox21 Enricher API (Step 7).

IV. Updating the Database

It will sometimes be necessary to update the database used by Tox21 Enricher. Included in the application's GitHub repository under the `/<project_root>/tox21enricher-database/update/` directory are several scripts that must be run to properly update the database.

The following instructions must be run to ensure the database update proceeds smoothly.
1. Navigate to the `/<project_root>/tox21enricher-database/update/` folder.
2. Make sure that three subdirectories, **Annotation**, **ReOrganized**, and **tmp**, all exist in this directory. If they do not, create them. **Note the capitalization of the names**.
3. Make sure the files **Tox21_CASRN_Names.anno** and **Tox21_CASRN_Names_Full.anno** exist in the **Annotation** folder. They should already be in this folder if you cloned the project from the GitHub repository.
4. Place your annotation files in the **Annotation** folder. There are a few specifications these files need to follow:
   a. You must place ALL annotation files in the **Annotation** folder, not just the new/updated files.
   b. Each line must be structured like this:
      `<CASRN>    semicolon and; space; separated; list; of; annotations`
      Note that the CASRN and annotations should be tab-separated, and each annotation should be separated by a semicolon AND a space. The last annotation on each line should be followed immediately by the newline character ("\n") and NO semicolon.

   c. Each file should be named like this:
      `DrugMatrix_EXAMPLE_NAME.txt`
      Note that each file must be prefixed with "DrugMatrix_", class names should be in all capital letters, and multiple-word names must be split by underscores ("_").

5. Run the first four scripts using Perl. Note that some or all of these scripts may take hours to days to complete, depending on
   a. Run the **first** script to reorganize the annotation data using the command `perl 01_ReOrganize_Annotation_Data_v2.1-20161003.pl`.
   b. Run the **second** script to perform the Fisher exact test and create most of the database files using the command `perl 02_Split-perform_Fisher-Exact-Test.pl`.
   c. Run the **third** script to create the final pairwise calculation file using the command `perl 03_Merge_Pvalues-and-calculate-qvalues.pl`.
   d. Run the **fourth** script to create the initial annotation matrix files using the command `perl 04_Create_complete_annotation_matrix_v1.0_11182020.pl`.

6. Run the next two scripts using Python.
   a. Run the **fifth** script to create the file **annotation-matrix-names.sql** which should store the names and IDs of each annotation in the matrix using the command `python 05_Create_annotation-matrix-names.py`.
   b. Run the **sixth** script to create the file **newmatrix.sql** which should store the new annotation matrix using the command `python 06_Create_annotation-matrix.py`.
7. The **seventh** script is optional and should only be run if you are updating the **mols_2** table. In most cases, you can ignore this script.
8. Before you can import the generated text files into the PostgreSQL database, a little bit of formatting needs to be done:
   a. The headers (first line) of each file must be deleted. The easiest way to do this is by running the **remove_headers.sh** bash script. Ensure this script has execute permissions.
   b. The table_annotation_class_v2.1.txt file needs to be padded with placeholder data that will be overwritten with a later script. To do this, run the **pad_annotation_class.py** script with Python.
9. You should now be able to import the files into the database. First, ensure that all database tables exist. They should be:
   a. annotation_class
   b. annotation_detail
   c. annotation_matrix
   d. annotation_matrix_terms
   e. annoterm_pairwise
   f. *chemical_detail*
   g. *enrichment_list*
   h. *fps_2*
   i. *mols_2*
   j. *queue*
   k. *status*
   l. term2casrn_mapping

   The *italicized* tables should not have to be updated.
10. You can use the PostgreSQL command `COPY <table_name> FROM '/path_to_file/file.txt';` to fill each table with the data from the respective text files. You should only copy from the files:
    a. table_annotation_class_v2.1.txt
    b. table_annotation_detail_v2.1.txt
    c. table_annoterm_pairwise_v2.1_FINAL.txt
    d. table_chemical_detail_v2.1.txt
    e. term2casrn_mapping_v2.1.txt
11. After filling in the above tables, run the eighth script, `08_Add_additional_info_annotation_class.sql`, with PostgreSQL to finish filling in the annotation_class table.
12. To fill the annotation_matrix and annotation_matrix_terms tables, run the **annotation-matrix-names.sql** and **newmatrix.sql** scripts.

V. Common Issues

Both the client and server code for Tox21 Enricher are still in continued development, so there are likely issues that may arise when trying to either perform certain tasks or set up the application. Known bugs in the code and planned features are documented on the project's GitHub repository at: https://github.com/hurlab/tox21enricher/issues. Other known issues not necessarily related to the code itself are documented in this section of the manual.

V.1. Setup Issues

These are known issues that you may experience when attempting to set up Tox21 Enricher.

- **Problem installing "xlsx" R package** – The **xlsx** package requires a Java installation on your computer. Specifically, the package requires that you install the **rJava** R package as well. Even if you have a correctly set-up installation of Java, the installation of the rJava package may fail and ask you to run the `R CMD javareconf` command. Running this command may also fail. To fix this, you must explicitly include the path of your JDK installation when running `R CMD javareconf`, like so: `R CMD javareconf JAVA_HOME=/<path_to_your_java_installation>/`.
- **Problem installing RDKit PostgreSQL cartridge** – Even if you successfully install RDKit using Conda, Conda may fail to install the RDKit PostgreSQL cartridge due to package version conflicts. A workaround for this is to create a new Conda environment using `conda create -n <environment_name>`, delete or move the contents of the new environment's `/bin/` directory, install the RDKit PostgreSQL cartridge to that environment, then finally move the newly installed files from the new environment's `/bin/` directory into the RDKit environment's /bin/ directory.
- **Problem installing "CePa" package – missing dependencies** – The **CePa** R package requires two dependency packages called **Rgraphviz** and **graph**. These cannot be installed with `install.packages()`. You must first install the **BiocManager** R package and then run `BiocManager::install("Rgraphviz")` and `BiocManager::install("graph")`.
- **Potential conflict with Python versions when using the "reticulate" R package** – The Tox21 Enricher API code file (plumber.R) sets the environment variable **RETICULATE_PYTHON** to the **python** path defined in the application's config.yml file. This may cause a conflict if you have another application running on the same machine using the **reticulate** package with a different version of Python. A fix or workaround for this may be implemented in a future update.

END OF THE SETUP GUIDE