

# LDAD Web rsync

Greg Jackson, WFO Midland, TX  
Version 2.33  
June 8, 2010

## Overview:

Though designed to meet Southern Region CMS needs, the Southern Region LDAD Web rsync script is designed as a general rsync utility that can be used not only to rsync content to the web but accomplish other rsync-related tasks as well, including grid upload. Though useful beyond the confines of the Southern Region, this documentation will give examples typical of Southern Region use.

As part of the effort to consolidate the NWS web farms and move to a content management system (CMS), the Southern Region has implemented changes to the procedures for uploading web content office websites. These changes will ultimately allow load-balancing between the web farms, provide multiple upload points, and extend service backup functionality to include web products. The goal is the elimination of single points of failure for web services.

The chosen method for content upload is rsync. The Linux rsync utility is very versatile and greatly limits the amount of logic required to synchronize web content between the local office and NWS web servers.

For the rsync procedure to work as intended, it is critical that only one computer in each office is assigned the role of synchronizing web site content via rsync.

The new Linux-based LDAD servers provide the best option for this role due to built-in redundancy, standardization, support, and AWIPS connectivity. The rsync scripts provided are designed to make failover to ls3 and recovery to ls2 automatic. Other machines may be used as sources for data pushed through LDAD to office websites, but it is recommended that only LDAD actually synchronize with the web server directly.

There should be adequate space in /data/ldad for the great majority of sites to use this procedure, particularly if outdated web content is removed. The /data/ldad partition has approximately 34 GB of total storage space.

For locations that use more than one machine to rsync information to the web farm, it is very important that the two machines are not synchronizing the same modules. This configuration is not supported. Those sites that do not have LDAD (such as CWSUs, the FAA Academy and SMG), can use the same scripts on a Linux machine or Windows XP (running Cygwin). For those sites using a non-LDAD Linux machine, an ordinary user other than "ldad" can be used to set up and use these scripts.

## Directory Layout:

A standard directory layout will be required for all office websites to ensure that proper support can be provided to Southern Region offices. The tables in Appendix C show the relationship between locations on the web, corresponding web server Linux file system paths, rsync module names, and LDAD (or local host) directories both before and after CMS implementation. A description of each location is also provided. The corresponding configuration options in `ldad_web_config.sh` are shown below each table. Typically, only RFCs would use the archive and `ahps2` modules.

There are file type restrictions that will apply to rsync on the CMS server and development CMS server. See Appendix D for information on excluded files.

Note that some paths may not be browsed on the web (like `apps-data`). This provides a basic level of information protection in these locations. These locations are read by site scripts to provide dynamic application data.

## Operational Overview:

The `ldad_web_push.sh` script is run once a minute to determine whether changes have occurred in any of the LDAD (local host) web directories.

An rsync is run by the script when a change is detected in a module and one of the following is true:

- 1) an “**update**” file is found in `/data/ldad/web/tmp` corresponding to the module name (i.e., `maf_html.update`). The update file is created by a user to activate synchronization of the web modules. An rsync only actually occurs if a new file has been added or an existing file has been modified. File deletions are not synchronized with an “update”.
- 2) the module is an operational module, which contains “live data”, like the `apps`, `rt_images`, `fxc`, `ahps2` or archive modules.
- 3) a “**force**” file (like `maf_html.force`) exists in the tmp directory. Since no changes are detected by the script when files are deleted and no files have been added or changed, a “force” file may be required to delete files. Though creation of a “force” file is okay in scripts that are manually triggered, it would be a bad idea for scripts that are triggered automatically (mindlessly). Note that file deletions will still not occur for any module listed as a `no_delete_module` in `ldad_web_config.sh`.
- 4) a “**pull**” file (i.e., `maf_images.pull`) can be used to trigger a one-time synchronization that pulls from the web server to the local machine before completing a normal rsync. This option can be used to get files that may only exist on the web server, like images stored using the CMS interface or operational files for a backup office.
- 5) the “**delete**” file (i.e., `maf_images.delete`) can be dropped into the tmp directory to force a one-time synchronization that includes the `--delete-after` option. The delete option is normally included in synchronizations by default; however, those modules listed in the `no_delete_modules` list in the configuration files as well as backup operational modules

do not normally include this option. This trigger can be used to accomplish “clean up” on the web server. It is advisable to pull content and delete unnecessary content before using this option to avoid accidental deletions.

- 6) The “*special*” file (i.e., maf\_images.special) in the tmp directory triggers an rsync using the contents of the file as rsync include/exclude rules using the rsync --include-from switch. When this control file is used to trigger synchronization of a web module, the “exclude” file (described later) is ignored, if it exists, so all necessary include/exclude rules must be in this file. The use of the “special” file can be useful to do things like rsync only a specified subdirectory, like images used for news headlines. The code below is an example maf\_images.special that could be used for that purpose:

```
+ /headline
+ /headline/**
- *
```

In plain English, this set of pattern rules says to include the news directory and all subdirectories and files within it, but exclude everything else. For more details, type “man rsync” in a Linux shell and look for the section on “EXCLUDE PATTERNS”.

There are seven situations that *prevent* an rsync from occurring, even for live data, and one situation that results in a non-fatal warning (which also indicates an incomplete rsync):

- 1) a “*block*” file has been placed in the tmp directory (i.e., maf\_html.block). This file is never automatically created or removed. It can be used to prevent attempts to update modules not in use or that do not exist on the web farm.
- 2) A “*wait*” file is in the tmp directory (i.e., maf\_images.wait). When a wait file is found that is less than a minute old, rsync will be skipped. After a minute, the file is ignored and deleted. This file can be used to briefly prevent an rsync while files are being created, usually by script (hence the short lifetime of the wait). This control file is generally not necessary, but this file might be used if both a “special” and a “delete” or “pull” control file are being created for a single rsync. To ensure that both the “special” and other control files are processed in the same run, the “wait” file can be created before the other control files are created. Once all required control files are in place, it is a good idea to delete the “wait” file, though it will be deleted once it becomes a minute old.
- 3) “*lock*” file corresponding to the module name is found in the tmp directory (i.e., maf\_html.lock). The lock is created when an rsync has been triggered for a module and is removed upon completion. The lock file’s purpose is to prevent two rsync processes from attempting to sync the same module simultaneously. If a lock file exists and the parent process no longer exists, it is ignored and syncs can continue. If a lock file exists for more than 15 minutes, the script will kill the corresponding rsync process if it continues to run.
- 4) an “*error*” file (i.e., maf\_html.error) has been created in the tmp directory. This file contains the rsync error number returned. As long as an error file exists, attempts to rsync the module occur only every 10 minutes due to the likelihood that the cause of an rsync failure is likely to last on the order of tens of minutes. Error files that have not been touched for a while are deleted and normal synchronization continues.

- 5) a “**huge**” file (i.e., tua\_archive.huge) has been created in the tmp directory. This file is created when an rsync times out. This will primarily occur when the size of the module is very big. As long as this file exists, attempts to rsync are scaled back according to the setting of huge\_module\_delay (defaults to 900 seconds). The file is removed, allowing more frequent updates, when a successful rsync is completed within the time specified by default\_timeout (normally 60 seconds).
- 6) an “**alive**” file (i.e., maf\_apps.alive) is created for every module every time ldad\_web\_push.sh runs on the cron, by default. This behavior can be disabled by setting block\_alive\_stat in the configuration file to 1. The alive files allow the administrator to identify currently functional control file names and view other module settings. The contents of the file (in order) include the configuration settings used, the module definition, the remote rsync server and module, the local directory that is synchronized with the module, and the module’s stat file identification. The alive files can help troubleshoot configuration problems. Below is an example of maf\_apps.alive:

```
operational_modules
apps|apps-data
martha2.srh.noaa.gov::maf_apps
/data/ldad/web/apps-data/maf
maf_apps.stat
```

- 7) (optionally) if the rsync server cannot be pinged; however, NOAAnet blocks ping traffic by default, preventing use of this option unless an exception has been obtained by opening a ticket with NOAAnet. The ping option should be disabled in the ldad\_web\_config.sh file unless an exception has been created and tested. If firewall exception have been created on NOAAnet, this is a useful option that help save execution time when servers are out for an extended period.
- 8) a “**warning**” file (i.e., crp\_images.warning) indicates that a non-fatal rsync error has occurred. This file contains the rsync error code that caused the warning. Though this condition is considered a successful rsync (a retry will not occur), it does indicate that some files could not be synchronized. A remote rsync server permissions issue (error 23) will cause a warning to be generated. Inspecting the log file will reveal which files could not be updated. It may take a system administrator’s assistance to result permission problems. Sites should monitor for generation of warning status files.

For those modules (currently images, fxc, rt\_images and apps) that are available for synchronization on both the operational server and the development server (sshdev.crh.noaa.gov), the script will normally synchronize with the operational server only. Synchronization to the development server can be triggered by placing a “**webdev**” file (i.e., maf\_images.webdev) in /data/ldad/web/tmp. As long as a webdev file exists for a module, it will be synchronized on the development server whenever synchronization with the operational server occurs. Creating a webdev file to trigger dual synchronization of the images module can be beneficial in expediting development of CMS web pages. Triggering dual synchronization may be beneficial as well for the rt\_images, fxc and apps modules at times to allow testing of scripts in development using live data. It makes sense to always sync the images module at both locations, but to save bandwidth, other modules should only be triggered when testing scripts in development.

Because the CMS is currently on a Central Region server and the operation web sites for other regions are on regional servers, without automatic replication to CRH, the web modules for the images and media modules must be replicated to the Central Region CMS server (taz.crh.noaa.gov) in addition to the regional server until a better solution is devised. This replication allows images and media to be found when working on the Central Region CMS. Synchronization with the CMS server “taz” occurs when a module definition is preceded by a plus “+” symbol immediately before the module definition (i.e., +images). Control and status files with “*secondary*” in the file name are automatically produced when secondary publication is enabled in this way. An example of such a file, also created in /data/ldad/web/tmp is maf\_images.secondary.update. This file is automatically created when a maf\_images.update file is produced and will result in synchronization of the images module with the Central Region server “taz” (or other server specified as secondary\_server in ldad\_web\_config.sh). Note that this dual publication is not necessary for a non-CMS web site.

Specific files and directories can be excluded from synchronization by placing an “*exclude*” file (i.e., maf\_html.exclude) in the tmp directory. This file remains until removed. This file is processed by the rsync command according to the rules that indicated for the rsync --exclude-from switch. Details about --exclude-from file patterns can be obtained by entering “man rsync” in a Linux shell. This control file should not be used except for if necessary to prevent synchronization for proper module operation. This situation is expected in few cases.

After a successful rsync occurs, a “*stat*” file (i.e., maf\_html.stat or maf\_images.webdev.stat) is left in /data/ldad/web/tmp with the file modification time set to the last modification time within the synced directory. This file is used to determine when updates have occurred in that directory tree. At LDAD sites, the corresponding directory on the secondary LDAD server is also updated using rsync after a successful module rsync. When ls3 is running the ls package, this does not occur as ls2 is likely down and failed over to ls3. The content of the “stat” file consists of the hostname of the LDAD server (or local host) where rsync was last successfully run for a package.

When LDAD failback occurs from ls3 to ls2, as determined by comparing a module’s “stat” file, the script will use rsync to pull updated content from the web rsync server. This is because this scenario will usually occur when ls2 has been down, in which case ls2 could not be updated by rsync.

When the last rsync for a module is at least 12 hours prior to a new rsync attempt, the script will pull new or updated files from the web as well. This is done to decrease the possibility of losing files that have been stored on the web server but not on the primary LDAD server (or local host). This behavior can, and should, be disabled by setting the **rsync\_only** variable in ldad\_web\_config.sh to **1** once the site is exclusively migrated to the use of RSYNC to update website contents.

**Note:** Sites attempting to continue use of FTP files from computers other than the primary LDAD parallel to use of the rsync scripts may lose content transferred to the web

farm by FTP. Serious thought should be put into the possible side-effects of this rsync procedure. Quick migration to an rsync-only configuration is strongly recommended.

## Prerequisites:

Before beginning the process of setting up the rsync scripts, sites must coordinate with the manager of any remote rsync server to ensure that the following has been done:

1. The LDAD (or local host) public IP address has been added to the list of sites allowed to rsync to the remote rsync server. For sites using a second machine (not recommended), its IP address must be provided as well.
2. The rsync modules have been set up for the site.
3. File ownership/permissions have been set appropriately for all files in the remote rsync module such that the rsync daemon can modify and delete any files in the remote module's location.

Though steps 2 and 3 must be completed by the remote rsync server's system administrator, it is necessary that the site provide the single IP address that should have rsync access to the remote module. When permission-related errors are observed, contact the remote rsync server's system administrator.

For sites using LDAD, the site must be in LDAD Mode B with both ls2 and ls3 up prior to configuration of rsync. As **root** on **ls3**, type **checkmode** to be certain that LDAD is running in mode B with no failover. The output should look similar to the following:

```
Checking ls2's heartbeat configuration... the configuration is for mode B.
Checking ls3's heartbeat configuration... the configuration is for mode B.
Checking IP addresses for mode B (this machine's current mode)...
LS1 = 192.168.1.10 - active shared cluster (actual name of computer: ls2-maf).
LS2 = 192.168.1.11 - active.
LS3 = 192.168.1.12 - active.
LS1-new = 192.168.1.13 - inactive. (This is as it should be for Mode B.)
LS1-old = 192.168.1.20 - active.
LDAD is running in Operating Mode B with ls2 as the primary Linux-cluster host.
```

Ensure that the root user can execute secure shell from ls2 to ls3, by executing the following commands on **ls2** as the **root** user:

```
ssh ls3 hostname
```

You should see **ls3-<officeid>**. If not, troubleshoot ssh before continuing.

In testing, one site experienced loss of files in /data/ldad on ls2 for reasons that have yet to be determined, so a backup of /data/ldad may be desired before continuing.

**Cygwin (Windows) sites, should use the Cygwin installation instructions near the end of this document and use Steps 1-4 as outlined in that section rather than the Steps 1-4 that follow.**

## Step 1 – Move Setup Files to LDAD or Linux Host (for Cygwin see Appendix A)

1. On **ls2** (or the Linux host) as the **root** user, copy **ldad\_web\_rsync.zip** to the **/root** directory. If FTP is used to move this file into LDAD, be sure to transfer the file in binary mode.
2. Type the following commands:
  - a. **cd /root**
  - b. **unzip ldad\_web\_rsync.zip**
  - c. **chmod -R u+rx ldad\_web\***
  - d. **chmod -R u+w ldad\_web\_config.sh**

## Step 2 – Rsync Setup Script

To prepare for web synchronization, directory structures must be created and permissions set on **ls2** and **ls3** (or the Linux host), executable files must be copied to their proper locations, and log rotation must be configured. Though this can be done manually, a script is provided to simplify the process and ensure a standard configuration. This script can be re-run without clobbering existing content. All module status files (like “force” or “stat” files) are removed from **/data/ldad/web/tmp** except for “block” files. Note that the output file may be useful in troubleshooting initial setup. The following steps should be completed as the **root** user on **ls2** (or Linux host):

1. **cd /root**
2. Copy **ldad\_web\_config.sh.pre\_cms** to **ldad\_web\_config.sh** if the site *is not* currently on the CMS. Copy **ldad\_web\_config.sh.cms** to **ldad\_web\_config.sh** if the site *is* currently on the CMS.
3. Modify **ldad\_web\_config.sh** for a site ID as well as backup site IDs (use ‘’ for the backup sites if none exist). Do not change any settings below the point indicated in the configuration file without consulting the author of the script.
4. **script -a -f ldad\_web\_setup.out**
5. **./ldad\_web\_setup.sh**
6. If the local Linux host is not LDAD, manually set user and group ownership to correspond to the user that will run the rsync scripts. The setup script sets the user and group ownership to **ldad** automatically on an LDAD system, but the script does not assume a specific user or group ID on other Linux systems. The setup script should indicate the Linux command that should be used, which should look like:

```
chown -cfR user:group /data/ldad/web
```

7. **exit**

### Step 3 – Import Current Module Contents

To ensure that the local rsync host contains all files currently stored in the remote module, it is a good idea to trigger a pull on the first synchronization. This is recommended whenever remote server content already exists. A “pull” trigger file can be created (empty) to flag that a pull from the remote server will occur the next time the `ldad_web_push.sh` script runs. Refer to the operational overview for details on trigger and stat files.

To pull current web contents on the local host do the following as root on the local rsync machine:

1. **su - ldad** (non LDAD sites can use an alternate ordinary user account)
2. **cd /data/ldad/web/tmp**
3. **touch xxx\_module.pull** (i.e., `maf_html.pull`)
4. **cd /data/ldad/web/bin**
5. **script -a -f ldad\_web\_pull.out**
6. **./ldad\_web\_push.sh xxx\_module** (i.e., `maf_html`)
7. **exit**
8. Review the script output carefully to ensure that a complete download occurred. In particular, look for **rsync error 12** and **timeouts**. If there is any question about the success of this initial pull, coordinate with the remote system administrator. The script output may be important to diagnosing issues. Error 12 normally indicates a module configuration issue or permissions issue on the remote server.
9. It does not hurt to repeat this procedure to be certain that a complete set of files was downloaded.
10. If space allows, consider creating a tar file of `/data/ldad/web` that can be backed up to optical media to prevent potential loss of critical data, or rsync this directory tree to another computer outside of the local host as a backup.

**Note:** There is a chance that timeouts will occur when pulling current module contents to LDAD (or the local host) due to the amount of data that is transferred. Should timeouts occur, just repeat this procedure until it can complete without a timeout.

## Step 4 – Set Up rsync Crons

*Only once confident that all files have been downloaded to the local host and several performance tests have been completed by manually running “ldad\_web\_push.sh”, as ldad on ls2 (or the selected rsync user on non-LDAD systems):*

**LDAD** sites modify **/etc/ha.d/cron.d/SITElscron** to include the following:

```
* * * * * ldad /data/ldad/web/bin/ldad_web_push.sh all > /dev/null 2>&1
```

For non-LDAD sites, add the following to the ordinary user account’s (used in step 3) cron:

```
* * * * * /data/ldad/web/bin/ldad_web_push.sh all > /dev/null 2>&1
```

The output file can become very big rather fast; however, an **ldad\_web\_logrotate** configuration file is copied to **/etc/logrotate.d** during the setup script so that daily cleanup occurs, as long as the cron log file path indicated in the example above is used.

At LDAD sites only, once SITElscron is edited, complete the following steps to activate the new cron entry and back it up to ls3 for failover:

1. **cp SITElscron /etc/cron.d**
2. **service crond restart**
3. **scp SITElscron ls3:/etc/ha.d/cron.d**

## A Note About Service Backup Modules:

There will likely be “stat” and “error” files created in your **/data/ldad/web/tmp** directory for the offices that you have specified as backup locations in **ldad\_web\_config.sh**. The ability to provide backup for other offices can be considered a feature to be implemented in the future. The modules required to provide backup of the operational modules do not exist at this time but will be created when needed.

To prevent attempts to sync modules that are not used, create “block” files in **/data/ldad/web/tmp** for each backup module that is resulting in the creation of “error” files in that directory. The “error” and “stat” files for blocked modules can be removed once the “block” file exists since they serve no purpose at that point. Below is an example of what you might do in the Linux or Cygwin shell to block the apps module:

```
cd /data/ldad/web/tmp
touch maf_apps.block
rm -f maf_apps.error maf_apps.stat
```

## What Next?

At the initial stage, everything is the same as before rsync on the surface in that your web site looks the same to our users as it did before. There are two things that are different “under the hood” at this point:

- 1) use of rsync rather than the FTP protocol to transfer content
- 2) narrowing the source machine for web updates from your office to one (and sometimes two) IP address.

Nothing else has happened, yet. These scripts do nothing to the organization or content of your web site.

In all likelihood, you will find a lot of files under the images directory. For most offices, not necessarily all, these files are not linked from any website. It is a good idea to make a backup of that directory before deleting anything, but once you have determined that the files in the images directory are not in use, the following procedure will empty the directory:

- 1) As the ldad user, touch /data/ldad/web/tmp/<sid>\_images.pull
- 2) Run /data/ldad/web/bin/ldad\_web\_push.sh
- 3) Make a backup of the images directory contents
- 4) Delete the unneeded files in the images directory
- 5) Touch /data/ldad/web/<sid>\_images.delete
- 6) Run /data/ldad/web/bin/ldad\_web\_push.sh

Once you are at the point that you are confident that the rsync approach works, it will be important to change all scripts that FTP data images so that this rsync method is used instead as soon as possible so that FTP access can be turned off. The big advantage of this rsync method to updating dynamic contents is that any programs that need to post data or images to the web can do so by simply copying files to the apps-data or rimages directory on LDAD. Local applications do not need to be concerned with the communications logic.

Once rsync is implemented you can also begin moving images to the "images" module and updating the referring web pages so that they point to images in their new location (/images/<sid>); however, note that this step is not required immediately and is not a prerequisite (indeed it may be an unnecessary burden) to building your new site on in the CMS development website – the images can be added to the new images directory later as the new CMS is populated; however, if images are moved to the new location prior to CMS implementation, references to the new images location do need to be relative to the web host (i.e., /images/maf/image.png rather than http://www.srh.noaa.gov/images/maf/image.png) since you don't necessarily know the name of the host once we are load balanced between the three web farms.

Eventually "live" updates to the html module go away as the CMS comes online. Once the CMS is live, some care needs to be used to update images to avoid accidental deletion of image data. One of the following approaches should be used:

- 1) update images strictly using rsync to both the operational and development web servers (see discussion of how to do this by touching the file /data/ldad/web/tmp/<sid>\_images.webdev in the operational overview).
- 2) as images and directories are created in the CMS interface, ensure that the same images and directories are created in the /data/ldad/web/images directory on LDAD
- 3) before going live, touch the file <sid>\_images.webdev.pull in the tmp directory this will ensure all image files on the development server are pulled to the local machine.

There is no way to look into the apps-data directory using a web browser. You can reference the apps-data directories in scripts (i.e., /www/apps-data/localwfo/<sid>).

The idea behind rt\_images may not be intuitively obvious. It can be thought of as similar to the images module, except it is intended for frequent automatic updates (likely done by scripts) of a relative few "operational" images. You may have hundreds or thousands of static, non-operational web page images. If the operational images were in the same place as the static images, there would be a greater burden on the network every time a new operational image is uploaded. There are also times, when working on significant changes to the website, for instance, when you may not want to rsync the whole images module when an "operational" or real time image is updated.

Before CMS implementation, HTML, PHP and CSS can be uploaded through the HTML module. After CMS implementation, that module will go away. The HTML, PHP, CSS and any other documents that belong in the HTML area will be copied to the development server to be approved. Only once they are approved will documents be put on the operational servers' HTML area.

## **Publishing Web Content:**

Content stored on the "operational" web modules (apps, rt\_images, fxc, ahps2 and archive) are published whenever a new file is copied to the LDAD directory containing these modules. These files are considered to contain real-time or near-real-time data. Once the LDAD cron is updated to include ldad\_web\_push.sh, no additional action is required to publish operational web module content.

Content stored on the static "web modules" (html, images and wwwdev) is only published when triggered by creation of an "update" or "force" file in /data/ldad/web/tmp. To publish the html module for "MAF", for instance, touch a new file /data/ldad/web/tmp/maf\_html.update. The ldad\_web\_push.sh script will then publish the maf\_html module at next run if files have been added or changed. Sites may wish to determine whether using the "force" files is a simpler option for scripts that are manually run to publish web content to ensure that file deletions are caught. Both options are provided for flexibility.

Shell scripts can be written for AWIPS to allow publishing from an AWIPS workstation and included in the Apps Menu by adding the script to /awips/fga/data/appsLauncher/local/conf on AWIPS workstations.

Such a script might have the following:

```
ssh ldad@ls1 touch /data/ldad/web/tmp/maf_html.force
```

If ls1 is made available to the office LAN using Samba, a windows command file can be run from /data/ldad/web to do the same thing. The following is an example publish.cmd script:

```
@echo off
cls
echo.
echo.
echo "Updated!" > \\ls1-maf\web\tmp\maf_html.force
echo "Updated!" > \\ls1-maf\web\tmp\maf_images.force
echo Rsync will be used to publish html and images directories.
echo Check web content in a few minutes.
echo.
echo.
pause
```

More complex scripts, possibly including a GUI, can be developed to simplify the publishing process for less experienced users, possibly by allowing selection of modules to publish. Eric Holweg has produced an excellent, straight-forward Java program that you may want to use. Eric's WebSync application and brief installation and usage notes are available on the Trac CMS Wiki.

Remember that a "**force**" file (maf\_html.force) can be created to force a sync even if the script does not detect a change since the last sync. A "**block**" file (sjt\_apps.block) can be created to prevent a sync, even if changes are detected.

A "**webdev**" file (maf\_images.webdev) can be created to publish to the development server in addition to the operational server. The images, apps and rt\_images modules are configured on the development server.

## Deleting Files:

It may not be apparent how to delete files from the web server. Though it would seem intuitive that deleting files from the local LDAD followed by touching the corresponding "update" file would accomplish this, it will not because file dates are used to determine when updates have occurred.

Another complication is that the ldad\_web\_push.sh script pulls from the web server before pushing when no updates have occurred in the previous 12 hours. This behavior is intended to

increase the chance that files moved to the web by FTP are updated on LDAD rather than deleted, as could occur when both FTP and rsync are in use. This behavior ceases when `rsync_only` is set to "1" in the `ldad_web_config.sh` file.

Finally, some modules may be specified as "no\_delete\_modules" or "web\_modules" in `ldad_web_config.sh`. Files are never deleted during synchronization for those modules.

The easiest way to ensure that deletion occurs on the next synchronization is to touch a "delete" file in `/data/ldad/web/tmp` corresponding to the module that you intend to delete files from. This will trigger a synchronization with deletion for both operational and web modules, whether or not those modules are in the "no\_delete\_modules" list.

### **CMS Website Activation:**

Once the new website has been built on the developmental CMS server, it is a good idea to pull images that exist on the developmental server to LDAD, particularly if multiple image upload methods have been used during site development. This will ensure that the local host has all the files in the images module in the same structure as on the developmental CMS server – recall that when an rsync occurs on the operational web images module, all images not on the local host will be removed from the operational web images module if they do not exist locally.

You can trigger a pull of images from the development module by dropping the file `<sid>_images.webdev.pull` in the tmp directory of the rsync installation.

Also refer to the example CMS site settings in Appendix C as there are likely changes required for the modules. Removing the html module, which does not exist once a site is on the CMS, is one example of a change required after moving to the CMS.

### **Important Note About SFTP to Development Server:**

It is not a good idea to use SFTP to copy files to any of the directories that are also updated by using rsync modules. Permission conflicts could result that prevent rsync from updating files on the development server. If SFTP must be used, it is advisable to ensure that all files created have permissions set to 777 (rwxrwxrw) to avoid permission problems using rsync.

## Appendix A – Cygwin Install

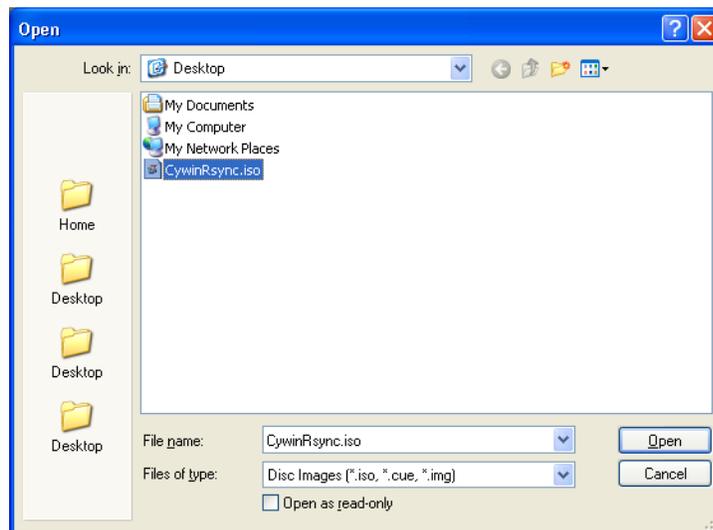
Offices that do not have LDAD and do not have a Linux computer running Redhat Linux can still utilize the rsync scripts for web page management through the use of the Linux emulator Cygwin. The Cygwin package should only be used by locations that have no alternative.

To ensure proper operation, use only the Cygwin package created for this purpose.

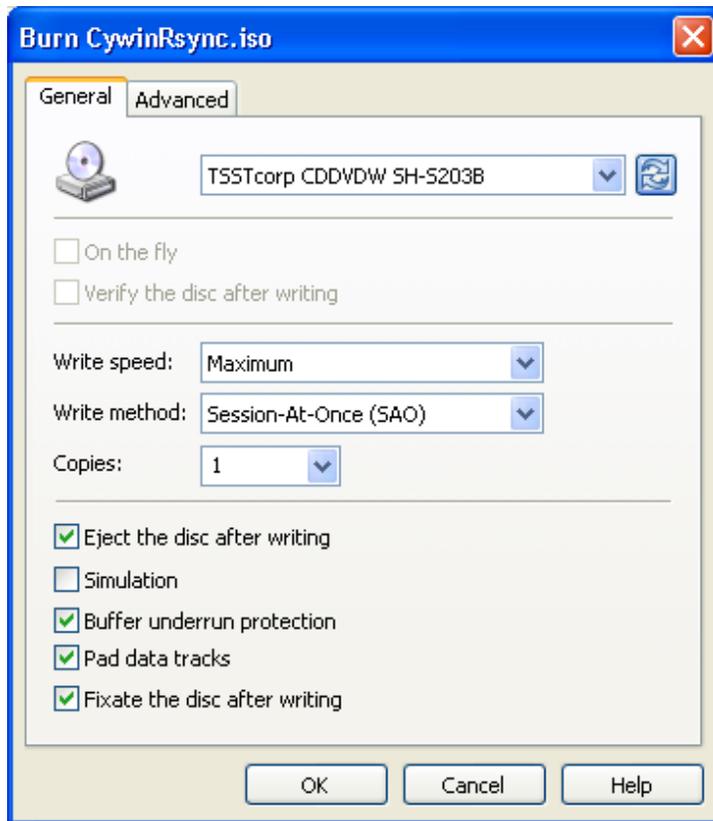
1. Download the rsync Cygwin package (as ISO CD image) at:  
<http://lucretia.srh.noaa.gov/maf/it/files/CygwinRsync.iso>  
Save CygwinRsync.iso to a chosen temporary location.
2. Burn the downloaded ISO CD image file to a CD with any CD burning package that supports burning ISO images. If a commercial product is not available, InfraRecorder can be downloaded from <http://infrarecorder.org/> (InfraRecorder V0.46.1 was used in testing):
  - a. Insert a blank CD in the CD drive.
  - b. Run InfraRecorder and click on the “Write Image”:



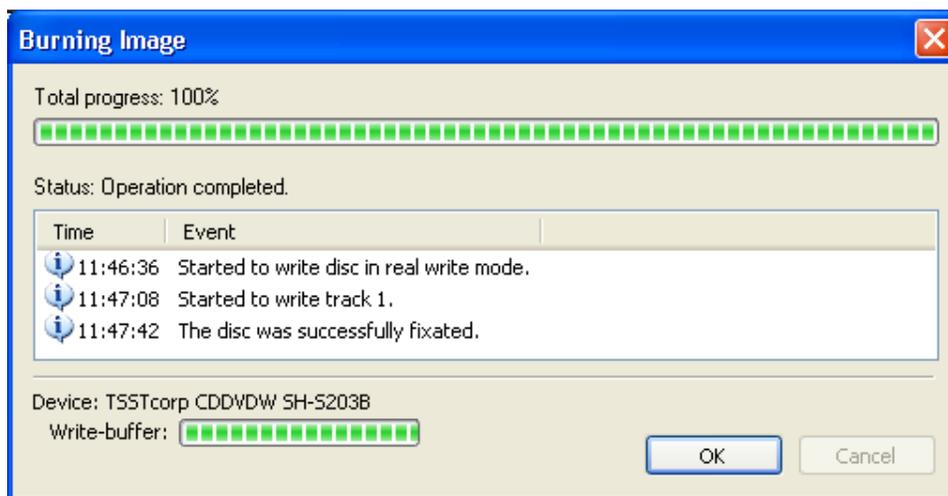
- c. Navigate to and select the ISO image downloaded in Step 1:



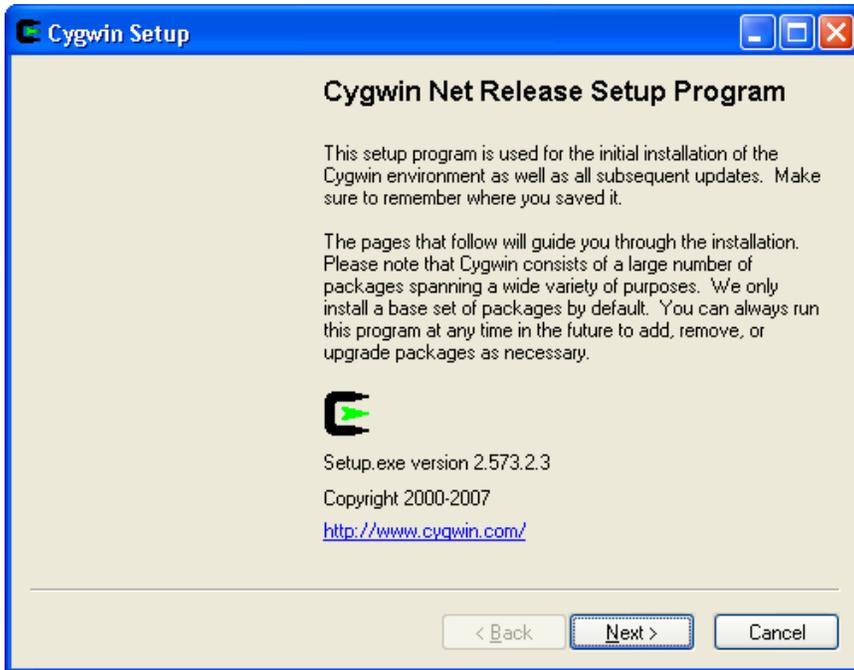
- d. Accept the defaults. Note that if the “OK” button is disabled, it is because a writable CD has not been detected. If a writable CD is in the selected drive, click the refresh button to the right of the drive selection drop-down menu.



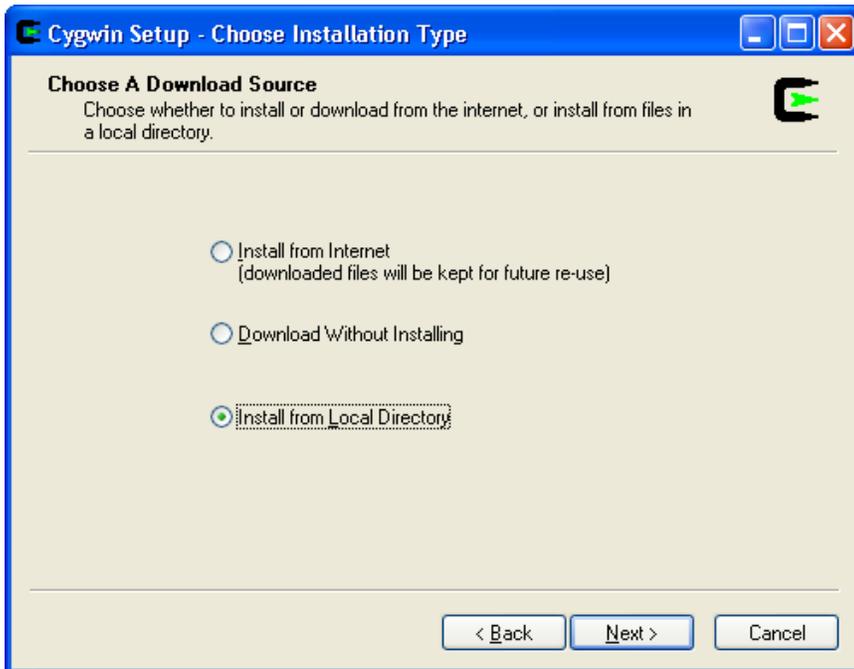
- e. Once the disk is done writing, you should get confirmation of a successful write, and the CDROM will eject:



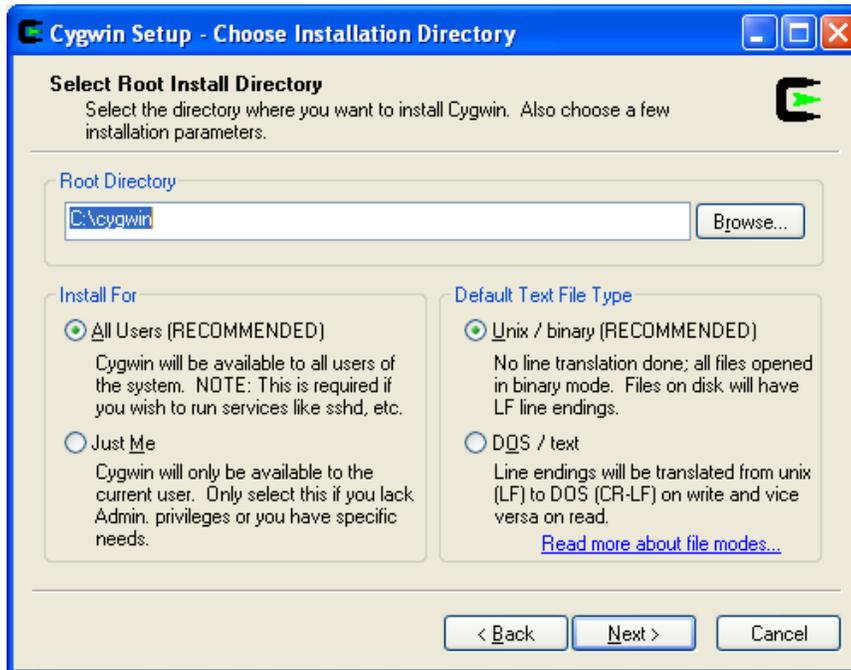
3. Log into the Rsync host machine as the administrator.
4. Insert the Cygwin installation CD in Step 2 and run Setup.exe. On the first screen, click the “Next” button.



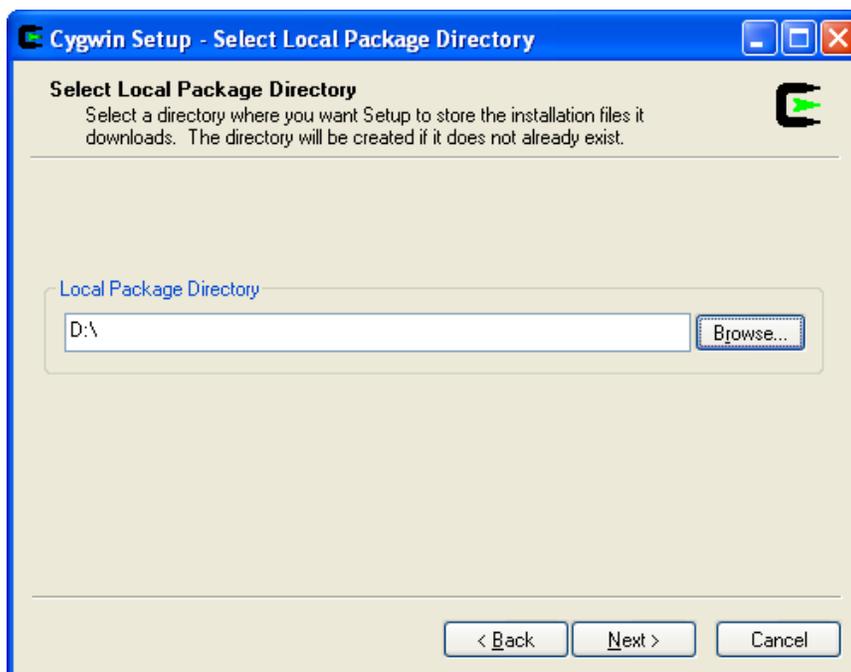
5. Select “Install from Local Directory” as the installation type and click “Next”.



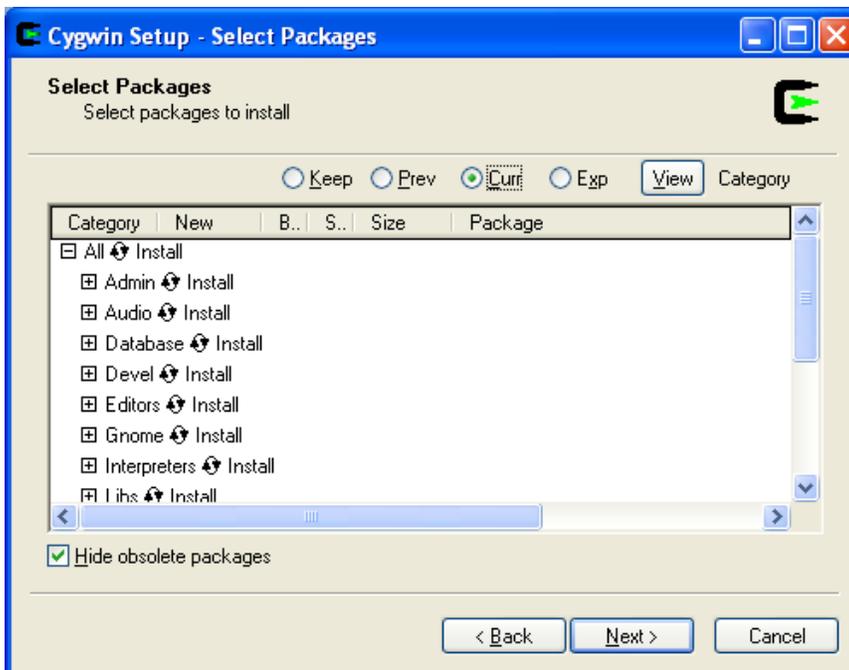
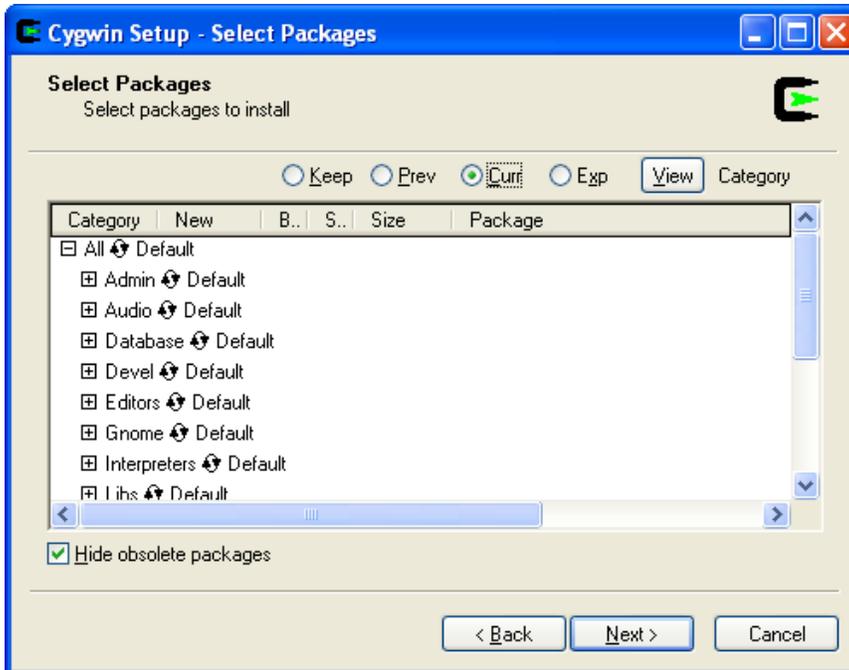
- Accept the default installation directory, make sure that the installation is available to all users and leave the default text file type “Unix”. Click “Next”:



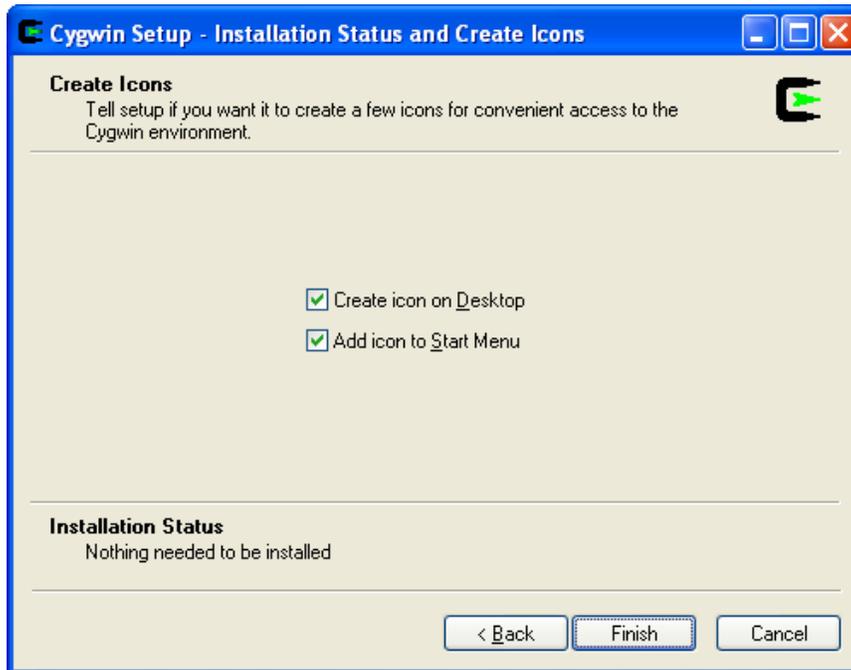
- Ensure that the root directory of the installation CD is selected as the “Local Package Directory” and click “Next” (setup may not respond for several minutes):



8. When prompted to select packages, click on “Default” next to “All”, at the top of the package list, until the “Install” option is shown. The packages on the CD were selected to support rsync, the bash shell, and a cron daemon. Omissions could cause an incomplete installation. Click “Next” when all packages show “Install”:



9. It may be convenient to create startup icons on the Desktop and in the Start Menu. Click “Finish” to begin installation:



10. Once the Cygwin install is complete, start Cywin by clicking on the shortcut on the Desktop or Start Menu, or click on Cygwin.bat in C:\cygwin.
11. To install the Cygwin cron daemon as a Windows service, enter the following (all on one line):

```
cygrunsrv -I cron -p /usr/sbin/cron -a -D -d "Cygwin cron"  
-e "CYGWIN=tty ntsec"
```

then,

```
net start cron
```

The “Cygwin cron” service can be managed in the Windows “Services” Administrative Tool.

At this point, the Cygwin Linux emulator is installed on Windows. All rsync script installation procedures should be completed within the Cygwin shell with the local Windows administrator logged on. These procedures are otherwise the same as the LDAD Linux installation instructions but are repeated in the following pages with Cygwin modifications for clarity and minor modifications.

Some interesting items to note about Cygwin:

1. The Cygwin root drive “/” is equivalent to C:\cygwin in Windows and is also /cygdrive/c/cygwin within the Cygwin Linux emulation. Any drive in Windows can be accessed within Cygwin using this convention as well!
2. Some Linux utilities in Cygwin are named the same as Windows counterparts (like “find” or “more”). When the Cygwin command is preferred, the full path may be required, or the PATH variable should be redefined to include Windows paths last.
3. You can use Windows-based editors to modify files in Cygwin file system (c:\cygwin and below) provided that they are “Linux-safe”. Notepad adds carriage returns to text files. Linux does not expect carriage returns in text files, including scripts. One safe editor is Notepad++, available at:

**<http://notepad-plus.sourceforge.net/uk/download.php>**

### **Step 1 – Move Setup Files to Cygwin**

1. As the local Windows administrator, copy **ldad\_web\_rsync.zip** to the C:\cygwin\tmp directory (which corresponds to /tmp in Cygwin).
2. In the Cygwin shell, type the following commands:
  - a. **cd /tmp**
  - b. **unzip ldad\_web\_rsync.zip**
  - c. **chmod u+rx ldad\_web\***
  - d. **chmod u+w ldad\_web\_config.sh**

### **Step 2 – LDAD Setup Script**

The first step to preparing the rsync client scripts for web synchronization is to set up the directory structure with the appropriate permissions and copy executable files to their proper location. Though this can be done manually, a script is provided to simplify the process and ensure a standard configuration. This script can be re-run without clobbering existing content. All module status files (like “force” or “stat” files) are removed from /data/ldad/web/tmp except for “block” files. From the Cygwin shell, do the following:

1. **cd /tmp**
2. Copy **ldad\_web\_config.sh.pre\_cms** to **ldad\_web\_config.sh** if the site *is not* currently on the CMS. Copy **ldad\_web\_config.sh.cms** to **ldad\_web\_config.sh** if the site *is* currently on the CMS.
3. Modify **ldad\_web\_config.sh** for a site ID as well as backup site IDs (if any). If there are no backup sites, set “backup\_sites” to an empty string (i.e., backup\_sites=’). If not familiar with Linux editing commands edit C:\cygwin\tmp\ldad\_web\_config.sh using a Linux file friendly Windows editor like Notepad++.
4. Set primary\_ldad to the name of the “short” Windows computer name (not fully-qualified, like MAF-W-WEB) and set secondary\_ldad to an empty string.

5. Do not change any settings below the point indicated in the configuration file without consulting the author of the script.
6. From the Cygwin shell enter:  
**./ldad\_web\_setup.sh**

### Step 3 – Import Current HTML Contents

To ensure that the Windows rsync client contains all files currently stored in the remote module, it is a good idea to trigger a pull on the first synchronization. This is recommended whenever remote server content already exists. A “pull” trigger file can be created (empty) to flag that a pull from the remote server with occur the next time the `ldad_web_push.sh` script runs. Refer to the operational overview for details on trigger and stat files.

To pull current web contents onto the local Windows machine run the Cygwin shell as the Windows local administrator:

1. **su - ldad** (or other user intended to be used for rsync in Cygwin)
2. **cd /data/ldad/web/tmp**
3. **touch xxx\_module.pull** (i.e., maf\_html.pull)
4. **cd /data/ldad/web/bin**
5. **script -a -f ldad\_web\_pull.out**
6. **./ldad\_web\_push.sh xxx\_module** (i.e, maf\_html)
7. **exit**
8. Review the script output carefully to ensure that a complete download occurred. In particular, look for **rsync error 12** and **timeouts**. If there is any question about the success of this initial pull, coordinate with the remote system administrator. The script output may be important to diagnosing issues. Error 12 normally indicates a module configuration issue or permissions issue on the remote server.
9. It does not hurt to repeat this procedure to be certain that a complete set of files was downloaded.
10. If space allows, consider creating a tar file of `/data/ldad/web` that can be backed up to optical media to prevent potential loss of critical data, or rsync this directory tree to another computer outside of the local host as a backup.

**Note:** There is a chance that timeouts will occur when pulling current module contents to the local Windows Cygwin host due to the amount of data that is transferred. Should timeouts occur, just repeat this procedure until it can complete without a timeout.

The `/data/ldad/web` directory where web contents, rsync scripts, and temporary work files reside correspond to the local directory `C:\cygwin\data\ldad\web`. It is recommended that those directories that others are allowed to update have proper permissions set. This can be done within Windows.

## Step 4 – Set Up LDAD Crons

*Only once confident that all files have been downloaded to the local machine and several performance tests have been completed by manually running “ldad\_web\_push.sh”, as the local Windows administrator, modify the administrator cron to include the following (all on one line):*

```
* * * * * /data/ldad/web/bin/ldad_web_push.sh all > /dev/null 2>&1
```

To modify the cron, enter “crontab -e” in the Cygwin shell. The editor is vim, so some “vi” knowledge is necessary. Use of vi is beyond the scope of these instructions, though there are various guides on the internet. One such guide is located at:

[http://www-ac.s.ucsd.edu/info/vi\\_tutorial.shtml](http://www-ac.s.ucsd.edu/info/vi_tutorial.shtml)

The output file can become very big rather fast; however, the script automatically rotates and deletes log files.

Note that other instructions in this document do generally apply to Cygwin implementations. Please continue with the section “A Note About Backup Modules” that follows the LDAD installation steps.

## Appendix B – Upgrade Instructions

Upgrading the rsync scripts is not laborious. In fact, it is relatively easy! In essence, update the rsync scripts by reinstalling the package. For Cygwin-based installations, Cygwin *does not* need to be reinstalled.

Before upgrading, be sure to back up the files in /data/ldad/web/bin (or C:\Cygwin\data\ldad\web\bin for Cywin installs) in case it becomes necessary to revert to the previous versions of the scripts.

To upgrade the rsync scripts, complete “Step 1” and “Step 2” of the original installation procedures. For Cygwin installations, see Appendix A, bypassing the Cygwin installation portion. The reason that it is a good idea to use this approach is that the ldad\_web\_setup.sh script updates both LDADs when it runs and creates any necessary directories.

It is *strongly advised* that the provided ldad\_web\_config.sh.pre\_cms or ldad\_web\_config.sh.cms script be used as a starting point, modifying as necessary for local use, before running ldad\_web\_setup.sh rather than using your existing configuration. There may be critical changes or additions that could be lost otherwise.

Whenever updates are installed, also refer to the Release Notes at the end of this document for any additional information that may be relevant to your site or configuration.

## Appendix C – Directory Structure

### Directory and Module Structure Prior to CMS Implementation

| Web Server Path and Linux File Path                    | rsync Module      | LDAD (Local) Directory         | Description  |
|--|-------------------|--------------------------------|--|
| /<sid><br>/www/html/<sid>                              | <sid>_html        | /data/ldad/web/html            | HTML, CSS & PHP documents (goes away after CMS implementation).  |
| /images/<sid><br>/www/images/<sid>                     | <sid>_images ‡    | /data/ldad/web/images          | Static website images  |
| (none)<br>/www/apps-data/localwfo/<sid>                | <sid>_apps ‡      | /data/ldad/web/apps-data/<sid> | Data and dynamic content used by local apps and web pages  |
| /images/rtimages/<sid><br>/www/images/rtimages/<sid> † | <sid>_rt_images ‡ | /data/ldad/web/rtimages/<sid>  | Real time (operational) images   |
| /images/fxc/<sid><br>/www/images/fxc/<sid>             | <sid>_fxc         | /data/ldad/web/fxc/<sid>       | Images for GraphiCast and other Regional and National programs.  |
| /images/fxc/<sid><br>/www/images/fxc/<sid>             | <sid>_fxc         | /data/ldad/web/fxcdev/<sid>    | Images for GraphiCast and other Regional and National programs on the <b>development server</b> (for training and practice). |
| /media/<sid><br>/www/apps-data/media /<sid>            | <sid>_media ‡     | /data/ldad/web/media/<sid>     | Articulate presentation files.   |
| /archive/<sid><br>/www/archive/<sid>                   | <sid>_archive     | /data/ldad/web/archive/<sid>   | Long-term archive (primarily RFC use)  |
| /images/ahps2/<sid><br>/www/images/ahps2/<sid>         | <sid>_ahps2       | /data/ldad/web/ahps2/<sid>     | AHPS graphics generated by RFCs  |
|  |                   | /data/ldad/web/bin             | Rsync support scripts  |
|  |                   | /data/ldad/web/logs            | Rsync script log file output   |
|  |                   | /data/ldad/web/tmp             | Rsync support script work area   |

‡ A few modules can be copied to both the active website and the development server (details in Operational Overview).

In `ldad_web_config.sh`:

```
dev_server='sshdev.crh.noaa.gov'
web_modules='html images'
operational_modules = 'apps|apps-data rt_images|rtimages fxc sshdev.crh.noaa.gov:fxc|fxcdev media archive ahps2'
no_delete_modules='archive'
```

## Directory and Module Structure After CMS Implementation

| Web Server Path and Linux File Path                  | rsync Module      | LDAP (Local) Directory         | Description  |
|--|-------------------|--------------------------------|--|
| /images/<sid><br>/www/images/<sid>                   | <sid>_images ‡ †  | /data/ldad/web/images          | Static website images.   |
| (none)<br>/www/apps-data/localwfo/<sid>              | <sid>_apps ‡      | /data/ldad/web/apps-data/<sid> | Data and dynamic content used by local apps and web pages  |
| /images/rtimages/<sid><br>/www/images/rtimages/<sid> | <sid>_rt_images ‡ | /data/ldad/web/rtimages/<sid>  | Real time (operational) images   |
| /images/fxc/<sid><br>/www/images/fxc/<sid>           | <sid>_fxc         | /data/ldad/web/fxc/<sid>       | Images for GraphiCast and other Regional and National programs.  |
| /images/fxc/<sid><br>/www/images/fxc/<sid>           | <sid>_fxc         | /data/ldad/web/fxcdev/<sid>    | Images for GraphiCast and other Regional and National programs on the <b>development server</b> (for training and practice). |
| /media/<sid><br>/www/apps-data/media /<sid>          | <sid>_media ‡ †   | /data/ldad/web/media/<sid>     | Articulate presentation files.   |
| /archive/<sid><br>/www/archive/<sid>                 | <sid>_archive     | /data/ldad/web/archive/<sid>   | Long-term archive (primarily RFC use)  |
| /images/ahps2/<sid><br>/www/images/ahps2/<sid>       | <sid>_ahps2       | /data/ldad/web/ahps2/<sid>     | AHPS graphics generated by RFCs  |
|  |                   | /data/ldad/web/bin             | Rsync support scripts  |
|  |                   | /data/ldad/web/logs            | Rsync script log file output   |
|  |                   | /data/ldad/web/tmp             | Rsync support script work area   |

‡ A few modules can be copied to both the active website and the development server (details in Operational Overview).

† Note that synchronization with taz.crh.noaa.gov may be necessary initially for image availability on the CMS (reason for + before images module).

### In `ldad_web_config.sh`:

```
dev_server='sshdev.crh.noaa.gov'
secondary_server='taz.crh.noaa.gov'
web_modules='+images'
operational_modules = 'apps|apps-data rt_images|rtimages fxc sshdev.crh.noaa.gov:fxc|fxcdev +media archive ahps2'
no_delete_modules='archive'
```

## Appendix D – Excluded Types by Module

Each module has an intended purpose. Though adherence to the intended purpose for each module is somewhat dependent on every site's cooperation, there are some file types that will not be allowed to rsync, in part to enforce the intended use of each module and the integrity of the web farm.

The exclusions are currently the same for each rsync module, though some modification may occur in the future. Also these exclusions apply to the CMS and development CMS servers. Prior to CMS implementation, no exclusions apply but sites should ensure that excluded file types are not being synchronized to avoid problems after CMS implementation.

As a general rule of thumb, currently excluded files are compressed files, scripts, and HTML pages. When HTML data is uploaded for script use in the apps module, this data should be uploaded as a "txt" file or other allowed flat-file type.

Below are the exclusions are currently configured:

### **html** (available pre-CMS only)

No exclusions

### **images**

tar, gz, php, html, shtml, cgi, pl, htm, sh, bsh, txt

### **apps**

tar, gz, php, cgi, pl, sh, bsh

### **rt\_images**

tar, gz, php, html, shtml, cgi, pl, htm, sh, bsh, txt

### **fxc**

tar, gz, php, html, shtml, cgi, pl, htm, sh, bsh

### **media**

tar, gz, php, cgi, pl, sh, bsh

### **ahps2**

tar, gz, php, html, shtml, cgi, pl, htm, sh, bsh

### **archive**

tar, gz, php, html, shtml, cgi, pl, htm, sh, bsh

## Appendix E – `ldad_web_config.sh` Options

There are several options in the `ldad_web_config.sh` script that should not be modified without coordination with the rsync scripts' author due to their ability to create unexpected behavior in the scripts. The other rsync scripts should not be modified at all.

There are several settings that must be changed for each office:

***site***                      The *site* variable defines the office based on the ID used on the web. For, instance, for the Midland WFO:

```
site='maf'
```

***backup\_sites***            The *backup\_sites* variable is a space-delimited list of backup sites for which rsync web backup services will be provided. These sites are also listed using the site IDs used on the web. Some sites, like CWSUs, may not currently offer backup services for another site. In that case, *backup\_sites* should be set to a null string (i.e., ""). The Midland WFO backs up San Angelo and El Paso:

```
backup_sites='sjt epz'
```

***primary\_ldad***            The *primary\_ldad* variable specifies the host name of the machine that normally runs the rsync scripts. For WFOs and RFCs, this would typically be 'ls2'. For sites like CWSUs, that either use a non-LDAD Linux or Windows machine, the *primary\_ldad* variable would be set to the host name of the actual machine (i.e., 'MAF-W-RSYNC' or 'MAF-LW-RSYNC'). Typical setting for a WFO or RFC:

```
primary_ldad='ls2'
```

***secondary\_ldad***        The *secondary\_ldad* variable specifies the host name of the machine that runs the rsync scripts when the machine listed as the *primary\_ldad* is not running as the primary LDAD. This setting is only applicable to a WFO or RFC environment using LDAD since the backup scheme is dependent on the AWIPS LDAD failover architecture for proper operation. Sites running the rsync scripts on a single non-LDAD machine need to set this variable to a null string (i.e., ""). For WFOs and RFCs:

```
secondary_ldad='ls3'
```

***block\_alive\_stat*** By default, “alive” status files are dropped in the tmp directory during each `ldad_web_push.sh` run. The *block\_alive\_stat* variable can be set to 1 to disable creation of these files. If creation of these files is desired, set *block\_alive\_stat* to 0 (typical):

```
block_alive_stat=0
```

***require\_ping*** The *require\_ping* variable can help script efficiency by allowing the script to test the availability of each rsync server before attempting to open an rsync session. To enable this feature set *require\_ping* to 1. Before enabling the feature, test pings should be made from the rsync script machine(s) to every remote rsync host. If test pings fail, a ticket can be opened with NOAAnet to request firewall rules allowing the necessary ping access. If ping access is not allowed or this feature is not desired, set *require\_ping* to 0 (typical):

```
require_ping=0
```

***rsync\_only*** While a site concurrently uses FTP and rsync to put files on the web, it is possible for files to be on the web that are not in the rsync script’s directory structure. This can be alleviated to some degree by setting *rsync\_only* to 0. When *rsync\_only* is set to 0, the rsync script will pull updated content from the web before attempt to push new content when more than 12 hours has elapsed since the last successful rsync. Once FTP is disabled, *rsync\_only* should be set to 1:

```
rsync_only=1
```

***default\_timeout*** The rsync command will timeout during very long transfers. If this is a routine problem due to the size and/or number of files typically transferred, the *default\_timeout* period can be increased. A setting of 60 seconds will be appropriate for most sites. If timeouts occur frequently, information from the `ldad_web_push` log file may be helpful in determining a more appropriate setting. For most locations:

```
default_timeout=60
```

***huge\_module\_delay*** When the script detects a timeout on a module, the module is considered a “huge” module. Attempts to rsync a huge module are scaled back to the period specified by *huge\_module\_delay*. Most sites will not experience “huge” modules, so a typical setting of 900 seconds between huge module rsync attempts will generally be good. Sites that do see modules being designated as huge (see the *ldad\_web\_push*) log file, should consider how often attempts to synchronize the offending module(s) should be made; otherwise, a typical setting:

```
huge_module_delay=900
```

***dev\_server*** The *dev\_server* variable specifies the name of the development server. When a “webdev” control file is found in the tmp directory for a module, that module will be synchronized with both the operational web filer and the development server. Presently, all sites would use the same setting:

```
dev_server='sshdev.crh.noaa.gov'
```

***secondary\_server*** The *secondary\_server* variable specifies the name of a secondary server to rsync modules to. Module definitions immediately preceded with a plus ‘+’ symbol (like ‘+images’) will synchronize to both the operational and a secondary server. This configuration will be needed for the images and media modules initially to make images and media available in the CMS at Central Region. Presently, all sites would use the same setting:

```
dev_server='taz.crh.noaa.gov'
```

***prepend\_server*** Setting *prepend\_server* to 1 results in the short hostname of servers explicitly included by name in either *web\_modules* or *operational\_modules* to be prepended to corresponding control files. This feature is designed to make it easier to identify control files when modules with the same name exists on more than one server.

```
prepend_server=1
```

***web\_modules*** The *web\_modules* variable is a space-delimited list of modules that support static web pages. Since these modules do not have dynamic content, synchronizations will not occur for modules listed as *web\_modules* unless a corresponding “update”, “force”, “delete” or “pull” control file is dropped in the rsync tmp directory, triggering a one-time rsync. A typical list of web modules (prior to CMS activation – html goes away afterward):

```
web_modules='html images'
```

File paths for web\_modules are normally built by adding the module name to local\_web\_root (described later). Since backup support is not implemented for these modules, the path to files in web\_modules does not include an office ID. For example, the file path for the images module would typically be /data/ldad/web/images.

The remote module actually synchronized with is assumed to be <sid>\_<module>. For example, files in the images module for the Midland WFO are synchronized from /data/ldad/web/images to the maf\_images module on the remote server.

If an alternate file path is required for a web\_module, a module can be specified by module name + “pipe” symbol + path. For example, if we added a module named “test” whose files are in the “localtest” directory relative to the local\_web\_root:

```
web_modules='html images test|localtest'
```

The rsync scripts can rsync with servers other than the operational web filer or development server by preceding the module name with the full remote server name followed by a colon as show in the example below:

```
web_modules='html images testserv.srh.noaa.gov:test|localtest'
```

Normally, the rsync script prepends the site ID to the defined module name. This behavior may not be appropriate for a shared module that is not owned by a specific office. To prevent automatic module name expansion, place an “at” (or “@”) character immediately before the module definition in *web\_modules*:

```
web_modules='html images @shared|common'
```

In this example, for “html” and “images” the remote module names would be expanded to “maf\_html” and “maf\_images”, while “shared” would remain named “shared”.

Note that to prevent unintentional deletion of files in shared modules, those modules that use the “@” directive are treated the same as a module listed in the *no\_delete\_modules* and *no\_pull\_modules* variables.

Publishing of modules to a secondary server, defined by *secondary\_server* can be accomplished by place a “+” directive immediately before the module definition. This approach is temporarily used for the images and media modules to publish to the CRH CMS server (taz.crh.noaa.gov) in addition to the operational server for those sites on the CMS.

The example below applies to CMS sites:

```
web_modules='+images'
```

In very rare instances when both a '+' directive and a '@' directive are required, the '@' directive appears first:

```
web_modules='+images @+ztl_images'
```

***operational\_modules*** The *operational\_modules* variable is a space-delimited list of modules that support dynamic web content (images and application data). Operational modules are synchronized whenever a file change is detected in the *operational\_modules* local files, so control files are not necessary to trigger publishing of data to *operational\_modules*; however, a “block” control file can be dropped in the tmp directory to prevent synchronization. This may be desired for backup sites until the appropriate configuration exists on the web filer for backup support.

It is assumed that a site will provide backup services for each office listed as *backup\_sites*. For this reason, the site ID is appended to each of the *operational\_modules* to determine local paths. For instance, WFO Midland’s files that are synchronized with the “fxc” module would be located in the directory fxc/maf, relative to *local\_web\_root*.

When the “@” directive, described previously in the *web\_modules* section, is used (for a shared module), the site ID is not appended to the module’s local file path and is not prepended to the remote module name.

A typical set of *operational\_modules* for WFOs:

```
operational_modules='apps|apps-data rt_images|rtimages fxc'
```

A typical set of *operational\_modules* for RFCs:

```
operational_modules='apps|apps-data rt_images|rtimages fxc archive ahps2'
```

As with *web\_modules*, alternate local paths and servers can be specified:

```
operational_modules='apps|apps-data rt_images|rtimages fxc sshdev.crh.noaa.gov:fxc|fxcdev'
```

Likewise, the '+' directive may be used:

```
operational_modules='apps|apps-data rt_images|rtimages +media'
```

***no\_delete\_modules*** By default, rsync is run with a --delete option on all modules, except operational modules for backup sites. There reasons why this option may not be desirable as a default. For instance, for site html and images

modules, automatic use of this option could wipe out an entire website if the local files were mistakenly removed prior to rsync. Other modules might grow to an excessive size if all files were retained on the local machine. The *no\_delete\_modules* is a space-delimited list of modules that should not implement the --delete option during rsync. The following settings are recommended:

```
no_delete_modules='archive images html'
```

The --delete option can still be used on a *no\_delete\_module* by dropping a “delete” control file corresponding to the module in the tmp directory to trigger a one-time rsync with the delete option.

***no\_pull\_modules***

The *no\_pull\_modules* variable is a space-delimited list of modules that should never be pulled from the web prior to synchronization, regardless of the *rsync\_only* setting. Modules that could exhaust local storage capacity should be included in this list:

```
no_pull_modules='archive'
```

***local\_web\_root***

The *local\_web\_root* specifies where the rsync script files reside. For most locations this variable should be set as follows:

```
local_web_root='/data/ldad/web'
```

***ldad\_module***

The *ldad\_module* variable specifies the name of the module that corresponds to /data/ldad on the secondary LDAD (ls3). For non-LDAD sites, this variable has no function. At WFOs and RFCs, the correct setting would be:

```
ldad_module='data_ldad'
```

***ldad\_web\_rel***

The *ldad\_web\_rel* variable specifies where the *local\_web\_root* is relative to the secondary LDAD’s *ldad\_module* location. At non-LDAD sites, this setting has no effect. For WFOs and RFCs:

```
ldad_web_rel='web'
```

***rsync\_server***

The *rsync\_server* variable specifies the host name for the remote rsync server that all modules are synchronized with when an alternate server is not specified in *operational\_modules* or *web\_modules*. The setting that is currently appropriate:

```
rsync_server='martha2.srh.noaa.gov'
```

***rsync\_pull\_opts***

The *rsync\_pull\_opts* are options used by rsync when pulling module data from a remote host. Since --delete option use is determined by the rsync script, these options should never include --delete. Since correct options are necessary for proper script operation, the defaults should not be changed:

```
export rsync_pull_opts='-t -lruv --progress --stats --bwlimit=256'
```

***rsync\_web\_opts***

The *rsync\_web\_opts* are options used by rsync when pushing module data to a remote host. Since --delete option use is determined by the rsync script, these options should never include --delete. Since correct options are necessary for proper script operation, the defaults should not be changed:

```
export rsync_web_opts='-lrtuv --progress --stats --bwlimit=256'
```

***rsync\_ldad\_opts***

The *rsync\_ldad\_opts* are options used by rsync when pushing module data to the secondary LDAD after successful rsync with a remote host. Since the --delete option use is determined by the rsync script, these options should never include --delete. Since correct options are necessary for proper script operation, the defaults should not be changed:

```
export rsync_ldad_opts='-lrtuv --progress --stats'
```

***max\_lock\_age***

The *max\_lock\_age* is the number of seconds that a the rsync task is allowed to lock a module using a “lock” control file for exclusive synchronization before a new rsync task kills the previous rsync task on a module to attempt rsync. Until this amount of time has elapsed, rsync tasks will not attempt to rsync a locked module. The *max\_lock\_age* variable should logically be at least as large as the *default\_timeout*. When an error occurs during rsync, an “error” file is created for a module. As long as that file exists, attempts to rsync that module will also occur only as often as indicated by the *max\_lock\_age* until the error condition is resolved to avoid excessive bandwidth and CPU utilization. The following setting is appropriate for most sites:

```
max_lock_age=900
```

## Appendix F – Control File Naming

The names of control and status files for existing modules can be easily identified by inspecting the script tmp directory (usually /data/ldad/web/tmp) for the “alive” control files. A status file ending with the “alive” suffix is created for each currently defined module as long as the *block\_alive\_stat* variable in *ldad\_web\_config.sh* is set to 0 (or is not defined). The contents of the “alive” status files give useful information about modules, so *block\_alive\_stat* should only be set to 1 once the local configuration is debugged and operating smoothly.

Here is an example from a status file called *sshdev\_maf\_fxc\_fxcdev.alive*:

|   |   |
|---|---|
| <code>operational_modules</code>            | Module defined in <i>operational_modules</i>    |
| <code>sshdev.crh.noaa.gov:fxc fxcdev</code> | Module definition in <i>operational_modules</i> |
| <code>sshdev.crh.noaa.gov::maf_fxc</code>   | Module server and remote module name            |
| <code>/data/ldad/web/fxcdev/maf</code>      | Local path synchronized with module             |
| <code>sshdev_maf_fxc_fxcdev.stat</code>     | Name of module’s status file                    |

There is a complex set of status and control file base filenames that can be difficult to discern, so it is easiest to use the “alive” files:

1. `<sid>_<module>` for all modules on the production server whose local file directory and remote module name are the same.
2. `<sid>_<module>_<dir>` for all modules on the production server whose local file directory and remote module names differ and for all modules having a remote server name explicitly defined in the module definition.
3. A “.webdev” extension is added to modules also synchronized with the development server.
4. `<module>_<dir>` for all modules whose definition is preceded by ‘@’ (meaning that the site ID should not be automatically prepended to the remote module name – use the module name as explicitly defined).
5. `<hostname>_<sid>_<module>_<dir>` for modules on a remote server explicitly defined in the module definition.
6. `<hostname>_<module>_<dir>` for modules on a remote server explicitly defined in the module definition preceded by ‘@’.

Again this naming convention is complex due to support of a legacy naming convention. Over the evolution of the *ldad\_web\_push.sh* script, new demands have required extensions to the initial naming convention. It is easiest to use the “alive” files as a reference.

## Appendix G – Note on Publishing to “taz”

For those sites on the CMS, there is initially no automatic replication from the operational web filers outside CRH to the CMS server at CRH. For this reason, images and media published to other regional operational web farms are not available for web design, including headlines, on the Central Region CMS server, currently taz.crh.noaa.gov.

Though automatic replication from regional web servers to CRH is planned, the details have not yet been worked out, so a workaround has been devised. The workaround is to publish some modules to both the regional servers and the CRH CMS server. These modules are the “images” and “media” modules.

To make this approach as painless as possible, sites must use version 2.30 or later of the `ldad_web_push.sh` script. Also, the following settings should be in the `ldad_web_config.sh` files for CMS sites *only* (be sure to maintain local modules using your current configuration as an example):

```
secondary_server='taz.crh.noaa.gov'  
web_modules='+images'  
operational_modules='apps|apps-data rt_images|rtimages fxc sshdev.crh.noaa.gov:fxc|fxcdev +media'
```

Some sites have temporarily used the `dev_server` variable to accomplish a similar result, though this prevents synchronization of modules to the development server for offline development. Any sites that have previously set `dev_server` to ‘taz.crh.noaa.gov’ should return to the following value in `ldad_web_config.sh`:

```
dev_server='sshdev.crh.noaa.gov'
```

## Appendix H – Common Rsync Error Codes

The following is a brief list of rsync error and warning codes that you might find in rsync module “*error*” or “*warning*” status files. Though this is not an exhaustive list, it does provide the errors and warnings most often seen when running the LDAD rsync scripts as well as a description of the most likely known causes.

Note that corrective actions on the remote (staging) server can only be made by SRH staff.

- Code 0** – Rsync command was completed successfully.
- Code 1** – Syntax or usage error. This error should not occur when rsync options in *ldad\_web\_config.sh* have not been changed from the tested suggested settings. The most likely problem is improperly set options *ldad\_web\_config.sh*.
- Code 5** – Error starting client-server protocol. There may be a mismatch between the IP address of the machines allowed to synchronize with the remote module and the current local machine’s IP address. This error indicates that there is not a corresponding remote rsync module that the local machine is allowed to access. Check module configurations in *ldad\_web\_config.sh*. If the local configuration file appears correct, the module may not be defined on the staging server or the local LDAD IP address does not correspond with the list IP addresses allowed to synchronize with the remote module.

When Code 5 suddenly occurs on modules that were previously functional, the iptables process may not be operating properly. When the normal iptables address translation fails on *ls1*, the apparent LDAD IP address shifts from the LDAD external IP address normally assigned to *ls1* by the LDAD firewall to the IP address of the LDAD firewall itself (which is not normally on the allowed list of rsync IP addresses). When this problem is occurring, connect to another Linux machine outside the LDAD DMZ and run “*who*”. This command will indicate the host that you are connecting from. If the indicated host is not the normal LDAD hostname (*ls1*) or LDAD external IP address, address translation is not occurring properly. More than likely the LDAD firewall host name or IP address is indicated instead. To ensure that all network services restart properly, *reboot ls1* to correct network address translation.

- Code 12** – Error in rsync protocol data stream. The LDAD IP address change issue described in Code 5 above may also apply, though it is more likely that the remote module is not properly defined or not defined at all. If the remote module appears to be correctly defined, the associated directory may not exist or permissions on the module’s directory on the staging server may not allow the rsync process to synchronize.
- Code 23** – Partial transfer due to error. The code 23 will trip a “*warning*” status file rather than an “*error*”. This code indicates that a remote file or directory cannot be overwritten or deleted. This is a permissions issue on the staging server. New files and files with proper permissions will synchronize while files with bad permissions cannot be updated or removed on the staging server.
- Code 24** – Partial transfer due to vanished source files. This is not considered an error by the rsync scripts. It simply means that a file on LDAD was deleted before it could be copied. This can happen under normal circumstances.

**Code 30** – Timeout in data send/receive. Timeouts are most likely to occur when file transfers are very large. Though small adjustments to timeouts or allowed bandwidth can be made in *ldad\_web\_config.sh*, inspect the data to be transferred to determine whether rsync is the best way to do it. For some applications, the LDAD rsync scripts may not be an appropriate method. Contact SRH for guidance if timeouts frequently occur as adjusting timeouts and bandwidth in an attempt to fix timeouts could have other unintended consequences.

## Appendix I – Service Backup

The LDAD rsync scripts allow updating of *operational* modules for service backup. It is assumed that except under extraordinary conditions, the primary office will always update static web images.

The rsync script will attempt to provide backup for all configured operational modules. Synchronization of backup operational modules occurs when new files appear (in absence of a corresponding “*block*” file). Backup module syncs do not delete existing files from the rsync server to avoid removal of pre-existing files on the web. Because deletion does not occur in backup mode, backup offices do not need to pull existing data before providing backup.

When an office is not providing service backup, “*block*” files are a good idea to allow configuration and testing of scripts and procedures that create the images and data for backup.

Due to the potential complexities of providing data for service backup by rsync, it is a good idea to devise a method to start and stop production of backup datasets through the use of scripts and clearly defined local procedures. These scripts and procedures should be based on the guidance described in the sections that follow.

### In order for service backup to work properly:

- Backup office *operational* rsync modules must be configured on the rsync server to allow access from the backup office LDAD IP addresses.
- The `ldad_web_push.sh` script must be at least version **2.33** for backup operations to go as intended.
- Ensure that the “*backup\_site*” configuration setting contains a space-delimited list of office IDs (as used in the offices’ rsync module names). This list can remain empty for any site not prepared to provide service backup.

### Beginning Operational Rsync Backup:

- Failed office requests that the backup office begin creating content for operational modules.
- Failed office creates “*block*” files for operational modules or modifies cron to disable rsync of those modules. The block file prevents deletion of files that are uploaded by the backup office during service backup.
- Backup office runs a local script, or follows a locally defined procedure, that removes the “*block*” files for all operational modules to be backed up. This step may or may not be necessary depending on the method used for initiating backup data production.
- The backup office executes backup scripts that create data for failed office operational modules.

### Ending Operational Rsync Backup:

- The failed office informs the backup office when it is ready to resume operational status.
- The backup office ceases creation of backup data.
- The backup office creates “*block*” files for backup operational modules.

- The failed office creates a “*pull*” control file for each operational module that was backed up to get any data that was produced by the backup office during service backup.
- The failed office removes “*block*” files for local operational modules.
- The backup office may clean out the contents of backup office rsync directories once the primary office is considered stable and fully operational.

# Release Notes

## New in Version 2.21

- Documentation (including table in Appendix C) reflects new web module for Articulate presentations.
- New setting `prepend_server` is added to allow better identification of control files (see Appendices E & F). Default is “0” if not defined, resulting in no change in function.
- Appendix F has been added to the documentation to describe the control and status file base file naming convention.

## New in Version 2.30

- Addition of `secondary_server` variable to `ldad_web_config.sh`.
- Modules immediately preceded by a plus ‘+’ symbol are now published to the server defined by `secondary_server` in addition to the `rsync_server`. This is to allow synchronization of modules needed on both the operational web server and the operational CMS, which are not hosted on the same servers.

## New in Version 2.30a

- Movement of script files to a zip file for easier download from website. Install procedure has been updated to reflect this change.
- Samba, WebSync, `get_current_stats.sh`, and `findWebOrphans.pl` documentation moved to separate documents.

## New in Version 2.32

- Incomplete transfers due to rsync error 23 will result in the generation of “warning” status files in the `tmp` directory. These are permission errors resulting from permission problems on the remote rsync server that may need to be resolved by a system administrator. Retries of partial rsync operations are not automatically retried.
- Incomplete transfers due to rsync error 24 will be ignored. These are generated when a file queued for transfer is removed before the transfer occurred.
- Changed example `ldad_web_config.sh.cms` example configuration script to include “images” in the no-delete modules list.
- **BUGFIX:** `ldad_web_push.sh` modified to correctly handle finding hung tasks with process IDs <10000. Thanks to Clark Safford for finding this one.

## New in Version 2.33

- Fix in service backup logic.