

Linvio Portal PHP Developer Guide

for  Appexchange

Version 1.1

Linvio, Inc. 2008

Table of Contents

Installing the Linvio Portal Sample Code	3
Prerequisites	3
Installing the Sample Code	3
Configuring the Site	3
The Portal Configuration File	3
Modifying the Sample Portal	7
Changing Style Sheets	7
Extending the SFProxy_Base and PortalUser_Base Classes	8
Useful URL Parameters	8
Displaying Content Items	9
content_item.php	10
list_view.php	10
Integrating Linvio Portal Features into an Existing Website	10
Querying Salesforce Data	11
Google Adwords for Salesforce	12
SFProxy and PortalUser Class Methods	12
Linvio Portal API Methods	13
logDocumentDownload	13
logPortalLogin	14
setupLogin	16
Additional Help	17

Installing the Linvio Portal Sample Code

Prerequisites

The following are required to install and configure the Linvio Portal sample portal PHP code:

1. Linvio Portal v1.0.8 installed in an Enterprise Edition (or higher) Salesforce account.
2. A web server running PHP 5.2.1 (or higher) with SSL and SOAP extensions enabled
3. You will also need a Salesforce account that will be used by the portal to connect to your Salesforce instance. This account will act as the proxy login account for all your portal users and for any content published to your site.

As a best practice, we recommend setting up a separate administrator account in Salesforce to use as your proxy account. You can then manage access permissions for this account as you see fit. For additional security you can also configure your proxy account for “API Only” access. Please consult the Salesforce system documentation for help in setting up API Only access.

Installing the Sample Code

To download the sample site, go to the Content Items tab in Linvio Portal. The default splash page for this tab includes instructions for getting started, and a “Download the sample site” link. Click this link, or if the splash page is hidden, visit <http://www.linvio.com/linvioportal/samplesite> .

1. Unzip the download file and upload the contents to a directory on your web server (for this example: **/portal**).
2. Ensure that your PHP configuration supports SSL and SOAP calls.
3. Check the write permissions on the `resize_temp` directory and the directory you plan to use for writing trace and error log messages from the portal (see portal settings below).
4. Configure the portal site as described below.

Configuring the Site

The Portal Configuration File

Here are the steps required to edit your Linvio Portal configuration file and set up the connection between the sample site and Salesforce.

- Edit the Linvio Portal configuration file `config.ini.php` in the **/portal** directory.

```

1 // Linvio Portal Configuration File
2 ;<?php
3 [SalesforceAPI]
4 sfUsername = "user@acme.com"
5 sfPassword = "password"
6 sfEndPoint = "https://www.salesforce.com/services/Soap/u/13.0"
7
8 [Organization Info]
9 sfOrgId = "0000000000000000"
10 orgName = "Acme"
11 copyrightNotice = "Copyright 2008, Acme, Inc. All rights reserved."
12 portalURL = "https://www.acme.com/portal"
13 siteURL = "http://www.acme.com"
14
15 [Administration]
16 pwdOverride = "override" ; Set this value to "" to turn off the password override feature
17 enableSFGoogleAdwords = true ; Enables Salesforce GoogleAdwords scripts on each page of the portal
18 siteAvailable = true
19
20 [Logging]
21 traceEnabled = false
22 logFile = "/path/to/writeable/directory/portal_log.txt"
23
24 [PayPal]
25 paypalMerchantAccountId = ""
26
27 [Google]
28 googleMerchantId = ""
29
30 :>

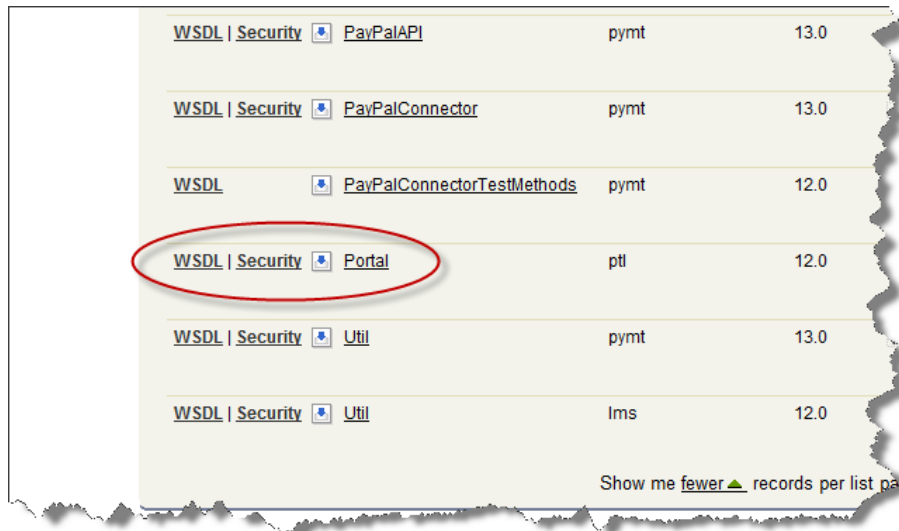
```

- Update the parameters with your own company and Salesforce account information:
- **sfUsername** and **sfPassword** should be the login credentials for the proxy admin account that will be used by Linvio Portal to access your account through the API.
- **sfOrgId** should be set to the org Id assigned to your Salesforce account. This will be used to enable Google Adwords for Salesforce tracking code, pages for submitting cases to Salesforce, and other org-Id driven features.
- Update the **copyrightNotice** value so your copyright will appear on portal pages.
- Set the **portalURL** and **siteURL** values to your full website domain and path values for both the portal and the main website so that links in the portal will point to the installed portal and to your main website.
- **pwdOverride** may be set to a secret password that administrators can use to login to any account. Leaving this parameter empty disables the password override feature.
- Set **enableSFGoogleAdwords** to activate Google Adwords for Salesforce tracking code in sample site pages.
- Leave **siteAvailable** equal to **true** unless you want to temporarily disable login access to the site (ie. When the site is down for repairs or updates).
- Set **traceEnabled** to **true** to enable writing of trace instructions to the site log file for debugging.
- Provide a full path for **logFile** for logging of errors and trace messages. Make sure PHP has access to write to files in this directory.
- Provide PayPal and Google Checkout merchant Ids if you plan to use the pre-built shopping cart pages.

The Linvio Portal WSDL File

There are two approaches you can take to setting up the Linvio Portal WSDL file with this site:

1. Generate an up-to-date Linvio Portal WSDL file from inside your Salesforce account by going to Setup>Develop>Apex Classes (after you have installed Linvio Portal off of the AppExchange).



Click on “WSDL” to download the version intended for you instance of Salesforce and save this file as `linviportal.wsdl.xml` in the `/soapclient` directory of the sample site.

2. **OR:** You may also edit the WSDL file that comes with the sample site directly if you choose; however this approach requires a bit more care.

To edit the WSDL file directly:

- a. Search for the string similar to <https://na5-api.salesforce.com/services/Soap/class/ptl/Portal> near the end of the file. (na5 has been used in this example)
- b. Replace the server name (“na5”) with the server name your instance of Salesforce is hosted on. For example: If you see “[https://na3.salesforce.com/...](https://na3.salesforce.com/)” in the address bar of your browser when you are logged into Salesforce (as shown in the screen shot below) you would change the WSDL to point to <https://na3-api.salesforce.com/services/Soap/class/ptl/Portal>

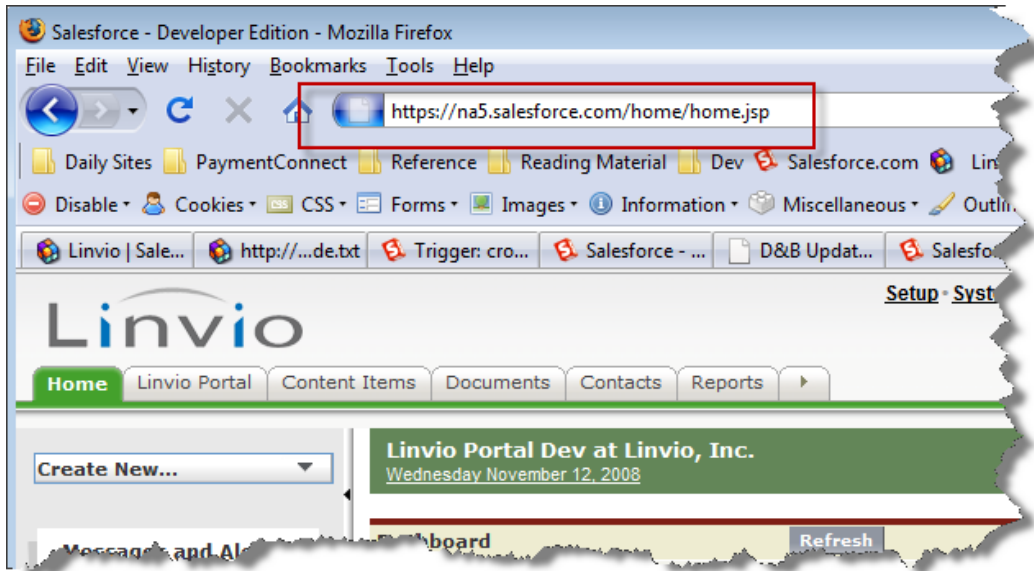
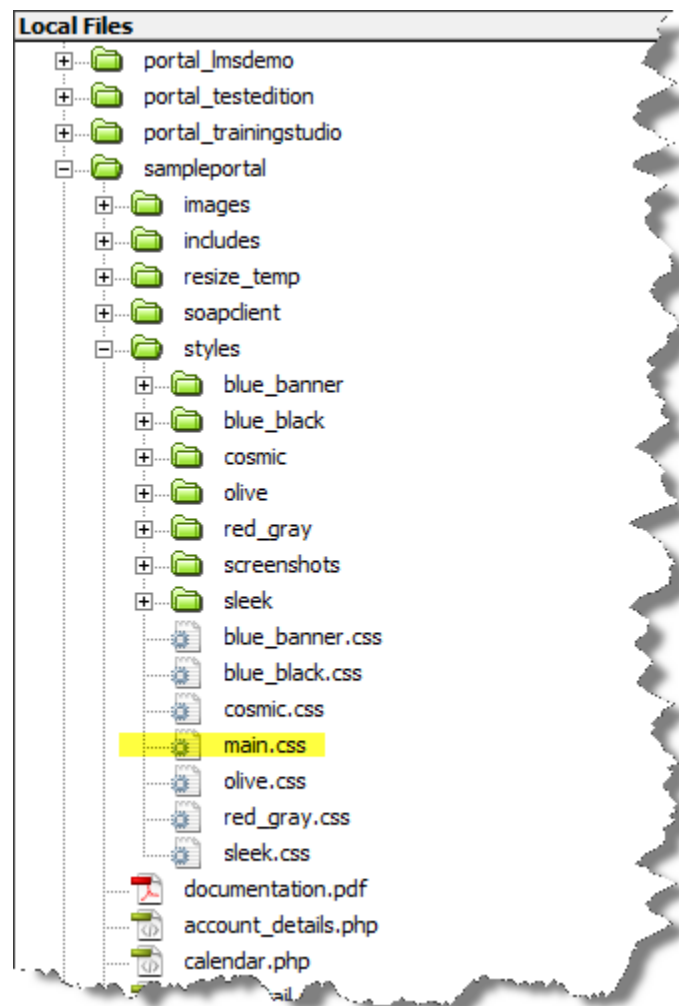


Figure 1. Determining your Salesforce server name.

Modifying the Sample Portal

Changing Style Sheets

Each page in the Linvio Portal sample site includes cascading style sheets that define the colors, fonts, and layout of the page. Developers can choose from a set of sample styles and modify the CSS content to change the site visual elements. Screenshots of the pre-built styles are included in styles/screenshots.



To apply a new set of style sheets site-wide, the top-level style sheet (styles/main.css – shown in the screen captures above) can be edited to point to a different style folder.

Here are the contents of main.css:

```
/* Linvio Portal Main Style sheet */
@import url(blue_banner/style_portlets.css);
@import url(blue_banner/date_picker.css);
@import url(blue_banner/portalnav_dhtml.css);
@import url(blue_banner/main.css);
@import url(blue_banner/calendar.css);
```

So for example, you might replace “blue_banner” with the name of the subdirectory containing your own custom style sheets for the site. Since main.css is included in every page on the site, this would effectively update all of the pages in the sample site.

Extending the SFProxy_Base and PortalUser_Base Classes

The Linvio Portal sample site includes proxy base classes (defined in sforce_portal_proxy_base.php) which are used to establish connections with Salesforce and perform common portal functions (such as checking user logins, resetting passwords and so on).

The sample site also includes stub classes which are subclasses of the proxy base class which you can modify to extend or override the methods provided in the base classes.

Class	Extends	File	Base Class File
SFProxy	SFProxy_Base	sforce_portal_proxy.php	sforce_portal_proxy_base.php
PortalUser	PortalUser_Base	sforce_portal_proxy.php	sforce_portal_proxy_base.php

We encourage you to add custom code to the PortalUser and SFProxy classes rather than in PortalUser_Base and SFProxy_Base as this will keep your code separate from Linvio code and make it easier to maintain.

The sample site code is already set up to include the classes SFProxy and PortalUser directly, so changes to these classes will be included automatically.

The base classes also include methods for retrieving a number of standard Salesforce objects (Contacts, Accounts, Tasks, etc), which can be used as we’ve defined them or overridden with your customizations in the subclasses. See *SFProxy and PortalUser Classes* on page 12 for more detail on the methods contained in the base classes.

Useful URL Parameters

Script	URL Parameters	Description
login.php	uid = username pwd = password retURL = return URL	The script login.php authenticates users for portal member access to the site. Use the uid and pwd parameters to allow auto-login. The retURL parameter tells the script which page to redirect to after successful login. If no retURL value is provided, login.php will redirect to the main page (index.php)
content_item.php	id = content item id tag = publish-to tag	The script content_item.php displays the details of a single ptl__Content_Item__c record. Provide either the Salesforce id for your content item, or a publish-to tag (e.g. portal:Sidebar) to retrieve and display a content item in full page format. If more than one matching record is found, only the first matching record will be displayed by this script.
list_view.php	target = publish-to-tag heading = string	This script list_view.php displays a list of ptl__Content_Item__c records that have been published to the specified target. The heading parameter is used to determine what you want the heading at the top of the list to say.

Displaying Content Items

When you install Linvio Portal in your Salesforce account, you also install a custom object called “Content Item” which is used for content management functionality and includes fields describing page titles, keywords, metatags and HTML content. The complete object name is ptl__Content_Item__c if you look under the covers at the object definition.

To retrieve content item data using the SFProxy class provided with Linvio Portal use the getContentItems() method. For example:

```
$sidebarResult = $sfProxy->getContentItems('portal:Sidebar');
$sidebarContent = new SObject($sidebarResult->records[0]);
$titleValue = $sidebarContent->fields->ptl__Title__c;
```

The sample site includes two generic template files that may be used to display these items; one as a detail view for a single item and the other as a list view of multiple items.

content_item.php

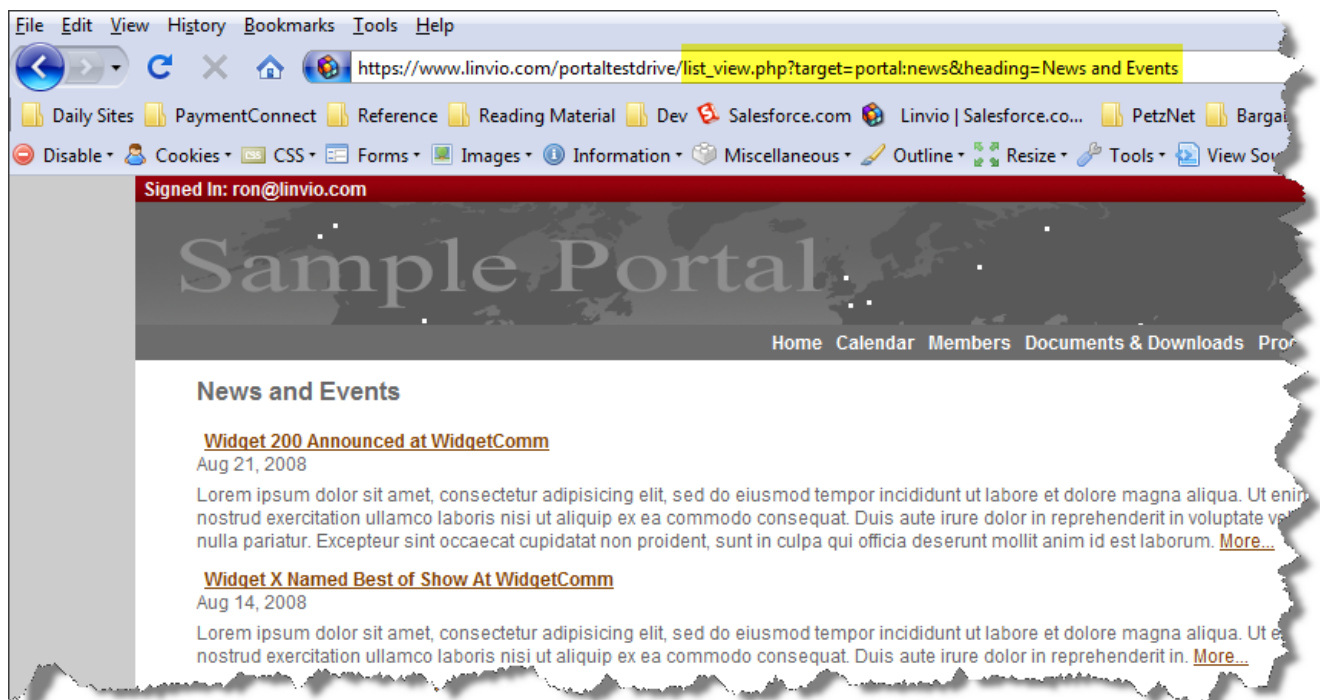
To display a single content item, call `contact_item.php?id=xxxxxx` where `xxxxxx` is the Salesforce record ID for the Content Item you wish to display. Or if you know the publish-to tag of the item you want to display, provide the “tag” URL parameter. For example:

```
contact_item.php?tag=portal:Sidebar
```

This PHP file pulls the metatags, keywords, title, and HTML content defined for the specified Content Item and embeds them in their respective areas on the page returned to the browser.

list_view.php

To display a list of content items, call `list_view.php?target=yyyyyy&heading=headingtext` where `yyyyyy` is the target keyword you want to match in the Content Item **Publish To** field and `heading` is the title you want to display above the list of items. The screen capture below illustrates how to list Content Items marked as **portal:news** items:



Integrating Linvio Portal Features into an Existing Website

In many cases, users may want to add Linvio Portal features to a public website, or simply convert an existing portal instead of using the sample site provided with Linvio Portal. The proxy class (SFProxy) included with the Linvio Portal PHP sample site can be used to convert your existing web pages into Salesforce-connected pages.

To enable a page, insert the following code snippet before any HTML content.

This code connects to Salesforce using the “proxy” Salesforce administrator login credentials you specify in the configuration file (config.ini.php) and checks to see if a portal user has been authenticated. If a portal user has not already been authenticated, the page redirects to a login page (login.php) to check login credentials against those stored in Salesforce.

```
<?php
require_once ('includes/sforce_portal_proxy.php');
session_start();
$sfProxy = SFProxy::init(); // Connect via proxy to SFDC
// Make sure the proxy has connected and the member is logged in before
showing page content.
if ($sfProxy == null) header( "Location: login.php");
if (!$sfProxy->portalUser->loggedIn) header( "Location: login.php");
$sfid = $sfProxy->portalUser->sfid;
$memberPortalAccess = true;
?>
```

Be sure that you have included the file login.php from the sample site (or your own version of it) to enable portal logins.

After the connection and login check, your code can access the SFProxy class and PortalUser class methods through the variables \$sfProxy and \$sfProxy->portalUser. See *SFProxy and PortalUser Classes* on page 12 for more detail on these classes and their methods.

For pages in your site that do not require login access, remove the login checks and use only the connection statements:

```
<?php
require_once ('includes/sforce_portal_proxy.php');
session_start();
$sfProxy = SFProxy::init(); // Connect via proxy to SFDC
?>
```

Querying Salesforce Data

Since Linvio Portal maintains the connection to Salesforce for you, all you have to do to submit a query is construct the query string and submit it through your portal proxy instance.

Assuming you have added the connection code at the top of your page, a sample query could be invoked as follows:

```
$query = "select id, Name, ptl__Portal_Nickname__c, MailingCity,
MailingState, MailingCountry from Contact where isDeleted = false and
ptl__First_Portal_Login__c != null order by ptl__First_Portal_Login__c
desc limit 5";

try {
    $queryResult = $sfProxy->mySforceConnection->query($query,
$queryOptions);
} catch (Exception $e) {
    Logger::logError($e, __CLASS__."::".__FUNCTION__."::Error
retrieving new members.");
}
```

See Salesforce API documentation for more information on submitting queries through the Force.com API.

Google Adwords for Salesforce

To enable Salesforce Google Adwords tracking on your pages, include the following code snippet immediately before the closing `</BODY>` tag on each page. The constant `ENABLE_SFGE_TRACKING` is set in the `config.ini.php` file and included with the portal includes at the head of each page, and can be used to turn on/off Adwords tracking.

```
<?php if (ENABLE_SFGE_TRACKING) { ?>
<!-- Begin Salesforce tracking code, Place immediately before closing
</BODY> tag -->
<SCRIPT type="text/javascript"
src="https://lct.salesforce.com/sfga.js"></SCRIPT>
<SCRIPT type="text/javascript">__sfga();</SCRIPT>
<!-- End Salesforce tracking code, Place immediately before closing
</BODY> tag -->
<?php } ?>
```

SFProxy and PortalUser Class Methods

Linvio Portal includes two proxy classes (`SFProxy_Base` and `PortalUser_Base`) which are used to establish connections with Salesforce and perform common portal functions.

We have also included empty subclasses of the proxy base class (defined in `sforce_portal_proxy.php`) which you can modify to extend or override the methods provided in the base classes.

PortalUser -> extends PortalUser_Base

SFProxy -> extends SFProxy_Base

See the proxy and portal user class documentation file included with the sample site code (`documentation.pdf`) for more information on these classes and methods.

Linvio Portal API Methods

The following web services API methods are included with Linvio Portal and can be access through the Salesforce API by exporting the Linvio Portal class WSDL from Salesforce and building your application around it.

For the examples code shown below, the WSDL file path is setup as follows:

```
$path = dirname(__FILE__);
$linvoPortalWsd1 = $path.'../../soapclient/linvioportal.wsdl.xml';
define('LINVIOPORTAL_WSDL', $linvoPortalWsd1);
```

If you are working with the sample portal code, the WSDL for these API methods has already been included and is used in the SFProxy ad PortalUser base classes.

logDocumentDownload

Creates a new `ptl__Portal_Log_Entry__c` record logging the download of a Salesforce document through the portal or website. If the Linvio Portal configuration settings have this feature disabled, the method will return error 301 (Document download logging disabled).

Parameter	Type	Notes
docId	String	The Salesforce record Id of the document that was downloaded.
Contacted	String (Optional)	The Salesforce record Id of the Contact who downloaded the document.
leadId	String (Optional)	The Salesforce record Id of the Lead who downloaded the document
Notes	String (Optional)	A note or commented related to the download.

```

/**
 * logDocumentDownload
 *
 * Calls Linvio Portal/Salesforce to log a document download request
 */
public function logDocumentDownload($documentId, $contactId=null,
$leadId=null, $notes="") {
    if (isNullOrEmpty($documentId)) return null;

    // Make API call to Linvio Portal
    try {
        $sessionId = $this->sfSessionId;
        $client = new SoapClient(LINVIOPORTAL_WSDL);
        $sforce_header = new
SoapHeader("http://soap.sforce.com/schemas/class/ptl/Portal",
        "SessionHeader",
        array("sessionId" => $sessionId ) );
        $client->__setSoapHeaders(array($sforce_header));
        $wsResponse = $client->logDocumentDownload(array("docId"=>$documentId,
"contactId"=>$contactId, "leadId"=>$leadId,"notes"=>""));

    } catch (Exception $ee) {
        Logger::logError($ee, 'Error logging document download (doc
Id: $documentId).');
        return null;
    }
    return $wsResponse;
}

```

logPortalLogin

Creates a new ptl__Portal_Log_Entry__c record logging the login of a member into the portal. If the Linvio Portal configuration settings have this feature disabled, the method will not record a log entry.

Parameter	Type	Notes
contacted	String (Optional)	The Salesforce record Id of the Contact who has logged in.
Notes	String (Optional)	A note or commented related to

```
/**
 * recordLoginTime
 *
 * Writes the current time/date to the 'Last Member Login' field in
 * Salesforce and logs a portal login event. Uses the Linvio Portal web
 * service method logPortalLogin(). If the Linvio Portal WSDL file
 * does not point to your SF server instance (na1, na2, na3 ...) this
 * function will fail and login times will not be recorded.
 */
public function recordLoginTime() {
    Logger::trace(__CLASS__."::".__FUNCTION__);
    if ($this->loggedIn) {
        Logger::trace(__CLASS__."::".__FUNCTION__." :Portal
User is logged in.");

        // Make API call to Linvio Portal
        try {

            $sessionId = $this->connection->getSessionId();
            $client = new SoapClient(LINVIOPORTAL_WSDL);
            $sforce_header = new
SoapHeader("http://soap.sforce.com/schemas/class/ptl/Portal",
            "SessionHeader",

            array("sessionId" => $sessionId ) );
            $client-
>__setSoapHeaders(array($sforce_header));

            $wsResponse = $client-
>logPortalLogin(array("contactId"=>$this->sfid,"notes"=>""));

        } catch (Exception $ee) {
            Logger::logError($ee, 'Error logging portal
login (contact Id: '.$this->sfid.')');
            return false;
        }

        Logger::trace('Successfully logged login');
        return true;
    }
}
```

```

    }
}

```

setupLogin

Sets up login access through Linvio Portal for the give Contact. This method will optionally send an email message to the Contact form inside Salesforce informing the contact of the new or updated account information. Linvio Portal provides a generic format for this email which can be overridden by entering an email template into the Linvio Portal Configuration settings record within Salesforce.

Parameter	Type	Notes
id	String	The Salesforce record Id of the Contact or Person-Contact.
username	String	Must be unique username (stored in the ptl__Portal_Username__c field on Contacts).
password	String	The unencrypted password for this user. The setupLogin method will encrypt and store this password in the Contact ptl__Portal_Password__c field.
Algorithm	String	Currently only MD5 encryption of passwords is supported by Linvio Portal. This value must be set to 'MD5'
emailNotification	Boolean	Determines whether or not an email notification should be sent to the Contact informing them of the new account or reset password.

```

/**
 * resetPassword
 *
 * Resets the current portal user's portal password using the Linvio
Portal setupLogin web service method. New password information is
emailed to the
 * contact from inside Salesforce.
 */
public function resetPassword($contactId, $username) {

```

```

        Logger::trace(__CLASS__."::".__FUNCTION__." :contactId =
        $contactId username = ".$username);
        if (isNullOrEmpty($contactId) || isNullOrEmpty($username)) return
        null;

        // Make API call to Linvio Portal
        try {

                $sessionId = $this->sfSessionId;
                $client = new SoapClient(LINVIOPORTAL_WSDL);
                $sforce_header = new
        SoapHeader("http://soap.sforce.com/schemas/class/ptl/Portal",

                "SessionHeader",

                array("sessionId" => $sessionId ) );
                $client->__setSoapHeaders(array($sforce_header));
                $wsResponse = $client->setupLogin(array("id"=>$contactId,"username"=>$username,"password"=>se
        lf::generatePassword(),"algorithm"=>"MD5","emailNotification"=>true));

        } catch (Exception $ee) {
                Logger::logError($ee, 'Error resetting password for
        $username. ');
                return null;
        }
        return $wsResponse;
}

```

Additional Help

For more information visit the Linvio website at www.linvio.com or our forum www.linvio.com/forum .

Consulting and integration services are also available from Linvio.

Linvio, Inc.
sales@linvio.com
 888-854-6846