XPages: Social Tools v1.01

Notes In 9

www.xpagescheatsheet.com Updated as of Jan 26 2012

XPages.Info

Created by Dan O'Connor, Niklas Heidloff, Phil Riand and David Leedy

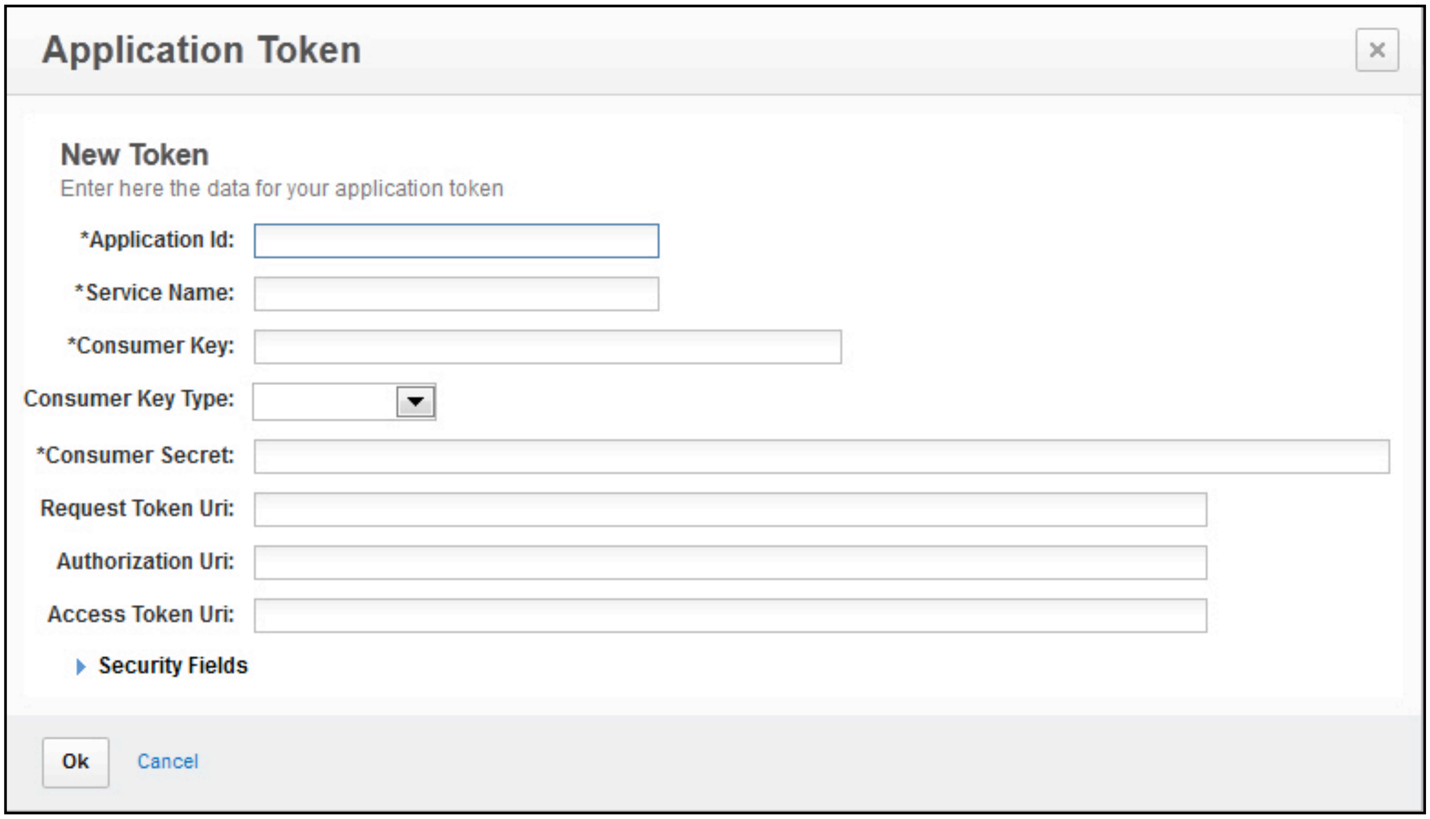
The XPages Social Enabler

The XPages Extension Library now contains functionality which allow users to connect to social networks, and REST services. In order to build "social capabilities" into your application there are new controls, data sources and other artifacts such as sample custom controls available through the Extension Library.
These capabilities are collectively referred to as the "XPages Social Enabler". The XPages Social Enabler consists of three parts:
➔ The XPages Social Enabler plug-in
➔ Token store application (WebSecurityStore.nsf)
➔ XPages Social Enabler sample application (XPagesSBT.nsf)

The XPages Social Enabler is freely available as a part of the XPages Extension Library. It can be downloaded from http://XPag.es/?XLibDownload (or by going to the Downloads section of http://extlib.openntf.org). Once downloaded please follow the setup instructions to install the XPages Extension Library on the Domino server, Notes and Domino Designer clients. The XPagesSBT.nsf application will need to be signed by the same ID as the ID used when creating Application OAuth Token documents within the WebSecurityStore.nsf application. Anonymous must not have access to WebSecurityStore.nsf. All users who are to use the XPages Social Enabler functionality need to have author access to the WebSecurityStore application, this is so that their user keys can be stored in the database. It is worth noting that the process of signing the database is not always immediate.

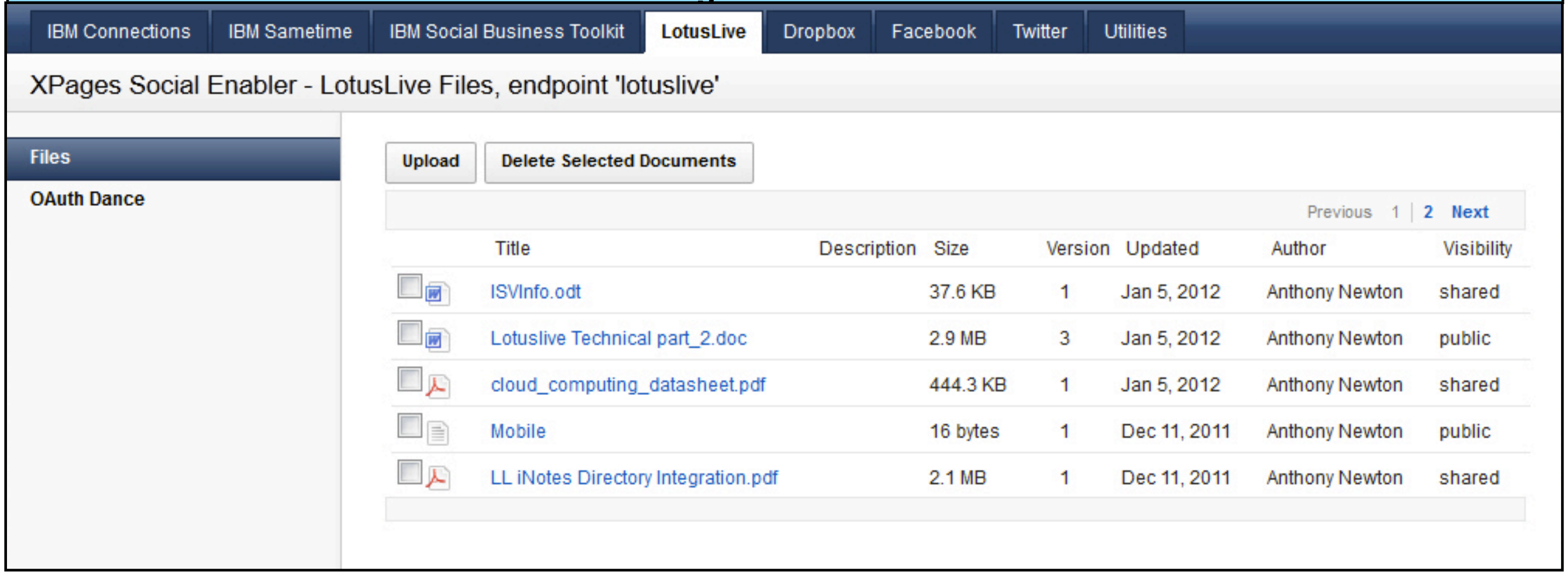
What are the Steps Required to Register an OAuth Application?

There are a number of common steps required when registering any OAuth application within the XPages Social Enabler:
• The system administrator must obtain OAuth credential from the online service provider. The credentials must include:
OAuth Consumer Key • OAuth Consumer Secret • OAuth Request URL • OAuth Authorization URL • OAuth Token URL
• The OAuth credentials must be entered in the WebSecurityStore application by the user who signed the application consuming this information (for instance XPagesSBT.nsf).
• Navigate to http://myserver.myco.com/WebSecurityStore.nsf
• In the Application Keys section click the Add Token button.
• The user must enter appropriate information in all of the fields in the Application Token dialog



• In the case of the XPages Social Enabler sample application the Application Id is XPagesSBT across all services. The Application Id field within the Application Token definition must correspond exactly to the appId property which is defined within the Endpoint definition (in faces-config.xml) for the service being registered.
The Service Name field must correspond to the serviceName property defined within the Endpoint definition (in faces-config.xml) for the service being registered.
• The Consumer key and Consumer secret are OAuth credentials which are given to the user when registering an Application within an OAuth provider (online service).
• The online API documentation for an online service should provide the remaining information that is required to register an Application Token within WebSecurityStore.nsf.

Connecting to External Services



XPages Social Enabler - LotusLive Files, endpoint 'lotuslive'

Title	Description	Size	Version	Updated	Author	Visibility
ISVInfo.odt		37.6 KB	1	Jan 5, 2012	Anthony Newton	shared
Lotuslive Technical part_2.doc		2.9 MB	3	Jan 5, 2012	Anthony Newton	public
cloud_computing_datasheet.pdf		444.3 KB	1	Jan 5, 2012	Anthony Newton	shared
Mobile		16 bytes	1	Dec 11, 2011	Anthony Newton	public
LL iNotes Directory Integration.pdf		2.1 MB	1	Dec 11, 2011	Anthony Newton	shared

IBM Connections and IBM Sametime

IBM Connections and Sametime are unique amongst the services configured in this document in that it is likely that they are the only services that reside within your firewall. IBM Connections and Sametime generally use Basic authentication so the setup is slightly different to the other (OAuth) services. IBM Connections and Sametime are configured through the Endpoint definition in faces-config.xml. They are configured by default to work with the IBM Connections server hosted on Lotus Greenhouse. In order to register with Lotus Greenhouse go to https://greenhouse.lotus.com/gh_next/lotusgreenhouserequests.nsf/MainDocumentSelf?openForm Once registered with Lotus Greenhouse and IBM Sametime servers you are free to use the IBM Connections and Sametime integration in the XPages Social Enabler sample application.
If you would like to use this functionality with your internal IBM Connections or Sametime server you will need to make a copy the IBM Connections Endpoint and/or the Sametime Endpoint in the faces-config.xml of the XPages Social Enabler sample application. To modify faces.config.xml perform the following steps:

1. Search for greenhouseBasic for IBM Connections and greenhouseSametime for IBM Sametime.
2. Copy/paste the whole endpoint definition element.
3. Change the value of managed-bean-name property to a value such that the endpoint will have a unique name (such as “myconnections” or "mysametime").
4. Replace the value of the url managed property with the URL location of your IBM Connections server or IBM Sametime server.
5. Save and close faces-config.xml.

6. Using the Resources Navigator view in Domino Designer open xsp.properties (located in the WebConent/WEB-INF folder within the application [XPagesSBT.nsf]). Modify xsp.properties to define which Connections Endpoint bean to use; e.g.: extlib.endpoint.connections=myconnections and also which Sametime Endpoint to use; e.g: extlib.endpoint.sametime=mysametime

7. Save and close xsp.properties.

8. On future attempts to access IBM Connections data or IBM Sametime it will be your company's IBM Connections and/or IBM Sametime server that will be accessed.

IBM Social Business Toolkit

URL	https://greenhouse.lotus.com/vulcan/security/provider/register.jsp?serviceProvider=vulcanToolkit
Application ID	XPagesSBT
Service Name	Greenhouse
Consumer Key	*<Unique to you>*
Key Type	HMAC-SHA1
Cons. Secret	*<Unique to you>*
Request Token	https://greenhouse.lotus.com:443/vulcan/security/provider/requestToken
Auth. URL	https://greenhouse.lotus.com:443/vulcan/security/provider/authorize
Access Token	https://greenhouse.lotus.com:443/vulcan/security/provider/accessToken

LotusLive

URL	https://apps.lotuslive.com/manage/account/isv
Application ID	XPagesSBT
Service Name:	LotusLive
Consumer Key	*<Unique to you>*
Key Type	PLAINTEXT
Cons. Secret	*<Unique to you>*
Request Token	https://apps.lotuslive.com/manage/oauth/getRequestToken
Auth. URL	https://apps.lotuslive.com/manage/oauth/authorizeToken
Access Token	https://apps.lotuslive.com/manage/oauth/getAccessToken

Twitter

URL	https://dev.twitter.com/apps/new
Application ID	XPagesSBT
Service Name	Twitter
Consumer Key	*<Unique to you>*
Key Type	HMAC-SHA1
Cons. Secret	*<Unique to you>*
Request Token	https://api.twitter.com/oauth/request_token
Auth. URL	https://api.twitter.com/oauth/authorize
Access Token	https://api.twitter.com/oauth/access_token

Dropbox

URL	https://dropbox.com/developers/apps
Application ID	XPagesSBT
Service Name	Dropbox
Consumer Key	*<Unique to you>*
Key Type	HMAC-SHA1
Cons. Secret	*<Unique to you>*
Request Token	https://api.dropbox.com/1/oauth/request_token
Auth. URL	https://api.dropbox.com/1/oauth/authorize
Access Token	https://api.dropbox.com/1/oauth/access_token

Facebook

URL	https://developers.facebook.com/apps
Application ID	XPagesSBT
Service Name	Facebook
Consumer Key	*<Unique to you>*
Key Type	HMAC-SHA1
Cons. Secret	*<Unique to you>*
Request Token	https://graph.facebook.com/oauth/request_token
Auth. URL	https://graph.facebook.com/oauth/authorize
Access Token	https://graph.facebook.com/oauth/access_token

Utilities

The XPages Social Enabler provides a number of XPages artifacts (such as data sources), utilities and new @Functions to the XPages Developer. Example driven unit tests of these utilities and @Functions are provided in the Utilities tab of the XPages Social Enabler sample application. Examples of how to use some of these utilities and @Functions are given on the final page of this document. The Utilities tab also provides a dump of the information stored in the userBean managed bean. For the adventurous, exploration of the userBean's functionality is a must, it provides a huge amount of flexibility to the XPages developer (and has many performance benefits also). The userBean is defined as a managed bean within the 'core' extension library, some interesting examples of its usage can be found be searching for "userBean" within the XPages Social Enabler sample application.

What is a Client Service?

An integral part of the XPages Social Enabler architecture is the ability to execute requests against a REST service and process the responses to those requests easily.
A ClientService is an object that has the ability to perform HTTP requests based on a URL, a set of parameters and options. For the most part each service supported by the XPage Social Enabler has a corresponding dedicated ClientService implementation:
ClientService

- **ActivityStreamsService**
- **AuthenticationService**
- **ConnectionsService**
- **DropboxService**
- **GenericService**
- **LotusLiveService**
- **TwitterService**

These classes can be called from within Server Side JavaScript. This model has far reaching implications, as it is possible to call services from these providers even if support for the service is not already built into the XPages Social Enabler application.

How does the Web Security Store Work?

The WebSecurityStore is a Domino Application (nsf) that is used by the XPages Social Enabler to store OAuth authentication information, and also user tokens. The WebSecurityStore is configured such that only the document author may read the user tokens, and only the application developer (signer of the design elements) may read the OAuth credential information. It is important to note that the same user ID should be used to sign the Design of the XPages Social Enabler sample application, and to enter the OAuth Application token credentials. During a typical request to a REST service, from the XPages Social Enabler application, the WebSecurityStore.nsf is queried for a user token for the logged in user. If one is not found the user credentials are requested and processed (through one of the supported authentication mechanisms such as OAuth), during this time the user logs into the REST service and a token is granted to the user which is stored in the WebSecurityStore.nsf and supplied for future requests.

Getting a User's Profile Image URL from IBM Connections

```
public String getPeopleImageUrl(String unid) {
   String url = null;
   Endpoint endpoint = (Endpoint) JSFUtil.getVariable("connections");
   GenericService svc = new GenericService(endpoint, "/profiles/atom/profile.do");
   HashMap<String, String> parameters =  new HashMap<String, String>();
   parameters.put("userid", unid);
      try {
         Object feed = svc.get(parameters, "xml");
         XmlNavigator navigator = new XmlNavigator((Document) feed);
         DataNavigator entry = navigator.get("feed").get("entry");
         DataNavigator links = entry.get("link");
         url = links.selectEq("@type", "image").stringValue("@href");
          }
          catch (ClientServicesException e) {
            e.printStackTrace();
          }
          return url;
   }
```

Note: This example is being used to show the coding principle. A better approach for this particular case would be to use the userBean to get the photo.

What is an Endpoint?

The XPages Social Enabler encapsulates each service into what is referred to as an Endpoint. The type of Endpoint used determines which form of authentication is to be used to authenticate with the service. The Endpoints are defined as managed beans, configured through faces-config.xml within the XPages Social Enabler application. The Endpoint architecture is sufficiently flexible that the Endpoint definitions can be externalized to an Endpoint store application.
Each service requires a limited number of steps to configure. All of the services support one form of authentication or another. There are two primary forms of authentication currently supported, Basic authentication and OAuth 1.0 authentication.

Sample Usage of Files Data Source with LotusLive Files

```
<xp:this.data>
   <xe:fileServiceData var="fileServiceData1">
        <xe:this.serviceType>
             <xe:lotusLiveFileData></xe:lotusLiveFileData>
        </xe:this.serviceType>
        <xe:this.urlParameters>
            <xe:urlParameter name="subscriberId">
                 <xe:this.value>
                     <![CDATA[${javascript:return userBean.lotusLiveSubscriberId;}]]>
                 </xe:this.value>
             </xe:urlParameter>
          </xe:this.urlParameters>
   </xe:fileServiceData>
</xp:this.data>
```

Note: If using this data source in your application be sure to enable the LotusLive People Data Provider in xsp.properties of the application:
extlib.people.provider=lotuslive

Getting User Information from LotusLive

```
if(!
@Endpoint('lotuslive').getOAuthProvider().acquireTok
en()) {
    viewScope.text = "Please login in order to use this
service"
    return;
  }
var svc = new
sbt.LotusLiveService(@Endpoint('lotuslive'),"manage/
oauth/getUserIdentity")
var msg = svc.get(null,"text");
viewScope.text = msg
```