
Z-Merge installation manual



Z-Merge is a realtime server based synchronisation framework. Using Z-Merge you are able to keep different kind of data on several systems in sync.

This document will give you general information about the functionality of Z-Merge and guide you through the installation and configuration.

Table of contents

Z-Merge installation manual.....	1
Introduction.....	2
Available Z-Merge connectors.....	2
Technical background on connectors.....	2
Installation.....	3
Topology aspects.....	3
Installing Z-Merge agent.....	3
Basic configuration of the agent.....	3
Database configuration.....	4
Connected server configuration.....	4
Conflict resolution configuration.....	5
Installing 3rd party connectors.....	6
Zarafa Connector configuration.....	6
Zarafa Server configuration.....	7
SugarCRM Connector configuration.....	8
Basic testing connectors.....	8
Securing connector access.....	8
Finalizing Installation.....	9
Preparing User-Mapping.....	9
Checking the Agent & Connectors configuration.....	9
Synchronise existing data.....	10
Installing LiveLog.....	10
Running.....	11
Starting the services.....	11
Reading logs.....	11
Logentries and their meaning.....	12
Monitoring.....	12
Troubleshooting.....	13
Z-Merge Agent error messages on checking configuration.....	13
General connector related error messages.....	13
Specific connector related error messages – Zarafa Connector.....	14
Specific connector related error messages – SugarCRM Connector.....	15
SugarCRM Connector: no delete request sent for some objects.....	16
Feedback & Contact.....	16

Introduction

Z-Merge is an autonomous server side service which interlinks several servers with each other. Z-Merge functions as central communication platform between two or more servers and provides an event based synchronisation to all connected systems in almost realtime. It is in constant communication and forwards events and their related data from one server to all others. It takes care of all conversion and mapping of data and objects. All servers continue with own databases and completely autonomous data, so that real objects and data are present on the system, as if it was a single server.

All connections base on web service technologies to transmit events and data. That way Z-Merge depends only on persistent and continuous TCP/IP connections. The Z-Merge Agent is connected to each server by one connection which handles events and data.

In this document the main Z-Merge instance, the **Z-Merge Agent**, may be referred only as **agent**. The agent's communication partners are the 3rd party connectors, referred only as **connector**.

Available Z-Merge connectors

Right now Z-Merge has following completely tested connectors:

- Zarafa Groupware Server
 - Supports Zarafa 5.20 and later
 - Package name: *z-merge-zarafa*

- SugarCRM – Customer Relationship Management
 - Supports versions 4.5.1e/i, 5.0.0c/e/f, 5.1.0/b/c and 5.2.0
 - Package name: *z-merge-sugar*

The provided connectors are developed in PHP. Each of them utilises an own webservice implementing parts of the Z-Merge protocol. The access to the target systems is established via designated interfaces (*PHP-MAPI* for Zarafa and the SugarCRM-libraries and custom hook methods for SugarCRM). That way the connector is highly integrated into the appropriate system without the need of major internal modifications on the target systems.

The connector is technically independent from the system. It is possible simply to install the connector, so integrating the server into a Z-Merge structure.

Note: Before updating the main applications to newer versions, please always consult the Z-Merge webpage in order to know if the version you want to upgrade to is already supported by the connector. Eventually it will be necessary to upgrade the connector before updating the application.

Technical background on connectors

The webservice of each connector is not standardised and therefore not exactly specified. This is not necessary, because each connector consists of two independent parts:

1. the connector/webservice on the 3rd party system itself providing an interface to the system and communicating by standardised methods
2. and an own conversion component inside the Z-Merge agent.

While the connector on the 3rd party system has to implement a limited number of methods, the transmitted data (DTD) can be chosen by each connector itself. The only component interpreting that data is special agent conversion classes.

If you are interested in integrating and connecting Z-Merge to a new system, please check the *Z-Merge connector specification*.

Installation

Topology aspects

As described above, Z-Merge is an autonomous agent. For small systems this agent can run on one of the servers on which the connector(s) is/are installed, so that no additional hardware is necessary. In these cases the agent will connect through `localhost` to the webservice.

Basically each connector (there have to be at least two to have a “syncable environment”) and the agent are completely separate instances, connected only through TCP/IP connections. The connectors have to be installed on the application servers. For Zarafa the connector has to be installed on a PHP-MAPI enabled system, for example on the system where the WebAccess is installed.

It is possible to have each connector and the Z-Merge agent running on an own physical machine or all on one, depending on your necessities. The only limitation is a stable TCP/IP connection from the agents to each of the connectors machine.

Installing Z-Merge agent

Download the `z-merge-agent` package. It is available as tarball and debian (or compatible) package. To install the deb file use the `dpkg` tool.

```
dpkg -i z-merge-agent_<version>-<build>.deb
```

The files are installed to the `/usr/lib/z-merge` directory. The configuration files are located at `/etc/z-merge`.

For non debian servers you can use the tarball. Please follow the instructions in the `INSTALL` file contained in the package.

Basic configuration of the agent

For new installations copy the `agent-cfg.inc.php.dist` to `agent-cfg.inc.php` using the command:

```
cp agent-cfg.inc.php.dist agent-cfg.inc.php
```

inside the `/etc/z-merge` directory.

When updating the agent to a newer version, please compare the `agent-cfg.inc.php.dist` to your existing `agent-cfg.inc.php` and update changed/new configuration parameters to your needs.

- Set the parameter `ZMERGE_DIR` to the location of the agent. The default value is `/usr/lib/z-merge`
- Set the parameter `ZMERGE_PID` to the location of the pid file. It prevents multiple starts of the agent. The default value is `/var/run/z-merge.pid`
- The parameter `ZMERGE_LOGFILE` specifies the default location of the log file. The default value is `/var/log/z-merge/z-merge.log`
- The parameter `ZMERGE_ERRORLOGFILE` specifies the default location of the error log file. The default value is `/var/log/z-merge/z-merge-errors.log`
- The parameter `ZMERGE_LOGLEVEL` specifies the loglevel of the agent. Possible values are `FATAL`, `ERROR`, `WARN`, `INFO`, `DEBUG`. The default level is `INFO`. When encountering problems you should set the loglevel to `DEBUG`

- The parameter `ZMERGE_LOGSIZE` specifies the size of the default logfile. The value represents the value in megabytes. Modifying this setting you can influence when a logrotate will be performed. In the `DEBUG` modus the agent could produce a lot of logs. In this case consider higher values, for example 50 MB.
- The parameter `ZMERGE_LOGCONFLICTREASONLEVEL` defines whether or at which level the reason of a conflict is written to the log file. The default value is `false` (disabled)
- If the parameter `ZMERGE_FORCELIVESYNC` is set to `true`, the synchronisation starts even if one or more servers are offline (not reachable) or returning errors. That option only applies, if there are more than two servers configured. The default value is `false` (do not start).
- When one or more servers go down, the agent will retry to reach the server after a period of time. That time can be specified in the option `ZMERGE_RECONNECT` in minutes.
- To receive error reports by mail, set the parameter `ZMERGE_SENDERRORS` to the email address of the administrator. To disable that functionality, set the parameter to `false`.
- Per request the agent will only send a limited amount of objects. That value is specified by the parameter `MAX_OBJECTS`. Large requests need a lot of memory on the connector's side. When more objects are available, the agent will send them consecutively in chunks, until all objects were processed correctly. The default value is 50.

Database configuration

As database layer used by the Z-Merge Agent is the `Pear MDB2` module. More information on MDB2 can be found at: <http://pear.php.net/package/MDB2>

- After some time of inactivity the agent could loose the database connection. Set the parameter `ZMERGE_DB_TRYCOUNT` to value bigger than three to re-establish the connection. The default value is 5.
- Set the `ZMERGE_DB_USERNAME` and `ZMERGE_DB_PASSWORD` to the correct values to connect to the Z-Merge agent database.
- Set `ZMERGE_DB_HOSTNAME` to the hostname or IP address where the database is located.
- Set `ZMERGE_DB_DATABASE` to the name of the database to use. Default is `zmerge_agent`
- `ZMERGE_DB_TYPE` specifies the database type which the agent is connected to. At the moment only the value `mysql` is tested and supported.
- The parameters `ZMERGE_DB_DEBUGLEVEL` and `ZMERGE_DB_PORTABILITY` are part of advanced `Pear::MDB` options. Please consult the MDB manual for details or just leave the default values.

Connected server configuration

In the minimal configuration Z-Merge could be used to synchronise two servers. In advanced configurations, you can connect several servers to be synchronised in *almost* realtime.

Each server has a unique `server profile` name which **never should be changed** for already synchronised servers. The profile name identifies the server internally and **can not** be changed afterwards without breaking the synchronisation.

The servers are configured in the `$servers` array. These parameters can be set for each server.

- `SERVER_NAME` – the internal “server profile name” - **never change on running systems!**
- `SERVER_TYPE` – connector type. The constants `SUGARCRM` and `ZARAF` are valid options
- `SERVER_URL` – the base URL of the Z-Merge connector on the server reachable from the Z-Merge agent environments. Please specify the directory, like <http://localhost/z-merge-zarafa/>

- `SERVER_USERNAME` & `SERVER_PASSWORD` – Administrative user's login credentials for this connector
- `SERVER_PROXY` – Proxy information necessary to connect to the server (optional)
- `SERVER_EVENTDRIVEN` – if the connector interface itself waits for change events, set the value to `true`
- `SERVER_NOEVENTQUERY` – if the connector's SOAP interface does not support events, this parameter specifies the time in seconds to wait before querying the server again for changes (default >5 seconds)
- `SERVER_TIMEOUT` – seconds before timing out trying to reach the server. Default > 10 seconds.
- `SERVER_WAITTIMEOUT` – time in seconds to wait for an answer from a connector. This value has to be bigger than the configured time waiting for events in the connector (see “configuring connectors”)
- `SERVER_DUPLICATOR` – if on a profile objects only have one user while on other profile(s) the same object is represented as several objects (for each user), this profiles is indicated to work as a profile to apply duplicates. In these cases set this parameter to true.

Note: It is possible to configure access the connector using Secure Socket Layers (SSL). The SSL connection is made using the `PHP-CURL` libraries which have to be installed.

Conflict resolution configuration

When the agent is running, all writing operations are executed following the “first comes, first served” principle. In fact, with Z-Merge all servers behave like working on the same database. That means, that the latest update request of an object will be valid and propagated to the other servers overwriting previous changes.

When running Z-Merge with three or more servers, it could happen that one server is not reachable. In that case the other two will stay synced by Z-Merge while the third server is offline. When that server comes back online, the synchronisation between the other two is stopped. In the next step the so called *offline-synchronisation* is started, synchronizing all servers to the same state.

If the third server was only not reachable by Z-Merge, (e.g. a VPN tunnel broke) but the server was generally working, there could be changes on the this machine for one object which was not seen by the other two. The same object could also be changed on one (or both) of the other two systems. In that cases, the *offline-synchronisation* has to decide which object version is valid for all servers.

The general behaviour in cases of data conflicts is to duplicate the object. The duplicated object is “marked” to be a duplicate. No data is lost and all systems are in a consistent state after the duplication. The user will then see the problem and can identify the unneeded data and resolve the conflict by simply deleting/editing the required object.

If an object was deleted on one server, but edited on another, the operation with the higher score wins which is the deletion. The object is then deleted on all systems. At the same time, the changed data will not be lost, but produce an object duplication which is also propagated to all servers.

The behaviour for data conflicts is configured using the `$conflict_resolution` parameter.

- `GENERAL_PROCEDURE` – the currently supported value is `DUPLICATE`
- `DUPLICATION_INDICATION` – when set to `false`, no additional modification to the object is made. If `true` the `DUPLICATION_INDICATE_FIELD` is used to modify the object indicating the conflict.
- `DUPLICATION_INDICATE_FIELD` contains a list of fields which should be modified (if set) when duplicating objects. More than one field can be modified for each object. Possible values can be found in the `z-merge/mergeObjects/SyncDefs.php` file. Each object, that should be modified, needs at least one field definition. It is possible to define a string that should be prepended or appended to the variable. Only variables of the type string can be modified (`STREAMER_VAR`).

```
ex. SYNC_POOMTASKS_SUBJECT =>
    array('CONFLICT: ', SYNC_POOMTASKS_SUBJECT)
```

The modified variable in Tasks is the subject. The value “Conflict:” is prepended to the variable.

Installing 3rd party connectors

The connector has to be installed into the webserver directory of the machine. The base configuration depends mostly of the requirements of each connector. The system must be reachable over TCP/IP by the agents machine.

Zarafa Connector configuration

In order to allow the connector to work, the webserver running the connector service has to be fully configured to work with Zarafa. That means especially, the correct installation and configuration of the PHP-MAPI library. Please consult the *Zarafa installation manual* for further informations:

<http://www.zarafa.com/?q=en/content/documentation>

Note: As the Zarafa *incremental change system* (ICS), used by Z-Merge to track changes, was introduced only with **Zarafa 5.20**, Z-Merge will work **not work** with earlier versions.

Download the z-merge-zarafa package. It is available as tarball and debian (or compatible) package. Use the dpkg tool to install the .deb file on the Zarafa system.

```
dpkg -i z-merge-zarafa_<version>-<build>.deb
```

The connector is automatically copied to the `/var/www/z-merge-zarafa` directory. The configuration files are located at `/etc/z-merge`. For new installations copy the `zarafa-cfg.inc.php.dist` to `zarafa-cfg.inc.php` using the command:

```
cp zarafa-cfg.inc.php.dist zarafa-cfg.inc.php
```

inside the `/etc/z-merge` directory.

For non debian servers you can use the tarball. Please follow the instructions of the `INSTALL` file contained in the package.

When updating the connector to a newer version, please compare the `zarafa-cfg.inc.php.dist` to your existing `zarafa-cfg.inc.php` and update changed/new configuration parameters to your needs.

- Update the `PHP_MAPI_PATH` to point to the PHP-MAPI share directory. Normally it is located at:

```
/usr/share/php/mapi/    (observe the final / )
```

- Point the `ZARAFADA_SERVER` to the socket or the url of the Zarafa server. In clustered/high available environments that could be another machine than the `localhost`.
- Add the MySQL settings of the Zarafa MySQL server. The connector accesses the Zarafa database directly to capture changes.
- Please set the correct MySQL credentials using `ZM_DB_SERVER`, `ZM_DB_USER` and `ZM_DB_PASS`
- Set the database name of the Zarafa database. The parameter for that value is `ZARAFADA_DB_NAME`
- The connector also needs an own database. The parameter for that value is `ZM_DB_NAME` with the default value `zmerge_zarafa`. If the database does not exists, it is automatically

created when performing a system check.

Note: The actual version of the Z-Merge Zarafa connector needs an own database located on the same database instance as the Zarafa database. The additional direct access to the database is necessary only for the current version and will be removed in future.

- The parameters `ZM_SOAP_NAMESPACE` and `ZM_SOAP_SERVICENAME` are internal values for the SOAP interface and don't require any further configuration.
- The parameter `ZM_LOGFILE` specifies the default location of the connectors log file. The default value is `/var/log/z-merge/zarafa-connector.log`
- The parameter `ZM_LOGLEVEL` specifies the loglevel of the connector. Possible values are `FATAL`, `ERROR`, `WARN`, `INFO`, `DEBUG`. The default level is `INFO`. When encountering problems you should set the loglevel to `DEBUG`.
- The parameter `ZMERGE_LOGSIZE` specifies the size of the connectors logfile. The value represents the value in megabytes. Modifying this setting you can influence when a logrotate will be performed. In the `DEBUG` modus the connector could produce a lot of logs. In this case consider higher values, for example 50 MB.
- Set the parameter `ZM_WAIT_FOR_EVENTS` to the highest value possible in order to minimize network traffic. **Please consider this value, when configuring the connector (profiles) in the configuration file of the Z-Merge agent.** For testing purposes a value of 30 seconds is recommended, in production environments up to 900 seconds were tested successfully. When stopping the agent it might take a considerable amount of time, depending on the value of this configuration.
- In order to avoid excessive memory usage, the connector only sends a certain number of objects per SOAP call. You can configure the maximum number of objects setting the parameter `ZM_MAX_OBJECTS`. The default value is 50.
- In order not to overload the database, the query is limited. You can adjust the maximum number of rows to query with the value of parameter `ZM_MAX_CHANGES`. The default value is 2000.
- The `$custom_tag_array` is used for custom MAPI variables transmitted by Z-Merge. Please consult the *Z-Merge connector specification* for further informations.

Zarafa Server configuration

In order to track changes inside the Zarafa system, the tracking of changes mechanism has to be enabled.

For Zarafa 6.20 or later, enable the `sync_log_all_changes` parameter in the Zarafa `server.cfg` configuration file and restart the Zarafa server using:

```
/etc/init.d/zarafa-server restart
```

Older versions have to use the `initialize` script to perform this operation. The script enables the tracking of changes on a per user basis. That way, it is necessary to execute this script when a new Zarafa user was created and changes should be tracked for this user. It is located in the `utils` directory of the Zarafa-connector and can be executed with:

```
php -f /var/www/z-merge-zarafa/utils/initialize.php
```

If executing the script as a non privileged user, a Zarafa admin user account has to be configured inside the initialize script.

SugarCRM Connector configuration

The SugarCRM connector has to be installed by the *Sugar Module Loader*. Please access your SugarCRM installation as administrator and choose the *Module Loader* in the administration interface.

You just have to select the zip file from your local hard disk and click on install. The *Module Loader* checks automatically if your version is supported and copies all necessary files.

The connector directory is inside the SugarCRM modules directory, located at

```
./sugar/modules/ZMerge.
```

There are additional configuration parameters that could be set. The location of the configuration file is `./sugar/modules/ZMerge/inc.config.php`. Modifications are not necessary but possible.

- Set the parameter `SG_WAIT_FOR_EVENTS` to the highest value possible to minimize network traffic. **Please consider this value, when configuring the connector (profiles) in the configuration file of the Z-Merge agent.** For testing purposes a value of 30 seconds is recommended, in production environments up to 900 seconds were tested successfully. When stopping the agent it might take a considerable amount of time, depending on the value of this configuration.
- In order to avoid excessive memory usage, the connector only sends a certain number of objects per SOAP call. You can configure the maximum number of objects setting the parameter `SG_MAX_OBJECTS`. The default value is 50.
- Execute the `check` script. Point your webbrowser to

```
<http://<server-ip>/sugar/index.php?module=ZMerge&action=check
```

The connector uses an own database table inside the SugarCRM database. This table is created automatically when installing and executing the check.

Note: Unfortunately the stable version SugarCRM contains bugs which prevents that **delete events** e.g. for contacts are sent to Z-Merge. **Please consult the troubleshooting area for instructions how to fix that problem!**

Basic testing connectors

Each connector provides a script called `check`. Depending on the architecture of the connector or the application, the location can change. Normally it is located directly in the connector's directory. You can then execute the script using your web browser, pointing it to:

```
http://<ip-of-your-server>/<connector-directory>/check.php
```

The script should show you some status informations. If some test fails, please check the Troubleshooting area at the end of this document.

Securing connector access

Generally the connectors are accessible from the internet. By knowing the correct URL and some parameters it is possible to access and modify all information stored inside your system. Each connector is responsible for implementing a login mechanism.

Additionally it is possible to secure the connector directory defining Apache access rules which limit the access to specific IP addresses. This should be done after the installation is completed and working.

Apache access rules can be defined by creating a `.htaccess` file in the connector's directory, allowing connections only from the IP of the system on which the Z-Merge agent is running.

Note: After enabling the `.htaccess` file, the connector-check script can not be executed by a webbrowser any more. It is recommended to test the connector as described in the section above before enabling the access protection. The connector's check by the check tool of the agent will still be possible.

If the agent will be installed on the same system, you should only allow connections from `localhost` to access the connector. To do so, create the `.htaccess` file with the content:

```
order deny,allow
allow from 127.0.0.1
deny from all
```

If the agent will be installed on another system, change `127.0.0.1` to the correct IP address.

Note: Please test the access to the `check` script in your webbrowser as described above. The host IP from the machine running the browser may not be in the allowed range of IPs in order to make sure, that the access is **NOT** possible.

Finalizing Installation

Some final adjustments have to be done before synchronisation can start.

Preparing User-Mapping

The users have to be mapped manually for the installation. Each user needs a counterpart on the other connected systems. Each user has to be configured on each system!

The check script can map the users for your systems automatically. This is only possible for a simple user matching. This means, that usernames have to be equal on all machines. The script will query each system for its users and then insert the usernames into the mapping table.

If you add users to your systems, just restart the check script on the agent's machine to add the new users.

Checking the Agent & Connectors configuration

The complete check can be started using the start-stop script from Z-Merge executing:

```
/etc/init.d/z-merge check
```

to test the local configuration of the agent and the (remote) connectors.

The check script looks at:

1. The configuration of the agent – directories, logfiles etc.
2. checks the database

The database and the necessary tables are created if needed

3. Tests each connector and shows status

Errors and status messages are returned. For detailed descriptions see the troubleshooting section.

4. Checks the status of the agents/servers modification ID's relation (used for synchronisation)

Using the modification ID's the agent determines which change has to be synchronised to which server. When starting a synchronisation environment for e.g. a Zarafa server could present an increased count of that ID if the system has been in use before.

5. Writes modification ID's according to the user's choice

If ID's for all connected servers are present, no user interaction is necessary. The tool will show, how many changes will be applied to each system. In ideal case that is 0.

If a connector/server presents changes and has no state information saved inside the agent, the user has to decide how to proceed:

- write '0' to configuration

All changes (from zero) will be applied to the other server(s). That could be several thousands and it is in fact no guarantee that **all** existing objects will be synchronised correctly (depends on the moment when the change tracking was enabled).

- Write latest ID from the connector.

When choosing this option, no data will be transferred initially between the servers. All objects written **from** that moment (even when an old object is edited) will be synchronised to the other servers. That is the recommended option when an initial-synchronisation is not requested.

- Do nothing (option 'n')

No data will be modified. When using that option, the agent can not be started because the configuration is incomplete.

Each check could return error messages when files are missing or when the configuration is incorrect. Please check the error messages in the Troubleshooting section.

Synchronise existing data

For the most installations, there already are objects/data in the databases. That data should be synchronised before starting the agent completing the configuration.

There are different approaches to realize that. In the default case, no data is synchronised initially. It is recommended to initialize the configuration using the “write ID from the connector” as shown in check configuration related chapter. All objects created or modified from that moment will be synchronised as expected.

If you already have objects with the “same” data (for e.g. a common contact) in your systems, these objects will get “duplicated” when edited. That happens because the agent has no knowledge about their relation and will treat them as completely different objects. Editing will be seen by the agent as a **new** object and so created on the other system producing a *duplicate*. The new object will be mapped correctly. So, editing the new object on the other system will result in an update. If the old (unmapped) object is modified, this modification will be seen again as a **new** object.

A tool to analyse all existing data and find potential matches is in an early stage of concept and development and not yet available.

Note: It is **not** possible to synchronise the servers using alternate migration tools. The agent will not be aware of the relations between these objects and users. Alternate tools do not provide the necessary meta data and that will result in duplicates in a later stage as described!

Since Z-Merge-beta5 a simple script to copy existing data from one server was introduced. It resides in the `tools` folder of the agent (default location `/usr/lib/z-merge/tools`). Open your console , change to that directory and run the script:

```
cd /usr/lib/z-merge/tools
./sync.php SourceProfileName
```

where `SourceProfileName` is a server you want the data to copy from. It **MUST** be equal to one of the `SERVER_NAME` values you have set in the agent's configuration file.

Note: Be aware that this script just gets all changes from one system via its connector and sends them to other connector(s) mapping the objects. The script **DOES NOT HAS** any algorithm for duplicates' detection. Using this script **can cause the duplicate entries** for some objects in your systems.

Installing LiveLog

LiveLog is a tool to provide live access to the Z-Merge Agent logs through your webbrowser for demonstrations. You can install it using the command.

```
dpkg -i z-merge-livelog_<version>-<build>.deb
```

The package provides some scripts in the `/var/www/livelog` directory. Using the *LiveLog* script you are able to start & stop the agent and look at the logfiles. The `start` and `stop` commands are executed using `sudo`. During the installation the `sudoers` config file is modified automatically.

In order to provide log information in the moment they occur, a small daemon needs to be started. For that, just execute:

```
/etc/init.d/z-merge-livelog start
```

And then point your webbrowser to:

```
http://<ip-of-your-server>/livelog/livelog.php
```

Note: It is not recommended to install LiveLog on productive servers. The Z-Merge agent could be stopped by anyone.

Running

Starting the services

The Z-Merge agent runs as daemon on Linux. The main process starts one child process per connected server, where each is responsible to connect to one connector/server.

The Z-Merge Agent is started and stopped using a script:

```
/etc/init.d/z-merge [start|stop|restart|check]
```

Use the `start` parameter to start the Z-Merge agent.

If a start-stop-script is not yet available for your system, you can start the agent executing

```
/usr/lib/z-merge/z-merge.php
```

The agent will fork itself and run in the background.

When starting for the first time, please take a look into the logfile, to see eventual errors.

As the agent consists of a main process and one sub process for each connector, in the default two server synchronisation there should be three Z-Merge processes easily identifiable using a tool like `ps`.

```
Try: ps -e | grep z-merge
```

Reading logs

The logs present all kind of possible problem occurring during the synchronisation. Please check the `logfile` and the `errorlogfile` for errors.

The agent can be configured on several error levels (see configuration for details).

In `DEBUG` mode, the logfile should present entries when a connector times-out or a change event was thrown. The log then shows for e.g. that `X ServerObjects/DataObject` were found, pass the object to the transformation layer, sends it to the other servers and then reconnects immediately again to the originating server to search/wait for changes. For no changes, that process should present itself like this:

```
DEBUG SugarCRM(SugServ) - SP(SugServ) returned 0 changed ServerObjects
```

```
DEBUG SugarCRM(SugServ) - SP(SugServ) returned 0 changed DataObjects
DEBUG MergeAgent - SP(SugServ) -> found 0 changed objects
DEBUG MergeAgent - Process-SP(SugServ) lowest modID:1614
DEBUG SugarCRM(SugServ) - enter getChanges(1614) - for SP(SugServ)
DEBUG SugarCRM(SugServ) - soap call getChanges()
```

After the soap call nothing should happen for the determined interval configured in the connector configuration.

Logentries and their meaning

```
WARN Mapping - getMaFrom3rd-getMaUserIds: NO USERIDs found for SP(ZaServ)
+'john'
```

An object was received, but the user `john` from the originating server could not be matched in the `UserMapping`. The originating server profile is the one named `ZaServ`.

Note: In these cases the changed object is ignored by Z-Merge. The agent supposes that objects from `john` are not of interest for the other servers.

```
ERROR SugarCRM(SugServ) - sendChanges: SP-> processReturnObjects() could
not save transmitted informations: detailed-error-message
```

An object was sent to the `SugServ` server, but could not be processed there. Generally that could happen if incomplete objects are sent. A detailed error message will appear.

```
ERROR SugarCRM(SugServ) - object processing on target server was NOT
SUCCESSFUL! Targetserver SP(SugServ) claimed 'create_fail:no userid set!'
```

The `SugServ` could not create the object, because the object was not assigned to any user. This could happen if the `UserMapping` is incorrect or incomplete.

The following messages are informations about changed objects found performing the `offline-synchronisation`.

```
INFO MergeAgent - runSynchroniseAgent - REAL CONFLICT: object will be
duplicated: MAid : 2046 modid: 1897 from SP(SugServ) conflicts with
SP(ZaServ) modid: 6899
```

A **real conflict** between the systems was found. While one or more systems were not connected through Z-Merge, both systems performed a change on the same object. The internal object ID of Z-Merge is 2046. That object was changed on `SugServ` (modification-id: 1897) and on `ZaServ` (modification-id: 6899). While one object will be maintained, for e.g. the one on `ZaServ` and overwritten on `SugServ`, the data from `SugServ` will be used to create a new object on `SugServ` that afterwards be synchronised to `ZaServ`. The newly created object will be previously modified accordingly to the in `$conflict_resolution` configured behaviour.

In order to get detailed information why the object was duplicated, change to the `DEBUG` loglevel. The conflicting field and its contents are logged in `DEBUG` mode.

```
INFO runSynchroniseAgent - switching objects: _toDelete object goes to
output_queue, changed data will be duplicated: MAid : 1987
```

As in the case before, a conflict was found for a object which was modified on both systems while the servers were not in sync. On one system the object was deleted while on the other system it was modified. The deleted object will be deleted on all systems, while the data from the changed object is used to create a duplicate as in the previous case.

ERROR Database - Database error: connect: [...] MySQL server has gone away
The MySQL connection was interrupted. The agent tries to reconnect several times before giving up. Generally the connection comes back and then the operation is performed. If not, the agent goes in `active wait` modus, retesting the connection every several minutes. If the connection comes back, the *offline-synchronisation* is executed.

Monitoring

When something unexpected happens, a child process (for a specific server) is stopped. The agent tries to contact the server after the configured period of time (`ZMERGE_RECONNECT`). Depending of the occurred problem, it's possible that the server doesn't come back. In any case the administrator will be informed by email and could then fix the problem.

Troubleshooting

Z-Merge Agent error messages on checking configuration

- 1) Configuration error! System claimed: Parse error: syntax error, unexpected T_STRING in /etc/agent-cfg.inc.php on line XX
The configuration file is broken. The line number could indicate where to look for the problem. Most of the time the error is some line located before the presented line, as the PHP processor just noticed the problem in the indicated line. Common errors are missing or duplicated string quotes (" and ') or missing semicolons.
Try to find and fix the problem. It is also possible to copy the `dist` file again and start the configuration from scratch.
- 2) Main directory: /usr/lib/z-merge doesn't exist or agent not found
The main directory and/or the agent could not be found. Please review your agents' configuration file.
- 3) Logfile: "/var/log/z-merge/z-merge.log" doesn't exist and directory isn't writable
When no logfile was created, the agent tries to do this with the local permissions. Make sure that the log directory is writeable.
- 4) `emailaddress` invalid or domain not reachable
In the configuration you could set an email address where errors are sent to. The email address is checked for syntax and if the local machine could resolve the DNS-MX record for the given domain. If you do not wish to receive errors by email, set the value to `false`.
- 5) unsuccessfully creating the database
When the database does not exist, the check script tries to create the database and the necessary tables. If the user has not enough permissions to perform such action, that error will be shown. Grant administrative rights to the user or create the configured database manually and restart the agent.
- 6) Database structure compromised
Some tables are missing from the configuration. This should never happen. Please look for the indicated table in your backup files. If you reinitialize the database, the object mapping between the systems will be lost. That way, every future change will produce duplicate entries in the other system(s). You should always backup the agents database before you try to fix a problem. Please contact you local support for further assistance.

General connector related error messages

- 1) Could not connect to the server "ZaServ" - Error: Response not of type text/xml

An error occurred while talking to the remote connector. Generally the exact error is described in detail.
- 2) Server "ZaServ" said: syntax error, unexpected T_STRING in /etc/z-merge/zarafa-cfg.inc.php line XX

The configuration file is not valid. You could find hints how to fix in the *general messages*, topic 1).
- 3) Authentication error: profile 'ZaServ': Authentication failed (401 Unauthorized)

The configured username and password in /etc/z-merge/agent-cfg.php are not valid or not of an admin user.
- 4) Server "ZaServ" said: require() [function.require]: Failed opening required 'filename.php'

A file could not be found in the connector's installation. Generally that indicates that the installation was not successful. Recheck the installation and add the connector's directory path to the PHP `include_path` variable. Please consult the PHP documentation for further informations.
- 5) Server "ZaServ" said: Cannot modify header information-headers already sent ..

This error means that serious problems were found while executing the connector. Please copy all available information and error messages (files, line numbers) and contact the developers. Please include all version numbers of installed applications (Application (Zarafa/SugarCRM etc.), Z-Merge & PHP).
- 6) The modificationID for 'Sugar' from 'Zarafa' could not be found!!

This is normal on new installations or when adding a new server. Please set the modification ID's to the last values of the other connector (Zarafa in that case). New/modified objects will then be synchronised to the new system.
- 7) The connector from 'ZaServ' has not reported an id. The connector is not installed correctly.

The installation of the connector was not completed. It may be that the auto-configuration of the connector could not be completed. Please check any additional informations regarding the connector and contact your local support for further assistance.

Specific connector related error messages – Zarafa Connector

- 1) Could not connect to the database.

The connector needs an additional database to save metadata in the current version of Z-Merge. Check if the configured database is up and running on the configured host.
- 2) The database '*dbname*' was not found and could not be created automatically. Please create it manually.

If the database is not found the connector tries to create it automatically. The error will be shown, if the configured database user has not enough permissions to create a new database, table or to insert into a table. Please grant additional permissions to the user or create the database manually. Restart the check after that.

It is also possible that there are other MySQL related errors, like a full disk. Please check your system.
- 3) The database '*dbname*' could not be selected.

The configured database was not found by the connector or is not readable. That might happen, if the configured user has not enough permissions to read from/write to the configured database.

- 4) Some tables are missing and could not be created automatically. Please create them manually.

The error will be shown, if the configured database probably user has not enough permissions to create a new table. Please grant additional permissions to the user or create the table manually. Restart the check after that.

- 5) Could not [SELECT|INSERT|UPDATE] (into) tables. Please check the permissions.

This error will be shown, if the configured database user has not enough permissions to perform the shown operation in the configured database. Please grant additional permissions to the user. The connector has to select, insert and update data continuously. Restart the check after that.

- 6) Unfortunately your Zarafa version is too old to use with Z-Merge.

As the Zarafa *incremental change system* (ICS) used by Z-Merge to track changes, was introduced only with **Zarafa 5.20**, Z-Merge will work **not work** with earlier versions.

- 7) User "username" is not a Zarafa admin user

The configured Zarafa user is not an administrative user. As Z-Merge has to open all user stores to read and write into them, an administrative user is a requirement.

- 8) There was an error retrieving user information. Please check your system.

An internal error occurred when logging into the Zarafa server with the configured user credentials. The store of the user is probably not available or broken. Please check the `Zarafa server.log` on the Zarafa server, try to log in using the Webaccess and/or contact your local Zarafa support to resolve this problem.

- 9) User "username" could not login to Zarafa

It was not possible to log in using the configured credentials. Please recheck the configured data and if your authentication layer is up and running (for e.g. the LDAP server). Please also check the status of the user with the `zarafa-admin` tool or try another administrative user.

Try: `zarafa-admin --details username`

- 10) Could not retrieve any change from Zarafa.

Probably you are using an empty database. Please log into the Webaccess and try to create a new object and execute a connector check again.

Specific connector related error messages – SugarCRM Connector

- 1) load Sugar entryPoint: failed

It was not possible to load the main SugarCRM library. Please recheck your installation, paths and file permissions.

- 2) load configuration: failed

It was not possible to load the `sugar/z-merge/inc.config.php` file by the webserver. Please recheck the file permissions for the connector. All files have to be readable at least for the webserver user (on debian `www-data`).

- 3) loading libraries: failed

It was not possible to read/load the connector libraries or some of the declared functions were not found. Recheck your installation and the file permissions.

- 4) **Check SugarCRM version:** This version of the Z-Merge SugarCRM connector was not designed and tested for SugarCRM *sugarversion*
The connector was tested only with certain versions of SugarCRM. Generally all versions of the same base version should be compatible. You can test the connector at your own risk setting the `SG_IGNORE_VERSION` parameter to true in the `sugar/modules/ZMerge/inc.config.php` file. Please contact the developers with the results of your tests, so that incorrect behaviour could be fixed and released after that.
- 5) **check database: failed**
The connector was not able to create the required table inside the SugarCRM database. Please check if the configured user has sufficient permissions, uninstall/reinstall the connector and retest the connector.
- 6) **instantiating sugar-admin-user: false**
No SugarCRM administrator was found or could not be instantiated. Please create an administrator account or contact your local SugarCRM support for assistance.
- 7) **perform ping: failed**
Ping is used to check if the server and the database connection are working. The ping can only fail if there is an error in your SugarCRM database connection. Please log in into SugarCRM and check your database parameters.
- 8) **select a change:** Please create/modify a 'Note', 'Contact', 'Appointment' or a 'Task' and re-check.
None already tracked change was found on the SugarCRM system. Please log into SugarCRM and create an object for a mapped user. After that, the check script tries to read and write that object using the connector and the check should succeed.
- 9) **get change using Z-Merge API: failed**
The connector was not able to retrieve the object using the connector's library. That is not a normal case. Please check your SugarCRM log files and contact your local support for further assistance.
- 10) **writing object using Z-Merge API: failed**
The connector was not able to write the before read object using the connector's library. In some cases (e.g. executing the check for several times) that operation could fail. You should try to start the agent anyway. If the problem persists, please check your SugarCRM log files and contact your local support for assistance.
- 11) **Z-Merge loop detection: failed**
The loop-detection prevents that objects changed by Z-Merge itself are replied back as original SugarCRM changes. Please check your configuration to see if special global variable rules apply in your PHP configuration. The loop-detection is done setting a global variable when instantiating the connector. It is normal for the loop-detection to fail if an previous check fails. Please contact your local support for further assistance.

SugarCRM Connector: no delete request sent for some objects

Unfortunately SugarCRM contains bugs causing this kind of scenario.

The connector uses the SugarBean to read and write data. For some objects, the *Contact* object for e.g., the overwritten *retrieve()* method is wrong in the subclasses.

This bug was already reported to the SugarCRM developers and is hopefully fixed in one of the next releases.

In the meantime you have to fix the bug yourself.

Please open the file `<sugar>/include/SugarObjects/templates/person/Person.php`, and locate the method

```
function retrieve($id = -1, $encode=true) {  
    $ret_val = parent::retrieve($id, $encode);
```

Replace these two lines by:

```
function retrieve($id = -1, $encode=true, $deleted = true) {  
    $ret_val = parent::retrieve($id, $encode, $deleted);
```

Note: Please be aware that these changes are *not update-safe*. This means that these changes are overwritten when you update your SugarCRM installation the next time.

Feedback & Contact

If you have trouble not described in this document or have feedback and suggestions, please contact the Z-Merge developers:

Sebastian Kummer <s.kummer@zarafa.com> and Manfred Kutas <m.kutas@zarafa.com>