Q-Messenger

Process Specifications Document

~~***Company Confidential***~~

Produced for
Quios.com

Prepared by
Matt Ericson, ~~Alexander~~ Bryan Scholz

Version1.0
8/1/~~00~~2XXX

~~For internal use only~~

Table of Contents:

Release History

Version	**Author**	**Date**
1.0	M. Ericson, B~~A~~. Scholz	8/1/~~00~~2XXX

Location and Dependencies

Document Location	http://intranet.quios.net/Docs/docs.shtml
Code Location	
Dependencies	

1. Overview

Q-Messenger has been redesigned. It is now broken into three modules + .mtpl

Separate functionality from Data as much as possible is a good programming practise. We have been able to modify Q-Messenger so that it is database driven, has increased modulartity. It Bit wise with a very fast rate of data exchange. With simple SQL formatting or through a soon to be designed UI, pages and flows can be easily swapped or changed to account for individual partner and brand prerequisites or unique features.

2. qmsgr.pm

Qmsgr.pm verifies validity of message, if there are problems with the message, and if the message is ok to send, it sends the message.

3. qmsgr_data.pm

Qmsgr_data.pm effectively holds data. It is basically an object. Everything to send message is stored in this perl module.

4. qmsgr_flow.pm

Qmsgr_flow.pm is the most interesting in that it is the data base driven module. It decides what page you see next based on qmsgr.pm responses, hence the flow.

It also reviews and cleans errors, i.e. such as lack of a data flow with fun ideas is an error, but with a regular data flow it isn’t. Errors come from qmsgr_data.pm. and are directed to qmsgr_flow.pm for error cleaning. It checks these against a series of database fields for validity in two stages. The first reviews certain preset parameters such as carrier id and phone number (full_address), the list of data in the data object reviewed is as follows:

```
%_fields = (
  qparms               => { },
  carrier_id           => undef,
  df_service_type          => undef,
  df_service_item          => undef,
  full_address           => undef,
  is_us_number            => undef,
  total_errors           => undef,
  fun_idea_switch          => undef,
  number_list            => [ ],
  successful_send          => [ ],
  recipient_array          => [ ],
  msg                  => undef,
  footer               => undef,
  message                => undef,
  errors                  => { },
  checked                   => undef,
);
```

It makes these checks in combination with action, that is to say, where the user is or what he has clicked. Based on this it is then accordingly able to review and clean errors.

Bit wise fast. Converts to bit wise map. Also keyed off of action so only has relatively few errors to sort through.

5. qmsgr.mtpl

Qmsgr.mtpl deals mostly with CGI Data and cleans numbers so that they all read in the same way.

The .mtpl module is a bit like the ringmaster. It transmits the action and errors to the qmesgr_flow, and sends back the information of whether a message can be sent, the destination page and clean errors to the qmsgr.pm that will actually send a message. It is a very much a data hub in the message sending process.

6. Database interaction

By having the information database driven instead of perl driven we are in a much better position to go into the database and make fundamental changes to the process flow. With a simple UI it is possible to change what pages appear in response to what errors. Parameters are separate and modular, it allows an ease of maintenance that is very important to our partners and branding strategy.

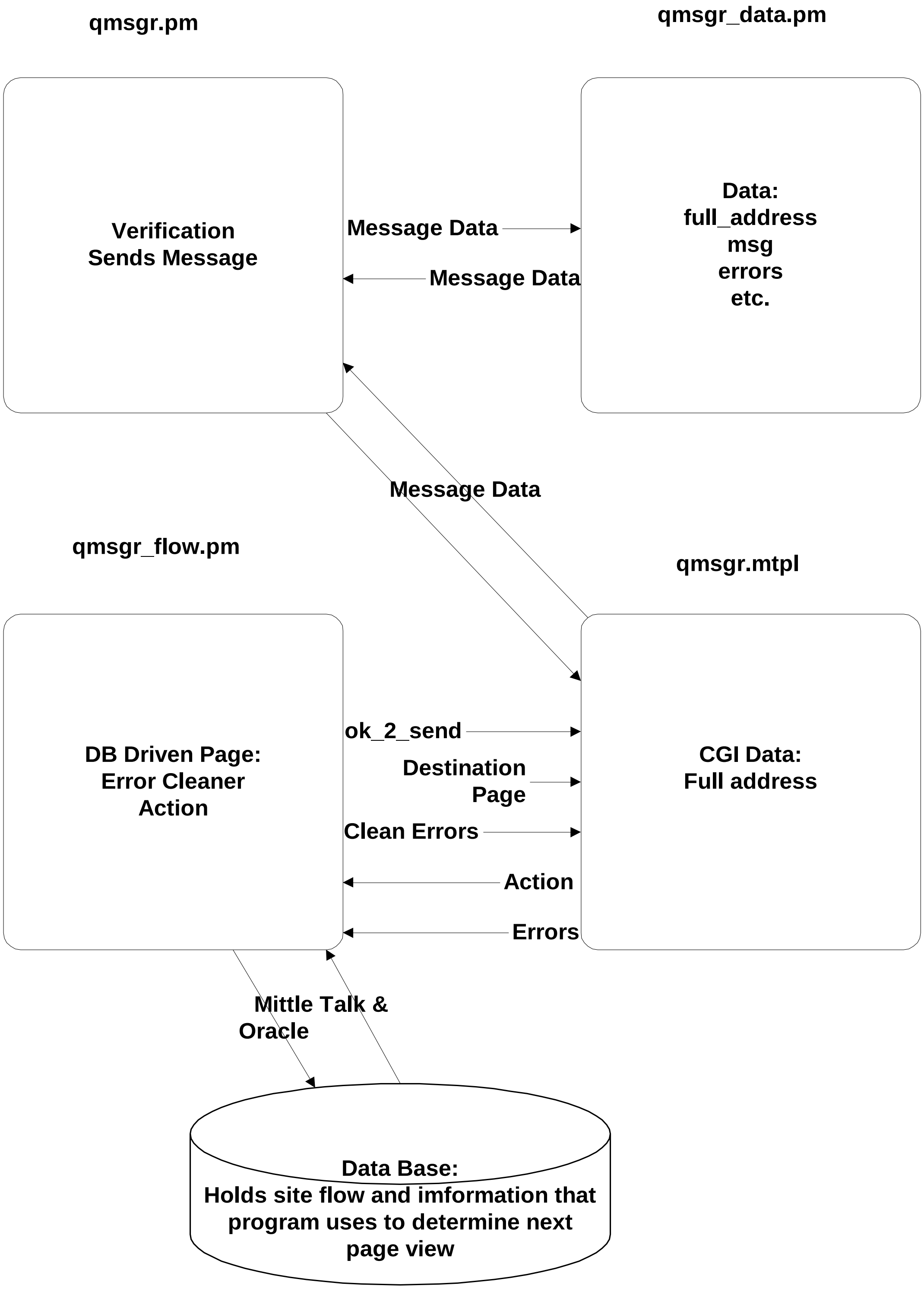
Database interacts through qmsgr_flow.pm. It checks as mentioned a series of parameters for validity as mentioned above. This is done by reviewing the object against a table that has the different possibilities for the action, sequence descritiption, page name, brand etc., as seen in the following data base view:

```
N confirm                  1 err_missing_message || err_mis
entry.tpl                 default    N Y Y N N N N N N N N N N N N
19-JUL-00 19-JUL-00 N
                              sing_number

N confirm                  2 err_checknumber_failed
bad_prefix.tpl                default    N N N N N N N N Y N N N N N N
19-JUL-00 19-JUL-00 N
N confirm                  3 err_rnl_failed
```

```
bad_rnl.tpl                   default    N N N N N N N Y N N N N N N N
19-JUL-00 19-JUL-00 N
N confirm                    4 err_number_wrong_length
bad_us.tpl                    default    N N N N N N Y N N N N N N N N
19-JUL-00 19-JUL-00 N
Y confirm                    5 err_detected_spam
spam.tpl                     default    N N N N N N N N N N N Y N N N
19-JUL-00 19-JUL-00 N
Y confirm                    6 err_message_limit
msg_limit.tpl                  default    N N N N N N N N N N N N Y N N
19-JUL-00 19-JUL-00 N
Y confirm                    7 err_blocked_address
msg_blocked.tpl                   default    N N N N N N N N N N Y N N N N
19-JUL-00 19-JUL-00 N
N confirm                    8 no_error
msg_preview.tpl                   default    Y N N N N N N N N N N N N N N
19-JUL-00 19-JUL-00 N
```

Q-Messenger Overview


qmsgr.pm
qmsgr_data.pm
Verification
Sends Message
Message Data
Message Data
Data:
full_address
msg
errors
etc.
Message Data
qmsgr_flow.pm
qmsgr.mtpl
DB Driven Page:
Error Cleaner
Action
ok_2_send
Destination
Page
Clean Errors
Action
Errors
CGI Data:
Full address
Mittle Talk &
Oracle
Data Base:
Holds site flow and imformation that
program uses to determine next
page view