Debugging / Tools

Sometimes it's necessary to run some tests or debug specific issues in Blesta. Here are some common things you may need to do, or may be instructed to do by support, when something goes wrong.

Check PHP CURL, outbound TLS Version

Blesta integrates with many 3rd party services and it may be necessary to ensure that your server is capable of making a TLS connection with certain minimum requirement. Here's sample code you can place on your server and access in your browser. It will return the most modern version of TLS that your server is capable of negotiating. If it say TLS 1.1 and the service you are connecting to requires TLS 1.2, then it will explain why a connection is not possible.

curl-check.php

```
<?php
    $ch = curl_init('https://www.howsmyssl.com/a/check');
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $data = curl_exec($ch);
    curl_close($ch);
    $json = json_decode($data);
    echo "<pre>TLS version: " . $json->tls_version . "</pre>\n";
?>
```

Copy the code into a text file named curl-check.php and upload to your Blesta installation directory. Access in your browser and it will display the most modern TLS version supported. e.g. **TLS version: TLS 1.2**

Enabling Error Reporting & Debugging

To enable error reporting, edit /config/blesta.php and change **Configure::errorReporting(0);** to **Configure::errorReporting(-1);** You may also wish to enable System Debug. To do so, change **Configure::set("System.debug", false);** to **Configure::set("System.debug", true);**

> **Do not leave System.debug enabled**
>
> Be sure to change these settings back when you are done, especially System.debug. Leaving System.debug enabled can cause other issues and may become responsible for other errors within the system.

Checking Error Logs

Starting with Blesta 4.1.0, errors are logged to disk by default. To see if your system is logging, check the path under Settings > System > General > Basic Setup for "Log Directory". This should be the full system path to your logs directory, and it should indicate to the right of the field that it is writable.

Then, navigate to this directory on your server. You should see some log files, including:

general-info-DATESTAMP.log
general-notice-DATESTAMP.log
general-warning-DATESTAMP.log
general-alert-DATESTAMP.log
genera-error-DATESTAMP.log
general-emergency-DATESTAMP.log

Support staff are mostly interested in the **error** and **emergency** logs, but depending on the issue the other logs may be important. The error log contains PHP exceptions, and the emergency log contains Blesta errors. Find the files with the most recent date, and open them to find the errors. If the error is reproducible and you have SSH access to the server, you can "tail" the log files while reproducing the error to more easily find what you need. For example: **tail -f general-*.log** will tail all of the log files at once.

Forcefully disable auto-debit for clients that do not have active services

Clients who have no active services, yet have open invoices with auto-debit enabled will be charged. This query will create a client setting for autodebit and set it to false for any clients that do not have any active services. **Backup your database first!**

```
INSERT INTO client_settings (client_settings.key, client_settings.client_id, client_settings.value) SELECT
'autodebit', `clients`.`id`, 'true' FROM `clients` LEFT JOIN `services` ON `services`.`client_id` = `clients`.
`id` AND `services`.`status` = 'active' LEFT JOIN `client_settings` ON `client_settings`.`client_id` =
`clients`.`id` AND `client_settings`.`key` = 'autodebit'  WHERE `services`.`id` IS NULL AND `client_settings`.
`key` IS NULL;
```

Fetch the Collation of Tables

This query can be useful for fetching the collation of your database tables. Replace DATABASE-NAME with your database name. The query should return each collation found and the tables that use it.

```
SELECT table_collation AS collation,GROUP_CONCAT(table_name) AS tables FROM information_schema.tables WHERE
table_schema='DATABASE-NAME' GROUP BY collation;
```

Upgrading MariaDB

The recommended requirements for Blesta 5.0+ includes MySQL 5.7.7+ or MariaDB 10.2.2+. If you are running MariaDB < 10.2.2, you can upgrade to 10.2.x using this script.

First, backup all databases.

```
mysqldump -u root -p --all-databases > all_databases.sql
```

On CentOS 7.x with no root password set. use "mysql_upgrade -u root -pPASSWORD" if there is a password.

```
#!/bin/bash
# Upgrade MariaDB to 10.2

# ------- REPLACE MARIADB -------
# Set up MariaDB 10.2 repo
cat >/etc/yum.repos.d/MariaDB10.repo <<EOL
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/10.2/centos7-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
EOL

# Stop MariaDB
systemctl stop mariadb

# Remove old MariaDB
yum -y remove mariadb-server mariadb mariadb-libs
yum clean all

# Install MariaDB 10.2
yum -y install MariaDB-server MariaDB-client
# UNCOMMENT IF YOUR PHP MYSQL PACKAGE IS php-mysql
#yum -y install php-mysql
systemctl enable mariadb
systemctl start mariadb
mysql_upgrade

# Re-install postfix!
# UNCOMMENT IF RUNNING VIRTUALMIN OR POSTFIX
#yum install postfix -y
#systemctl enable postfix
#systemctl start postfix
```