# TapirAddOn

AddOnMechanism > TapirAddOn

## The Tapir E-Theses Add-On

In place of describing the structure and installation of the Tapir again, here are some links to some resources which already do this. Later I will try and distill this to a more simple set of steps:

1. Jones, . 2004. "The Tapir: Adding E-Theses Functionality to DSpace". Ariadne, Issue 41. http://www.ariadne.ac.uk/issue41/jones/
2. Jones, . 2005. "Incorporating Local Developments to DSpace". DSpace User Group Meeting 2005 http://www.ub.uib.no/prosj/OA/Presentasjoner/J-DSUG-2005.ppt
3. DSpace Installation and Systems Administration Procedures for the Edinburgh esearch Archive (EA) (Slightly dated version) http://www.thesesalive.ac.uk/archive/EAInstallation-1.9.pdf

(1) describes more the development approach that we took for the Tapir, and does not deal so heavily with installation; it does suggest some ways in which DSpace may be better designed to deal with add-ons, but in no deep way.

(2) is the paper I gave at DSUG 2005, which describes how the tapir is structured (in a very general way) and the installation requirements; it proposes an alternative installation also, with some comments as to the pros and cons of each method.

(3) is a comprehensive installation guide for the previously live version of EA; since then Tapir and EA have been upgraded, but the documentation for this is not yet available online (I will endeavour to do this ASAP).

The Tapir source code can be found at http://sourceforge.net/projects/tapir-eul
Further Tapir related documentation can be found at http://www.thesesalive.ac.uk/arch_eul-dspace.shtml

## Tapir Add-On Techniques

Here the current installation mechanism for the Tapir is described in order to see where this process could be improved by a build-time add-on installation process. In italics I have placed suggestions as to how this might impact a generalised add-on mechanism and what solutions *might* be appropriate.

## Pre-installation preparations

Tapir uses some specific qualified Dublin Core style metadata elements. These need to be pre registered in the DC egistry before installation begins, or there will be errors thrown once the software is installed.
*Perhaps one feature of an add-on mechanism would be to provide a tool to automatically register required DC elements during the installation process*

## Source code compilation

Tapir has it's own ant build file, which knows how to install a compile a jar file into the pre-installation DSpace lib directory. This is done with a command like the following:

```
% ant -Ddslib=dspace/lib -Dtarget=dspace-source/lib -Dconfig=dspace/config/dspace.cfg install
```

This defines the dspace lib to build against, since Tapir uses some DSpace classes, the target directory into which the resulting jar should be placed and the live dspace config file from which Tapir can get any of the extra details it needs. The ant target is then "install".
Perhaps one way of managing add on installation is to have a series of build files passed to the main dspace build file as arguments to be run with specified values once the initial compilation is complete. This would allow Tapir, for example, to be compiled against the DSpace jar pre-installation lib directory and pulled in before the final deployment/creation of the war file is done. This would reduce the three stage installation process (build dspace, build tapir, re-build dspace) to one stage"

## Database updates

Tapir also brings with it some schema changes to the database. These are installed by another build target called "database" and is invoked thus:

```
% ant -Ddslib=dspace/lib -Dconfig=dspace/config/dspace.cfg database
```

Again this requires to know where the live DSpace library is so it can use the database management classes, and it needs access to the live config for any further information that it needs.
*A standardised way of applying schema changes should probably be used here. Tapir uses the native DSpace InitializeDatabase class to execute an sql file against the database. Perhaps a standardised form of an add on would have an sql directory which the DSpace build file would test for and run the scripts in during build time*

## Web.xml and taglib

Tapir requires several new servlets and replaces some of the DSpace tags with new ones. This requires manual intervention at the moment, although a patch file alternative would also be possible. It is currently necessary to manually insert the servlet mappings, filter mappings and url patterns into dspace-web.xml and update the tags in dspace-tags.tld prior to re-building DSpace for the final time before installation is complete.
"A couple of solutions seem likely:
1 Pure patch files: these could be placed in a "patches" directory in the add on, for example, and run automatically by the DSpace build file.
2 A full XML management solution whereby an installation process knows how to read in correctly formatted web.xml files from the add-on and combine them with the dspace-web.xml file to create the fully featured web.xml file"

## JSP installation

This section is the most confusing because in some cases it is OK just to copy the jsps from Tapir to the jsp/local directory, while in other cases it is necessary to manually merge similar JSPs from each package.
*The obvious solution is for add-ons to provide a patch file for the JSPs, although this would not be reliable if multiple add-ons were being installed. The DSpace build file could then install patches from the "patches" directory of the add-on source, say.*

## Updating configuration

Updating the configuration options in DSpace is straightforward in Tapir, since it does not change any settings, it merely adds new ones. Therefore, the configuration is updated thus:

```
% cat tapir/config/tapir-dspace.cfg >> dspace/config/dspace.cfg
```

Mostly it would seem unlikely that add-ons would modify configurations, so patching would not be necessary here. Instead documentation regarding the new configuration should be provided, and a module.cfg file could be catted onto the end of the standard dspace.cfg file easily. Each add-on should probably therefore have a "config" directory in which this is contained"

## Other configuration options

Tapir also needs to copy a couple of other files into the live configuration directory of DSpace. This covers all the new licencing system features that come with it. At the moment this is done manually from the Tapir config directory to the *live* DSpace config directory.
*DSpace has some noticeable issues with regard to how configuration is managed. After the initial installation there is no process other than a total fresh_install which copies config files from the source to the live directories. Perhaps this needs to be addressed, since add-ons may be installed any time, and will want their config options in the live directory for sure, and probably also in the source directory for completeness. That way the add-on config options could be installed into the source config directory and then re-deployed when DSpace is re-build for the final time.*

## Upgrading from previous versions of Tapir

Since Tapir has database schema changes, new versions sometimes require an update script to be run. This takes the form of a command like:
{{

```
Unknown macro: {% ant -Dconfig=dspace/config/dspace.cfg upgrade_0.3_0.4}}
```

}
So there is an ant target "upgrade_0.3_0.4" which then runs an InitializeDatabase process using some more sql commands.
*This is the sort of sql that you probably don't want to run until after all the other installation has completed. Perhaps it would be a good idea to have, in the "sql" directory of the add-on a "pre" and "post" directory for scripts to be run during the installation and after the installation. There is also the possibility that some of these targets perform other operations such as calling custom java processes, in which case perhaps it would be better to have these ant targets called from the DSpace build file rather than it executing purely SQL directly*

## Tapir cron jobs

Tapir has a couple of cron jobs which need to be set up in order to perform basic system maintenance.
*This sort of thing is probably best left to the add-on documentation. Perhaps after the build the user should be presented with a list of files regarding documentation that they need to read in order to finish up the installation themselves*

## Final thoughts

Based on the above thoughts, it would make sense for the Tapir add-on package to be structured with some or all of the elements as follows:

```
/tapir #top level add-on directory
/tapir/build.xml #the build file
/tapir/build #the target build directory for the source code
/tapir/src #the top level directory for the java source
/tapir/jsp #the top level directory for the new JSPs

    1. This should be sub-structured similarly to the DSpace JSP directory
       /tapir/config #the top level config directory
       /tapir/config/dspace.cfg #the tapir config options for dspace
       /tapir/config/additional #directory for module specific config files
       /tapir/etc/sql #top level directory for SQL scripts
       /tapir/etc/sql/pre.sql #SQL script to be executed during installation
       /tapir/etc/sql/post.sql #SQL script to be executed after installation
       /tapir/etc/web.xml #The web.xml file which needs to be combined with the dspace-web.xml
       /tapir/etc/tags.tld #the tags that need to be combined with the dspace-tags.tld
       /tapir/etc/dcfields.xml #an XML representation of the required DC fields for the add on
       /tapir/patches/web.xml.diff #a diff to patch the dspace-web.xml file
       /tapir/patches/tags.tld.diff #a diff to patch the dspace-tags-tld file
       /tapir/patches/jsp.diff #patch file for JSPs which need replacing
```

Therefore, a quick stab at suggesting the way that the DSpace build file might be redone follows:
First of all, the installation process might take the following form, in order to make life a little easier:

```
% ant
% ant fresh_install
% ant -Dinstall=add on directory addon
% ant -Dcopy_configs=true update
% ant -Dinstall=other add on directory addon
% ant -Dcopy_configs=true update
```

This is quite a few commands, but unless we can pass an array, multiple values per variable, or a comma delimited string to ant parameters (which I don't think you can after a minute or so of googling), there is not an obviously easy alternative.
So the DSpace traditional build will work much as before, although it will have the additional copy_configs flag which would default to false unless set otherwise by the calling user. Meanwhile we set a new build target "addon" which would perform the operations needed sequentially to the add on directory. These would be approximately:

```
1 % /dspace/bin/dsrun org.dspace.app.itemimporter.DCImporter addon/etc/dcfields.xml
2 % ant -f addon/build.xml -Dcfg=dspace/config/dspace.cfg -Ddsclass=dspace/build/classes -Ddslib=dspace/lib install
3 % ant -f addon/build.xml -Dcfg=dspace/config/dspace.cfg dbschema
4 % patch -u dspace/etc/dspace-web.xml addon/patches/web.xml.diff
5 % patch -u dspace/jsp/WE-INF/dspace-tags.tld addon/patches/tags.tld.diff
6 % cp -r addon/jsp/* dspace/jsp/local
7 % patch -u addon/patches/jsp.diff # or however it is that patching an entire directory works
8 % cat addon/config/dspace.cfg >> dspace/config/dspace.cfg
9 % cp -r addon/config/additional/* dspace/config
10 % ant -f addon/build.xml -Dcfg=dspace/config/dspace.cfg finally
```

There are some alternative methods of doing some of these as shown here:

```
3 % /dspace/bin/dsrun org.dspace.storage.rdbms.InitializeDatabase addon/etc/sql/pre.sql
4 % /dspace/bin/dsrun org.dspace.administer.WebXMLConfigurator addon/etc/web.xml dspace/etc/dspace-web.xml
5 % /dspace/bin/dsrun org.dspace.administer.TagLibConfigurator addon/etc/tags.xml dspace/jsp/WE-INF/dspace-tags.tld
10 % /dspace/bin/dsrun org.dspace.storage.rdbms.InitializeDatabase addon/etc/sql/post.sql
```

So that's basically just a pile of half-formed ideas at the moment. I'll come back and rationalise this some more shortly. In the mean time a couple of last things:

- the order in which stuff is done will be important, in case of failure part way through
- The choice of which alternative solutions would be, for me, I think to defer always to the add on build file
- patching JSPs directly might be a bad idea. Possibly creating copies in local and then patching that would be better