Busy Administrator's guide

What can Searchy do for you

Searchy's purpose is not very easy to understand. Searchy's main task is to give a framework in which federate different information systems. For example, you may have several data bases with different structures, Searchy lets you perceive those data bases as an homogeneus entity with an uniform structure of your choice.

However, in a higher level of complexity, you may have not only data bases, but also directories, semi-structures information sources like web pages, mailing lists and many more information systems. You can use Searchy to integrate all them, the user may iterate with all those systems in a transparent way, dealing with only one system, with one interface but in fact he is dealing with several, distributed systems.

This scenario may be even more complex. You may need not only integrate you information systems, but also integrate your systems with some partner of a different department or even of another organization. Searchy may help you in this task. And it is as simple as installing a small piece of software on the top of your systems, this guide will help you in this -we wish- easy task.

Searchy final user may be typically an application, it may be a web application, a heavy desktop program or a specific software that uses Searchy as information source. You, as administrator, only have to deal with your agent editing its XML config file, regardless of who uses it.

Figure 1 may help you to understand how Searchy looks like.

Searchy overview

The main unit in Searchy is the Agent, indeed, you may be reading this lines just because you need to set up one. All you need to know is some theoretical stuff and the format of the XML config file.

Let's begin with the begin. Any agent has three part, as you can see in figure 2.

- *Core.* It is the common part for any agent.
- *Provider.* It is a sort of "plug-in" that extends Searchy to support new data sources, it deals with the particulars.

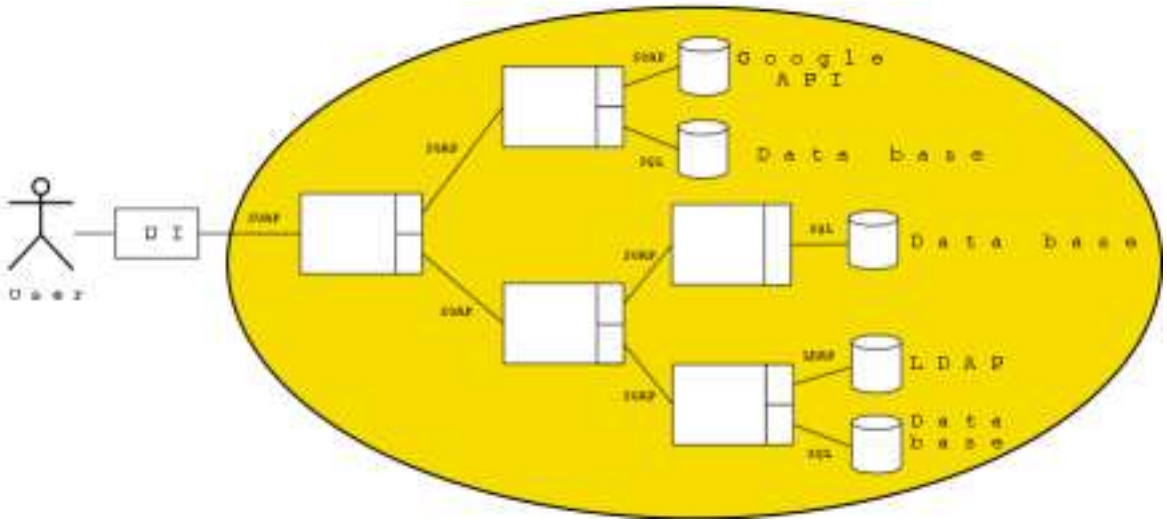

Figure 1: Searchy system

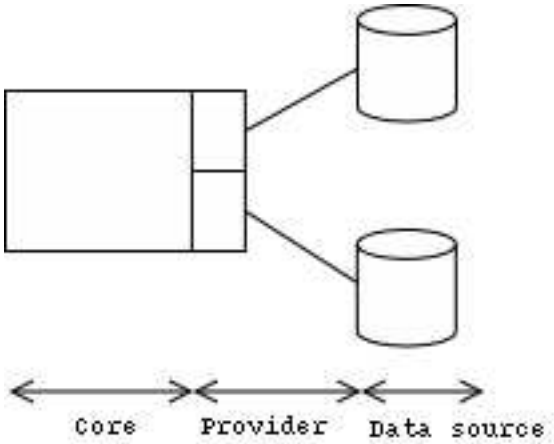

Figure 2: Searchy agent architecture

- *Data source.* It is not part of Searchy, but defining it may be useful. It is the information system that Searchy federates. it may be a relational data base, a LDAP directory, Google, any thing that bring some data may be used as data source.

There is another critical concept that you must know, the mapping. You have to define some shared vocabulary between all the agents, it will be typically Dublin Core or vCard. There must be some mechanism to translate the local information format into this shared vocabulary. It is done by the mapping. A mapping says, for example, that field SONGNAME in a relational data base have to be mapped into field title of Dublin Core. Don't be afraid (yet), we will deep this later.

Warning: defining mapping is the most difficult task you will have to deal with.

Any Searchy deployment should define a high level policy describing the vocabulary that agents should use, and some mandatory terms that any agent has to understand. In particular, we encourage using Dublin Core, but any other vocabulary is supported.

Common configuration

Well, it's time to put hands on work. We are going to describe step-by-step all you have to do to set up an agent, however, those steps depends on the providers that your agent needs. In this section we explain the configuration common to all agents. Let $SEARCHY be the directory where Searchy is located.

1. Download Searchy. It is available in Searchy's web site at http://jsearchy.sourceforge.net and you can download a zip file or a tgz file.

2. Decompress Searchy in $SEARCHY. This step depends of your operating system.

   Under UNIX:

   (a) *unzip searchy-version.zip* or *tar -zxf searchy-version.zip*

   (b) *chmod u+x $SEARCHY/bin/searchy.sh*

   Under Windows:

   Use winzip, winrar or your preferred compression software

3. Edit $SEARCHY/agent.xml with any text editor, it is a simple XML file. Searchy is distributed with a verbosely commented example config file, we recommend read this text and edit agent.xml at the same time, they are complementary. There are evident tags, but we have to point out a couple of things.

(a) Searchy's core configuration is kept in the block with tag *transport*, there must be just one transport section and it configures Searchy's core.

(b) Attributes *name* and *target* in *transport* tag are critical and you have to edit them.

   i. *Name* attribute sets the agent's name, you should use a univo-queunivocal name, it may e quite helpful in debugging tasks.

   ii. Target attribute sets the providers used by this agent, there may be several providers separated by a comma. You can find more information about this important tag in the provider section.

(c) Tag *max-connections* defines the max number of clients that the agent may attend at the same time. A big number of clients implies a big amount of resources, you have to set a trade of.

(d) Tag *log-config* defines the file that stores logger configuration. You may find useful using file log4jdebug.conf instead of file log4j.conf to ease debugging of the configuration.

(e) Tag *vocabulary* define vocabularies available for the agent, it must point to a RDF or OWL file. By default, Dublin Core and vCard vocabularies are included.

Note: Configuration file is case sensitive

Provider configuration

Provider are specialized depending of the data source it address, so configuration differs from one sort of providers or others. If you deploy a SQL provider you will need some fields but if you deploy a LDAP provider you will need another ones. However, all this stuff is stored under a section with a *provider* tag and you should anyway edit its three attributes:

1. Edit attribute *Name*. It is provider's identification within the configuration file, when you have to refer to this providers, you must use this name. For example, *target* attribute in transport section uses this name.

2. Edit attribute *Type*. It specify the sort of provider it is and there are some fixed values it may take: SQL, LDAP, Google, harvest and Searchy.

3. Edit attribute map. It identifies the mapping that the provider uses; in this way several providers may share a mapping, making administration easier for you.

Note: There may be several different provider blocks, but they only will be considered if they are pointed by the target attribute in transport section

Now, you are prepared to jump directly to the provider you are interested in.

Google provider configuration

Google provider uses Google API to perform Internet wide queries in the same way you use Google. This is the easiest provider developed for Searchy, there are only one step you must follow.

1. Set your Google key in tag *key*. If you do not have a key, you may get one for free in http://www.google.com/apis/.

Mapping in a Google provider may use fields *url*, *title*, *snippet*, *summary* and *hostName*.

Searchy provider configuration

Using this provider you can connect two Searchy providers. You should use this provider with careful: no loops should be created. Configuration is quite easy.

1. Set agent's URL in the *url* tag. Follow format http://host:port

In almost all situation you will use a trivial map with this provider, just for simplicity :).

SQL provider configuration

You can access any SQL database server, the only limitation is the existence of a valid JDBC driver. By default, a MySQL and postgres drivers are provided with Searchy, to add new drivers you only have to copy the far file into $SEARCHY/lib.

1. Set the database's URL in *url* tag. Follow format jdbc:server://host/table
2. Set the JDBC driver in driver tag.
3. Set user and password in user and password tags.

Database field selected by the SQL query are available to map responses with the table appended. For example, if you submit "SELECT books.title, author FROM creators WHERE 1", fields books.title and creators may be used in the response mapping.

LDAP provider configuration

This one requires a little bit of job (but not too much ;)).

1. Set directory's URL in *url* tag. Follow format ldap://host:port
2. Set the LDAP context in *context* tag.
3. You may want to limit the response size, if so, edit *answerLimit* tag.

4. Select query scope in scope tag. It make take values subtree, onelevel and object.

5. If you want to disable referrals set *referral* tag to ignore.

Mappings

Now you should understand how queries are processed by Searchy. Let's study some theory. Agents receive queries with two parameters: the query string and the term query refers to. Terms are actually URIs (it is like URLs), so, for example, term http://purl.org/dc/elements/1.1/title refers to concept "title". You may be wondering where that URL comes from, well, don't panic, it is defined in the vocabulary file.

Agents will receive, for example, a query like "Hamlet" referred to http://purl.org/dc/elements/1.1/title, so, it ask for resources whose title is "Hamlet". If the agent access to a SQL server, it must generate an SQL sentence with the query, it may be, for example

```
SELECT bookTitle, author, description FROM books WHERE (title = 'Hamlet')
```

If you use any other information system, the process is the same. On the other hand, agents have to do the inverse job, i.e., translate database response into agent's vocabulary.

```
books.bookTitle -> http://purl.org/dc/elements/1.1/title
books.author -> http://purl.org/dc/elements/1.1/author
bools.description -> http://purl.org/dc/elements/1.1/description
```

How this translation is done is your task, we sorry, we cannot help you here. You have to choose which terms in you information system suits better in the terms of any given vocabulary. Well, It's time to come back to the computer.

You can set different mappings using the map tag in config file. If you understood the theory, it is not too hard, you just have to follow there steps:

1. Select which terms will understand the agent. It should be a shared criteria between all the agents involved in the deployment.

   Note: You can list available terms in a vocabulary by executing "*$SEARCHY/bin/searchy.sh vocabulary [file]*".

2. Set mapping's name in name attribute.

3. Set vocabularies that this mapping is going to use in *vocabulary* tag, you may set several vocabularies just placing a comma between them.Vocabulary name is the one used in transport section.

4. Set mapping's type, it is the method used to map terms. Most likely you will want to use a *filter* mapping unless if you use the map with a Searchy agent, in such a situation consider using better a *trivial* mapping.

5. Set query mapping rules. XML tag contains the term that rule applies to. Query rules has attribute filter with value "query". We encourage using XML name spaces. Let prefix dc be http://purl.org/dc/elements/1.1/, then query used in our example would be

```
<dc:title filter=''query''>SELECT bookTitle, author, description FROM books WHERE (ti
```

   String %query% is substituted by the real query, "Hamlet" in our example.

6. Set response mapping rules, they have *filter* attribute with value "response". Each agent term shall have an entry in the map section, tag value contains the term that rule refers to. It is better understood with an example.

```
<dc:title filter=''response''>%bookTitle%</dc:title>
<dc:creator filter=''response''>%author%</dc:creator>
<dc:description filter=''response''>%description%</dc:description>
```

   These rules substitutes %title% by the title field given by the provider. It may be, for example, the title field in a relational data base. You may use more complex chains, for example

```
<dc:creator filter=''response''>Author is: %name% %surname%</dc:creator>
```

7. There is a special tag in any mapping: *URL*, works in the same way but it has a special meaning identifing the resource. It must be an URL and different resources must have different meanings. In general you won't need to change its default valour (%URL%). Our advice, if it works, do not touch it, but if client begins to miss resources, then it's time to change this field.

Starting Searchy

Just use one of the following commands

```
(UNIX)    $SEARCHY/bin/searchy.sh start
(Windows) $SEARCHY/bin/searchy.bat start
```

Testing Searchy

1. Test if the agent is up by running this command

```
(UNIX)    $SEARCHY/bin/searchy.sh test
(WINDOWS) $SEARCHY/bin/searchy.bat test
```

Your agent's name should be printed on the screen

2. Test doing some query, it is quite useful to debug mappings.

```
(UNIX)    $SEARCHY/bin/searchy.sh client [-a agent] -e element -q query
(WINDOWS) $SEARCHY/bin/searchy.bat client [-a agent] -e element -q query
```

For example, if you want to query resources whose title is "Hamlet" on an agent in localhost under UNIX (for windows is quite similar), type:

```
$SEARCHY/bin/searchy.sh -e http://purl.org/dc/elements/1.1/title -q Hamlet
```

and in the agent is placed on www.example.com:33333

```
$SEARCHY/bin/searchy.sh -a www.example.com:33333 -e http://purl.org/dc/elements/1.
```

Point out that terms are they-self URLs, you can check available terms by executing, for example,

```
$SEARCHY/bin/searchy vocabulary $SEARCHY/conf/vocabulary/dcmes.rdfs
```

Dublin Core Vocabulary

Dublin Core is a vocabulary suites fine to describe resources of any type: documents, pictures, music, etc. The most significant terms are briefly described here, for a complete reference execute "$SEARCHY/bin/searchy vocabulary $SEARCHY/conf/vocabulary/dces11.rdfs"

http://purl.org/dc/elements/1.1/title A name given to the resource. Typically, Title will be a name by which the resource is formally known.

http://purl.org/dc/elements/1.1/creator An entity primarily responsible for making the content of the resource. Examples of Creator include a person, an organization, or a service. Typically, the name of a Creator should be used to indicate the entity.

http://purl.org/dc/elements/1.1/subject A topic of the content of the resource. Typically, Subject will be expressed as keywords, key phrases or classification codes that describe a topic of the resource. Recommended best practise is to select a value from a controlled vocabulary or formal classification scheme.

http://purl.org/dc/elements/1.1/description An account of the content of the resource. Examples of Description include, but is not limited to: an abstract, table of contents, reference to a graphical representation of content or a free-text account of the content.

http://purl.org/dc/elements/1.1/identifier An unambiguous reference to the resource within a given context. Recommended best practice is to identify the resource by means of a string or number conforming to a formal identification system. Formal identification systems include but are not limited to the Uniform Resource Identifier (URI) (including the Uniform Resource Locator (URL)), the Digital Object Identifier (DOI) and the International Standard Book Number (ISBN).

vCard Vocabulary

This vocabulary may be used to describe people. A complete recerence may be found executing "$SEARCHY/bin/searchy vocabulary $SEARCHY/conf/vocabulary/vCard.rdfs"

http://www.w3.org/2001/vcard-rdf/3.0#ROLE To specify information concerning the role, occupation, or business category of the object the vCard represents.

http://www.w3.org/2001/vcard-rdf/3.0#LABEL To specify the formatted text corresponding to delivery address of the object the vCard represents.

http://www.w3.org/2001/vcard-rdf/3.0#LABEL To specify the formatted text corresponding to delivery address of the object the vCard represents.

http://www.w3.org/2001/vcard-rdf/3.0#MAILER To specify the type of electronic mail software that is usedby the individual associated with the vCard.

http://www.w3.org/2001/vcard-rdf/3.0#ADR To specify the components of the delivery address for the vCard object.

http://www.w3.org/2001/vcard-rdf/3.0#TITLE To specify the job title, functional position or function of the object the vCard represents.

http://www.w3.org/2001/vcard-rdf/3.0#TEL To specify the telephone number for telephony communication with the object the vCard represents.

http://www.w3.org/2001/vcard-rdf/3.0#EMAIL To specify the electronic mail address for communication with the object the vCard represents.

http://www.w3.org/2001/vcard-rdf/3.0#ORG To specify the organizational name and units associated withthe vCard.