

# Subversion Release Manager

## Plugin Information

No information for the plugin 'svn-release-mgr' is available. It may have been removed from distribution.

This plugin allows you to set up a job in Jenkins for building specific revisions of a project.

Be sure to select both "Use Subversion Release Manager" in the top section of the job configuration page, and select "Subversion Release" for the SCM type, rather than the standard Subversion SCM.

Here is a screenshot of this project configured to use the plugin.

**Hudson** search ?

Hudson > SubversionReleasePlugin > Releases ENABLE AUTO REFRESH

[Back to Dashboard](#)  
[Status](#)  
[Changes](#)  
[Workspace](#)  
[Build Now](#)  
[Delete Project](#)  
[Configure](#)  
[Modules](#)  
[Releases](#)

**Build History** (trend)  
#0 Mar 27, 2009 3:03:12 PM  
Production  
[for all](#) [for failures](#)

## Releases

Compare

	Revision	Author	Date	Message
<input type="checkbox"/>	16633	piascikj	Fri Mar 27 14:19:12 CDT 2009	[maven-release-plugin] prepare for next development iteration
<input checked="" type="checkbox"/>	#0		Fri Mar 27 15:03:12 CDT 2009	<b>Production</b>
<input type="checkbox"/>	16631	piascikj	Fri Mar 27 14:18:51 CDT 2009	[maven-release-plugin] prepare release svn-release-mgr-0.1
<input type="checkbox"/>	16630	piascikj	Fri Mar 27 14:15:35 CDT 2009	prepare for release with modified release number in POM
<input type="checkbox"/>	16621	piascikj	Fri Mar 27 00:00:26 CDT 2009	[maven-release-plugin] prepare for next development iteration
<input type="checkbox"/>	16619	piascikj	Thu Mar 26 23:59:07 CDT 2009	[maven-release-plugin] prepare release svn-release-mgr-1.0
<input type="checkbox"/>	16618	piascikj	Thu Mar 26 23:55:51 CDT 2009	re-release
<input type="checkbox"/>	16616	piascikj	Thu Mar 26 23:48:56 CDT 2009	[maven-release-plugin] prepare release svn-release-mgr-1.0
<input type="checkbox"/>	16615	piascikj	Thu Mar 26 23:45:28 CDT 2009	fix description before release
<input type="checkbox"/>	16614	piascikj	Thu Mar 26 23:30:58 CDT 2009	[maven-release-plugin] prepare for next development iteration
<input type="checkbox"/>	16612	piascikj	Thu Mar 26 23:29:35 CDT 2009	[maven-release-plugin] prepare release svn-release-mgr-1.0
<input type="checkbox"/>	16611	piascikj	Thu Mar 26 23:03:49 CDT 2009	re-release of 1.0
<input type="checkbox"/>	16610	piascikj	Thu Mar 26 21:37:58 CDT 2009	[maven-release-plugin] prepare for next development iteration
<input type="checkbox"/>	16608	piascikj	Thu Mar 26 21:34:31 CDT 2009	for release
<input type="checkbox"/>	16601	piascikj	Thu Mar 26 20:33:18 CDT 2009	[maven-release-plugin] prepare release svn-release-mgr-1.0-BETA
<input type="checkbox"/>	16595	piascikj	Thu Mar 26 20:18:10 CDT 2009	Back to SNAPSHOT for release
<input type="checkbox"/>	16585	piascikj	Thu Mar 26 17:34:09 CDT 2009	Fixed xstream serialization error.
<input type="checkbox"/>	16584	piascikj	Thu Mar 26 15:43:58 CDT 2009	update version to BETA
<input type="checkbox"/>	16583	piascikj	Thu Mar 26 15:26:53 CDT 2009	Changed build now link, and added ProjectReleaseAction to Run.
<input type="checkbox"/>	16517	piascikj	Wed Mar 25 01:13:45 CDT 2009	maxRevisions and compare validation
<input type="checkbox"/>	15973	piascikj	Tue Mar 03 11:43:22 CST 2009	Added url to pom.xml

Hudson ver. 1.293

Today it uses a copy of SubversionSCM named SubversionReleaseSCM, but would only require a small change to SubversionSCM.CheckOutTask to work with the core code.

Here is the change that is needed. Just look for the java comments that start with EDIT.

1. Add a String field for the revision
2. In the constructor, retrieve the revision number from the environment variable.
3. Set the revision to check out based on the value stored in revision field if not null

Here is a link to the patch request: [JENKINS-3207](#)

## SubversionSCM.java CheckOutTask

```
/**
 * Either run "svn co" or "svn up" equivalent.
 */
private static class CheckOutTask implements FileCallable<List<External>> {
    private final ISVNAuthenticationProvider authProvider;
    private final Date timestamp;
    // true to "svn update", false to "svn checkout".
    private boolean update;
```

```

private final TaskListener listener;
private final ModuleLocation[] locations;
//EDIT (1) next line added by piascikj for building specific revision
private String revision;

public CheckOutTask(AbstractBuild<?, ?> build, SubversionSCM parent, Date timestamp, boolean update,
TaskListener listener) {
    this.authProvider = parent.getDescriptor().createAuthenticationProvider();
    this.timestamp = timestamp;
    this.update = update;
    this.listener = listener;
    this.locations = parent.getLocations(build);
    //EDIT (2) next line added by piascikj for building specific revision
    this.revision = build.getEnvVars().get("REVISION");
}

public List<External> invoke(File ws, VirtualChannel channel) throws IOException {
    final SVNClientManager manager = createSvnClientManager(authProvider);
    try {
        final SVNUpdateClient svnuc = manager.getUpdateClient();
        final List<External> externals = new ArrayList<External>(); // store discovered externals to
here
        //EDIT next line removed by piascikj for building specific revision
        //final SVNRevision revision = SVNRevision.create(timestamp);
        //EDIT (3) next 6 lines added by piascikj for building specific revision
        SVNRevision revision = SVNRevision.create(timestamp);
        try {
            if (this.revision != null) revision = SVNRevision.create(Long.parseLong(this.revision));
        } catch (NumberFormatException e) {
            listener.getLogger().println("Unable to parse revision number from value: " + this.
revision + ", checking out HEAD revision.");
        }
        if(update) {
            for (final ModuleLocation l : locations) {
                try {
                    listener.getLogger().println("Updating "+ l.remote);

                    File local = new File(ws, l.local);
                    svnuc.setEventHandler(new SubversionUpdateEventHandler(listener.getLogger(),
externals,local,l.local));
                    svnuc.doUpdate(local.getCanonicalFile(), l.getRevision(revision), true);

                } catch (final SVNException e) {
                    if(e.getErrorMessage().getErrorCode()== SVNErrorCode.WC_LOCKED) {
                        // work space locked. try fresh check out
                        listener.getLogger().println("Workspace appear to be locked, so getting a fresh
workspace");
                        update = false;
                        return invoke(ws,channel);
                    }
                    if(e.getErrorMessage().getErrorCode()== SVNErrorCode.WC_OBSTRUCTED_UPDATE) {
                        // JENKINS-1882. If existence of local files cause an update to fail,
                        // revert to fresh check out
                        listener.getLogger().println(e.getMessage()); // show why this happened.
Sometimes this is caused by having a build artifact in the repository.
                        listener.getLogger().println("Updated failed due to local files. Getting a
fresh workspace");
                        update = false;
                        return invoke(ws,channel);
                    }
                }

                e.printStackTrace(listener.error("Failed to update "+l.remote));
                // trouble-shooting probe for #591
                if(e.getErrorMessage().getErrorCode()== SVNErrorCode.WC_NOT_LOCKED) {
                    listener.getLogger().println("Polled jobs are "+ Hudson.getInstance().
getDescriptorByType(SCMTrigger.DescriptorImpl.class).getItemsBeingPolled());
                }
                return null;
            }
        }
    }
} else {

```

```

Util.deleteContentsRecursive(ws);

// buffer the output by a separate thread so that the update operation
// won't be blocked by the remoting of the data
PipedOutputStream pos = new PipedOutputStream();
StreamCopyThread sct = new StreamCopyThread("svn log copier", new PipedInputStream(pos),
listener.getLogger());
sct.start();

for (final ModuleLocation l : locations) {
    try {
        listener.getLogger().println("Checking out "+l.remote);

        File local = new File(ws, l.local);
        svnuc.setEventHandler(new SubversionUpdateEventHandler(new PrintStream(pos),
externals, local, l.local));
        svnuc.doCheckout(l.getSVNURL(), local.getCanonicalFile(), SVNRevision.HEAD, l.
getRevision(revision), true);
    } catch (SVNException e) {
        e.printStackTrace(listener.error("Failed to check out "+l.remote));
        return null;
    }
}

pos.close();
try {
    sct.join(); // wait for all data to be piped.
} catch (InterruptedException e) {
    throw new IOException2("interrupted",e);
}

try {
    for (final ModuleLocation l : locations) {
        SVNDirEntry dir = manager.createRepository(l.getSVNURL(),true).info("/",-1);
        if(dir!=null) { // I don't think this can ever be null, but be defensive
            if(dir.getDate()!=null && dir.getDate().after(new Date())) // see http://www.nabble.
com/NullPointerException-in-SVN-Checkout-Update-td21609781.html that reported this being null.
            listener.getLogger().println(Messages.SubversionSCM_ClockOutOfSync());
        }
    }
} catch (SVNException e) {
    LOGGER.log(Level.INFO,"Failed to estimate the remote time stamp",e);
}

return externals;
} finally {
    manager.dispose();
}
}

private static final long serialVersionUID = 1L;
}

```

## Change Log

### Version 1.2 (2010-11-03)

- Fix a couple help links

### Version 1.1 (2010-03-01)

- Thread safety fix for compare and build actions
- Change maxRevisions field validation to client-side
- Fix help link and add content
- Misc cleanup and javadoc updates
- Bump version up to 1.1 since there's a 1.0 out there even though 0.2 was the last release

### Version 0.2 (2009-12-29)

- Fix broken image
- Update uses of deprecated APIs

Version 0.1 (2009-03-27)

- Initial release