STEP

STEP SYSTEM ADMINISTRATION GUIDE

Release 9.1-MP6 (July 2019)

StiboSystems
MASTER DATA MANAGEMENT

Table of Contents

STEP System Administration

There are many administrative tasks that are needed to keep a STEP system running at optimum levels. This section of the documentation introduces those responsible to the common tasks, challenges, and issues that often occur when working with administration of a complete STEP system and infrastructure.

Some of the information presented may not be applicable to all systems. Any questions should be directed to your Stibo Systems' representative or the Stibo Systems' Technical Support department.

Patching STEP

The system architecture of the STEP platform is split up into separate components, each of which may access other components through a set of component APIs. This component-based architecture satisfies the otherwise contradictory requirements for longer time between releases and fast introduction of new improvements. Customers can choose to upgrade specific components in order to take advantage of new features and important updates, whilst keeping the core of STEP and other unrelated components unchanged. This approach reduces the risk and workload involved in testing new updates.

When upgrading components individually, the customer may choose not to upgrade to the newest STEP release. If the feature is available in a new component, that component can simply be installed on its own. If the feature is available as an upgrade to an existing component, that component may simply be upgraded while keeping other components as they are.

Note: Available component updates are made visible on a STEP system similarly to the way updates are to mobile phone apps, i.e., with release notes detailing the new features and fixes available relative to the current installation, and with instructions on how to perform the update.

Components have separate release cycles limited only by the dependencies introduced when one component uses another component. Each component declares its dependency on other components through principles where a given component version may depend on a specified range of versions of another component.

For additional information on patching STEP components, see the following topics within this documentation:

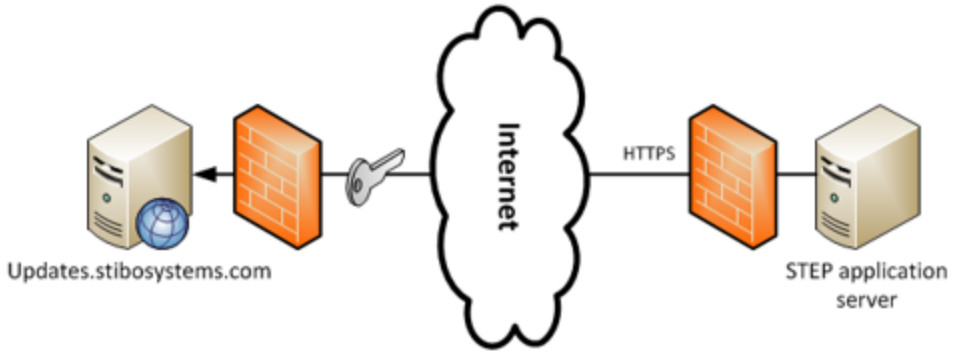
- Patching Methods
- SPOT Program
- STEP Patching Procedures
- Patching Security
- Configuring a Private Updates Mirror

Patching Methods

Patch operations in the STEP system are defined by the specific component(s) being installed / upgraded. These component updates are downloaded either directly from one of the Stibo Systems Global Updates Mirrors (Release Server) or from a private updates mirror at the customer can be used to execute these operations. The connection to either of the two uses an encrypted network connection over HTTPS. Connections are always initiated from the customer side. The update mirror will at no time initiate a connection to the STEP environment.

Direct Connection to Release Server

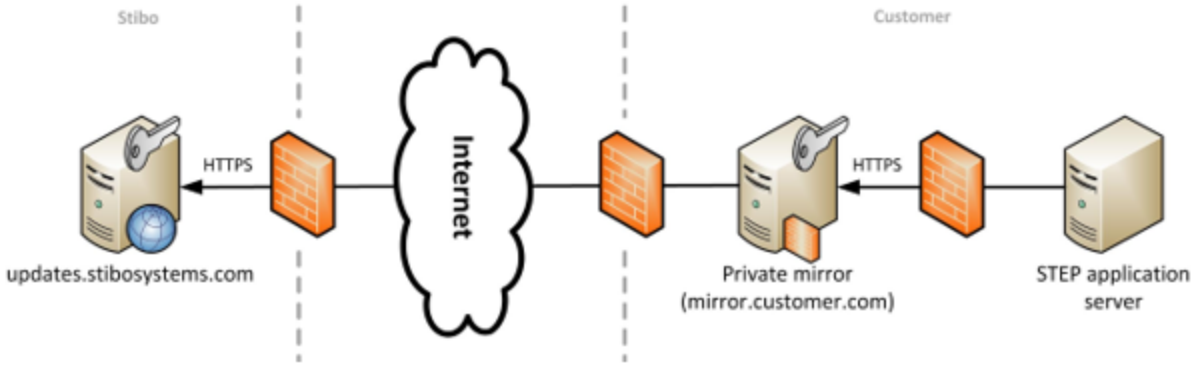
Downloading updates directly from a Release Server is the default method for patching. Using this method, the STEP environment is configured to allow an encrypted connection by HTTPS to the release server. This method offers the best security.



Advantages to using the Direct Connection method include: faster support from Stibo Systems by providing complete version information and a simplified infrastructure.

Private Updates Mirror

As an alternative to accessing the release server directly, it is possible to set up a Private Updates Mirror and configure SPOT on the internal STEP servers to use the mirror instead.



Advantages

The advantages of using the Private Updates Mirror method are:

- If the internet connection or the global updates server breaks down, the already downloaded files will still be available.
- The internet connection bandwidth consumed is reduced by avoiding repeated downloads.
- The network configuration is simpler as only the mirror needs to access the updates server, while the individual SPOT instances can be configured to talk only to the private mirror on the internal network.

Requirements

To run a private mirror server, you need:

- A 64-bit Linux host, not shared with STEP.
- java 8 64-bit (and updated version will be installed by SPOT, so the OS version is okay for bootstrapping).
- Enough storage to hold the entire mirror (400 GB will suffice).
- Outgoing internet access to the Stibo Systems updates servers on port 443.
- Incoming aces from the private network on port 443 for the SPOT hosts.
- A DNS entry on the local network that can be expected to never change, so *mirror.customer.com* would be preferable to *pc2016-02-13-room7-linux-test-dl120g9.dhcp.customer.com*.

Upstream Root Mirrors

The root mirrors that the private mirror connects to can be listed using `spot --mirrors`, but these are the current hosts:

- *dk1.updates.stibosystems.com*: Primary root mirror.
- *dk2.updates.stibosystems.com*: Secondary root mirror.
- *updates.stibosystems.com*: Fail-over mirror on a shared IP between the two root mirrors.

Outgoing TCP access on port 443 must be allowed to each of the root mirror IP addresses from the private mirror, this way the mirror has more upstream mirrors to pick from if one fails.

SPOT Program

The Stibo Patch Operations Tool (SPOT) program initiates an encrypted exchange between the customer site and the update mirrors at Stibo Systems. This program runs off either the STEP application server or on a dedicated SPOT support installation PC.

The communication sequence between the SPOT program and the update mirror is as follows:

1. SPOT stores the current thin snapshot of version information to updates.stibosystems.com.
2. SPOT fetches the desired recipe of the software bundles to download.
3. SPOT downloads the actual bundles.
4. SPOT stores the updated thin snapshot of the version information to updates.stibosystems.com.

Storing the thin snapshots to updates.stibosystems.com serves two purposes. It enables:

- Stibo Systems to support the STEP environment by providing complete version information.
- Easy creation of the exact software configuration for additional Test / QA environments and in the case of disaster recovery.

Both the metadata (including the thin snapshot) and the bundle recipe, together with the actual bundles, are cached by SPOT and only the files that are actually needed are ever downloaded, so the amount of data transferred is as low as possible.

The SPOT program can be found in the home directory of the STEP installation on the application server. On a Linux server, this will typically be in `/opt/stibo/step`. On a Windows server, this will typically be in `E:\stibo\step`.

Using the Upgrade Command

To help simplify the analysis process and make it easier to work with components, the `--upgrade` command can be executed to look for possible upgrades to the components installed on a STEP system. Users can also run the command to search for a component not already installed to verify availability and compatibility with their version of STEP.

The `--upgrade` (or `-u`) command is used to calculate the newest possible version of the listed components that can be installed given different restrictions on how large of an upgrade is allowed. This command never changes the STEP system or performs any automatic upgrading.

To further facilitate the process, the upgrade options output that is displayed upon running the `--upgrade` command includes a recipe file that can later be applied to the system.

Outlined below are the different upgrade command options with examples. The name of the actual component(s) should be used in place of what is shown in the examples.

Upgrade Options	Examples
Upgrading one component	To upgrade the Experian component, use: `--upgrade=experian`
Upgrading several components	To upgrade both Experian and Loqate, use: `--upgrade=experian,local-loqate`
Upgrading the baseline (the STEP version such as 9.0, 9.1)	The baseline can also be upgraded by using the component name *step*: `--upgrade=step` Pick the release of the baseline by specifying a prefix: `--upgrade=step:9.1` When a prefix is specified, the newest version matching the prefix will be tried.

Installation candidates (components that have not yet been installed) can also be found by using the `--upgrade` command, as described above.

Upgrade levels

The calculation used to determine upgrades can potentially produce suggestions for up to five levels of upgrade. Only the upgrades that bring newer versions of the listed components will be shown in the result. Below, the options shown are sorted by how aggressive the update would be with regard to introducing new component versions.

Level	Description
Listed	This is the most conservative upgrade possible where only the listed components are touched.
Dependents	This level allows upgrading of: • The listed components • The components that depend on the listed components
Dependencies	This level, listed with `DEPENDENCIES_BUT_NOT_BASELINE` in the file name, allows upgrading of: • The listed components • The components that depend on the listed components

Level	Description
	• The components that the listed components depend on, but not STEP baseline
Baseline within Maintenance Patch	This level, listed with `BASELINE_WITHIN_MP` in the file name, allows upgrading of all components, including the STEP baseline, but only to the latest maintenance patch of the same release as the one currently installed. For example, if the system has step-8.2-mp1 installed, then this level would look for the newest MP of that release, possibly 8.2-mp3, but not 8.3.
Baseline	This is the least conservative upgrade level, which allows upgrading of all components, including the STEP baseline to the latest released version.

The upgrade options are shown on screen with the upgrade file recipes listed. Users can use standard commands to view a detailed change log, prepare for an installation, and to apply changes to their STEP system.

For example: `--upgrade=inmemory`

```
Found 3 possible upgrades to choose from:
========================================================================
Option 1: Upgrade only the listed components
Components:
       * assetloader: Keep at 7.0.14 (newest available: 7.0.24)
       * inmemory: Upgrade from 7.0.10 (newest available: 7.0.23)
       * spot: Keep at 7.0.48 (newest available 7.0.65)
File: /home/step/admin/spot/recipes/upgrade/upgrade.LISTED.2017-11-01-15-24-
28.spr
========================================================================

Option 2: Upgrade to latest maintenance patch within the same STEP release
+ All components
Components:
       * assetloader: Keep at 7.0.14 (newest available: 7.0.24)
       * inmemory: Upgrade from 7.0.10 to 7.0.15 (newest available: 7.0.23)
       * spot: Keep at 7.0.48 (newest available 7.0.65)
       * step: Upgrade from 8.0-mp3-2016-09-06-14-12-00 to 8.0-mp4-2016-10-04-10-10-
27 (newest available: 8.2-mp3-2017-11-02-07-39-51)


File: /home/step/admin/spot/recipes/upgrade/upgrade.BASELINE_WITHIN_MP.2017-11-
01-15-24-28.spr

========================================================================

Option 3: Upgrade to latest STEP release (full upgrade)
+ All components
```

```
Components:
       * assetloader: Keep at 7.0.14 (newest available: 7.0.24)
       * inmemory: Upgrade from 7.0.10 to 7.0.14 (newest available: 7.0.23)
       * spot: Keep at 7.0.48 (newest available 7.0.65)
       * step: Upgrade from 8.0-mp3-2016-09-06-14-12-00 to 8.1-mp5-2017-10-02-16-10-
00 (newest available: 8.2-mp3-2017-11-02-07-39-51)


File: /home/step/admin/spot/recipes/upgrade/upgrade.BASELINE.2017-11-01-15-24-
28.spr
```

If the system is ignoring any components or if the system cannot find a way to upgrade the components specified, the applicable messaging will be shown on the screen. All ignored versions will not be considered when trying to find an upgrade.

Starting with STEP 8.3 and with all subsequent versions, the `--upgrade` command can be used in place of the installation commands. For example, `--upgrade=wikimetadata` or `--upgrade=acrolinx`.

STEP Patching Procedures

All commands listed are valid for any STEP environment, counting single application server setups and clusters.

Back Up the STEP Database and Application

Before patching STEP, a fallback procedure should be developed to mitigate any risk. Ideally, full back ups of the STEP database should be maintained, and the option to restore the database to a specific point in time should be available.

Note: The requirement for each individual patch may vary. Refer to the relevant release note for more information.

Back ups of the STEP application should also be maintained, including all files provided in STEP_HOME/config.properties and a snapshot of the STEP system itself.

To take a snapshot of the STEP system:

```
cd /opt/stibo/step
./spot --snapshot=/workarea/<snapshot-env-date>.spr
```

Prepare the Patch

The patch should be downloaded in advance to avoid unnecessary downtime for deployment.

A STEP core patch may look like the following command:

```
./spot --prepare=to:step/trailblazer/step-trailblazer-<release>.spr
```

Sometimes, customers have their own components in addition to the STEP core, and the command could look something like this:

```
./spot --prepare=to:step/trailblazer/step-trailblazer-
<release>.spr,to:customer/<customer>/<customer>-addon/7.0/<customer>-addon-
7.0.x.spr
```

Install the Patch

The patch should be installed by the following command:

```
./spot --apply=to:step/trailblazer/step-trailblazer-<release>.spr
```

With customer components included, the command looks like this:

```
./spot --apply=to:step/trailblazer/step-trailblazer-
<release>.spr,to:customer/<customer>/<customer>-addon/7.0/<customer>-addon-
7.0.x.spr
```

STEP will automatically stop and start during the patch session.

In case of any deprecated parameters in the configuration, follow the instructions on the screen that explain how to correct and restart STEP.

```
./spot --start
```

Fallback

In the event of errors during patching, it may be necessary to restore STEP to a previous state.

Depending on the contents of the patch the following steps should be completed when reverting the patch:

1. Stop STEP

   ```
   ./spot --stop
   ```

2. Restore database
3. Restore configuration files
4. Redeploy STEP using a snapshot

   ```
   ./spot --apply=/workarea/<snapshot-env-date>.spr --sync --syncmode=delete
   ```

Using the snapshot and the above `--sync --syncmode=delete` command will entirely recover STEP and delete any files related to a failed patch-session.

Note: Refer to the relevant release note to check if restoring the database is required for the patch in question.

Patching Security

Stibo Systems only distributes software via the updates.stibosystems.com server or one of the official mirrors.

The update mirror web server is configured to only communicate via HTTPS (never plain HTTP) on port 443, with only the high security cipher suites using the Apache SSLCipherSuite 'HIGH' option and only communicating with clients which have a proper client certificate issued by the build system certificate authority (CA) of Stibo Systems. This Stibo-specific CA was created solely for the purpose of certifying various STEP-related infrastructures.

Unlike a standard website where an external CA-signed certificate is used for ease of access by multiple clients (users), the updates server has only one client that is allowed to communicate with it: the SPOT client. For this reason, Stibo Systems believes this to be a safer and stronger security approach – over using an external CA certificate – as it is not possible for a cyberattacker to use a fake certificate from a compromised external CA to gain access.

By taking this approach, some auditing tools may register a false positive and flag the server's certificate as self-signed. Because of this, security teams should configure these tools to trust Stibo Systems' CA to certify *stibosystems.com* domains

The client certificate required for communicating with the update mirror is included in the STEP installation package, and is used by the SPOT program to fetch both the software required for the initial installation and future application updates. Only the certificate used by the updates server will be trusted by SPOT for downloading these installation bits and updates.

All the certificates involved use 2048-bit RSA keys, so the system is considered secure against any man-in-the-middle attacker for the foreseeable future. Even with a valid client certificate, the operations allowed are severely limited to downloading only the licensed software produced by Stibo Systems and to saving customer-specific thin snapshots that do not contain software, so a compromised client would not be able to affect other customers or compromise other clients.

The SPOT program caches all files locally and validates contents using a SHA-1 hash before using the cached files, so the amount of traffic is kept as low as possible while ensuring the integrity of the cached files.

At no point will the STEP software communicate customer data back to the update mirrors at Stibo Systems. The thin snapshots uploaded to the release server contain only a list of versions of the installed STEP software components and they are only used by Stibo Systems to provide the best support to the STEP system.

Configuring a Private Updates Mirror

Requirements for running a private mirror server include:

- A 64-bit Linux host, not shared with STEP.
- 64-bit Java 8 (an updated version will be installed by SPOT, so the OS version is okay for bootstrapping).
- Enough storage to hold the entire mirror (currently 400 GB will suffice).
- Outgoing internet access to the Stibo Systems updates servers on port 443.
- Incoming access from the private network on port 443 for the SPOT hosts.
- A DNS entry on the local network that can be expected to never change, so *updates.example.com* would be preferable to *pc2016-02-13-room7-linux-testdl120g9.dhcp.example.com*.

The root mirrors that the private mirror connects to can be listed using `spot --mirrors`, but these are the current hosts:

- *dk1.updates.stibosystems.com*: Primary root mirror.
- *dk2.updates.stibosystems.com*: Secondary root mirror.
- *updates.stibosystems.com*: Fail-over mirror on a shared IP between the two root mirrors.

Outgoing TCP access on port 443 must be allowed to each of the root mirror IP addresses from the private mirror, this way the mirror has more upstream mirrors to pick from if one fails.

The mirror server listens on three ports:

- *10080*: The Admin port of dropwizard, which is used to serve HTTP requests that allow monitoring the health of the server. The init script uses this port to check if the server is running.
- *10081*: The stop port of Jetty. The init script uses this port to shut down the server in an orderly fashion. This port should not be accessed from outside the machine itself.
- *10082*: The HTTPS service port that serves the actual mirror. This port should not be accessed from outside the machine itself.

These ports are all internal to the host that the server runs on and external systems should not connect directly to them (with the possible exception of having a monitoring system talking to port 10080.)

Important: Do not configure any STEP systems to talk to the mirror on port 10082. Port redirection (as described in the next section) must be set up.

IPTables Rules

It is impossible to listen to port 443 when running the Java process as an unprivileged user. To account for this, a set of iptables rules must be used.

There are two ways to install the required rules: either run the mirror script as root when starting the server or set up iptables at the OS level. If the init script is called by root, then it will install the needed port redirection, but if the administrator tasked with maintaining the mirror does not have sudo access to this script, then the rules can be inserted into the */etc/sysconfig/iptables* config file, allowing the OS to load the rules at boot time.

These rules redirect all incoming requests to TCP port 443 over to port 10082 where the server listens.

To configure iptables on the server, switch to the root user and run the following command to view the current settings:

```
[root@mirror mirror]# /sbin/iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source      destination
1    ACCEPT     all  --  0.0.0.0/0   0.0.0.0/0   state RELATED,ESTABLISHED
2    ACCEPT     icmp --  0.0.0.0/0   0.0.0.0/0
3    ACCEPT     all  --  0.0.0.0/0   0.0.0.0/0
4    ACCEPT     tcp  --  0.0.0.0/0   0.0.0.0/0   state NEW tcp dpt:22
5    REJECT     all  --  0.0.0.0/0   0.0.0.0/0   reject-with icmp-host-
prohibited

Chain FORWARD (policy ACCEPT)
num  target     prot opt source       destination
1    REJECT     all  --  0.0.0.0/0    0.0.0.0/0   reject-with icmp-host-
prohibited

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source       destination
```

In the output, there will be a line that displays `REJECT` as the `INPUT` type, and in its first column `(num)`, the line number is shown ('5' in the above example.) This line number will be the starting line for adding entries to the iptables configuration.

Once this information is known, run the following commands to add the needed port-opening entries:

```
[root@mirror mirror]# /sbin/iptables -I INPUT <line_number> -p tcp -m tcp --dport
443 -j ACCEPT
[root@mirror mirror]# /sbin/iptables -I INPUT <line_number> -p tcp -m tcp --dport
10082 -j ACCEPT
```

In the example above, the line number shown is '5', and therefore, the commands would look like the following commands:

```
[root@mirror mirror]# /sbin/iptables -I INPUT 5 -p tcp -m tcp --dport 443 -j
ACCEPT
[root@mirror mirror]# /sbin/iptables -I INPUT 6 -p tcp -m tcp --dport 10082 -j
ACCEPT
```

Afterwards, add the entries for redirection by executing these commands:

```
[root@mirror mirror]# /sbin/iptables -t nat -A PREROUTING -p tcp -m tcp --dport
443 -j REDIRECT --to-ports 10082
[root@mirror mirror]# /sbin/iptables -t nat -A OUTPUT -o lo -p tcp -m tcp --dport
443 -j REDIRECT --to-ports 10082
```

Once that has been done, the added entries can be checked by running the commands that follow:

```
[root@mirror mirror]# /sbin/iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source       destination
1    ACCEPT     all  --  0.0.0.0/0    0.0.0.0/0   state RELATED,ESTABLISHED
2    ACCEPT     icmp --  0.0.0.0/0    0.0.0.0/0
3    ACCEPT     all  --  0.0.0.0/0    0.0.0.0/0
4    ACCEPT     tcp  --  0.0.0.0/0    0.0.0.0/0   state NEW tcp dpt:22
5    ACCEPT     tcp  --  0.0.0.0/0    0.0.0.0/0   tcp dpt:443
6    ACCEPT     tcp  --  0.0.0.0/0    0.0.0.0/0   tcp dpt:10082
7    REJECT     all  --  0.0.0.0/0    0.0.0.0/0   reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT)
num  target     prot opt source       destination
1    REJECT     all  --  0.0.0.0/0    0.0.0.0/0   reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source       destination
```

```
[root@mirror mirror]# /sbin/iptables -L -n --line-numbers -t nat
Chain PREROUTING (policy ACCEPT)
num  target     prot opt source       destination
1    REDIRECT   tcp  --  0.0.0.0/0    0.0.0.0/0    tcp dpt:443 redir ports 10082

Chain INPUT (policy ACCEPT)
num  target     prot opt source       destination

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source       destination
1    REDIRECT   tcp  --  0.0.0.0/0    0.0.0.0/0    tcp dpt:443 redir ports 10082

Chain POSTROUTING (policy ACCEPT)
num  target     prot opt source       destination
```

If everything looks to be correct, save the configuration so that it will be loaded each time the system reboots using these commands:

```
[root@mirror mirror]# /sbin/service iptables save
[root@mirror mirror]# /sbin/service iptables stop
[root@mirror mirror]# /sbin/service iptables start
```

Important: If the mirror server is a RHEL 7.x system, the above `/sbin/service iptables stop` and `/sbin/service iptables start` commands should be replaced with the following: `/bin/systemctl stop iptables` and `/bin/systemctl start iptables`.

Once complete, the */etc/sysconfig/iptables* config file should look similar to the following:

```
# Generated by iptables-save v1.4.7 on Tue Jun 21 14:16:10 2016
*nat
:PREROUTING ACCEPT [4595:497811]
:INPUT ACCEPT [1:28]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A PREROUTING -p tcp -m tcp --dport 443 -j REDIRECT --to-ports 10082
-A OUTPUT -o lo -p tcp -m tcp --dport 443 -j REDIRECT --to-ports 10082
COMMIT
# Completed on Tue Jun 21 14:16:10 2016
# Generated by iptables-save v1.4.7 on Tue Jun 21 14:16:10 2016
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [434:47393]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 10082 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
# Completed on Tue Jun 21 14:16:10 2016
```

Installing a Private Mirror

1. Satisfy all requirements mentioned above.
2. Make a note of the DNS name which all the SPOT hosts will be using. In this example we will call it *mirror.customer.com*.

3. Contact Stibo Systems Technical Services and request that a system name is created for the mirror. This must be human readable and unique. The system name in this example is *your-mirror*.
4. Create a directory for the mirror.
5. Unzip the SPOT foothold (must be newer than the March 2016 release).
6. Run: `./spot --enroll=mirror:your-mirror:mirror.customer.com`.
7. Run: `./spot --apply=to:updates/mirror/latest.spr`.
8. Edit the *mirror.yaml* file and review the options in the file. Some of them, particularly those dealing with mailing of errors, will need to be changed.
9. Run: `./mirror start`.
10. Your mirror should now be running on *mirror.customer.com*.
11. On a system with STEP installed, run the following command:

    ```
    ./spot --updates=https://mirror.customer.com --ping
    ```

12. As `root`, create a symlink to the mirror script into the appropriate sysv init directories using a command similar to the following:

    ```
    ln -s <mirror_home>/mirror /etc/rc3.d/S90stibo-updatesmirror
    ```

For example:

```
ln -s /home/mirror/mirror /etc/rc3.d/S90stibo-updatesmirror
```

Important: Do not run any of these commands as `root`. Make sure an unprivileged user exists for this mirror — e.g., `mirrorsw`.

Once the private mirror has been configured, it can be verified by running the following command on the STEP application server as the `stibosw` (or equivalent) user:

```
[stibosw@app1 step]$ ./spot --mirrors
Stibo Patch Operations Tool
Priority Id          Name                          Url                                [X]
100      <customer>  <customer> local mirror       https://mirror.customer.com         X
30       global      Auto failover mirror          https://updates.stibosystems.com
20       dk1         Primary mirror in Aarhus DK   https://dk1.updates.stibosystems.com
10       dk2         Secondary mirror in Aarhus DK https://dk2.updates.stibosystems.com

Please use spot --mirrors --updates={Url} to set the upstream mirror
```

The mirror can also upgrade itself using the init script by simply running: `./mirror upgrade`. The upgrade command calls the `spot --apply=to:updates/mirror/latest.spr` and `./mirror restart` commands.

If the iptables' rules have been added to the RHEL config file, the init script no longer needs root access and can be started by an unprivileged user. This is done by editing said user's *crontab* entries (such as `crontab -e`) and adding the following line:

```
@reboot <mirror_home>/mirror start
```

For example:

```
@reboot /home/mirror/mirror start
```

Preemptive Download

The mirror server can download files before the STEP systems ask for them. Doing this allows most files to be served from the local mirror without waiting for the upstream mirror, so better performance can be expected. This comes at the cost of more disk space being utilized and the possibility of downloading files that end up never being needed.

The download option has three possible values:

- `download: HISTORIC`: Downloads all the files available from the upstream mirror, regardless of age. This requires about 1.5 TB of space.
- `download: RELEASED`: Downloads newly released code as soon as it becomes available, this is the default and will steadily consume space. About 2 GB are consumed per month.
- `download: ON_DEMAND`: Nothing is downloaded until a client asks for it.

Note: When new content (e.g., monthly maintenance patches, add-on components, hotfixes, etc.) is downloaded to the mirror, it will be saved to the server's `<mirror_home>/content/takeout` directory (e.g., `/home/mirror/content/takeout`).

As no user is actively waiting for the preemptive downloads to complete and because the downloads can be quite large, the bandwidth consumed by the background downloads can be limited via the `bulkDownloadSpeedInMbitPerSecond` configuration option. The default limit is 10 Mb/s, so the expected lag after a release of STEP until the mirror is in sync should be less than an hour.

The bulk download speed limit is applied to the download of newly released files and historically released files separately, so if a historic download is running, then the two bulk processes can consume twice the speed limit in total.

If downloads take a long time to complete, it could be because the network or the upstream mirror is overloaded. To avoid contributing to the problem the bulk download threads will sleep for a while after completing a download. The amount of time to sleep after a download can be specified using the `bulkBackoffFactor` option, which defaults to '1.5'.

For example, if a download takes two seconds then a *bulkBackoffFactor* of '1.5' means that the process will sleep three seconds before downloading the next file.