
Liferay Application Integration Strategies

**Note: All of the configuration examples in 3rd-party software (i.e. – Apache, Oracle Java) in this document are examples only. Please read the appropriate documentation and apply the settings that are appropriate for your system. Liferay does not recommend these exact same settings for all systems.*

Contents

Overview	3
The Need for Application Integration.....	4
Integration Levels	4
Liferay Application Integration	6
IFrame Integration	7
WEB Proxy Integration	9
WSRP Integration	11
Full Liferay/Portlet Integration.....	12
Summary	15
Additional Information	15
Moving Forward	15
Liferay Training	15
Liferay Enterprise Edition Support.....	15
Liferay Professional Services	15

Overview

Integrating standalone web applications into a cohesive web-based experience is not an easy task. There is no magic recipe for a good integration in a very short time without having to make compromises. However, Liferay makes it possible to achieve a near-native integration with minimal effort. This paper gives a brief overview of various integration strategies using Liferay Portal, and discusses the associated benefits and tradeoffs of each method.

Integrating Liferay Portal together with existing corporate and IT assets can involve many different types of efforts:

- You want to integrate Liferay with your existing SSO authentication or authorization policy engine. This might involve development of a custom SSO connector to the service, or the re-use of an out-of-the-box Liferay authentication connector.
- You want your existing ERP or CRM workflows to be managed and executed through your Liferay Portal. This can range from a simple presentation-level IFrame to a Web Proxy clipping portlet to a custom portlet.
- You have multiple existing web-based applications for which you want to provide a consistent user experience.
- You want to integrate with consumer social networking so that you have a Facebook or Twitter presence.

All of these kinds of integrations are possible with Liferay. In this paper, we concentrate on integrating existing enterprise web applications into Liferay Portal in order to provide a seamless user experience and capitalize on existing corporate IT assets.

The Need for Application Integration

In most enterprises, a multitude of IT systems and applications serve the needs of the business. These range from low-level database services, order processing and fulfillment, CRM solutions, and the like. A Portal and integrated workflows are typically used to tie these together. However, through acquisitions, changes in IT leadership, or other influences, the systems are brought online at different times, with different design goals, and different integration mechanisms. Simple applications such as stock quotes or traffic maps do not depend on other business processes or data and can simply be added to the user interface at will (potentially being customized per-user). More complex applications need to be integrated together to provide critical business systems at the heart of the enterprise.

Application integration can take on many forms and levels of complexity. Before undertaking an integration project, one must consider many factors, including

- What tasks do users need to accomplish using the solution?
- What integration points are offered by the applications being integrated?
- What existing features in the Portal (or from a 3rd party) can help me?
- How long will this solution be needed?
- What expertise or other experience is needed?
- How secure does this solution need to be?

Taking time to answer these questions will help you to understand the overall cost vs. benefit of a given solution and ultimately to make the right choices during implementation.

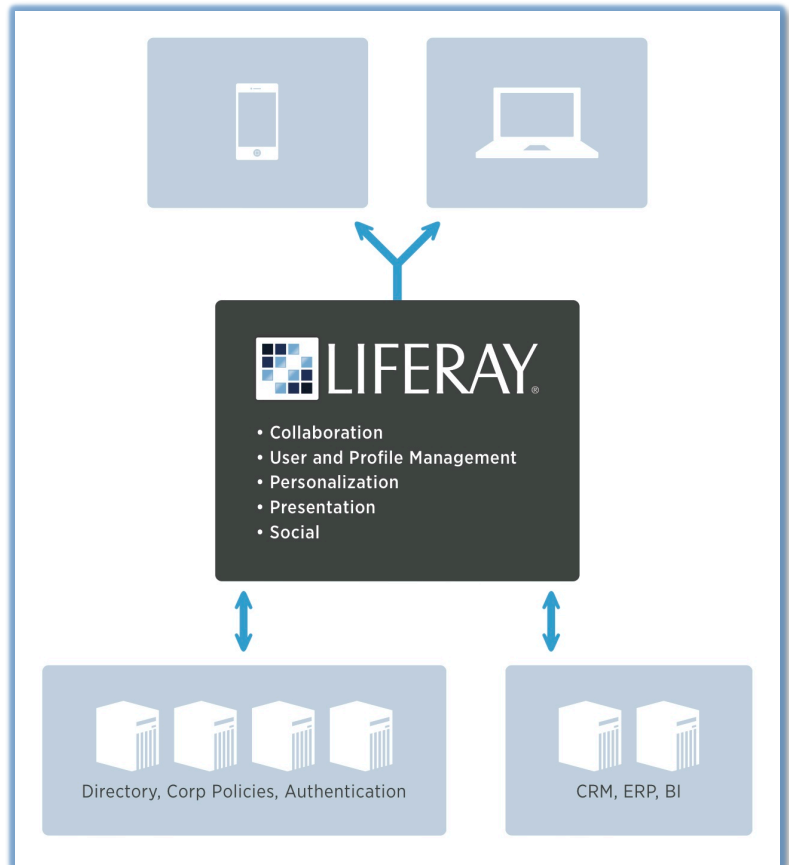


Figure 1: Typical Integration of Enterprise Systems using Liferay

INTEGRATION LEVELS

Integrating applications together using a Portal means exposing, or surfacing, the applications to the end user in a way that they can easily interact with them to get their job done. This is the job of a Portal, which presents users with a (hopefully) unified interface that is intuitive, responsive, and accurate. The place at which a user typically interacts is called the presentation layer. For example, in Liferay Portal, the presentation layer is seen using a web browser. Integrating applications at the presentation layer alone is usually the easiest and cheapest option. It requires very little development effort to get up and going, and is very non-invasive to the applications being integrated. The application's native user interfaces are maintained, and no modifications of the applications are needed. In almost all cases, the applications themselves have no knowledge that they are being integrated. However, there are some serious drawbacks in this approach:

- Information (its state) entered in an application is lost when navigating away from the page

- Individual applications have different interface styles (look and feel), different decorations, and different data entry paradigms
- No ability to interact with the underlying portal
- Little to no control of the target application's content

Some of these drawbacks can be mitigated by a little extra effort, but if these drawbacks prevent you from accomplishing your design goals, a different solution is needed.

A deeper (and more complex) integration option is via a proxy. A proxy allows you to fetch content from an external application on the server side, and inject it into the portal environment. The biggest benefit of this option is allowing you to modify the content before it is presented to the user. For example, you can remove headers and footers, or retrieve content from remote sites through specialized gateways that restrict access.

While this solution addresses some of the problems of pure presentation-layer integrations, it introduces some of its own. Since this effectively combines the portal presentation environment with the underlying application's presentation environment (at a lower level than before), there is a good chance of a conflict between the two, resulting in one or both systems not functioning.

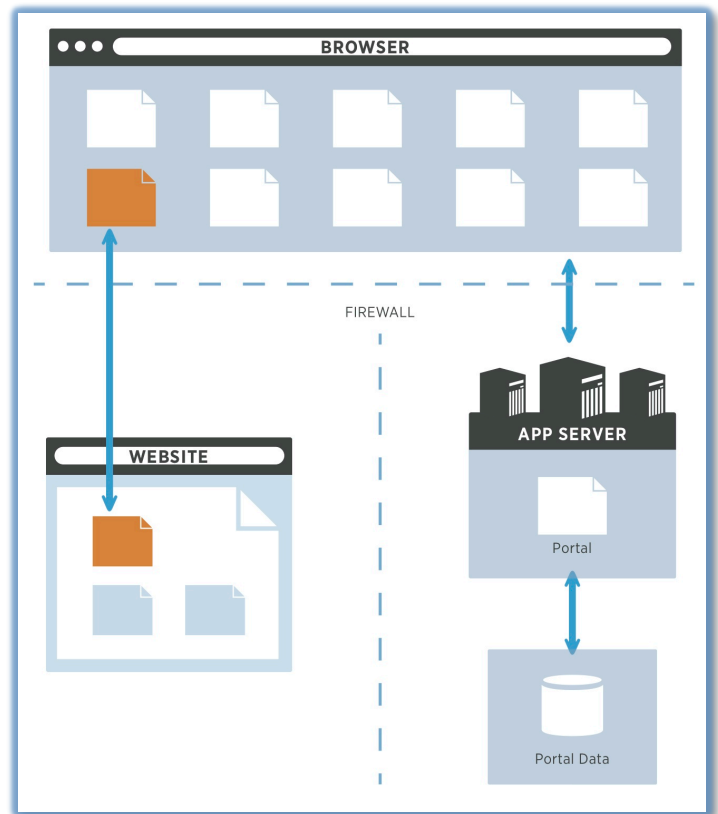


Figure 2: Presentation-Layer Integration

The most seamless and powerful solution is the portal-level integration. It is also the most complex and costly when designing from scratch. In this solution, native portal applications (portlets) are designed and developed, along with any needed supporting services, to support the desired workflows. The portlet environment is fully utilized: portlets communicate with each other via Eventing or Public Render Parameters, view modes are designed appropriately, and underlying calls to the applications being integrated are made either via direct Java APIs, or remote invocation via SOAP, REST, or other remote networking protocols. Realize that full portal integration is not a one-size-fits-all solution. It is important to understand business requirements, capabilities of underlying enterprise applications, and the true need for integrating them together, before undertaking a true integration project.

In some cases, much of the work of integrating an application

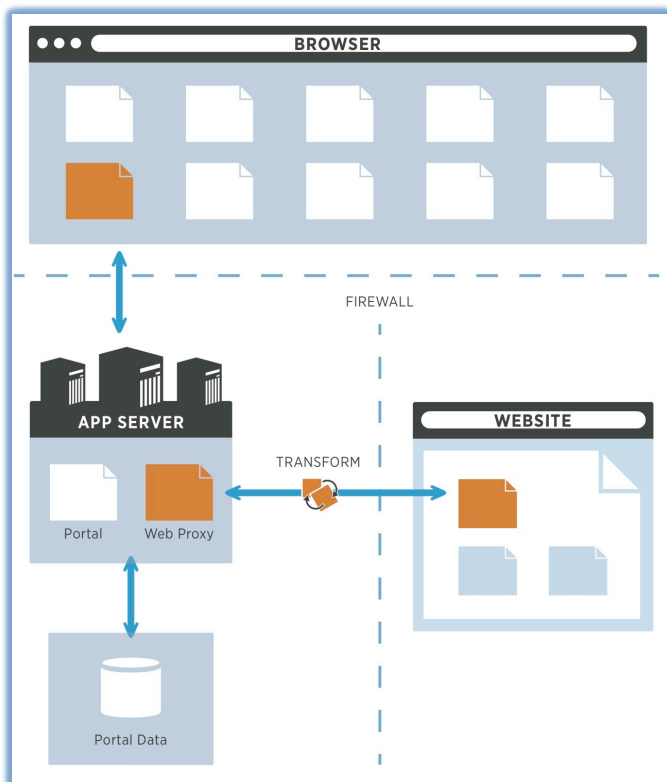


Figure 3: Web Proxy Integration

into a portal environment has been done for you. For example, the application has already had portlets written for it, or the application supports WSRP. In this case, it is straightforward to integrate these into a portal environment. Indeed, native portal integrations are available for popular enterprise applications. For example, SyncEX has portlets available for Microsoft Outlook, which enables mail and calendar integration into an existing portal environment.

The following table summarizes key attributes of each of the discussed integration levels:

	PRESENTATION LEVEL	PROXY	FULL / NATIVE
Relative Strengths and Weaknesses of various integration levels			
Complexity	Low	Medium	High
Cost	Low	Low	High
Conflict Potential	None	High	None
Control over User Experience	None	Medium	High
Portal Artifact Access	None	Low	High

Finally, it should be noted that there isn't a one-stop solution for application integration. The final solution may even involve a mixture of the above strategies, in order to meet the requirements of the problems you're trying to solve and the resources at hand. With this broad understanding of integration strategies, we next discuss specific features of Liferay Portal and how it can help you to make the most of your investment.

Liferay Application Integration

The Liferay Portal environment provides several facilities designed to help integrate external applications into a single, cohesive user experience. These facilities range from simple presentation layer integrations to full on portlet integrations.

The following table summarizes the key differences between integration options:

	IFRAME	WEB PROXY	WSRP	FULL INTEGRATION
Key differences between Liferay's integration options				
Difficulty	Easy	Easy	Medium	Complex
Authentication Options	HTTP Basic, Form	HTTP Basic, NTLM	Custom	Native
Authorization Options	None	None	Custom	Native
Customizations	None	Medium	None	Easy
Resource Usage	Low	Low to Medium	High	Low
Access to Liferay Artifacts	None	Low	High	High

IFRAME INTEGRATION

An IFrame is a very inexpensive and simple way to integrate applications at the presentation level. In Liferay, this is implemented using an IFrame portlet. The portlet makes it possible to embed another HTML page inside the current page. Furthermore the user can navigate through the embedded page without losing the context of the portal page.

The IFrame portlet uses the HTML IFrame tag that is part of HTML 4 and is supported in all major browsers. The IFrame portlet will adjust to the size of the HTML page if that page is hosted in the same server. The browser will add scrollbars if the embedded page does not fit within the size of the IFrame.

To get started with the IFrame portlet, you can add an instance of it to any page using Liferay's built-in applications. Once added to a page, click the configure icon to configure the address of the page you wish to embed. For example, type in <http://www.google.com> and save the changes. Navigate back to the page on which the IFrame portlet was added, and it will now show Google's search page inside the portlet.

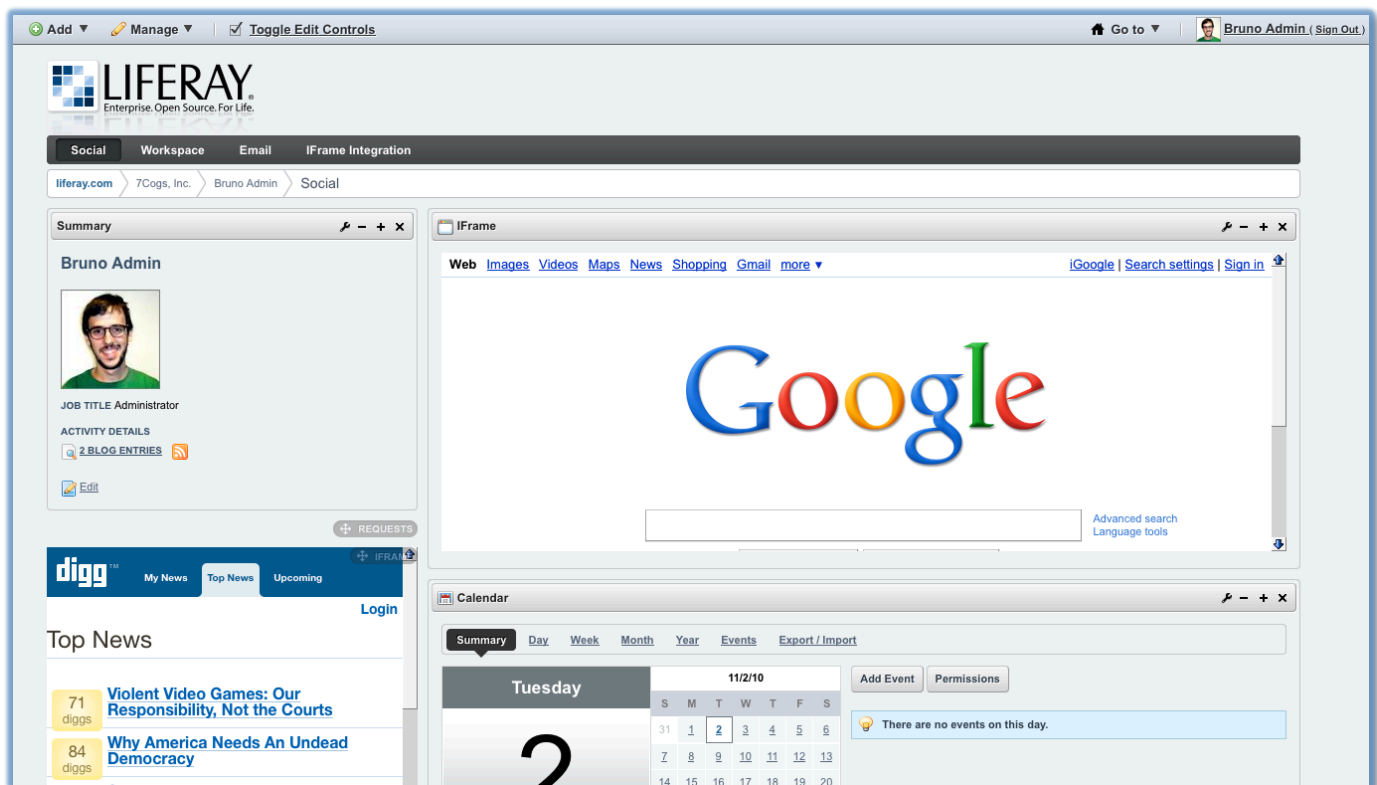


Figure 4: Example IFrame Integration

You can interact with this page as though it was the only page you were viewing. The window acts as a miniature browser for that page alone.

IFrame Authentication

If the embedded web page requires authentication the IFrame portlet can be configured to provide authentication information. This section describes the different options offered to the user.

Basic vs. form based authentication

There are several ways to provide the authentication information to the embedded page. The IFrame portlet supports two of them:

- **Basic authentication:** This method is described in the HTTP specification. The user name and passwords are usually provided as HTTP headers but it's also possible to embed them in the URL with the syntax <http://username:password@www.myurl.com> This method is simple but it's not supported in many websites.
- **Form authentication:** Is the name given to websites that authenticate the user by making him fill in a form with his user name and password. It's the one used in most web sites. There are two variants of this: GET and POST. The first one sets the username and password in the URL and the second one in the body of the request. Usually they make little difference and both will probably work with any website.

When form-based authentication is used, the IFrame portlet will first make a (hidden) request to a portal page that recreates a form and then submits it to the configured URL. To learn the details of this form take a look at the following file from Liferay's sources:

`portal-web/docroot/html/portlet/iframe/proxy.jsp`

Be aware that *neither* of these methods are secure unless they are sent through an HTTPS connection. If you want to make sure that the user and password information cannot be seen by third parties only use authentication with sites that support HTTPS and use only URLs in the IFrame that start with `https://`.

Automatic vs. configured user name and password

Liferay provides two ways to retrieve the user name and passwords that will be used to authenticate against the embedded web page:

- **Automatic:** The user name and password of the current user that is viewing the page with the IFrame portlet will be used. It's only supported by the 'basic authentication' method described in the previous section.
- **Configured:** The administrator will configure a user name and password and they will be used for all users.

To use the automatic way just leave the 'user name' and 'passwords' fields blank. Otherwise fill them with the values that will be used for authentication. You may use `@email_address@`,

The screenshot shows the 'IFrame - Configuration' window. The 'Setup' tab is selected. The 'General' section has 'Source URL' set to 'http://somesite.com'. The 'Authentication' section has 'Authenticate' checked and 'Authentication Type' set to 'Form'. The 'Form Method' is 'Post'. The 'User Name' section has 'Field Name' 'userName' and 'Value' '@email_address@'. The 'Password' section has 'Field Name' 'passwd' and 'Value' '@password@'. The 'Hidden Variables' section has 'site=foo,org=baz'. The 'Advanced' section has 'HTML Attributes' with a list of attributes: 'alt=', 'border=0', 'bordercolor=#000000', 'frameborder=0', 'height-maximized=600', 'height-normal=300', 'hspace=0', 'margin=0'. A 'Save' button is at the bottom.

Figure 5: Sample IFrame Configuration Screen

@screen_name@, or @user_id@ as tokens that represent the currently logged-in user information. Liferay will replace the tokens with the information from the currently logged-in user when the IFrame portlet is displayed.

Hidden variables

When using form-based authentication, the IFrame portlet allows the administrator to configure a set of additional parameters that will be sent with the form. These parameters must be set in the *Hidden Variables* field and its format must be:

```
xxxx=blah;abc=formSubmit1
```

Cross-Domain Concerns

There is often a need for dynamic communication between applications when integrating them together at the presentation level using IFrames. For example, when a user clicks on the name of a sales contact from their list of contacts in one window, a separate window should update to show the geolocation of that contact on a map. Another common situation is resizing the IFrame to snugly fit around the content. However, when applications are being hosted and provided through separate hosts, modern browsers will prohibit direct communication or execution of code between the two sites hosted in the IFrames. They will also prevent properties (such as the content size) from being read at all, thereby making resizing the surrounding frame impossible.

There are some cases where this can be bypassed, but requires that you have the ability to modify the application being embedded, in order to send messages which can be read by code in Liferay. There are many off-the-shelf open source solutions available for this, most of which rely on the `window.postMessage` JavaScript API. However, the complexity and sustainability of this kind of implementation further illustrates that presentation-level integration may not always be the best solution.

WEB PROXY INTEGRATION

Integration via web proxy is similar in nature to the IFrame integration. Web Proxy integration ultimately accomplishes the same goal (presenting disparate applications and workflows in a seamless user experience), however does so via a slightly different approach, which gives you, then integrator, more control over the final user experience. Advantages include:

- Where an IFrame is like a window onto another application, the Web Proxy Portlet behaves more like a portlet, providing a block of HTML into your portal page.
- URLs in an embedded IFrame application would take you out of the portal, when clicked. In contrast, links in an embedded application that are served via the Web Proxy portlet are automatically re-written, such that when clicked, the user is not taken out of the portal context.
- With IFrames, the end user's browser must have direct access to the embedded application. With Web Proxy, applications can be accessed through a firewall or other proxy that are not otherwise reachable by the end user's browser.
- Besides URLs, other content can be re-written (e.g. removal of headers/footers, or other "clipping" operations).

To use Liferay's Web Proxy Portlet, simply add the portlet to the desired page. Similar to the IFrame portlet, the configuration screen of the Web Proxy Portlet is used to configure the application to embed via the portlet, and any necessary authentication information. When the Web Proxy Portlet is displayed, the portlet will first fetch markup from the configured URL. It will then apply a transformation via the default XSL transform (which will rewrite links, e.g. <http://myapp.com/link1> will become something like http://myportal.com/pbhs/324adfe_34/link1), or a custom XSL transform specified via configuration. Once transformed, the

final markup is displayed in the Web Proxy portlet.

Custom Stylesheets

A custom XSL transform would be needed when integrating an application that has non-standard content. For example, Microsoft Exchange will emit markup that has special attributes that need to be re-written. Custom stylesheets can also be used to inject custom JavaScript code, for example to automatically log the end user into the embedded application when its pages are accessed. These would be handled via a custom XSL transform. To configure a custom XSL transform to use, enter the transform in the “Stylesheet” text area.

Web Proxy Authentication

If the embedded web page requires authentication the Web Proxy portlet can be configured to provide authentication information. Liferay's Web Proxy support provides two kinds of authentication:

- **Basic** – Similar to the IFrame portlet, this style of authentication should only be used when HTTPS is being used. To configure basic authentication, enter the “Proxy Authentication Username”, and “Proxy Authentication Password” fields.
- **NTLM** – NT Lan Manager – an authentication protocol used by Microsoft Windows networks. To configure NTLM, enter the “Proxy Authentication Host” and “Proxy Authentication Domain” for your network.

Scope

By default, the Web Proxy portlet will modify/rewrite *all* links in the embedded application in order to keep the user in the portal when a link is clicked in the embedded application. In some cases, there may be a need to not rewrite links pointing at certain locations. For example, it is common to only re-write URLs that are within the intranet of the application. Other URLs should not be touched, as the URLs may not even be accessible from the server due to firewalls or other network configurations preventing access. In this case, one would specify the scope as a regular expression which selects only those URLs on the intranet. For example, the following URL might match an intranet:

```
.*intranet.corp.com.* | .*site2.corp.com.* | .*hr.corp.com.*
```

Which would rewrite any links that matched the above regular expression. All other links would remain untouched.

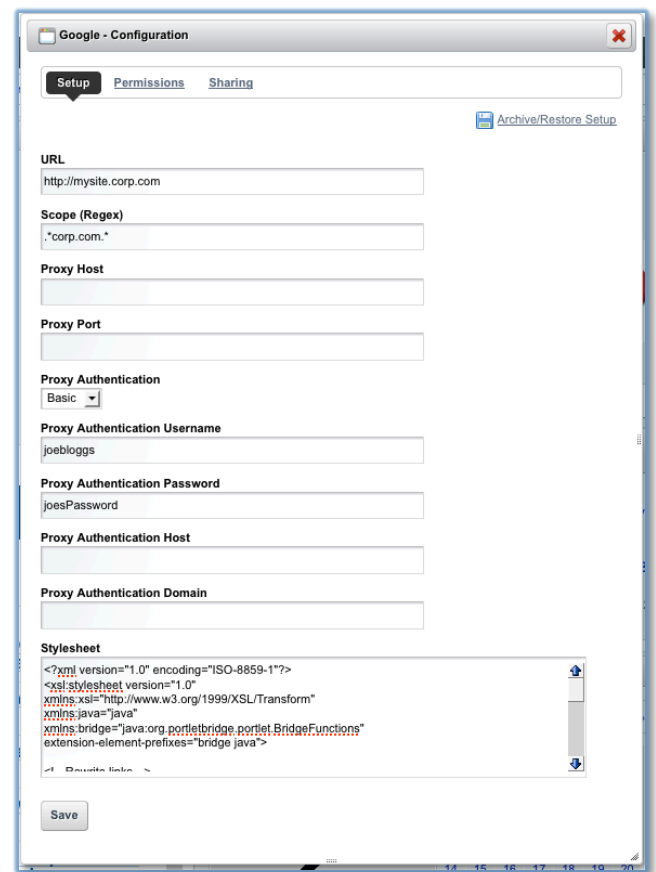
The image shows a web browser window titled "Google - Configuration". It has three tabs: "Setup", "Permissions", and "Sharing", with "Setup" being the active tab. In the top right corner of the "Setup" tab, there is a link that says "Archive/Restore Setup". The configuration form includes several fields: "URL" with the value "http://mysite.corp.com"; "Scope (Regex)" with the value ".*corp.com.*"; "Proxy Host" (empty); "Proxy Port" (empty); "Proxy Authentication" set to "Basic" via a dropdown menu; "Proxy Authentication Username" with the value "joeblogs"; "Proxy Authentication Password" with the value "joesPassword"; "Proxy Authentication Host" (empty); and "Proxy Authentication Domain" (empty). At the bottom, there is a "Stylesheet" text area containing an XSL transform code snippet. The code starts with an XML declaration and an XSL stylesheet declaration, followed by namespace declarations for "xsl" and "xhtml". It then defines a function "bridge" and a "bridgeFunctions" object. The "Save" button is at the bottom left of the form.

Figure 6: Web Proxy Configuration

WSRP INTEGRATION

Web Services for Remote Portlets (WSRP) is an OASIS-approved network protocol standard designed for communications with remote portlets. All enterprise editions of Liferay support WSRP 2.0, as does Liferay Portal 6.x Community Edition.

Integration via WSRP is similar in concept to Web Proxy and IFrame – you simply place WSRP portlets on your page and configure them to point to the WSRP Producer endpoints of the application you wish to integrate. **An important assumption is that the application you are trying to integrate already makes WSRP-compliant portlets available to consume.** Unlike the IFrame and Web Proxy solutions, WSRP actually respects most of the Portlet artifacts like Window States, Modes, and a Two Phase Commit. Much of the processing of portlet events and resource handling is transparent to the consumer. Theoretically this seems like a great, seamless technology. In practice however, this is hardly the case. There are enough ambiguities in the WSRP specification that make it pretty difficult to consume more than the simplest of portlets. Anything requiring SSO, file uploads, or other complex interaction is not possible without some proprietary integration.

User Identity and Authentication

One of the historical difficulties when using WSRP is around User Identity. The WSRP specification does not address this topic at all, and has been one of the major drawbacks to this approach. In a WSRP integration, a WSRP Consumer consumes content and data from a WSRP Producer. This exchange is done under the guise of the logged-in user. However, the user has only explicitly logged into one side of the exchange (the Consumer). The user has not logged into the Producer system (which may not even have a user interface for logging in, or even be a true portal for that matter!). During the WSRP Producer/Consumer exchange, a user identity *is* propagated, which enables the producer to store preferences and other persistent state related to the user. However, it is not guaranteed that this identity is the same identity that would be asserted if the user were to actually log into the producer side (assuming it was possible to log in).

Liferay's WSRP Consumer implementation propagates the user identity that was used to log into Liferay – for example, if the login method is by email, then when

joe.bloggs@7cogs.com logs into the portal and accesses a

page that is consuming WSRP portlets, the user identity joe.bloggs@7cogs.com is propagated as part of the WSRP exchange (along with other ancillary information, such as full name, birthdate, etc.)

It is possible to assert a greater degree of control over the user identities involved in WSRP, but using other specifications and implementations. In particular, OASIS has specifications for propagating user identity using UserNames, X509 Certificates, or SAML tokens. However, these specifications are not part of the WSRP standard, which has resulted in interoperability problems between WSRP implementations. It is for these reasons that the final option, a full portlet integration, is sometimes required.

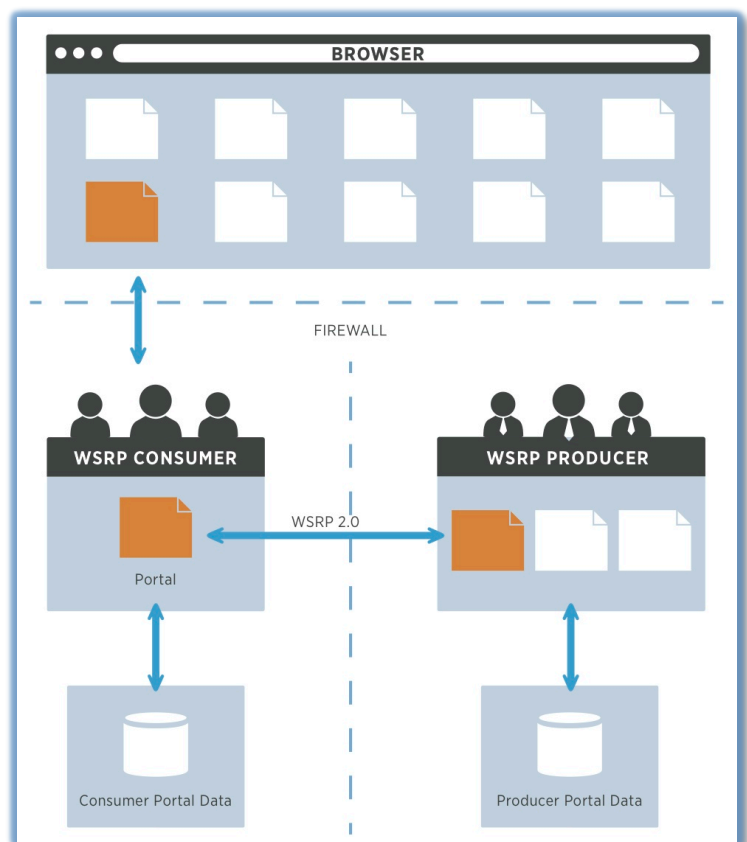


Figure 7: WSRP Producer and Consumer Integration

This is also the best route when the application you wish to integrate does not provide WSRP-compliant portlets, and cannot be easily integrated via IFrame or Web Proxy solutions.

FULL LIFERAY/PORTLET INTEGRATION

This integration method involves writing native portlets or other runtime artifacts which access the application you wish to integrate through low-level data-driven APIs, and presenting a user interface using native Liferay UI mechanisms. For example, the email application that ships with Liferay can integrate any IMAP-compliant mail system (such as Outlook or Google Mail) by using the IMAP protocol to exchange information. Other integration mechanisms are available depending on the application in question (e.g. REST interfaces for integrating with Facebook, or SOAP interfaces for integrating with Salesforce.com).

This integration method is best used when the application either has no web-based interface (e.g. a legacy enterprise apps), or has a very minimal or non-intuitive interface, and the application has lower-level APIs available. It requires an in-depth knowledge of the application being integrated, and there is no one-size-fits-all approach to its integration.

Various integration methods can be used depending on the type of application being integrated. Consider the following topics when planning your integration:

- **Authentication and Authorization**

Liferay has a built-in authentication and user management system. A user logs in (or creates a new account) using an identifier and credentials, usually a username and password. The user's profile is stored in the Liferay database. Users can be managed through the Liferay Control Panel, for example to assign a Role to a user, add or remove them from a community, and so on. This is fully self-contained.

When integrating an external application, it is usually the case that the application has its own notion of authentication and user management. For enterprise applications, this usually means that the applications are already integrated with an authentication system that is combined with a user store. For example, using OpenSSO for authentication and LDAP for user directory and password management.

Liferay has a flexible and fully customizable authentication and authorization system. Through

its *Authentication Pipeline*, external authentication systems can be used to augment or completely replace Liferay's built-in notion of authentication. The pipeline is essentially a serialized set of authenticators that must successfully validate a user and their credentials before the user is authenticated. To affect how Liferay authenticates users, you must insert (or replace) using your custom authenticators.



Figure 8: Sesame Street on Liferay: Integrated with Alfresco CMS, MySQL, Solr Search, and Nagios.



Figure 9: Liferay Authentication Pipeline

Liferay has built-in support for various 3rd-party authentication systems and user directory systems, including standard LDAP (with pre-configured settings for Apache DS, Fedora DS, Microsoft Active Directory, Novell eDirectory, and OpenLDAP), CAS, Facebook, NTLM, OpenID, OpenSSO, and SiteMinder.

- **Search**

As with Authentication, Liferay comes with a default search implementation using the Lucene search engine. This engine retrieves and indexes all content within Liferay. There are other search engines available (such as the popular Solr search engine from the Apache project, GSA, FAST, Coveo, etc). Regardless of which engine you use, there are various ways of integrating 3rd-party applications with Liferay Search:

1. Search Portlet

One easy option is to create a portlet that presents a simple search UI, and delegates searching content to the 3rd-party application. Similar to the IFrame style of integration, it has its pros and cons. It's very easy to implement, but the drawback here is that search results are not combined (or federated) with the overall portal search functionality, and will appear completely disjoint from other content. It is also difficult to do proper authorization, for example, to only see search results that are available to the given user.

2. OpenSearch

Liferay's built-in search functionality makes use of the OpenSearch API. Using this API, your custom portlets could declare that they participate in OpenSearch. When this is done, when the user uses the Liferay search facilities, results from the portal are combined with results from whatever external system your OpenSearch implementation is able to access, and shown in a federated result set. Your application may also provide an OpenSearch implementation (such as Alfresco).

3. Search Engine

As described above, Liferay's pluggable search allows you to swap out the default Lucene engine with the engine of your choice. Liferay makes available a Solr search plugin, which automatically configures Liferay to use an external Solr search engine for indexing and querying. The applications you integrate into your portal can also be integrated with Solr, such that when activities occur within the application (for example, a contact is added to address book), this information can be indexed and made available through Liferay search.

- **User Interaction**

This is highly dependent on the nature of the application being integrated, but you should consider the workflows you wish to accomplish in your portal before designing the user interactions. A successful integration will allow your users to accomplish their workflows using intuitive and efficient user interfaces.

When multiple applications are to be integrated together, Liferay provides the foundation for creating a compelling interaction model using its modern, flexible user interface components.

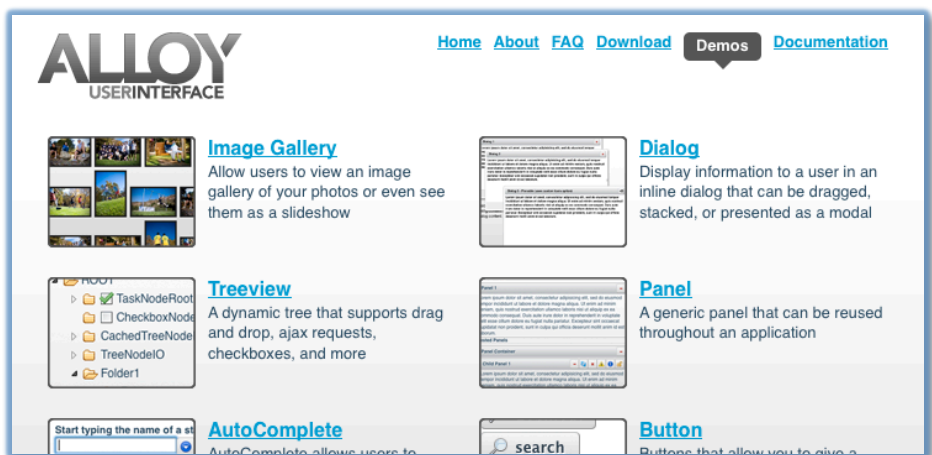


Figure 10: Example AlloyUI Components

1. **Out-of-the-box, customizable portlets.** These portlets can be used as-is, or customized via Liferay's plugin architecture to suit your needs. For example, the Calendar portlet could be used to show data from a legacy backend corporate calendaring system.
2. **AlloyUI** provides simple and complex interface components, which can be used to create rich, responsive interfaces.
3. **Full support of industry-standard JSR-286 Portlets.** Using inter-portlet communication, sophisticated yet intuitive interfaces can be built that bring together your enterprise applications in a cohesive user experience.

- **Collaboration and Social Integration**

Social features are increasingly important to consider when implementing any web-based enterprise application. Liferay has been built with social in mind, and provides integrators with many options for enabling these features during application integration. These include:

1. **Out of the box collaboration portlets.** As described earlier, many of these can be used for easy integration of external sources of content or data, such as Microsoft Exchange (via the out-of-the-box Mail/Calendar applications), or through RSS subscriptions to the Blog applications.
2. **Activity Stream integration** via the Activity Tracker API. When an activity occurs in the context of using your composite application (or in the context of an individual integrated application), Liferay's Activity Tracker API can be called to register such actions on behalf of the users. Other users who are connected to this user can see this activity when visiting their activity stream. Further, an integration can be made with Liferay's Social Equity system, giving "points" to users for undertaking certain tasks.
3. **OpenSocial.** Rather than writing portlets, if you are more familiar with (or have a requirement to integrate) gadgets, Liferay can host your gadgets directly on the portal, via Liferay's support of OpenSocial. The OpenSocial concepts, entities, and actions defined in its specification are mapped to the equivalent concepts in Liferay. The social-oriented APIs of OpenSocial mean that much of the hard work of socially-enabling your application is done for you!

- **Advanced Integration Techniques**

Liferay provides even more integration options when the basics don't work. For example, many enterprise data systems are integrated using some form of a messaging bus, such as an ESB, or even in conjunction with Liferay's own message bus plus web-oriented architectures. These messaging solutions deal with network fault tolerance, differing application data interchange formats, and the inevitable change that comes with a long-lived enterprise system. Liferay can integrate with these systems, and proxy enterprise data into your Liferay environment for further processing.

Summary

Through acquisition, organizational changes, and environmental factors, today's enterprises are a very heterogeneous mixture of legacy applications, enterprise data warehouses, applications, and IT staff. Coordinating these resources and creating a cohesive user experience is necessary to realize the full value of organizational assets.

All but the most basic of enterprise use cases require some form of integration to be truly useful. Integration cost, complexity, and effectiveness should be evaluated and measured against goals before deciding how to proceed with an integration. In some cases, a simple presentation-oriented integration will cover 80% of the use cases, and be "good enough" to not justify the cost of a more complex undertaking. In other cases, a deeper integration of products and portals is necessary to achieve the desired experience, or to take advantage of features already offered by the products being integrated.

Liferay Portal is in a unique position to offer a single point of access for organizations' data, content, and information, whether originating from Liferay itself, or from existing in-house applications (ERP, CRM, etc). Liferay's flexible and modern architecture exposes integration points at many levels, including presentation, proxy, WSRP, and fully exposed and supported web services (SOAP), JSON, RMI and its own tunneling classes. As long as your existing applications are open and provide support through some service layer or API, Liferay can integrate with those applications. There is a range of options for application integration depending on your needs, from web services and a Liferay IFrame portlet for lighter integration scenarios, to a web proxy, WSRP, or native application for more complex integration scenarios.

Additional Information

For more information on Liferay integration options and Designing With Liferay, please see the Liferay Portal Developer's Guide at: <http://www.liferay.com/documentation/liferay-portal/6.0/development>

Moving Forward

LIFERAY TRAINING

Liferay offers a Liferay Developer course, where you can learn about and get hands-on with Liferay development and integration strategies. Please see <http://www.liferay.com/services/training/topics/developer-training> or contact training@liferay.com for more information.

LIFERAY ENTERPRISE EDITION SUPPORT

Liferay Enterprise Edition ensures stability and reliable technical support for your Liferay Portal installation and your organization's team, including a customer portal, product bulletins, security alerts, plus the support of over 60 partners worldwide.

LIFERAY PROFESSIONAL SERVICES

Liferay offers enterprise application integration and architecture assistance for your system. Since specific integration needs may vary from system to system, Liferay recommends thorough evaluation of your environment and business requirements before applying any of the techniques mentioned in this paper.

Contact sales@liferay.com for more information.