Users Guide

Table of contents

1 Overlays..... 2
2 Directory Structure for Users..... 2
3 Files..... 2
4 Directory Structure for Developers..... 3
5 Environmental Variables..... 4
6 Configuration File..... 4

Note:
This section is seriously out of date!

1. Overlays

An XLattice user is expected to take part in more than one **overlay** or network. These are not networks in a conventional sense. They are overlay networks, application-specific networks built on top of the Internet and other networks. Each such overlay has a name. Resources are allocated to the overlay under that name. A user has at least one **EndPoint** (address) in each overlay that the user participates in.

By default each user participates in the universal overlay and has one node in that overlay.

2. Directory Structure for Users

In UNIX terminology, XLattice directories may be either at the root level or at the user level. In the first case, XLattice expects to find its files under /usr/local/xlattice. In the second case, which is preferred for security reasons, XLattice will look under $HOME/xlattice, where $HOME represents the user's home directory.

If, for example, user Fred with home directory /home/fred participated in three overlays, universal, strat_game, and freernet, Fred's directory structure would look like this:

```
home
  fred
    xlattice
      bin
      xlattice.xml
      overlays
        universal
        strat_game
        freernet
      lib
```

The bin subdirectory contains runnable scripts. In a Java environment, the lib directory contains jars. Lattice configuration and data files are organized under the overlay subdirectory.

3. Files

Each node has a configuration file, xlattice.xml, that specifies the **NodeID,** its **RSA key,** and the overlays that the node participates in. The NodeID is a 160-bit number that is unique to the node. The RSA key is its **cryptographic identity,** used by the node to prove its identity to other nodes.

Each of the overlay directories contains a configuration file, two subdirectories reserved for network files, and possibly other application-specific files.

The two subdirectories are .global and .cache. The first is used by the network. The second is used by the application running on the overlay to store local, usually temporary, data. The amount of space allocated to these two subdirectories is set in the configuration file.

With these files, Fred's directory structure looks like this:

```
home
  fred
    xlattice
      xlattice.xml
      bin
      overlays
        universal
          .cache
          .global
        strat_game
          .cache
          .global
          strat_game.xml
        freernet
          .cache
          .global
          freernet.cfg
      lib
```

The .cache and .global subdirectories are hidden because of the leading dot ('.') and so are not normally visible when the directory is listed.

4. Directory Structure for Developers

Anyone who downloads the XLattice source code either from CVS or by extracting downloaded tarballs will have a number of **additional** files and directories, including those shown below.

XLattice is being developed as a number of components. Each component has its own separate development tree. The example below shows the directory structure for corexml. There will be a similar subtree for each XLattice component.

```
home
  fred
    xlattice
      corexml
        build.sh
        build.xml
        classpath.sh
        project.xml
        src
          java
            org
```

```
              xlattice
                corexml
          test
            org
              xlattice
                corexml
        target
          classes
          test-classes
```

build.xml is an Ant build file, Java's equivalent to the C/C++ Makefile. Ant is run and the software built and tested by typing

```
./build.sh test
```

under UNIX/Linux or similarly invoking build.bat in a Windows environment.

Java source code is organized below, in this case, src/java/org/xlattice/corexml. Tests are in the parallel src/java/org/xlattice/corexml subdirectory. Compiler output for source code is directed to subdirectories under target/classes/org/xlattice/corexml; compiled test classes are found in the parallel directory structure under target/test-classes.

5. Environmental Variables

The user needs to define at least one environmental variable, JAVA_HOME, and should define one more, XLATTICE_HOME. In addition, it is convenient to add $XLATTICE_HOME/bin to the path. Under UNIX or Linux and bash, these can be accomplished by something similar to

```
export JAVA_HOME=/usr/local/java
export XLATTICE_HOME=$HOME/xlattice
export PATH=$XLATTICE_HOME/bin:${PATH}
```

6. Configuration File

Although the configuration files can be created and edited manually, normally they will be created by the system from command line arguments. Long numeric values in the configuration file (such as the NodeID and RSA key) are base-64 formatted. IP addresses are in dotted-quad notation.

Note:

The example that follows is wrong. Look at the CryptoServer release for a better model. However, it is likely that this entire approach will be replaced by something like the JCE keystore.

```
<overlay name="freernet" cache="756M" global="10G">
  <node name="freddyBoy" id="0123456789abcdef">
    <key type="..." public="..." private="..."/>
    <address type="ipv4" value="1.2.3.4:5678"/>
  </node>
```

```
    <neighbors>
      <neighbor id="123456789abcdef0" keytype="..." public="...">
        <address type="ipv4" value="2.3.4.5:6789" protocol="tcpip"/>
      </neighbor>
      <neighbor id="23456789abcdef01" keytype="..." public="...">
        <address type="ipv4" value="3.4.5.6:7890" protocol="tcpip"/>
      </neighbor>
    </neighbors>
  </overlay>
```