# TinyTSC Documentation

Latest: 20061030

## Background

This is a tiny prototype from which TSC4J will evolved. It represents the smallest possible body of code that satisfies the core architectural requirements of TSC. At this time, it imports legacy TSC (old Forth-based) code, and exports to its own XML format <note>not implemented yet</note>.

See **Envisionment Trace** below for a glimpse of the XML format in tinyTSC.

## Architecture

<tbd>

## Knowledgebase

Right now, we are just using the old TSC KBs from earlier experiments. They have been cleaned up (sortof) and are found in the directory /legacy/test .

TODO: wire import and export so we can move to an internal XML format for future work.

## Installation

Just configure the directory /tinytsc according to the following directory structure. There are three jar files currently required in the /lib directory:

* jdbm.jar
* log4j.jar
* xpp3.jar

The system includes a database engine (JDBM[[1]](#footnote-1)) that builds a database in the /database directory. At this time, the system does nothing directly with that database. It will come into use once we extend this development platform to include mappings from other sources of knowledge structures (frames).

### Directory Structure

/root

/tinytsc

/classes

/doc

/database

/images

/legacy (raw data)

/lib

/logs

/src

build.xml

logfile.log

todo.txt

tsc-props.xml

build.xml includes several *ant* targets, for compiling, building a jar file, building the javadocs, and running tinyTSC.

## Configuration

The TSC directory structure includes the file tsc-props.xml. In that file is an important property:

<parameter name="DatabasePath" value="/database" />

DatabasePath is presently set to the *base relative* path /database. At this time, TSC is hard-wired to prepend that value with the path to the install directory, /tinytsc . A future change will add a new property: PathIsAbsolute with "1" = true and "0" = false. This would allow for installation of the database data files anywhere desired on the same computer.

## Running

Several ways to run tinyTSC. From a shell script, you need to link in the jars in /lib, then call org.nex.tinytsc.Main

If %ANT\_HOME% is configured, simply call from /tinytsc directory

ant run

There are a number of shortcuts taken in this code, one of which is that the code expects tsc-props.xml to be found in the install directory from where the classpath originates: /tinytsc.

**Graphical user interface, application, Word

Description automatically generatedOn Bootup**

Using: File:Import Legacy, then navigate to /legacy/test/\_testLoader results in the following imported Ontology.

**Graphical user interface

Description automatically generated with low confidence**

**After Importing Legacy**

Graphical user interface, text, application, email

Description automatically generated

**Console after Importing Legacy**

Select the Model Browser tab and observe the empty model. Double click on it and observe the Episode Editor.

**Graphical user interface, application

Description automatically generatedEmpty Model after Loading**

<note>This tab should, after running tasks, show the full envisionment as a tree of nodes. Right now, it fails to do so</note>

<note>Ontology Browser tab should, after running tasks, show envisionments below their models. Not showing. Perhaps due to only class/subclass links being painted</note>

**Graphical user interface, text, application, email

Description automatically generatedModel Browser with Initial Conditions**

Using TSC:Run Task... the system will exercise the imported tasks, all those installed in the Agenda. That results in the following Console trace.

**Graphical user interface

Description automatically generatedConsole Trace after running Tasks**

This also results in a new view in the Model Browser when double clicking the first episode. <note>some bug in the code (nullpointerexception at org.nex.tinytsc.engine.Episode.compareEpisode(Episode.java:222)) seems to prevent the display of the two generated episodes, but they do appear in the Episode Editor as follows</note>

**Graphical user interface, text, application, email

Description automatically generatedEpisode Editor on Initial Conditions showing 2 Next Episodes**

Click Refresh on the Envisionment tab to see the XML trace of the result of a task firing.

**Graphical user interface, text, application, email

Description automatically generatedXML trace of the Envisionment**

### Envisionment Trace

.A model is taken as the initial conditions for an envisionment. It is then followed by episodes. As such, it is the root of a tree.

Shape, arrow

Description automatically generated

<model id="myExperiment">

<instanceOf>model</instanceOf>

<slot name="actors">

<sentence>

<predicate>foo</predicate>

<subject>foo1</subject>

<truth>true</truth>

</sentence>

<sentence>

<predicate>foo</predicate>

<subject>foo2</subject>

<truth>true</truth>

</sentence>

<sentence>

<predicate>bar</predicate>

<subject>bar1</subject>

<truth>true</truth>

</sentence>

</slot>

<slot name="relations">

<sentence>

<predicate>abuts</predicate>

<subject>foo1</subject>

<object>foo2</object>

<truth>true</truth>

</sentence>

<sentence>

<predicate>abuts</predicate>

<subject>foo2</subject>

<object>bar1</object>

<truth>true</truth>

</sentence>

</slot>

<slot name="states">

<sentence>

<predicate>happy</predicate>

<subject>foo2</subject>

<truth>true</truth>

</sentence>

</slot>

<nextEpisode>

<rule>rule-1</rule>

<node>E1161825719281</node>

</nextEpisode>

</model>

<episode id="E1161825719281">

<instanceOf>episode</instanceOf>

<myMechanism>rule-1</myMechanism>

<slot name="actors">

<sentence>

<predicate>foo</predicate>

<subject>foo1</subject>

<truth>true</truth>

</sentence>

<sentence>

<predicate>foo</predicate>

<subject>foo2</subject>

<truth>true</truth>

</sentence>

<sentence>

<predicate>bar</predicate>

<subject>bar1</subject>

<truth>true</truth>

</sentence>

</slot>

<slot name="relations">

<sentence>

<predicate>abuts</predicate>

<subject>foo1</subject>

<object>foo2</object>

<truth>true</truth>

</sentence>

<sentence>

<predicate>abuts</predicate>

<subject>foo2</subject>

<object>bar1</object>

<truth>true</truth>

</sentence>

</slot>

<slot name="states">

<sentence>

<predicate>sad</predicate>

<subject>foo1</subject>

<truth>true</truth>

</sentence>

<sentence>

<predicate>happy</predicate>

<subject>foo2</subject>

<truth>true</truth>

</sentence>

</slot>

<previousEpisode>

<rule>rule-1</rule>

<node>myExperiment</node>

</previousEpisode>

</episode>

<note>missing </model></note>

1. JDBM: http://jdbm.sourceforge.net/ behaves essentially as a persistent hashmap. [↑](#footnote-ref-1)