

Upgrading Mods With Git

[Set up Git](#)

[Obtain the Git Archive](#)

[Roll Back Git](#)

[Apply Changes](#)

[Commit Changes](#)

[Update the Archive](#)

[Resolve Conflicts](#)

[Commit Resolutions](#)

[Re-integrate Changes](#)

[When You're Done](#)

* * * * *

Follow along with these instructions if you already have Git installed. Two major directories are involved in this process; the “addon” archive is the AddonSoftware archive obtained from BASIS and the “mods” directory is the client’s project area containing customizations.

Note: The examples below use the fictitious username of `jd` for John Doe.

Set up Git [Back to top](#)

Enable three-way conflict resolution with the following command:

```
git config --global merge.conflictstyle diff3
```

Obtain the Git Archive [Back to top](#)

The Git addon archive is available at `ssh://git@git.basis.com/addon`. In order to access the archive, find or generate a public key and send it to shaney@basis.com. Your key will be added, and then you will be able to obtain the archive with the following command:

```
git clone ssh://git@git.basis.com/addon
```

Roll Back Git [Back to top](#)

Follow along to roll back the Git archive to the version of AddonSoftware in which the client’s changes were modified.

Note: Skip this step for subsequent upgrades to continue where you left off from your previous upgrade.

At a command line, `cd` to the top-level directory of the addon archive. For example, if John Doe keeps his archive at `/home/jdoe/merge/addon/`, the command would be:

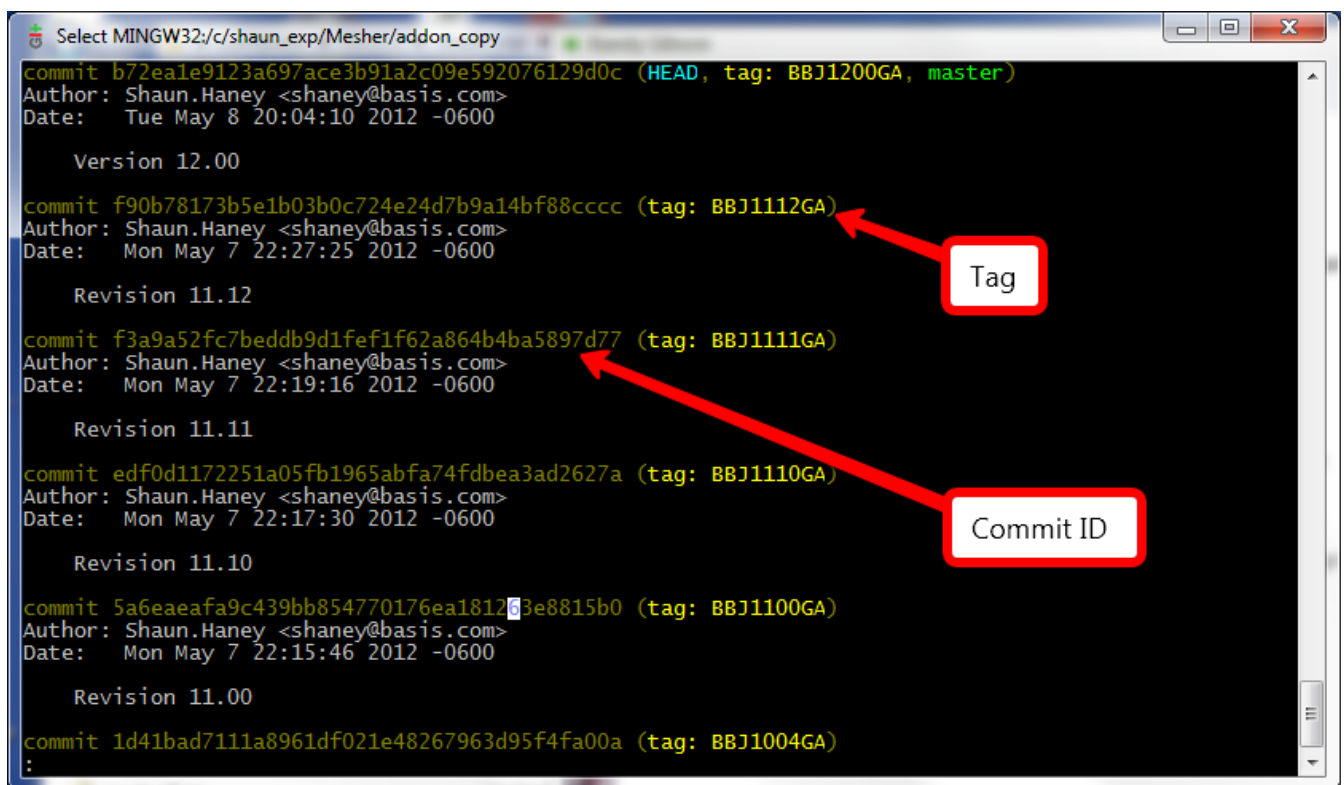
```
cd /home/jdoe/merge/addon
```

The files in this newly obtained archive, known as your workspace directory, is the latest revision of AddonSoftware that the BASIS archive conveniently tags. In order to find the tag for the rollback revision you want, execute the following command:

```
git log --decorate
```

This command shows the history of the archive from its present state, all the way back to its creation. The `--decorate` option tells Git to display any tags entered at a particular commit. The tag is simply a shorthand for the commit id, which can also identify a particular revision to roll back to.

In the example below, you could use tag `BBJ1111GA` to roll back to revision 11.11 or the commit id `f3a9a52fc7beddb9d1fef1f62a864b4ba5897d77`.



The screenshot shows a terminal window with the following output:

```
Select MINGW32:/c/shaun_exp/Meshier/addon_copy
commit b72ea1e9123a697ace3b91a2c09e592076129d0c (HEAD, tag: BBJ1200GA, master)
Author: Shaun.Haney <shaney@basis.com>
Date: Tue May 8 20:04:10 2012 -0600

Version 12.00

commit f90b78173b5e1b03b0c724e24d7b9a14bf88cccc (tag: BBJ1112GA)
Author: Shaun.Haney <shaney@basis.com>
Date: Mon May 7 22:27:25 2012 -0600

Revision 11.12

commit f3a9a52fc7beddb9d1fef1f62a864b4ba5897d77 (tag: BBJ1111GA)
Author: Shaun.Haney <shaney@basis.com>
Date: Mon May 7 22:19:16 2012 -0600

Revision 11.11

commit edf0d1172251a05fb1965abfa74fdbea3ad2627a (tag: BBJ1110GA)
Author: Shaun.Haney <shaney@basis.com>
Date: Mon May 7 22:17:30 2012 -0600

Revision 11.10

commit 5a6eaeafa9c439bb854770176ea1812b3e8815b0 (tag: BBJ1100GA)
Author: Shaun.Haney <shaney@basis.com>
Date: Mon May 7 22:15:46 2012 -0600

Revision 11.00

commit 1d41bad7111a8961df021e48267963d95f4fa00a (tag: BBJ1004GA)
:
```

Two red arrows point from labels to the output. One arrow points from a box labeled "Tag" to the tag `BBJ1112GA`. Another arrow points from a box labeled "Commit ID" to the commit ID `f3a9a52fc7beddb9d1fef1f62a864b4ba5897d77`.

The command to roll back to revision 11.11 is:

```
git reset --hard BBJ1111GA
```

Apply Changes [Back to top](#)

BASIS has written a few convenient Java programs to help move code back and forth between your mods directory and the addon archive. While these Java programs require Java 7, the source is available to recompile it for Java 6.

Two programs are important for this step: `CallpointMerger.jar` and `ProgMerger.jar`. `CallpointMerger` copies “before” callpoints that contain “SKIP” and “CUSTOM” callpoints from the mods directory to their corresponding callpoints in the addon archive. This is done in such a way that Git recognizes any changed code in these callpoints and upgrades it accordingly. `ProgMerger` copies all programs from the prog directory into its corresponding spot in the addon archive.

To show how to use these two programs, let’s say John Doe’s addon archive is located at `/home/jdoe/merge/addon` and his mods directory is at `/home/jdoe/merge/mods`.

The first program copies the callpoints and takes the `cdf` directory from the mods directory and the Addon archive as follows:

```
java -jar CallpointMerger.jar /home/jdoe/merge/mods/data/cdf /  
home/jdoe/merge/addon/data/cdf
```

The second program copies the prog files and takes the prog directory under mods and the root directory of the addon archive as follows:

```
java -jar ProgMerger.jar /home/jdoe/merge/mods/prog /home/jdoe/  
merge/addon
```

Commit Changes [Back to top](#)

After running the scripts to copy the callpoints and the prog files, it is time to commit the changes.

```
git add .  
git commit
```

When issuing the Git commit, an editor appears in which you can log your change. You will also see numerous files that have been changed and how they’ve been changed. At the very top of the file, enter a comment that will help you remember 10 years from now that this is where you integrated code from the mods archive into the addon archive. Adding this comment does not affect the BASIS archive at all; you are working with a local copy that tracks of your changes. The only way to affect the BASIS archive would be to perform a “push,” which we disabled for this reason.

Update the Archive [Back to top](#)

Now, you’re ready to perform the actual update to the latest version! The following command brings in the latest changes, but won’t commit anything if conflicts result.

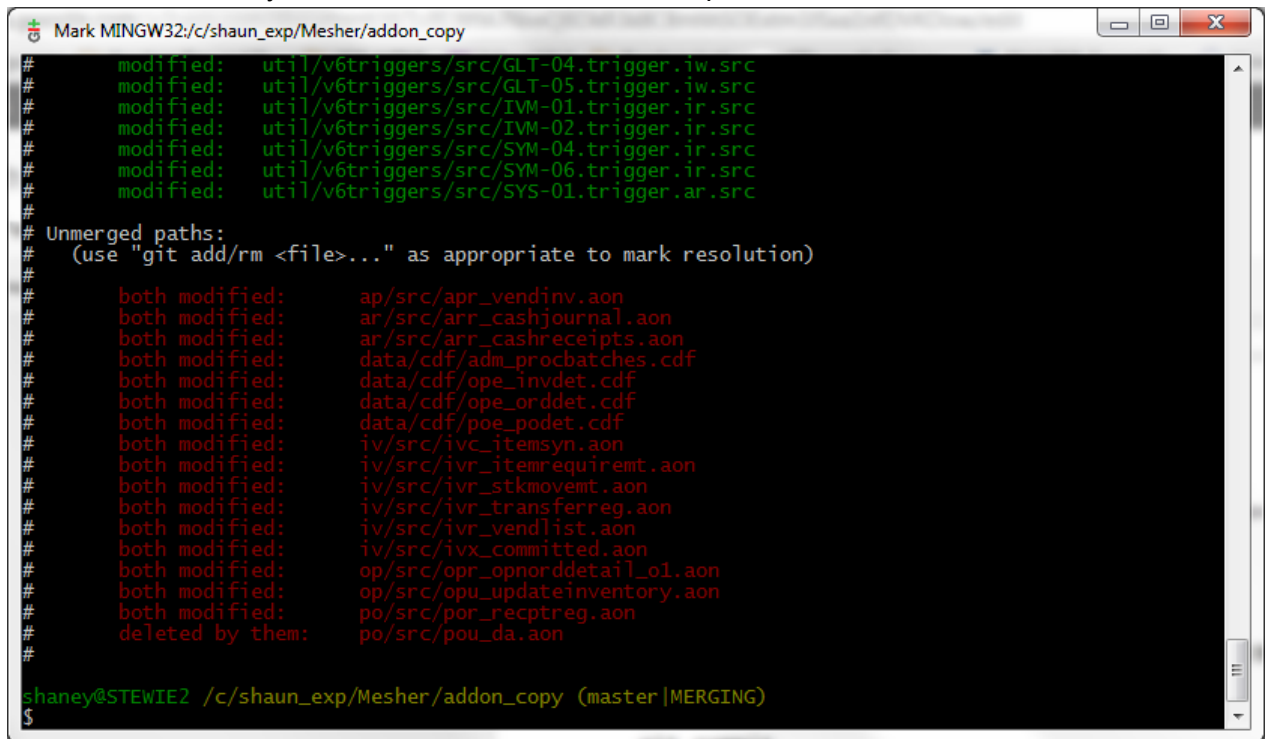
```
git pull ssh://git@git.basis.com/addon
```

Once you issue this command, you will see text pour across the screen indicating what Git is doing to each file in the archive. You will notice that Git takes care of updating most of the files itself, just leaving you with a handful of conflicted files to resolve.

Find out exactly what files contain conflicts with the following command:

```
git status
```

This command lists your conflicts at the end of the output as shown here:



```
Mark MINGW32:/c/shaun_exp/Mesher/addon_copy
#      modified:   util/v6triggers/src/GLT-04.trigger.iw.src
#      modified:   util/v6triggers/src/GLT-05.trigger.iw.src
#      modified:   util/v6triggers/src/IVM-01.trigger.ir.src
#      modified:   util/v6triggers/src/IVM-02.trigger.ir.src
#      modified:   util/v6triggers/src/SYM-04.trigger.ir.src
#      modified:   util/v6triggers/src/SYM-06.trigger.ir.src
#      modified:   util/v6triggers/src/SYS-01.trigger.ar.src
#
# Unmerged paths:
#   (use "git add/rm <file>..." as appropriate to mark resolution)
#
#      both modified:   ap/src/apr_vendinv.aon
#      both modified:   ar/src/arr_cashjournal.aon
#      both modified:   ar/src/arr_cashreceipts.aon
#      both modified:   data/cdf/adm_procbatches.cdf
#      both modified:   data/cdf/ope_invdet.cdf
#      both modified:   data/cdf/ope_orddet.cdf
#      both modified:   data/cdf/poe_podet.cdf
#      both modified:   iv/src/ivc_itemsyn.aon
#      both modified:   iv/src/ivr_itemrequiremt.aon
#      both modified:   iv/src/ivr_stkmovmnt.aon
#      both modified:   iv/src/ivr_transferreg.aon
#      both modified:   iv/src/ivr_vendlist.aon
#      both modified:   iv/src/ivx_committed.aon
#      both modified:   op/src/opr_opnorddetail_01.aon
#      both modified:   op/src/opu_updateinventory.aon
#      both modified:   po/src/por_recptreg.aon
#      deleted by them: po/src/pou_da.aon
#
shaney@STEWIE2 /c/shaun_exp/Mesher/addon_copy (master|MERGING)
$
```

Resolve Conflicts [Back to top](#)

Each file with conflicts will embed the conflict at the location where it occurs in the format shown below:

```
<<<<<<< HEAD
Changes from the mod directory.
||||| merged common ancestors
The original file before the merge took place.
=====
The latest Addon changes
>>>>>> b72ea1e9123a697ace3b91a2c09e592076129d0c
```

The portions above highlighted in green are the standard lines that Git uses to show a conflict. The last line shows the commit id is actually involved in the conflict.

The best way to demonstrate how to solve a conflict is to show an example. Suppose the file `ivr_stkmovmnt.aon` contains the following conflict:

```
<<<<<<< HEAD
rem --- Copyright (c) 1981-2007 AddonSoftware
||||||| merged common ancestors
rem --- Copyrights 1981-2011, BASIS International Ltd. All Rights
Reserved.
=====
rem --- Copyright BASIS International Ltd. All Rights Reserved.
>>>>>>> b72ea1e9123a697ace3b91a2c09e592076129d0c
```

This conflict occurs on the copyright line, and shows that the latest copyright has been set to:

```
rem --- Copyright BASIS International Ltd. All Rights Reserved.
```

The way to resolve this conflict is fairly trivial: Just erase everything in the conflict except for the latest change. In this case, delete both the AddonSoftware copyright and the BASIS copyright that specifies years. Of course, also delete the auxiliary lines that Git has placed in the file to assist in locating and resolving the conflict.

Once a conflict is resolved in a file, add the file back and commit the changes with the following lines:

```
git add iv/src/ivr_stkmovement.aon
```

Commit Resolutions [Back to top](#)

Once all conflicts are resolved, commit the resolutions with:

```
git commit
```

Enter a comment about the resolved conflicts. Now you're ready to bring the changes back into the mods directory.

Re-integrate Changes [Back to top](#)

Two programs, CopyBackCallpoints.jar and CopyBackProgs.jar, copy the changes back into the mods directory. To use these programs, follow these examples for John Doe:

```
java -jar CopyBackCallpoints.jar /home/jdoe/merge/mods/data/cdf /home/jdoe/merge/addon/data/cdf
java -jar CopyBackProgs.jar /home/jdoe/merge/mods/prog /home/jdoe/merge/addon
```

When You're Done

Keep your work! The Git archive now contains your local changes, independent of the BASIS archive. Keep your archive so that you're not starting from scratch with the next AddonSoftware upgrade. Continue to use this archive so you won't have to revisit previous conflicts each time you upgrade. For the next upgrade, skip the Roll Back Git step since your changes will be made against the archive in its current state.
