OpenERP Custom Integration

Contents

Overview

This document outlines the steps required to integrate the OpenERP with Swivel using dual or single channel authentication.

The Swivel install requires configuring an agent on the Swivel server and setting up a shared secret with the code being added to OpenERP 7.0 server to allow communication for authentication.

Due to the complexity of OpenERP, several files need to be changed (or replaced) on the server, then recompiled. Integration with Swivel is made through Agent-XML.

NOTE: This document uses the files from version 7.0 of OpenERP, however, the principle is that either customized, or from another version, the functions should be similar, making integration easyer.

For an article specific for version OpenERP 7.0, please check article Open_ERP7_Integration

Prerequisites

OpenERP server (version 7.0 assumed in this example)

PINsafe server

Text Editor or Code editor

It is recommended to read the following Kowledgebase articles (to better understand how Swivel Agent-XM works):

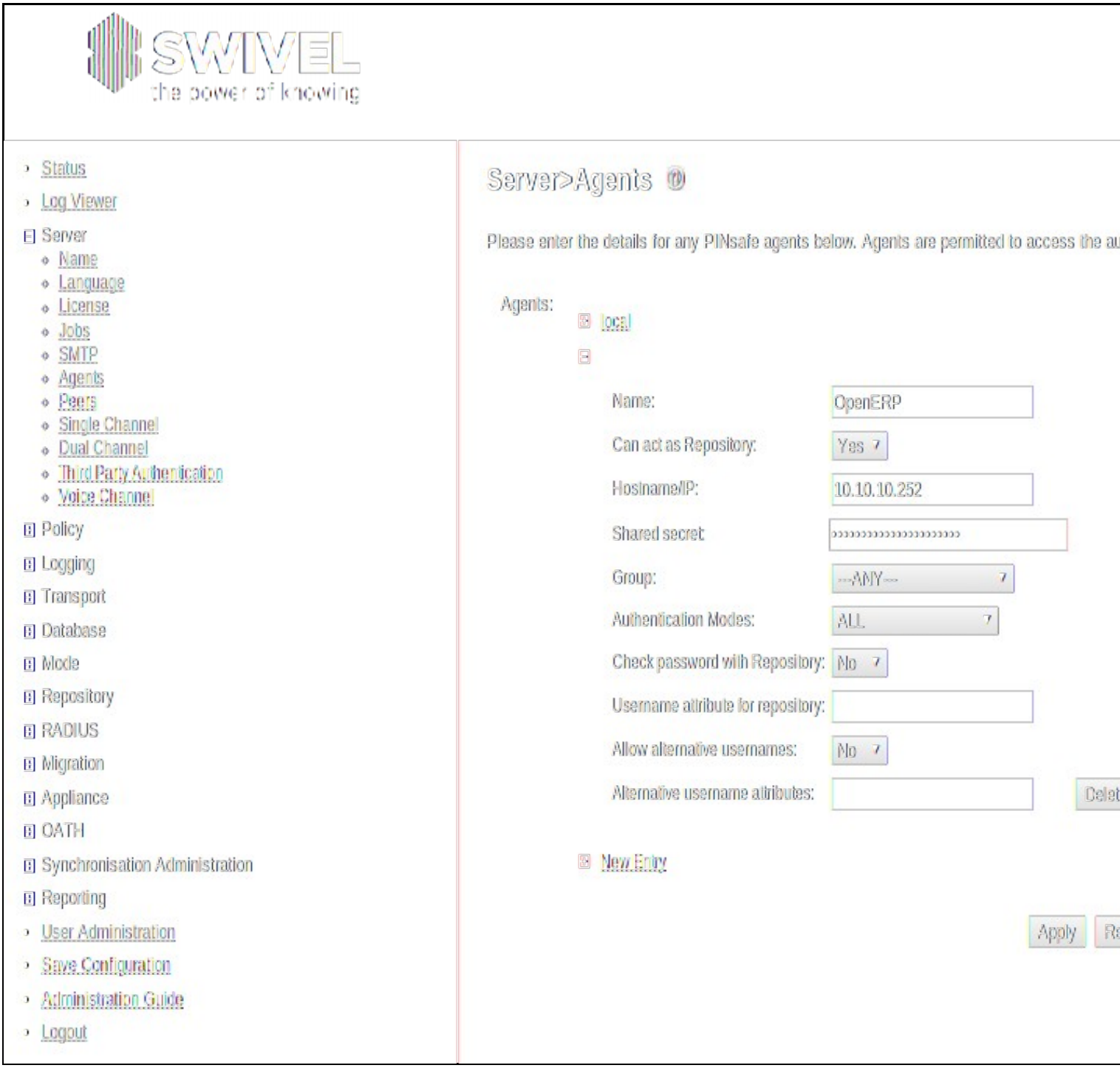
https://kb.swivelsecure.com/wiki/index.php/Agent-XML

https://kb.swivelsecure.com/wiki/index.php/AuthenticationAPI]

Swivel Configuration

On the Swivel server configure the agent that is permitted to request authentication. On the Swivel Administration Console select from the server menu Agents and enter the details of the OpenERP IP address and a shared key, then click on apply. Example:

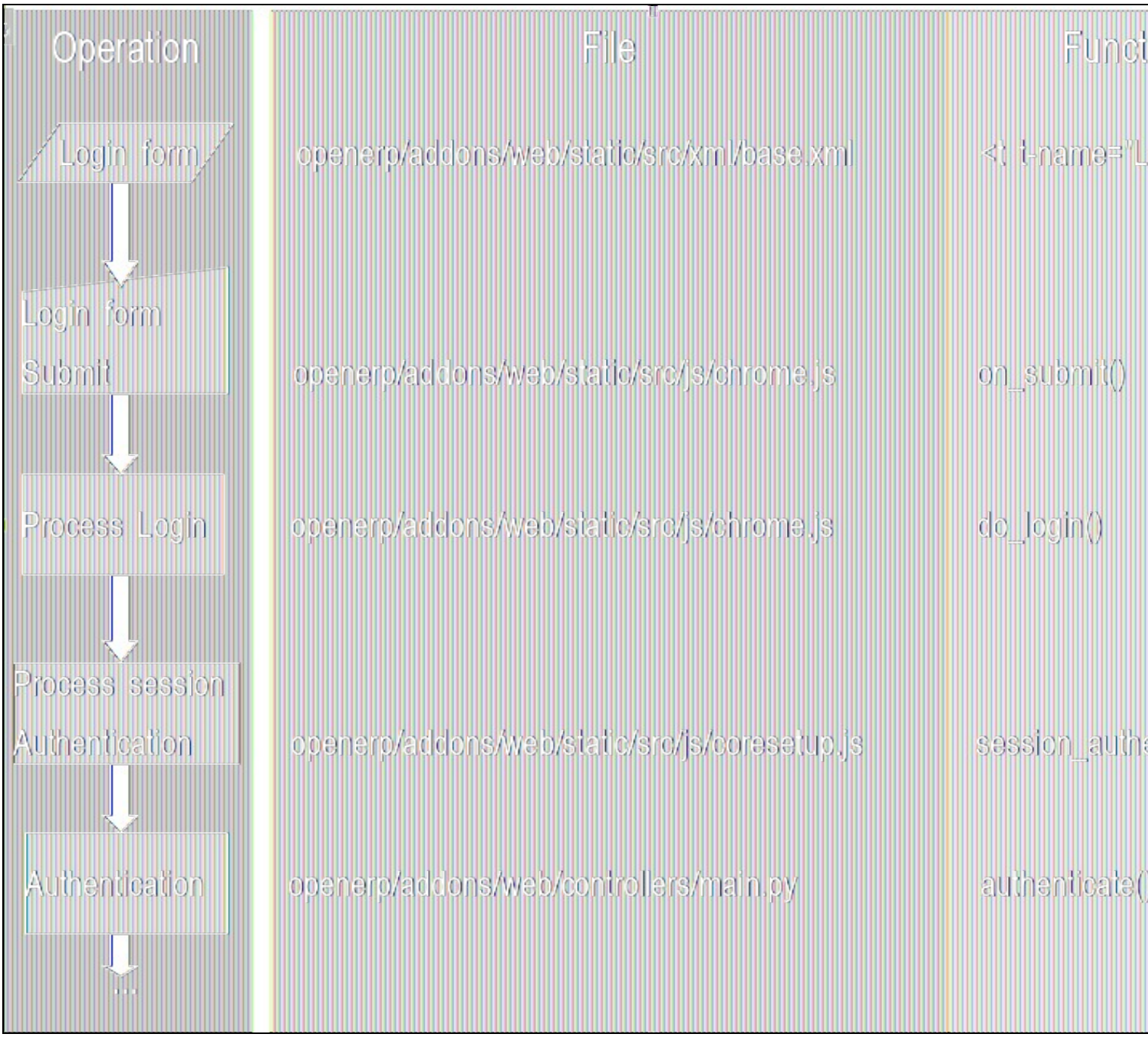
```
Name : OpenERP,
Hostname/IP : 10.10.10.252,
shared secret : secret
```



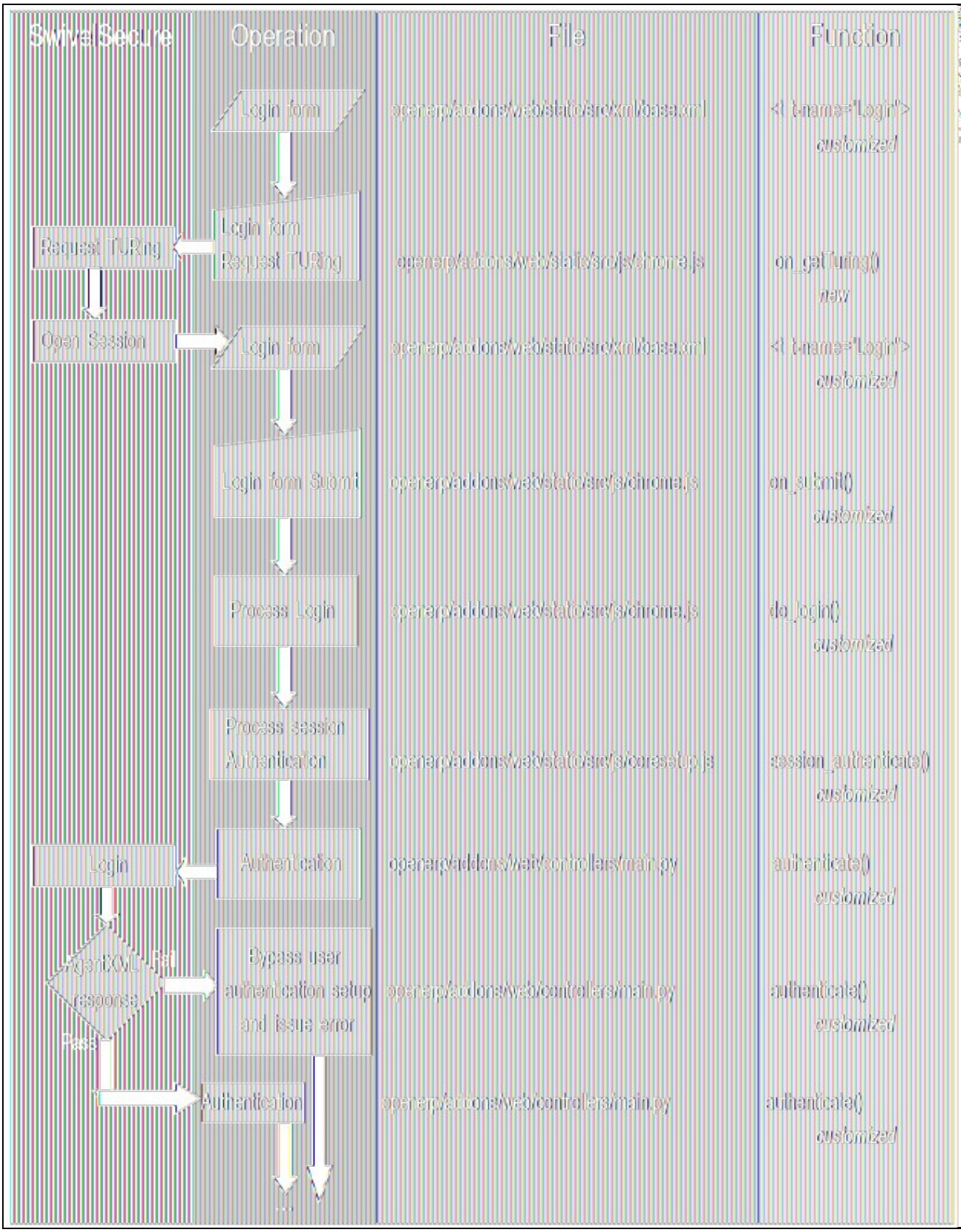
If Single Channel communication is to be used, select from the Swivel Administration Console Single Channel, and set the Allow image request by username to Yes then click on apply.

OpenERP User Authentication flow Integration

The Standard OpenERP User Authentication flow



The Swivel Integrated OpenERP User Authentication flow



Configuring the OpenERP Files

openerp/addons/web/static/src/xml/base.xml

Unchanged code in black

Added code in green

Changed code in orange

In this file we are transforming the form to support the Get TURing button and the OTC password field.

Search for the login form code, usually called ?<t t-name="Login">? and place in line nr 61

Code

```
<t t-name="Login">
    <div class="oe_login">
        <div class="oe_login_bottom"> </div>
        <div class="oe_login_error_message"/>
        <div class="oe_login_pane">
            <div class="oe_login_logo"><img src="/web/static/src/img/logo2.png"/></div>
            <form action="" method="post">
                <div class="oe_login_dbpane" >
                    Database:
                    <input name="db" t-att-value="widget.selected_db || ''"/>
                </div>
                <ul>
                    <li>Username</li>
                    <li><input name="login" type="text" value="" autofocus="autofocus"/></li>
                    <li>Password</li>
                    <li><input name="password" type="password" value=""/></li>
                    <li><button name="getTURing">Request TURing</button></li>
                    <div class="TURing"><img name="TURingImg" class="myTURinging" /></div>
                    <li>OTC</li>
                    <li><input name="password2" type="password" value=""/></li>
                    <li><button name="submit">Log in</button></li>
                </ul>
            </form>
            <div class="oe_login_footer">
                <a href="#" class="oe_login_manage_db">Manage Databases</a> |
                <a href="http://www.openerp.com" target="_blank">Powered by <span>OpenERP</span></a>
            </div>
        </div>
    </div>
</t>
```

openerp/addons/web/static/src/js/chrome.js

Unchanged code in black

Added code in green

Changed code in orange

In this file we will be adding the functions that support automation and control over the form displaying the TURing image and Get TURing button.

First we need to hide the TURing image and register the link from the Get TURing button to the respective function.

Search for the form initiation routine, usually called ? start: function() {? and place in line nr 640

Code

```
start: function() {
    var self = this;
    self.$el.find("form").submit(self.on_submit);
    this.$('div.TURing').hide();
    self.$el.find("form button[name=getTURing]").click(self.on_getTuring);
    self.$el.find('.oe_login_manage_db').click(function() {
        self.do_action("database_manager");
    });
    self.on("change:database_selector", this, function() {
        this.database_selected(this.get("database_selector"));
    });
    var d = $.when();
    if ($.param.fragment().token) {
        self.params.token = $.param.fragment().token;
    }
    // used by dbmanager.do_create via internal client action
    if (self.params.db && self.params.login && self.params.password) {
        d = self.do_login(self.params.db, self.params.login, self.params.password);
    } else {
        d = self.rpc("/web/database/get_list", {})
            .done(self.on_db_loaded)
            .fail(self.on_db_failed)
            .always(function() {
                if (self.selected_db && self.has_local_storage && self.remember_credentials) {
                    self.$("[name=login]").val(localStorage.getItem(self.selected_db + '|last_login') || '');
                }
            });
    }
    return d;
},
```

Then we need to create the function. Find space right after the ?on_db_failed: function? on lines 703 to 707.

Change the IP address on the example with your swivel appliance IP or domain name.

Code

```
on_db_failed: function (error, event) {
    if (error.data.fault_code === 'AccessDenied') {
        event.preventDefault();
    }
},
on_getTuring: function() {
    var login = this.$('form input[name=login]').val();
    var SwivelURL = 'https://10.10.10.201:8443/proxy/SCImage?username=' + login + '&random=' + Math.floor(Math.random() * 10000);
    this.$('.myTURinging').attr('src',SwivelURL);
    this.$('div.TURing').show();
    return false;
},
```

Finally, we need to change the functions that support the original form submit, so we can add OTC (called "password2" in this example).

Function ?on_submit: function(ev)? should be immediately after your recently added function code, on line nr 715.

You will also update function ?do_login?, placed immediately after ?on_submit? function.

Code

```
on_submit: function(ev) {
    if(ev) {
        ev.preventDefault();
    }
    var db = this.$("form [name=db]").val();
    if (!db) {
        this.do_warn(_t("Login"), _t("No database selected !"));
        return false;
    }
    var login = this.$('form input[name=login]').val();
    var password = this.$('form input[name=password]').val();

    var password2 = this.$('form input[name=password2]').val();
    this.do_login(db, login, password, password2);
},
/**
 * Performs actual login operation, and UI-related stuff
 *
 * @param {String} db database to log in
 * @param {String} login user login
 * @param {String} password user password
 */

do_login: function (db, login, password, password2) {
    var self = this;
    self.hide_error();
    self.$(".oe_login_pane").fadeOut("slow");

    return this.session.session_authenticate(db, login, password, password2).then(function() {
    self.remember_last_used_database(db);
        if (self.has_local_storage && self.remember_credentials) {
            localStorage.setItem(db + '|last_login', login);
        }
        self.trigger('login_successful');
    }, function () {
        self.$(".oe_login_pane").fadeIn("fast", function() {
            self.show_error(_t("Invalid username or password"));
        });
    });
},
```

openerp/addons/web/static/src/js/coresetup.js

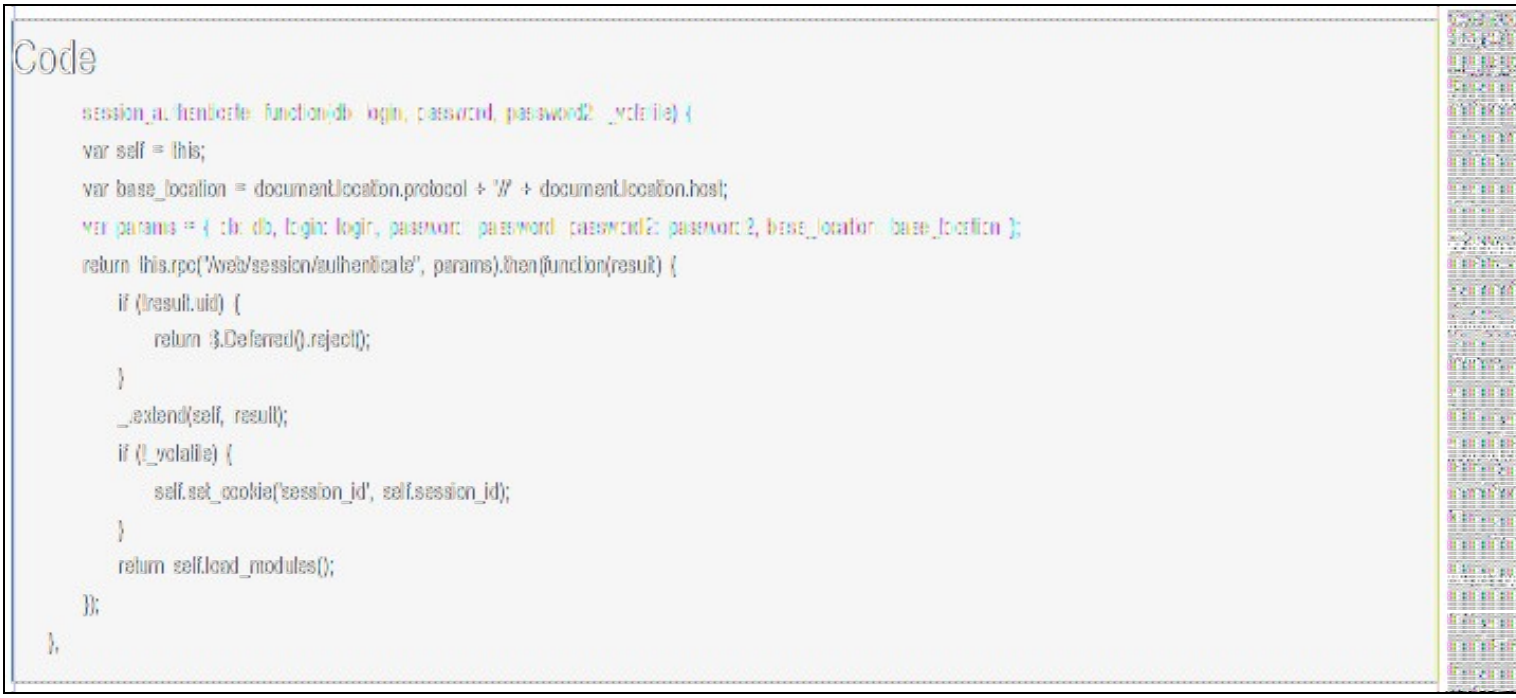
Unchanged code in black

Added code in green

Changed code in orange

In this file we will be adding the parameter ?password2? to the function that creates the bridge between the user interface (in XML, javascript and jquery) and the OpenERP server core (in python).

Search for the ?session_authenticate: function?, usualy place on line nr101



openerp/addons/web/controllers/main.py

Unchanged code in black

Added code in green

Changed code in orange

In this file we will be changing the OpenERP core authentication function, so that it calls the Swivel Secure server with a login function request, and then continues on the consideration of the reply given by swivel software.

First we need to add a library module to handle and parse the XML sent by the Swivel Agent-XML.

This is added in the very beginning of the code file.

Code

```
# -*- coding: utf-8 -*-

import ast
import base64
import csv
import glob
import itertools
import logging
import operator
import datetime
import hashlib
import os
import re
import simplejson
import time
import urllib
import urllib2
import urlparse
import xmlrpclib
import zlib
from xml.etree import ElementTree
from cStringIO import StringIO

from xml.dom.minidom import parseString

import babel.messages.pofile
import werkzeug.utils
import werkzeug.wrappers
try:
    import xlwt
except ImportError:
    xlwt = None

import openerp
import openerp.modules.registry
from openerp.tools.translate import _
from openerp.tools import config

from .. import http
openerpweb = http
```

Finally, we need to change the ?def authenticate? function to handle the the new ?password2? parameter, call the Swivel Server, parse the response and make decision based on the response.

The ?def authenticate? function should start in line nr 857 or 858

Don't forget to change the server IP in the code by your own Swivel server IP or domain path and the "secret".

Code

```
@openerpweb.jsonrequest
def authenticate(self, req, db, login, password, password2, base_location=None):

    url = "https://10.10.10.201:8080/pinsafe/AgentXML"
    params = '<%3Fxml+version%3D"1.0"+encoding%3D"UTF-8"%3F><SASRequest><Version>3.6<%2FVersion><Secret>secret<%2FSecret>
<Action>login<%2FAction><Username>' + login + '<%2FUsername><Password><%2FPassword><OTC>' + password2 + '<%2FOTC><%2FSASRequest>'
    data = urllib.urlopen(url + "?" + params).read()
    dom =  parseString(data)
    xmlTag = dom.getElementsByTagName('Result')[0].toxml()
    xmlData = xmlTag.replace('<Result>','').replace('</Result>','')
    if xmlData == 'PASS':
        wsgienv = req.httprequest.environ
        env = dict(
            base_location=base_location,
            HTTP_HOST=wsgienv['HTTP_HOST'],
            REMOTE_ADDR=wsgienv['REMOTE_ADDR'],
        )
        req.session.authenticate(db, login, password, env)
        return self.session_info(req)
    if xmlData == 'FAIL':
        return {'error': _('Error, Invalid credentials !'), 'title': _('No access')}
```

Pyhton 3.x changes

openerp/addons/web/controllers/main.py

Unchanged code in black

Added code in green

Changed code in orange

If running Python 3.0, then you have a call to the urllib module.

The urllib <http://docs.python.org/2/library/urllib.html#module-urllib> module has been split into parts and renamed in Python 3 to urllib.request, urllib.parse, and urllib.error.

This is shown in RED in the example from the changes to be done on file ?openerp/addons/web/controllers/main.py?

Do not forget to change the IP an "secret" in the code by your own Swivel server IP or domain path.

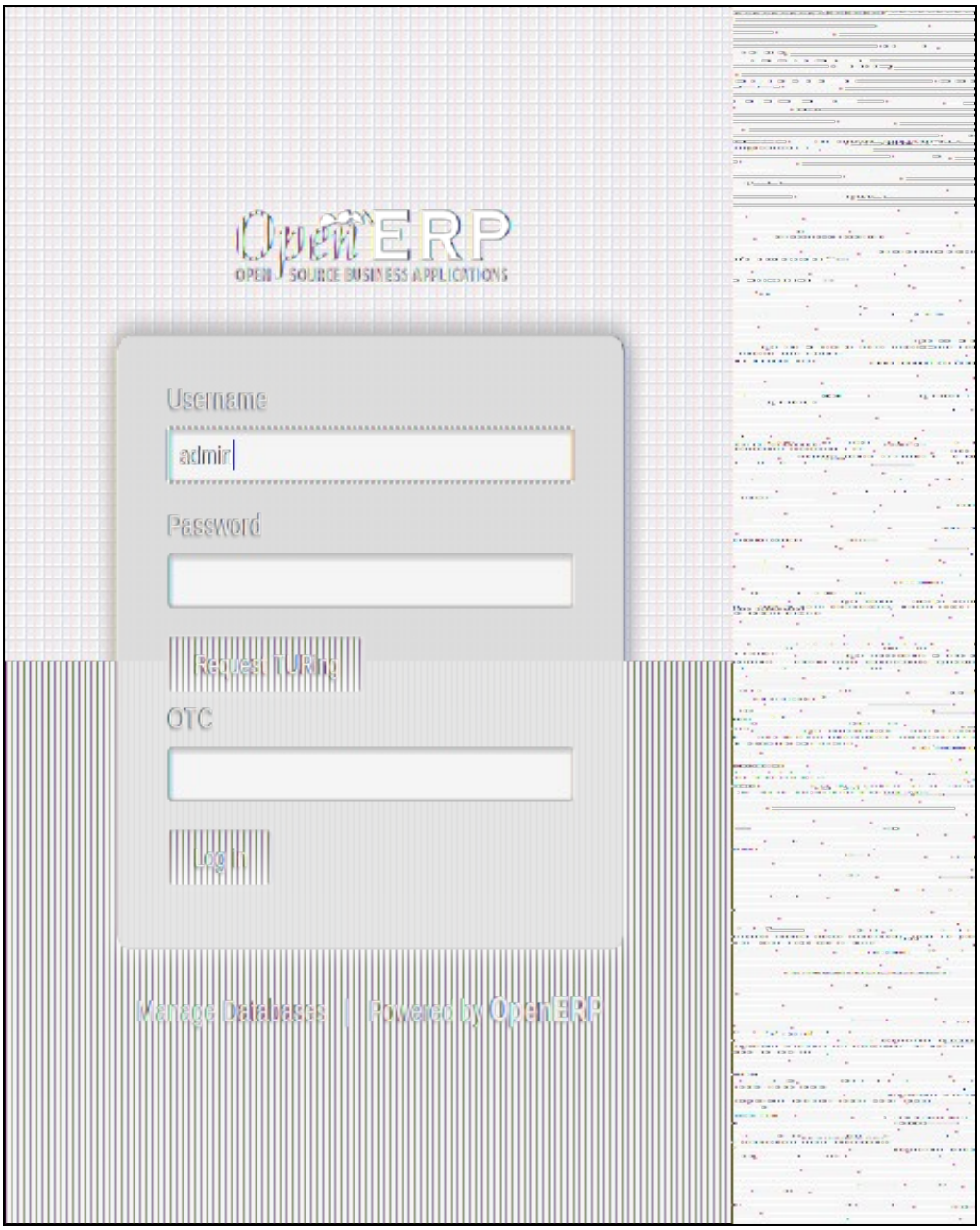
Code

```
    @openerpweb.jsonrequest
    def authenticate(self, req, db, login, password, password2, base_location=None):

        url = 'https://10.10.10.201:8080/pinsafe/AgentXML'
        params = '<%3Fxml+version%3D"1.0"+encoding%3D"UTF-8"%3F><SASRequest><Version>3.6<%2FVersion><Secret>secret<%2FSecret>
<Action>login<%2FAction><Username>' + login + '<%2FUsername><Password><%2FPassword><OTC>' + password2 + '<%2FOTC><%2FSASRequest>'
        data = urllib.request.urlopen(url + '?' + params).read()
        dom =  parseString(data)
        xmlTag = dom.getElementsByTagName('Result')[0].toxml()
        xmlData = xmlTag.replace('<Result>','').replace('</Result>','')
        if xmlData == 'PASS':
            wsgienv = req.httprequest.environ
            env = dict(
                base_location=base_location,
                HTTP_HOST=wsgienv['HTTP_HOST'],
                REMOTE_ADDR=wsgienv['REMOTE_ADDR'],
            )
            req.session.authenticate(db, login, password, env)
            return self.session_info(req)
         if xmlData == 'FAIL':
            return {'error': _('Error, Invalid credentials !'), 'title': _('No access')}
```

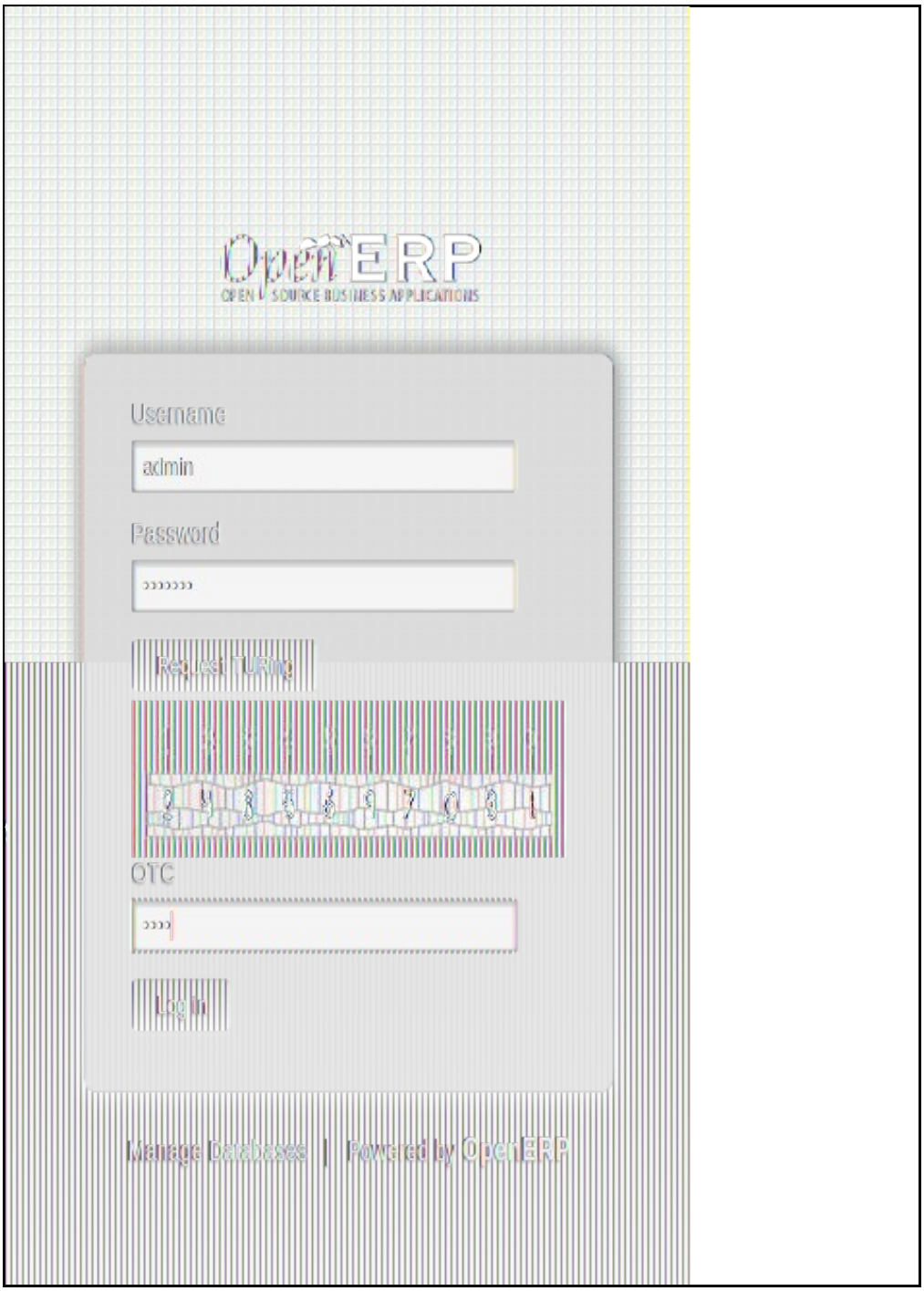
Testing

Open a browser and point to http://yourserverip:8069



After clicking the ?request TURing? button, the swivel appliance log should show ?127.0.0.1:Session started for user: admin.?

And this should be the result:



Then after login, the swivel appliance log should read:

?OPENERP SERVER IP? OpenERP:Login successful for user: admin.

Error Messages

On OpenERP stack trace

"ImportError: No module named urllib.urlopen"

Please refer to the this article, section Python 3.0 changes.

On the Swivel Log

AgentXML request failed, error: The agent is not authorized to access the server

User fails to authenticate with the above error message in the Swivel log.

This means that an Agent on Swivel server has not been defined for the OpenERP server.

Go to Server/Agents in the PINsafe admin console, and add a new entry, using the IP address of the OpenERP server.

Make sure the agent secret is the same as on the OpenERP configuration.