HCL Leap and Enterprise Directory (LDAP)

Introduction

The corporate enterprise directory, or LDAP, is the most common target of integration with HCL Leap. HCL has provided a transport and a few sample service descriptions that can be used to perform this integration.

A Transport is an OSGI bundle, written in Java, that implements HCL Leap extension points. Service Descriptions are XML files that define the inputs and outputs that are used by the transport. To discover more about implementing your own Transports *reference our documentation.*

When the transport and associated service description XML files are deployed, then application designers can setup their applications to auto-populate user information. This document describes how to configure and deploy all the artifacts necessary to enable applications to communicate with LDAP.

Transport Properties

The transport defines properties that must be referenced in the service description XML files. These properties define how the transport behaves. All the properties are listed in the tables below.

Input Parameters

url	Required	The URL to the LDAP
basedn	Required	The basedn of the LDAP query
filter	Required	The LDAP filter to use for the search. For example: <constant> <id>filter</id> <value>(&amp; (objectClass=person)(managerSerialNumber={0}*)(managerCountryCode={1}*))</value> </constant>
filter-arg-x	Required	For every search parameter in the filter this defines what will be used for the search. For example: <mapping target="transport:filter-arg-0" source="parameter:searchSerialNumber" /> <mapping target="transport:filter-arg-1" source="parameter:searchCountryCode" /> These mapping statements will substitute the "0" in the LDAP filter with the value of the searchSerialNumber value passed to the service and "1" with the value of the searchCountryCode value passed to the service.

result-attributes	Required	Comma separated list of the LDAP attributes to return in the search results. If unspecified then all attributes are returned.
bind-user	Required	The username required to connect to the LDAP. Required if using this transport with WAS Liberty.
bind-password	Required	The password required to connect to the LDAP. Required if using this transport with WAS Liberty.
result-count	Optional	Limits the number of records returned from LDAP. If unspecified the default is 50.
sub-filter-*x*	Optional	A secondary LDAP query to execute for each record of the first result set.
sub-filter-arg-*x*	Optional	Value should start with "result-attribute:" to reference an attribute from the first result set, this will be used to replace the integers in the filter. `<constant> <!-- OPTIONAL: The filter variables must be specified in this format "result-attribute:<name-of-ldap-attribute>" the "result-attribute" value tells the transport to pull the attribute from the parent result set. --> <id>sub-filter-arg-0</id> <value>result-attribute:hrOrganizationCode</value> </constant>`
sub-result-attributes	Optional	Comma separated list of the LDAP attributes to return in the search results. If unspecified then all attributes are returned.
sub-result-count	Optional	Limits the number of records returned from LDAP. If unspecified the default is 50.
return-multiple-attr	Optional	If true, then all attributes will be returned including those with multiple values. If false then only the first attribute will be returned. Valid values are true or false. Default is false.

Output Parameters

resultXML	The XML containing the LDAP search results. `<searchresults> <searchresult> <ldapattrribute1/> <ldapattrribute2/> <subsearchresults> <subsearchresult> <ldapattrribute1/> <ldapattrribute2/> </subsearchresult> </subsearchresults> </searchresult> </searchresults>`

```
<searchresults>
    <searchresult>
        <ldapattrribute1/>
        <ldapattrribute2/>
        <subsearchresults>
            <subsearchresult>
                <ldapattrribute1/>
                <ldapattrribute2/>
            </subsearchresult>
        </subsearchresults>
    </searchresult>
</searchresults>
```

	Note: "ldapattributex" will need to be replaced with the actual attributes in your LDAP.

Deploying the Transport

The transport is a .jar file that must be deployed into the file structure for HCL LEAP. All extensions are placed into the extensions directory. The default location is /opt/HCL/Leap/extensions or C:\HCL\Leap\extensions. You can discover more by referring to our documentation.

Once the jar file has been placed in the extensions directory you will see a message written to the log file indicating that it has been loaded and is ready for consumption:

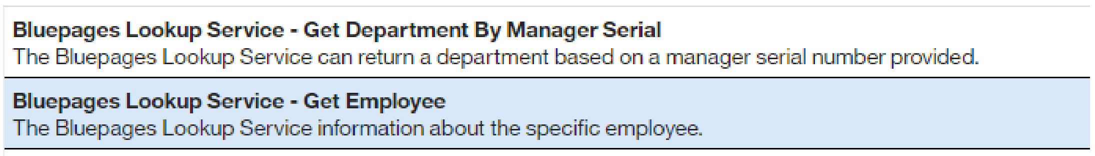
> *[6/11/18 10:07:14:550 PST] 0000007a DirectoryWatc I com.ibm.form.platform.service.framework.DirectoryWatcher processArtifact FSPDI5: The Bundle, <ExtensionName>.jar loaded successfully.*

Configuring the Service Description

Creating a service description is a one-time action that once completed, provides the function to all HCL Leap application designers. Three service descriptions have been provided as examples of the kind of functionality that can be provided with an integration with LDAP. This document will use the most common service as the example.

Setting up the Global Parameters

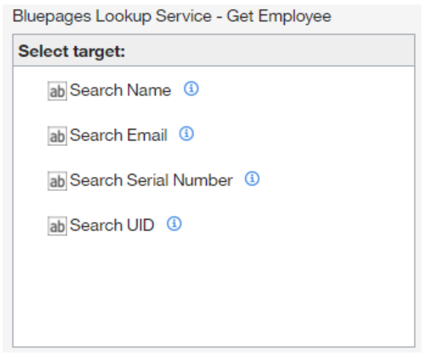
1. Open the LDAPLookup_GetEmployee.xml in a text editor. Do not use Notepad or Wordpad, instead use a proper editor like NotePad++, UtraEdit, etc.
2. The <id>, found on line 3, must be a unique value (not used by any other xml files deployed with Leap) as this is how the service is registered with Leap.
3. The <transportId>, on line 6, must be set to CustomLDAPServiceTransport
4. The <name>, on line 13, is the name of the service that will appear in the service configuration dialog within Leap, therefore insure it is something that your users can recognize.
5. The <description>, on line 14, will appear in the service configuration dialog and should provide a clear description of what the service provides.


Bluepages Lookup Service - Get Department By Manager Serial
The Bluepages Lookup Service can return a department based on a manager serial number provided.
Bluepages Lookup Service - Get Employee
The Bluepages Lookup Service information about the specific employee.

Setting up the Inbound Parameters

1. The <inbound> element is where all the parameters are defined that will be passed to the transport. In this example, there are three: searchName, searchEmail and searchUID. Each parameter follows the same format

```
<parameter>
        <id>searchName</id>
        <name xml:lang="en">Search Name</name>
        <description xml:lang="en">The employee name to search
for.</description>
        <mandatory>false</mandatory>
        <type>STRING</type>
</parameter>
```


Bluepages Lookup Service - Get Employee

Select target:

Search Name

Search Email

Search Serial Number

Search UID

Configure each parameter.

- The <id> must be unique, and will be referenced in the inbound <servicemapping> section
- The <name> and <description> will appear in the dialog within Leap
- If <mandatory> is true, then the service cannot be saved without specifying this value.
- <type> defines the data being provided. The other types are documented, but in this case, all will be STRING.

Setting up the Constants

1. Define the URL to the LDAP server on line 45.
2. Define the baseDN that will be used for the LDAP queries on line 50.
3. Define the filter that will be used to define the LDAP queries on line 55.

```
<value>(&amp;(objectClass=person)(cn={0}*)(emailAddress={1}*)(uid={2}*))</value>
```

The 0,1 and 2 shown in this filter reference the input parameters defined at the start of the service description. In this case 0 is searchName, 1 is searchEmail and 2 is searchUID. When this filter is executed the numbers will be dynamically replaced with the value passed in from the Leap application calling the service.

4. Define all the LDAP attributes, on line 61, that you want to be returned from the LDAP query. This is a comma-separated list. **Note:** For every attribute listed here you must create a corresponding parameter definition in the <outbound> section of the XML file.
5. Define all the inbound mappings, lines 70-81. The inbound mappings connect the parameters and constants defined in the XML file to the transport parameters. The only mappings that you may have to modify are the filter-arg-x as your search parameters may differ.

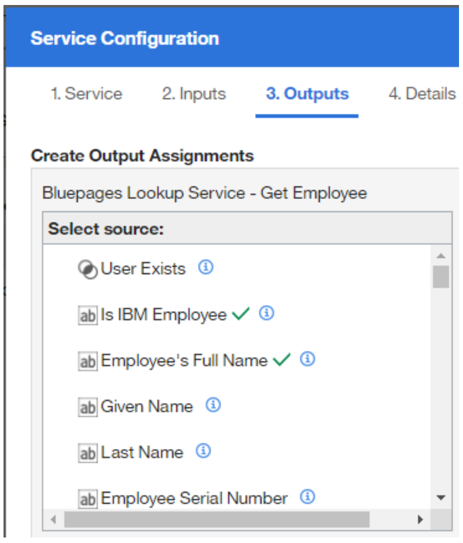
Setting up the Outbound Parameters

You must define an outbound parameter for every attribute listed in the result-attributes. The outbound parameters all follow the same structure.

```
<parameter>
        <id>cn</id>
        <name xml:lang="en">Employee's Full Name</name>
        <description xml:lang="en"></description>
        <type>STRING</type>
</parameter>
```

Configure each parameter.

- The <id> must be unique, and will be referenced in the outbound <servicemapping> section.
- The <name> and <description> appear in the service dialog
- The type defines what is being returned.



All the parameters defined within the <outbound> section will appear in the service dialog on the Outputs tab.

For every outbound parameter, there must be a corresponding <mapping>. To properly configure the service mappings, you must understand what is being returned from the LDAP query. In this case, the LDAP will return XML that looks like:

```
<searchresults>
  <searchresult>
    <ldapattrribute1/>
    <ldapattrribute2/>
    <subsearchresults>
      <subsearchresult>
        <ldapattrribute1/>
        <ldapattrribute2/>
      </subsearchresult>
    </subsearchresults>
  </searchresult>
</searchresults>
```

Now that we know the structure we can create the mapping that places individual elements from the result and assigns them to the corresponding output parameters. Take the following mapping as an example:

```
<mapping source="transport:resultXML" target="parameter:cn"
sourceRef="string(searchresults/searchresult[1]/cn)" sourceType="xml" />
```

This mapping assigns the “cn” value from the xml result to the “cn” parameter.

The sourceRef attribute enables the use of XPath to manipulate the XML that is returned from the service. This allows us to pull content out of the XML structure and assign it directly to an outbound parameter.

Once all the mappings are defined you are ready to deploy the service and start using it.

Deploying the Service Description

The XML files must be deployed to the ServiceCatalog/1 directory.

Once the xml file has been placed in the ServiceCatalog/1 directory you will see a message written to the log file indicating that is has been loaded and is ready for consumption:

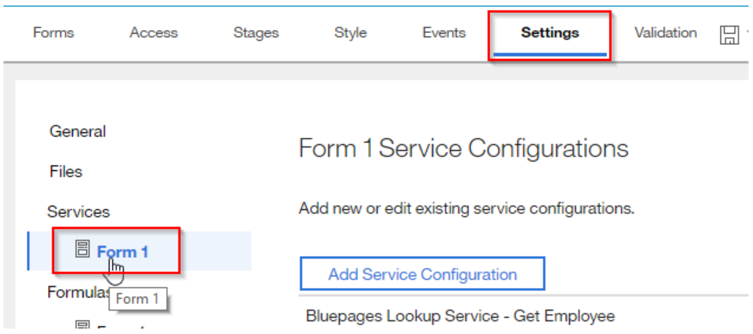
> *[6/11/18 10:07:45:904 PST] 0000008a ServiceFileCa I com.ibm.form.nitro.service.services.impl.ServiceFileCatalogDirectory add Registering service description in C:\HCL\Leap\ServiceCatalog\1\<ServiceFileName>.xml.*

If you do not see this message within 60 seconds, then starting and stopping the Leap application (not WebSphere) will also cause the file to be picked up. **Note:** If there is a syntax error in the XML file then you may also see a message in the log and the service will not appear in Leap.

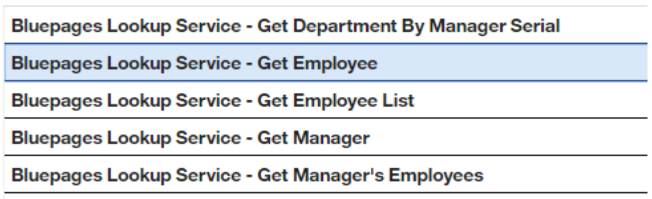
Using a Service within HCL Leap

Now that the service is deployed you can begin using it within your applications.

1. Edit an application.
2. Click on the Settings tab.
3. Under Services, on the left-side, click one of the forms.

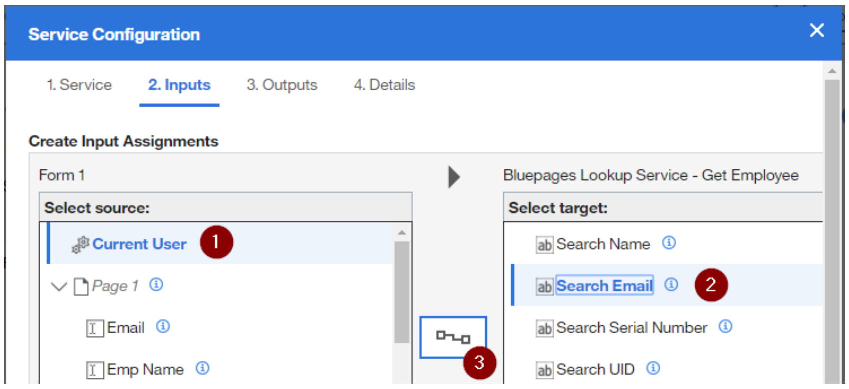

Forms
Access
Stages
Style
Events
Settings
Validation
General
Files
Services
Form 1
Formulas
Form 1
Form 1 Service Configurations
Add new or edit existing service configurations.
Add Service Configuration
Bluepages Lookup Service - Get Employee

4. Click Add Service Configuration.
5. Select "Or, select a service"
6. Set the Service Catalog to General.
7. Locate and click on the service in the list below



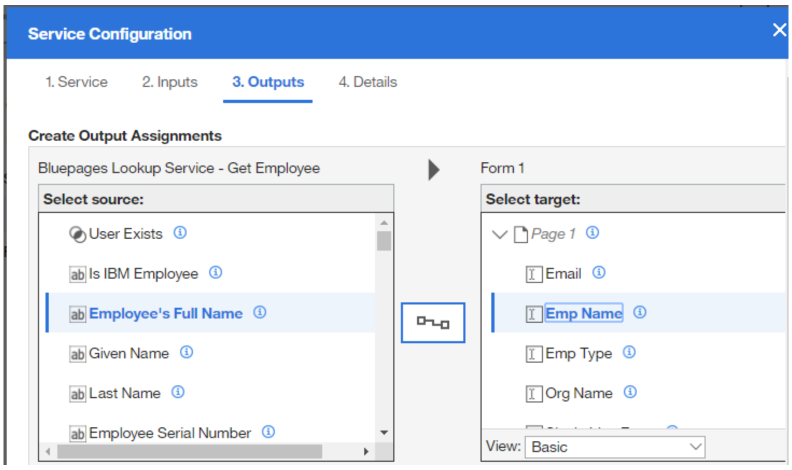
Bluepages Lookup Service - Get Department By Manager Serial
Bluepages Lookup Service - Get Employee
Bluepages Lookup Service - Get Employee List
Bluepages Lookup Service - Get Manager
Bluepages Lookup Service - Get Manager's Employees

8. Click Next.
9. Decide how you want to perform the employee lookup. Will it be based on the current user? Or will the user of the form be entering some key information (like email address or uid) to then load details about that person? In this example, I want to automatically load the name of the current user. Click Current User and then connect it to the appropriate search parameter. In my implementation of Leap user's login with their email address, therefore I would connect the Current User with the Search Email.

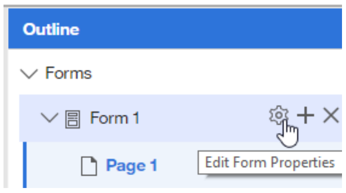


10. Click the Outputs tab across the top or click Next (at the bottom).

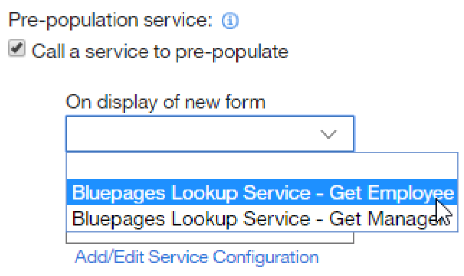
11. Connect the LDAP result values (shown on the left) to the fields you have defined in your form (shown on the right).



12. Repeat the action of connecting the fields until all the desired fields are connected.
13. Click Next.
14. Give the service a meaningful ID and Description and click OK.
15. Now that the service is created we need to call it. In this case, we will call the service when the form is just launched by a new user. In the Outline View click on the gear icon for the Form.



16. Click on Advanced.
17. Click Call a service to pre-populate
18. In the "On display of New form" dropdown, select the service.


Pre-population service:

☑ Call a service to pre-populate

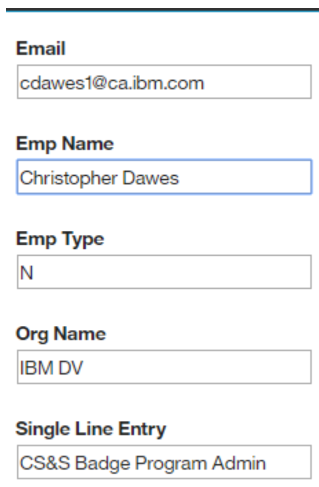
On display of new form

Bluepages Lookup Service - Get Employee

Bluepages Lookup Service - Get Manager

Add/Edit Service Configuration

19. Click OK.
20. Save and preview the form. When the form opens, you should notice that your information is automatically loaded into your form.



Email	cdawes1@ca.ibm.com
Emp Name	Christopher Dawes
Emp Type	N
Org Name	IBM DV
Single Line Entry	CS&S Badge Program Admin