

Teppefall User Guide

This guide applies to:

Teppefall Capture
Teppefall Colorspace
Teppefall Layout
Teppefall FX
Teppefall SDK (Also known as Darkstar and DSR)

About

This is a simple user guide to Teppefall software. These configurations have been tested on Windows Vista, XP, Fedora, CentOS and Mac OS X.

Teppefall application parameters (Java)

-Dteppefall.antialias=true|false

Use anti aliased text in Teppefall components.

-Dteppefall.chrome=true|false

Turn on/off native chrome.

-Dteppefall.console-date=true|false

Add date to console output.

-Dteppefall.fullscreen=true|false

Display primary frame as full screen on startup.

-Dteppefall.heavy-menus=true|false

Heavyweight menus.

-Dteppefall.offline=true|false

Turns off the update system.

-Dteppefall.language=no|en

Force language.

-Dteppefall.memorymanager=true|false

Turn on/off memory manager. Writes an error message when the system has less than one megabyte of memory left.

-Dteppefall.cursor-monitor=true|false

Fix for a Swing bug where the resize cursor gets stuck when resizing a frame.

-Dteppefall.nx=true|false

Set by executable files. Used by DSR to override the overridden application parameters.
Example -Xchrome and -Dteppefall.chrome.

-Dteppefall.override=true|false

Override the default LAF setup. Add this if you need to set swing.defaultlaf.

-Dteppefall.quality-rendering=true|false

Use quality rendering in Teppefall components.

-Dteppefall.state-verbose=true|false

Display a message every time application state is written to disk or cached.

-Dteppefall.tooltips=true|false

Turns tooltips on/off.

-Dteppefall.script.timeout

Define how long the application should wait for the scripting system to initialize. A system that runs very slowly might require a higher value. Default is 60 seconds.

-Dteppefall.darkstar-license

Lets you set the license key.

-Dteppefall.disable-appleextras=true|false

Not supported.

-Dteppefall.layout.noerase

Not supported.

-Dteppefall.layout.translet

Not supported.

Teppefall executable parameters (native)

-Xinsecure

FX, Layout and DSR runs under a security policy by default (DSR.policy). This parameter disables it.

-Xchrome

Turns on native chrome. Default is off. Remote desktop viewers like VNC might have problems with non-native chromed applications.

-Xgraphite

Turns on Substance Graphite look and feel. Default is Teppefall Titanium. Graphite is faster than Titanium.

-Xnative

Turns on the native look and feel.

-Xoffline

Turns off the update system.

JAVA_HOME

The Java home environment variable must be defined under Linux in order for the executables to work.

Teppefall Titanium Look And Feel

Titanium is a Swing Look And Feel based on Substance. It comes in a dark and white version.

This is how you launch a Teppefall application, with Titanium, from the command line.

```
java  
-Dteppefall.chrome=false  
-Dteppefall.override=true  
-Dswing.defaultlaf=com.teppefall.ds.look.TitaniumLAF  
-jar teppefalllayout.jar
```

Teppefall Layout

The runtime source code is included in the Teppefall Layout distribution. In theory the vendor lock in is minimal. I say theory, because some people can not even recognize that this is all based on stone age JavaBean logic from Sun. You are simply using Sun API's with a little sugar on top. Look up XMLDecoder if you do not get it. The Importer source code is located in the Teppefall Layout source/ folder.

Static import

```
Assembly assembly = Importer.decode(getClass()); // Hello.java + Hello.xml  
Assembly assembly = Importer.decode(getClass(), "random.xml");  
Assembly assembly = Importer.decode(getClass(), InputStream);  
Jpanel panel = assembly.getPanel();  
Jcomponent component = assembly.getComponentById("id")
```

Dynamic import

```
add(new Layout("hello.jfc"), BorderLayout.CENTER);
```

JFC markup language

```
jfc (component,layout,gridlayout,action,object,group,listener,vector,hashtable)  
@title  
@width  
@height
```

```
component (component,layout,gridlayout,rigid,glue,filler,attribute,property)  
@id  
@name
```

@text
@enabled
@action
@mnemonic
@icon
@focus
@constraint

Notes:

ID's are used through the element and parameter "idref". In Java code you use the `Assembly.getComponentById(String)` method.

layout (component,layout,gridlayout,rigid,glue,filler)

@link
@class
@constraint

property

@name
@value
@type=boolean|byte|char|short|double|float|int|long|string|hashtable|vector|object|idref|
component|color|font|insets|dimension|rectangle|point|icon|tree
@deriveFrom (font only)

Notes:

color=#ABCDEF
insets|dimension|rectangle|point=0,0
hashtable|vector|component=Require child elements.
font=Use @deriveFrom=\$ID in order to get the parent font.

rigid

@value=0,0

glue

@type=vertical|horizontal

filler

@minimum
@maximum
@preferred

action | object

@id
@name
@class

group(idref)

idref
@id
@name

listener
@type=caret|undo|redo|selected
@source
@undo
@redo
@button

attribute
@name
@value

gridlayout
@columns
@rows

vector | hashtable (item)
@id

item
@name
@value

Teppefall FX

A Teppefall FX Javascript renderer must implement D2DRenderer. The current scripting engine does not create a new scope on reload and you will encounter many strange problems if you depend on globally scoped variables. If you need a fresh scope you must open a new instance of Teppefall FX by pressing CTRL-SHIFT-N.

Teppefall FX supports rendering over RMI, rendering via a J2EE Servlet (mapped up as an image in web.xml) and local PNG/JPEG export. It can import images through Quicktime and supports video libraries such as VXP and JMyron.

Script context

```
context.getApplication()  
context.getApplication().getApplicationFrame()  
context.getApplication().getApplicationName()  
context.getDarkstar()  
context.getStatus()
```

```
context.info("message")  
context.warn("problem")  
context.status("Yo!")
```

Example use:

```
importPackage(Packages.java.awt)
importPackage(Packages.com.teppefall.ds.render2d)

var renderer = {
  render : function(g, size) {
    g.setColor(Color.BLACK)
    g.fillRect(0, 0, size.width, size.height)
  }
}
context.getDarkstar().setRenderer(new D2DRenderer(renderer))
```

Teppefall FX Server

Teppefall FX supports rendering over RMI. This is very slow to start up and we only recommend you use it in prototypes. There is a Servlet based FXServlet that is much faster and J2EE compatible. We use it in our CAPTCHA system. Contact Teppefall for more information.

Teppefall Runtime (DSR/SharedVM)

Teppefall DSR is an Internet aware application launcher for Darkstar applications. It makes it easier to test multiple builds on multiple platforms. It also supports launching multiple Darkstar applications at the same time through SharedVM. SharedVM has no separation between applications, no security and one application can take down all the other applications if it fails. SharedVM should only be used for testing or non critical applications. If you need launch on demand functionality you can test out SharedVMLauncher and Erratic. It launches applications over a socket based system.

Please note that SharedVM only properly exists the last remaining application. Your shutdown code might not work. This is due to the threading complexity of adding multiple shutdown hooks into a shared VM instance.

Java

```
java -jar dsr.jar example.ExampleApplication arguments
```

Java SharedVM

```
java -jar dsr.jar example.ExampleApplication1,example.ExampleApplication2
```

Executable

```
Usage: DSR example.ExampleApplication arguments
```

Executable SharedVM

```
Usage: DSR example.ExampleApplication1,example.ExampleApplication2
```

Web start launcher

/app/Example?class=example.ExampleApplication&args=arguments&jars=1,2&mime=jnlp

The Web Start launcher lets the user add an icon on their desktop that launches the application in the same way as a desktop application.

Teppefall Scripting

The Javascript based scripting system is based on BSF and Rhino. You can modify Teppefall applications through the context.js file that is run on startup. Since starting the engine is very slow it runs asynchronously to the application itself and is only available after the application has been made visible.

Non standard behavior

Some of Colorspace's functionality depends on the Substance look and feel. Running Colorspace under a native look and feel causes the color picker initialization to fail.

Understand that DSR is not completely transparent. Some parameters may fail to operate in the intended manner and the differences may differ on a build to build basis. This is due to internal changes in the application substructure forced by operating system and VM specific bugs. The source code for DSR is available in the SDK if you need more information.

Copyright

Copyright 2008 Teppefall Digital Layout

<http://www.teppefall.com>

<http://www.teppefall.no>

<http://labs.teppefall.com>

<http://app.teppefall.com>