# Build

Here are the steps to build trunk locally.

1. Install Ant 1.9# and make sure it is in your path. Verify by running ant -version.
2. Install Java 8 64-bit JDK and make sure it is in your path. Verify by running java -version. Ensure JAVA\_HOME points to where ever you installed it.  
   (This was tested with the latest version of Java 1.8.0\_162)
3. These are the only dependencies required and dev should no longer need Tomcat or Java 5.   
   (Note: The jenkins builds still require Java 5 due to the dependency on the old pack/unpack utility. We are working on removing it.)
4. Checkout the following svn repo [http://svn.techrx.com/erx/t](http://svn.techrx.com/erx/branches/feature/jboss)runk
5. At the root of the repo, run the following command:  
   ant

That should build the whole project, run all unit tests and produce all the artifacts.

# Deploy

Here are the steps to deploy the local build to jboss.  
We will specify steps to deploy both with and without docker. On Win 7 platforms, the non-docker option might be faster.  
First run the steps common to both deployments. Then follow docker or non-docker instructions.

### Environment Set up

1. Open the following file:  
dev/domains/local/erx.env  
Edit the following entries in the file to set your local DEV01 personal schema credentials (sid:ndcd01)  
db\_user=  
db\_password=  
Save the file once done.

### Standard Ant deployment (Non-docker option)

1. At the root folder run the following commands:

* ant install-jboss  
  This will download and install the correct version of JBoss EAP
* ant configure-jboss  
  This will configure the jboss installation for this branch of ERX
* ant deploy-jboss  
  This will deploy the EnterpriseRx.ear file. If you are using exploded ear file format, this will be the first step, followed by the rest of the steps for exploded ear deployment (see below).
* ant plsql-load-erxpackages    
  (Some folks wanted a target to simply sqlplus load the db packages . Run this only if you don't want to use your existing scripts to load packages. This will also clean the pipelined objects in your local schema prior to load.)
* ant start-jboss  
  This will start jboss with the ear file deployed. It will stop jboss if already running and configure it too if needed.   
  If you prefer running in exploded ear style, read below for more instructions.  (Note that the default behavior of start-jboss is to run exploded.  To run normally, you'd have to switch the build/dev.xml  start-jboss params to auto\_deploy\_zipped to true and auto\_deploy\_exploded to false.)

1. Once installed and configured, you should only need to run the start-jboss target.
2. This installs the JBOSS Enterprise Edition EAP 7.1.0. That is the version we are going with.
3. Feel free to play around with your jboss instance, if you messed something up, just re-run install-jboss and configure-jboss and  
   it will set everything back up.
4. Edit the http and management ports to something other than the defaults in dev/domains/local/erx.env if those ports conflict with services you already run locally. Restart jboss after making those port changes.
5. This should run a local jboss instance with our app deployed.
6. Ctrl-C will shut down the server.
7. On Windows, you have two ways to shut down the appserver.

* Ctrl-C in the window you ran start-jboss in. This will shut down the ant process. Then run ant stop-jboss to stop the server.
* In another cmd window, run ant stop-jboss. This will shut down both the app server and the ant process.

### IDE Setup

### Eclipse

There is now an ant target that will create an eclipse project for you.

1. First make sure you have run the full build by running:  
   ant
2. Next, in your root trunk folder, run  
    ant eclipse  
   You will need to run this anytime folks change the compile classpath say by adding or removing jars.  
   So when you find your eclipse project no longer compiling due to missing jars etc, re-run this command and refresh your eclipse project.
3. It is a helpful practice to call your checkout folder the same as the branch you are checking out → trunk, 8.1 etc. This is especially helpful when working with multiple branches for support, back-porting code etc.  
   If however, your root folder is not called trunk, edit the root.projdir property in build/dev.xml and rename it to whatever your root folder is and re-run step #1
4. Open eclipse ( I suggest upgrading to the latest version - oxygen)
5. In Eclipse Window>Preferences>Java>Installed JREs has the default set to the latest Java 8 JDK 64-bit.
6. Open File→Import Files
7. In the import wizard, expand General  and click Existing projects into workspace and click Next
8. This should make eclipse recognize the project we created in Step 1. Click finish.
9. Eclipse will now refresh and rebuild the project with no errors.
10. By default, this project will write output classes to trunk\classes so as not to conflict with trunk\bin where the ant build script output is written to.
11. Follow the instructions below for exploded ear deployment to change the output folder to the exploded TRexOneEnterpise.jar folder

The reason I went the route of a target instead of just checking in static project files is so that this will help the project stay current with library and classpath changes.

A static project will quickly get out of date and someone will have to keep maintaining it. The ant target uses the actual build compile path to build the project.

So when we make library changes, we would only need to re-run step 2 (the ant eclipse target) above.

### Exploded ear deployment

Follow these instructions if you want to run in exploded ear format

1. After the deploy-jboss task above, goto the jboss.home/standalone/deployments (dev/server/jboss-eap-71/standalone/deployments) folder and use your favorite unzip utility to unzip the ear file.
2. Delete the ear file and call that unzipped folder EnterpriseRx.ear
3. Inside the ear's lib folder, unzip the TRexOneEnterprise.jar
4. Delete TRexOneEnterprise.jar and call the unzipped folder as TRexOneEnterprise.jar
5. Now you can point your IDE's output or classes folder to this TRexOneEnterprise.jar folder and it should start up with classes compiled in your IDE.   
   Once exploded, the eclipse project default output path should look something like this:  
   trunk/dev/server/jboss-eap-7.1/standalone/deployments/EnterpriseRx.ear/lib/TRexOneEnterprise.jar
6. If your IDE of choice is eclipse, note that Eclipse uses its own compiler (an open source project called JDT core) and will rebuild all the classes the first time. It should be incremental and faster after that.
7. In **build/dev.xml**, look for the start-jboss target and set the env variables (Default settings are for exploded)
   1. auto\_deploy\_zipped to false and
   2. auto\_deploy\_exploded to true    
      for windows or linux depending on where you are running this.
8. Run ant start-jboss
9. It should run with all your exploded classes and classes from your IDE output now.
10. The server is set up to listen for debug on port 6666. Change it if needed.

### Client setup

1. Open either client.bat or client.sh depending on your OS
2. Edit the IDE\_CLASSES entry to point to the eclipse output path you set above. You might need to provide the complete path to that folder.
3. Now both your client and server should be using your eclipse output to help with local development.
4. The client is set up to listen for debug on port 5555 by default. Change it if needed.

### New App Server

1. The jboss instance is set up with an admin user.  
   userid : admin  
   password : jboss
2. The default jboss console is at: [http://localhost:9990](http://localhost:9990/)
3. The app url is at: <http://localhost:8080/TRexOneWeb>
4. If not using docker, jboss will be installed locally under dev/server and the log files etc will be in the jboss install there.

### Client

There is a client launch script for linux and windows included now under dev/client  
This is just to make it easy for new folks joining our team.  
You can continue to use your existing scripts and launch files and they  
should work just as well. You will have to change the port 7001 to 8080 though.

To run the included client,

1. Cd to dev/clients folder and run client.sh or client.bat depending on your platform and you should be able to login to your store.
2. You can run it as clients.sh store# or without the store# in which case our client will prompt you for the store#.

# Code changes related to Java 8 compilation

1. I have noticed regex failures. Regex that used to match no longer does in some cases. I have had to fix this both in code  
   and build scripts. Java 8 includes changes to the regex classes likely causing this. So be aware of that.
2. When you convert collections like Set and Map to Arrays, the order of elements in the returned array is different.  
   We have had a few instances where our code depended on this order. Change such dependencies to just check for contains() or   
   some order-agnostic method like that.
3. List now includes the sort method so any subclasses will have to take that into account and delegate or override the method.
4. Jboss is more strict about jndi naming conventions so the jndi names in our beans had to be changed to meet the j2ee standard.
5. Some unit tests which were simulating timeouts failed on my slow win7 laptop. I had to increase the wait time in those tests.

# Build changes related to Java 8

### Changes

1. The ear manifest was modified to add a reference to the erx config files

### Improvements

1. Going back to creating only ONE ear file merging both the app and interface endpoints.
2. Depends on only one version of Java - Java 8. Dropped most dependencies on Java 5.
3. Drop the need to install tomcat4 and external references to tomcat 4.
4. Removed the copy of jaxb generated source code back into the source tree. This corrupts the svn check in view and was  
   totally unnecessary.
5. **The ear file is now renamed to simply EnterpriseRx.ear**
6. Removed the following unused targets

* compileFax
* packageBoomi
* packageConsumerPortal
* package TRexOneMMS.war
* package TRexOnePPISimulator.war