Pazpar2 - User's Guide and Reference

Sebastian Hammer, Adam Dickmeiss, Marc Cromme, Jakub Skoczen,
Mike Taylor, and Dennis Schafroth

COLLABORATORS

	TITLE : Pazpar2 - User's Guide and Reference	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Sebastian Hammer, Adam Dickmeiss, Marc Cromme, Jakub Skoczen, Mike Taylor, and Dennis Schafroth	October 1, 2021	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Abstract

Pazpar2 is a high-performance metasearch engine featuring merging, relevance ranking, record sorting, and faceted results. It is middleware: it has no user interface of its own, but can be configured and controlled by an XML-over-HTTP web-service to provide metasearching functionality behind any user interface.

This document is a guide and reference to Pazpar2 version 1.14.1.

Chapter 1

Introduction

1.1 What Pazpar2 is

Pazpar2 is a stand-alone metasearch engine with a web-service API, designed to be used either from a browser-based client (JavaScript, Flash, Java applet, etc.), from server-side code, or any combination of the two. Pazpar2 is a highly optimized client designed to search many resources in parallel. It implements record merging, relevance-ranking and sorting by arbitrary data content, and facet analysis for browsing purposes. It is designed to be data-model independent, and is capable of working with MARC, Dublin Core, or any other XML-structured response format -- XSLT is used to normalize and extract data from retrieval records for display and analysis. It can be used against any server which supports the Z39.50, SRU/SRW or Apache Solr protocol. Proprietary backend modules can function as connectors between these standard protocols and any non-standard API, including web-site scraping, to support a large number of other protocols.

Additional functionality such as user management and attractive displays are expected to be implemented by applications that use Pazpar2. Pazpar2 itself is user-interface independent. Its functionality is exposed through a simple XML-based web-service API, designed to be easy to use from an Ajax-enabled browser, Flash animation, Java applet, etc., or from a higher-level server-side language like PHP, Perl or Java. Because session information can be shared between browser-based logic and server-side scripting, there is tremendous flexibility in how you implement application-specific logic on top of Pazpar2.

Once you launch a search in Pazpar2, the operation continues behind the scenes. Pazpar2 connects to servers, carries out searches, and retrieves, de-duplicates, and stores results internally. Your application code may periodically inquire about the status of an ongoing operation, and ask to see records or result set facets. Results become available immediately, and it is easy to build end-user interfaces than feel extremely responsive, even when searching more than 100 servers concurrently.

Pazpar2 is designed to be highly configurable. Incoming records are normalized to XML/UTF-8, and then further normalized using XSLT to a simple internal representation that is suitable for analysis. By providing XSLT stylesheets for different kinds of result records, you can configure Pazpar2 to work against different kinds of information retrieval servers. Finally, metadata is extracted in a configurable way from this internal record, to support display, merging, ranking, result set facets, and sorting. Pazpar2 is not bound to a specific model of metadata, such as Dublin Core or MARC: by providing the right configuration, it can work with any combination of different kinds of data in support of many different applications.

Pazpar2 is designed to be efficient and scalable. You can set it up to search several hundred targets in parallel, or you can use it to support hundreds of concurrent users. It is implemented with the same attention to performance and economy that we use in our indexing engines, so that you can focus on building your application without worrying about the details of metasearch logic. You can devote all of your attention to usability and let Pazpar2 do what it does best -- metasearch.

Pazpar2 is our attempt to re-think the traditional paradigms for implementing and deploying metasearch logic, with an uncompromising approach to performance, and attempting to make maximum use of the capabilities of modern browsers. The demo user interface that accompanies the distribution is but one example. If you think of new ways of using Pazpar2, we hope you'll share them with us, and if we can provide assistance with regards to training, design, programming, integration with different backends, hosting, or support, please don't hesitate to contact us. If you'd like to see functionality in Pazpar2 that is not there today, please don't hesitate to contact us. It may already be in our development pipeline, or there might be a possibility for you to help out by sponsoring development time or code. Either way, get in touch and we will give you straight answers.

Enjoy!

Pazpar2 is covered by the GNU General Public License (GPL) version 2. See Appendix A for further information.

1.2 Connectors to non-standard databases

If you need to access commercial or open-access resources that don't support Z39.50 or SRU, one approach would be to use a tool like SimpleServer to build a gateway. An easier option is to use Index Data's MasterKey Connect service, which will expose virtually *any* resource through Z39.50/SRU, dead easy to integrate with Pazpar2. The service is hosted, so all you have to do is to let us know which resources you are interested in, and we operate the gateways, or Connectors for you for a low annual charge. Types of resources supported include commercial databases, free online resources, and even local resources; almost anything that can be accessed through a web-facing user interface can be accessed in this way. Contact info@indexdata.com for more information. See Section 3.8 for an example.

1.3 A note on the name Pazpar2

The name Pazpar2 derives from three sources. On one hand, it is Index Data's second major piece of software that does parallel searching of Z39.50 targets. On the other, it is a near-homophone of Passpartout, the ever-helpful servant in Jules Verne's novel Around the World in Eighty Days (who helpfully uses the language of his master). Finally, "passe par tout" means something like "passes through anything" in French -- on other words, a universal solution, or if you like a MasterKey.

Chapter 2

Installation

The Pazpar2 package includes documentation as well as the Pazpar2 server. The package also includes a simple user interface called "test1", which consists of a single HTML page and a single JavaScript file to illustrate the use of Pazpar2.

Pazpar2 depends on the following tools/libraries:

YAZ The popular Z39.50 toolkit for the C language. YAZ *must* be compiled with libxml2/libxslt support.

It is highly recommended that YAZ is also compiled with ICU support.

In order to compile Pazpar2, a C compiler which supports C99 or later is required.

2.1 Installation from source on Unix (including Linux, MacOS, etc.)

The latest source code for Pazpar2 is available from `http://ftp.indexdata.com/pub/pazpar2/`. Most Unix-based operating systems have the required tools available as binary packages. For example, if libxml2/libxslt libraries are already installed as development packages, use these.

Ensure that the development libraries and header files are available on your system before compiling Pazpar2. For installation of YAZ, refer to the Installation chapter of the YAZ manual at `http://www.indexdata.com/yaz/doc/installation.html`.

Once the dependencies are in place, Pazpar2 can be unpacked and installed as follows:

```
tar xzf pazpar2-VERSION.tar.gz
cd pazpar2-VERSION
./configure
make
sudo make install
```

The `make install` will install manpages as well as the Pazpar2 server, `pazpar2`, in PREFIX`/sbin`. By default, PREFIX is `/usr/local/` . This can be changed with configure option `--prefix`.

2.2 Installation from source on Windows

Pazpar2 can be built for Windows using Microsoft Visual Studio. The support files for building YAZ on Windows are located in the `win` directory. The compilation is performed using the `win/makefile` which is to be processed by the NMAKE utility part of Visual Studio.

Ensure that the development libraries and header files are available on your system before compiling Pazpar2. For installation of YAZ, refer to the Installation chapter of the YAZ manual at `http://www.indexdata.com/yaz/doc/installation.html`. It is easiest if YAZ and Pazpar2 are unpacked in the same directory (side-by-side).

The compilation is tuned by editing the makefile of Pazpar2. The process is similar to YAZ. Adjust the various directories `YAZ_DIR`, `ICU_DIR`, etc., as required.

Compile Pazpar2 by invoking nmake in the `win` directory. The resulting binaries of the build process are located in the `bin` of the Pazpar2 source tree - including the `pazpar2.exe` and necessary DLLs.

The Windows version of Pazpar2 is a console application. It may be installed as a Windows Service by adding option `-install` for the pazpar2 program. This will register Pazpar2 as a service and use the other options provided in the same invocation. For example:

```
cd \MyPazpar2\etc
..\bin\pazpar2 -install -f pazpar2.cfg -l pazpar2.log
```

The Pazpar2 service may now be controlled via the Service Control Panel. It may be unregistered by passing the `-remove` option. Example:

```
cd \MyPazpar2\etc
..\bin\pazpar2 -remove
```

2.3 Installation of test interfaces

In this section we show how to make available the set of simple interfaces that are part of the Pazpar2 source package, and which demonstrate some ways to use Pazpar2. (Note that Debian users can save time by just installing the package `pazpar2-test1`.)

A web server, such as Apache, must be installed and running on the system.

Start the Pazpar2 daemon using the 'in-source' binary of the Pazpar2 daemon. On Unix the process is:

```
cd etc
cp pazpar2.cfg.dist pazpar2.cfg
../src/pazpar2 -f pazpar2.cfg
```

And on Windows:

```
cd etc
copy pazpar2.cfg.dist pazpar2.cfg
..\bin\pazpar2 -f pazpar2.cfg
```

This will start a Pazpar2 listener on port 9004. It will proxy HTTP requests to port 80 on localhost, which we assume will be the regular HTTP server on the system. Inspect and modify pazpar2.cfg as needed if this is to be changed.

The pazpar2.cfg file includes settings from the file `settings/testserver.xml` to use for searches. That uses a single local target. If you do not yet have a local information server running (e.g. yaz-ztest) then adjust the target settings to refer to a known public server. Alternatively, adjust the `default.xml` to refer to a different set of targets (e.g. `settings/edu.xml`).

The test UIs are located in `www`. Ensure that this directory is available to the web server, either by copying `www` to the document root, using Apache's `Alias` directive, or creating a symbolic link: for example, on a Debian or Ubuntu system with Apache2 installed from the standard package, you might make the link as follows:

```
cd .../pazpar2
sudo ln -s `pwd`/www /var/www/pazpar2-demo
```

This makes the test applications visible at `http://localhost/pazpar2-demo/` but they can not be run successfully from that URL, as they submit search requests back to the server from which they were served, and Apache2 doesn't know how to handle them. Instead, the test applications must be accessed from Pazpar2 itself, acting as a proxy to Apache2, at the URL `http://localhost:9004/pazpar2-demo/`

From here, the demo applications can be accessed: `test1`, `test2` and `jsdemo` are pure HTML+JavaScript setups, needing no server-side intelligence; `demo` requires PHP on the server. The `jsdemo` also needs `pazpar2/js/pz2.js` to be available to the webserver.

If you don't see the test interfaces, check whether they are available on port 80 (i.e. directly from the Apache2 server). If not, the Apache configuration is incorrect.

In order to use Apache as frontend for the interface on port 80 for public access etc., refer to Section 2.6.

2.4 Installation on Debian GNU/Linux and Ubuntu

Index Data provides Debian and Ubuntu packages for Pazpar2 and YAZ. Refer to these directories: `http://ftp.indexdata.com/pub/pazpar2/debian/` and `http://ftp.indexdata.com/pub/pazpar2/ubuntu/`.

2.5 Installation on RedHat / CentOS

Index Data provides CentOS packages for Pazpar2 and YAZ. Refer to `http://ftp.indexdata.com/pub/pazpar2/redhat/centos` for CentOS packages.

2.6 Apache 2 Proxy

Apache 2 has a proxy module which allows Pazpar2 to become a backend to an Apache 2 based web service. The Apache 2 proxy must operate in the *Reverse* Proxy mode.

On a Debian based Apache 2 system, the relevant modules can be enabled with:

```
sudo a2enmod proxy_http proxy_balancer
```

Traditionally Pazpar2 interprets URL paths with suffix `/search.pz2`. The ProxyPass directive of Apache must be used to map a URL path to the Pazpar2 server (listening port).

Note
The ProxyPass directive takes a prefix rather than a suffix as URL path. It is important that the Java Script code uses the prefix given for it.

Example 2.1 Apache 2 proxy configuration

If Pazpar2 is running on port 8004 and the portal is using `search.pz2` inside portal in directory `/myportal/` we could use the following Apache 2 configuration:

```
<IfModule mod_proxy.c>
 ProxyRequests Off

 <Proxy *>
  AddDefaultCharset off
  Order deny,allow
  Allow from all
 </Proxy>

 ProxyPass /myportal/search.pz2 http://localhost:8004/search.pz2
 ProxyVia Off
</IfModule>
```

Chapter 3

Using Pazpar2

This chapter provides a general introduction to the use and deployment of Pazpar2.

3.1 Pazpar2 and your systems architecture

Pazpar2 is designed to provide asynchronous, behind-the-scenes metasearching functionality to your application, exposing this functionality using a simple webservice API that can be accessed from any number of development environments. In particular, it is possible to combine Pazpar2 either with your server-side dynamic website scripting, with scripting or code running in the browser, or with any combination of the two. Pazpar2 is an excellent tool for building advanced, Ajax-based user interfaces for metasearch functionality, but it isn't a requirement -- you can choose to use Pazpar2 entirely as a backend to your regular server-side scripting. When you do use Pazpar2 in conjunction with browser scripting (JavaScript/Ajax, Flash, applets, etc.), there are special considerations.

Pazpar2 implements a simple but efficient HTTP server, and it is designed to interact directly with scripting running in the browser for the best possible performance, and to limit overhead when several browser clients generate numerous webservice requests. However, it is still desirable to use a conventional webserver, such as Apache HTTP Server, to serve up graphics, HTML documents, and server-side scripting. Because the security sandbox environment of most browser-side programming environments only allows communication with the server from which the enclosing HTML page or object originated, Pazpar2 is designed so that it can act as a transparent proxy in front of an existing webserver (see Pazpar2 conf(5) for details). In this mode, all regular HTTP requests are transparently passed through to your webserver, while Pazpar2 only intercepts search-related webservice requests.

If you want to expose your combined service on port 80, you can either run your regular webserver on a different port, a different server, or a different IP address associated with the same server.

Pazpar2 can also work behind a reverse Proxy. Refer to Section 2.6 for more information. This allows your existing HTTP server to operate on port 80 as usual. Pazpar2 can be started on another (internal) port.

Sometimes, it may be necessary to implement functionality on your regular webserver that makes use of search results, for example to implement data import functionality, emailing results, history lists, personal citation lists, interlibrary loan functionality, etc. Fortunately, it is simple to exchange information between Pazpar2, your browser scripting, and backend server-side scripting. You can send a session ID and possibly a record ID from your browser code to your server code, and from there use Pazpar2's webservice API to

access result sets or individual records. You could even 'hide' all of Pazpar2's functionality between your own API implemented on the server-side, and access that from the browser or elsewhere. The possibilities are just about endless.

3.2 Your data model

Pazpar2 does not have a preconceived model of what makes up a data model. There are no assumptions that records have specific fields or that they are organized in any particular way. The only assumption is that data comes packaged in a form that the software can work with (presently, that means XML or MARC), and that you can provide the necessary information to massage it into Pazpar2's internal record abstraction.

Handling retrieval records in Pazpar2 is a two-step process. First, you decide which data elements of the source record you are interested in, and you specify any desired massaging or combining of elements using an XSLT stylesheet (MARC records are automatically normalized to MARCXML before this step). If desired, you can run multiple XSLT stylesheets in series to accomplish this, but the output of the last one must be a representation of the record in a schema that Pazpar2 understands.

The intermediate, internal representation of the record looks like this:

```
<record xmlns="http://www.indexdata.com/pazpar2/1.0"
  mergekey="title The Shining author King, Stephen">

  <metadata type="title" rank="2">The Shining</metadata>

  <metadata type="author">King, Stephen</metadata>

  <metadata type="kind">ebook</metadata>
  <!-- ... and so on -->
</record>
```

As you can see, there isn't much to it. There are really only a few important elements to this file.

Elements should belong to the namespace `http://www.indexdata.com/pazpar2/1.0`. If the root node contains the attribute 'mergekey', then every record that generates the same merge key (normalized for case differences, white space, and truncation) will be joined into a cluster. In other words, you decide how records are merged. If you don't include a merge key, records are never merged. The 'metadata' elements provide the meat of the elements -- the content. The 'type' attribute is used to match each element against processing rules that determine what happens to the data element next. The 'rank' attribute specifies specifies a multipler for ranking for this element.

The next processing step is the extraction of metadata from the intermediate representation of the record. This is governed by the 'metadata' elements in the 'service' section of the configuration file. See the section called "server" for details. The metadata in the retrieval record ultimately drives merging, sorting, ranking, the extraction of browse facets, and display, all configurable.

Pazpar2 1.6.37 and later also allows already-clustered records to be ingested. Suppose a database already clusters for us and we would like to keep that cluster for Pazpar2. In that case we can generate a `cluster` wrapper element that holds individual `record` elements.

Cluster record example:

```
  <cluster xmlns="http://www.indexdata.com/pazpar2/1.0">
    <record>
      <metadata type="title" rank="2">The Shining</metadata>
<metadata type="author">King, Stephen</metadata>
<metadata type="kind">ebook</metadata>
    </record>
    <record>
      <metadata type="title" rank="2">The Shining</metadata>
<metadata type="author">King, Stephen</metadata>
<metadata type="kind">audio</metadata>
    </record>
 </cluster>
```

3.3 Client development overview

You can use Pazpar2 from any environment that allows you to use webservices. The initial goal of the software was to support Ajax-based applications, but there literally are no limits to what you can do. You can use Pazpar2 from Javascript, Flash, Java, etc., on the browser side, and from any development environment on the server side, and you can pass session tokens and record IDs freely around between these environments to build sophisticated applications. Use your imagination.

The webservice API of Pazpar2 is described in detail in Pazpar2 protocol(7).

In brief, you use the 'init' command to create a session, a temporary workspace which carries information about the current search. You start a new search using the 'search' command. Once the search has been started, you can follow its progress using the 'stat', 'bytarget', 'termlist', or 'show' commands. Detailed records can be fetched using the 'record' command.

3.4 Ajax client development

Pazpar2 offers programmers a simple Web Service protocol that can be used (queried in a request/response fashion) from any, server- or client-side, programming language with an XML support. However, when programming a Web-based client to Pazpar2, to achieve certain level of interactivity and instant notification of latest changes in the result set, Ajax (Asynchronous JavaScript and XML) technology may be used. An Ajax client allows user to browse the results before the lengthy process of information retrieval from the back-end targets is finished. Blocking and waiting for usually slow back-end targets is one of the biggest functionality issues in a federated search engine.

Pazpar2 comes with a small JavaScript library called `pz2.js`. This library is designed to simplify development of an Ajax-based pazpar2 client and alleviate the programmer from the low-level details like polling the web service, fetching and parsing returned XML output or managing timers, sessions and basic state variables.

The library supports most major browsers including Firefox 1.5+, IE 6+, Safari 2+, Opera 9+ and Konqueror.

The library can work in two modes: a session-aware mode and a session-less mode.

In the session-aware mode, the library assumes that the pazpar2 daemon is contacted directly (preferably via Apache proxy to avoid security breaches) and tracks the session IDs internally.

In the session-less, mode the library assumes that the client is identified on the server and the session IDs are not managed directly. This way of operation requires a more sophisticated pazpar2 proxy (preferably a wrapper written in a server-side scripting language like PHP that can identify clients and relate them to open pazpar2 sessions).

Using `pz2.js`

Client development with the `pz2.js` is strongly event based and the style should be familiar to most JavaScript developers. A simple client (jsdemo) is distributed with Pazpar2's source code and shows how to set-up and use `pz2.js`.

In short, the programmer starts by instantiating the pz2 object and passing an array of parameters to the constructor. The parameter array specifies callbacks used for handling responses to the pazpar2 commands. Additionally, the parameter array is used to configure run-time parameters of the `pz2.js` like polling timer time-outs, session-mode and XSLT stylesheets.

Command callbacks

Callback naming is simple, and follows "on" prefix plus command name scheme (like onsearch, onshow, onrecord, ... etc.). When the programmer calls a function like show or record on the pz2 object, `pz2.js` will keep on polling pazpar2 (until the backend targets are idle) and with each command's response an assigned callback will be called. In case of Pazpar2's internal error, an error callback is called.

```
my_paz = new pz2 (
{
 "pazpar2path": "/pazpar2/search.pz2",
 "usesessions" : true,

 // assigning command handler, turns on automatic polling
 "onshow": my_onshow,
 // polling period for each command can be specified
 "showtime": 500,

 "onterm": my_onterm,
 // facet terms are specified as a comma separated list
 "termlist": "subject,author",

 "onrecord": my_onrecord
}
);
```

Each command callback is a user-defined function that takes a hash object as a parameter. The hash object contains parsed pazpar2 responses (hash members that correspond to the elements in the response XML document). Within the handler, the programmer further processes the data and updates the viewed document.

```
function my_onstat(data) {
 var stat = document.getElementById("stat");
 stat.innerHTML = '<span>Active clients: '+ data.activeclients
  + '/' + data.clients + ' | </span>'
  + '<span>Retrieved records: ' + data.records
  + '/' + data.hits + '</span>';
}

function my_onshow(data) {
 // data contains parsed show response
 for (var i = 0; i < data.hits[0].length; i++)
  // update page with the hits
}

function on_record(data) {
 // if detailsstylesheet parameter was set data array
 // will contain raw xml and xsl data
 Element_appendTransformResult(someDiv, data.xmlDoc, data.xslDoc);
}
```

`pz2.js` on runtime

The search process is initiated by calling the search method on the instantiated pz2 object. To initiate short status reports and per-target status information, methods stat and bytarget are called accordingly.

```
my_paz.search (query, recPergPage, 'relevance');
```

Managing the results (keeping track of the browsed results page and sorting) is up to the client's programmer. At any point the show method may be called to bring up the latest result set with a different sorting criteria or range, without re-executing the search on the back-end.

```
my_paz.show (1, 10, 'relevance');
```

To retrieve a detailed record, the record command is called. When calling the record command, one may temporarily override its default callback by specifying the handler parameter. This might be useful when retrieving raw records that need to be processed differently.

```
my_paz.record (recId, 2, 'opac', { "callback": temp_callback, "args",  ↩
   caller_args});
```

PARAMETERS ARRAY

pazpar2path server path to pazpar2 (relative to the portal). When pazpar2 is installed as a package this does not need to be set.

usesessions Boolean. When set to true `pz2.js` will manage sessions internally, otherwise it is left to the server-side script. Default true.

autoInit Boolean. Sets auto initialization of pazpar2 session on the object instantiation. Default true. Valid only if usesession is set to true.

detailstylesheet path to the xsl presentation stylesheet (relative to the portal) used for the detailed record display

errorhandler callback function called on any (pazpar2 or `pz2.js`) internal error

oninit specifies init response callback function

onstat specifies stat response callback function

onshow specifies show response callback function

onterm specifies termlist response callback function

onrecord specifies record response callback function

onbytarget specifies bytarget response callback function

onreset specifies reset method callback function

termlist comma separated list of facets

keepAlive ping period, should not be lower than 5000 usec

stattime default 1000 usec

termtime

showtime

bytargettime

METHODS

stop () stop activity by clearing timeouts

reset () reset state

init (sessionId, serviceId) session-mode, initialize new session or pick up a session already initialized

ping () session-mode, intitialize pinging

search (query, num, sort, filter, showfrom) execute piggy-back search and activate polling on every command specified by assigning command callback (in the pz2 constructor)

show (start, num, sort) start or change parameters of polling for a given window of records

record (id, offset, syntax, handler) retrieve detailed or raw record. handler temporarily overrides default callback function.

termlist () start polling for termlists

bytarget () start polling for target status

stat () start polling for pazpar2 status

`pz2.js` comes with a set of cross-browser helper classes and functions.

Ajax helper class

pzHttpRequest a cross-browser Ajax wrapper class

constructor (url, errorHandler) create new request for a given url

get (params, callback) asynchronous, send the request with given parameters (array) and call callback with response as parameter

post (params, data, callback) asynchronous, post arbitrary data (may be XML doc) and call callback with response as parameter

load () synchronous, returns the response for the given request

XML helper functions

document.newXmlDoc (root) create new XML document with root node as specified in parameter

document.parseXmlFromString (xmlString) create new XML document from string

document.transformToDoc (xmlDoc, xslDoc) returns new XML document as a result

Element_removeFromDoc (DOM_Element) remove element from the document

Element_emptyChildren (DOM_Element)

Element_appendTransformResult (DOM_Element, xmlDoc, xslDoc) append xsl transformation result to a DOM element

Element_appendTextNode (DOM_Element, tagName, textContent) append new text node to the element

Element_setTextContent (DOM_Element, textContent) set text content of the element

Element_getTextContent (DOM_Element) get text content of the element

Element_parseChildNodes (DOM_Element) parse all descendants into an associative array

3.5 Unicode Compliance

Pazpar2 is Unicode compliant and language and locale aware, but relies on character encoding for the targets to be specified correctly if the targets themselves are not UTF-8 based (most aren't). Just a few bad behaving targets can spoil the search experience considerably if for example Greek, Russian or otherwise non 7-bit ASCII search terms are entered. In these cases some targets return records irrelevant to the query, and the result screens will be cluttered with noise.

While noise from misbehaving targets can not be removed, it can be reduced using truly Unicode based ranking. This is an option which is available to the system administrator if ICU support is compiled into YAZ, see Chapter 2 for details.

In addition, the ICU tokenization and normalization rules must be defined in the master configuration file described in the section called "server".

3.6 Load balancing

Just like any web server, Pazpar2 can be load balanced by a standard hardware or software load balancer as long as the session stickiness is ensured. If you are already running the Apache2 web server in front of Pazpar2 and use the apache mod_proxy module to 'relay' client requests to Pazpar2, this set up can be easily extended to include load balancing capabilites. To do so, you need to enable the mod_proxy_balance module in your Apache2 installation.

On a Debian based Apache 2 system, the relevant modules can be enabled with:

```
sudo a2enmod proxy_http
```

The mod_proxy_balancer can pass all 'sessionsticky' requests to the same backend worker, as long as the requests are marked with the originating worker's ID (called 'route'). If the Pazpar2 serverID is configured (by setting an 'id' attribute on the 'server' element in the Pazpar2 configuration file) Pazpar2 will append it to the 'session' element returned during the 'init' in a mod_proxy_balancer compatible manner. Since the 'session' is then re-sent by the client (for all pazpar2 request besides 'init'), the balancer can use the marker to pass the request to the right route. To do so, the balancer needs to be configured to inspect the 'session' parameter.

Example 3.1 Apache 2 load balancing configuration

Having four Pazpar2 instances running on the same host, having port range of 8004-8007 and serverIDs of: pz1, pz2, pz3 and pz4 respectively, we could use the following Apache 2 configuration to expose a single Pazpar2 'endpoint' on a standard (`/pazpar2/search.pz2`) location:

```
<Proxy *>
  AddDefaultCharset off
  Order deny,allow
  Allow from all
</Proxy>
ProxyVia Off

# 'route' has to match the configured pazpar2 server ID
<Proxy balancer://pz2cluster>
```

```
        BalancerMember http://localhost:8004 route=pz1
        BalancerMember http://localhost:8005 route=pz2
        BalancerMember http://localhost:8006 route=pz3
        BalancerMember http://localhost:8007 route=pz4
      </Proxy>

      # route is resent in the 'session' param which has the form:
      # 'sessid.serverid', understandable by the mod_proxy_load_balancer
      # this is not going to work if the client tampers with the 'session' ↩
          param
      ProxyPass /pazpar2/search.pz2 balancer://pz2cluster lbmethod= ↩
         byrequests stickysession=session nofailover=On
```

The 'ProxyPass' line sets up a reverse proxy for request '/pazpar2/search.pz2' and delegates all requests to the load balancer (virtual worker) with name 'pz2cluster'. Sticky sessions are enabled and implemented using the 'session' parameter. The 'Proxy' section lists all the servers (real workers) which the load balancer can use.

3.7 Relevance ranking

Pazpar2 uses a variant of the fterm frequency–inverse document frequency (Tf-idf) ranking algorithm.

The Tf-part is straightforward to calculate and is based on the documents that Pazpar2 fetches. The idf-part, however, is more tricky since the corpus at hand is ONLY the relevant documents and not irrelevant ones. Pazpar2 does not have the full corpus -- only the documents that match a particular search.

Computatation of the Tf-part is based on the normalized documents. The length, the position, and terms are thus normalized at this point. Also the computation is performed for each document received from the target - before merging takes place. The result of a TF-compuation is added to the TF-total of a cluster. Thus, if a document occurs twice, then the TF-part is doubled. That, however, can be adjusted, because the TF-part may be divided by the number of documents in a cluster.

The algorithm used by Pazpar2 has two phases. In phase one, Pazpar2 computes a tf-array: This is being done as records are fetched from the database. In this case, the rank weight `w`, and the rank tweaks `lead`, `follow` and `length`.

```
   tf[1,2,..N] = 0;
   foreach document in a cluster
      foreach field
         w[1,2,..N] = 0;
         for i = 1, .. N:  (each term)
            foreach pos (where term i occurs in field)
               // w is configured weight for field
               // pos is position of term in field
               w[i] += w / (1 + log2(1+lead*pos))
               if (d > 0)
                   w[i] += w[i] * follow / (1+log2(d)
         // length: length of field (number of terms that is)
   if (length strategy is "linear")
            tf[i] += w[i] / length;
```

```
        else if (length strategy is "log")
           tf[i] += w[i] / log2(length);
        else if (length strategy is "none")
           tf[i] += w[i];
```

In phase two, the idf-array is computed and the final score is computed. This is done for each cluster, as part of each show command. The rank tweak `cluster` is in use here.

```
// dococcur[i]: number of records where term occurs
// doctotal: number of records
for i = 1, .., N (each term)
  if (dococcur[i] > 0)
     idf[i] = log(1 + doctotal / dococcur[i])
  else
     idf[i] = 0;

relevance = 0;
for i = 1, .., N: (each term)
   if (cluster is "yes")
      tf[i] = tf[i] / cluster_size;
   relevance += 100000 * tf[i] / idf[i];
```

For controlling the ranking parameters, refer to the rank element of the service definition. Refer to the rank attribute of the metadata element for how to control ranking for individual metadata fields.

3.8 Pazpar2 and MasterKey Connect

MasterKey Connect is a hosted connector, or gateway, service that exposes whatever searchable resources you need. Since the service exposes all resources using Z39.50 (or SRU), it is easy to set up Pazpar2 to use the service. In particular, since all connectors expose basically the same core behavior, it is a good use of Pazpar2's mechanism for managing default behaviors across similar databases.

After installation of Pazpar2, the directory `/etc/pazpar2/settings/mkc` (location may vary depending on installation preferences) contains an example setup that searches two different resources through a MasterKey Connect demo account. The file mkc.xml contains default parameters that will work for all MasterKey Connect resources (if you decide to become a customer of the service, you will substitute your own account credentials for the guest/guest). The other files contain specific information about a couple of demonstration resources.

To play with the demo, just create a symlink from `/etc/pazpar2/services-enabled/default.xml` to `/etc/pazpar2/services-available/mkc.xml` and then restart Pazpar2. You should now be able to search the two demo resources using JSDemo or any user interface of your choice. If you are interested in learning more about MasterKey Connect, or to try out the service for free against your favorite online resource, just contact us at info@indexdata.com.

Chapter 4

Reference

The material in this chapter is drawn directly from the individual manual entries.

4.1 Pazpar2

pazpar2 — Metasearch daemon.

Synopsis

`pazpar2` [`-d`] [`-D`] [`-f` *config*] [`-h` *ip:port*] [`-l` *logfile*] [`-m` *timeformat*] [`-p` *pidfile*] [`-R` *recfile*] [`-t`] [`-u` *uid*] [`-v` *level*] [`-V`] [`-w` *dir*] [`-X`] [`-install`] [`-remove`]

DESCRIPTION

pazpar2 is the Pazpar2 Metasearch daemon and server. In normal operation it acts as a simple HTTP server which serves the Pazpar2 protocol. The HTTP listener address may be given on the command line using option `-h` or in the main configuration file. The main configuration must be specified using option `-f`.

OPTIONS

`-d` Enables dump of XML records to the current log file. It is useful if stylesheets are being debugged. Using this option twice makes Pazpar2 also dump full HTTP responses.

This option may also be used together with option `-t` in which case the configuration, after include processing, will be dumped to stdout.

`-D` Puts the Pazpar2 server in the background.

`-f` *`config`* Specifies main configuration. This option must be specified in order for Pazpar2 to operate normally.

-h ***ip:port*** Specifies the HTTP listener binding address. The *ip* may be a hostname, or `@` for "any" address. The *port* is an integer.

-l ***logfile*** Specifies log file. The log file must be specified when Pazpar2 is running in the background (-D).

-m ***timeformat*** Sets the format of time-stamps for logging. Refer to the strftime(3) man page for the format.

-p ***pidfile*** Specifies PID file. If Pazpar2 is started and configured properly, the file given holds the process ID of the Pazpar2 process.

-R ***recfile*** If this option is given, HTTP requests are logged to file named *recfile* and predictable sessions are enabled. Using special argument, dash (−), will make Pazpar2 use predictable sessions only (no recording). This is necessary when playing HTTP communication against pazpar2 with the pazpar2_play program. Refer to Pazpar2_play(1).

-t Checks parameters and configuration. No service or daemon is started. Useful for checking a new configuration before a Pazpar2 is restarted.

The configuration, after include processing, may also be dumped to stdout by supplying option `-d` as well.

Note

In Pazpar2 1.2 and earlier releases, option -t specified a local target settings file.

-u ***uid*** Makes the Pazpar2 server change user ID to the *uid* given. This, normally, requires root privilege.

-v ***level*** Sets log level (YAZ log level system).

-V Shows Pazpar2 version, and versions of some of the components that it is using (ICU and YAZ). Pazpar2 will exit immediately after displaying the version information (no daemon started).

-w ***dir*** Changes working directory to *dir*.

-X Makes the Pazpar2 server operate in debugging mode. This prevents Pazpar2 from making separate threads and processes. This option should not be used in production.

-install This is an option which is only recognized on Windows. It installs Pazpar2 as a Windows Service.

Note

Pazpar2 only supports Windows Service options if Pazpar2 is linked against YAZ 3.0.29 or later.

-remove This is an option which is only recognized on Windows. It removes a Pazpar2 - Windows Service.

EXAMPLES

The Debian package of pazpar2 starts the server with:

```
pazpar2 -D -f /etc/pazpar2/pazpar2.cfg -l /var/log/pazpar2.log -p /var/ ↩
   run/pazpar2.pid -u nobody
```

(one line).

This will put pazpar2 in the background (-D), read config from from `/etc/pazpar2/pazpar2.cfg`, log messages to `/var/log/pazpar2.log`, create PID file `/var/run/pazpar2.pid`. When the daemon is properly started, the server will change effective user ID to nobody.

The server can be terminated with:

```
kill `cat /var/run/pazpar2.pid`
```

If Pazpar2 is to be debugged using GDB, we use option -X:

```
cd pazpar2/src
gdb ./pazpar2
(gdb) run -X -f ../etc/pazpar2.cfg
```

FILES

`/usr/sbin/pazpar2`: pazpar2 daemon

`/usr/share/pazpar2`: pazpar2 shared files

`/etc/pazpar2`: pazpar2 config area

SEE ALSO

Pazpar2 configuration: pazpar2_conf(5)

Pazpar2 protocol: pazpar2_protocol(7)

Pazpar2 player: pazpar2_play(1)

4.2 Pazpar2 protocol

pazpar2_protocol — The webservice protocol of Pazpar2

DESCRIPTION

Webservice requests are any that refer to filename "search.pz2". Arguments are GET-style parameters. Argument 'command' is always required and specifies the operation to perform. Any request not recognized as a webservice request is forwarded to the HTTP server specified in the configuration using the proxy

setting. This way, a regular webserver can host the user interface (itself dynamic or static HTML), and Ajax-style calls can be used from JS (or any other client-based scripting environment) to interact with the search logic in Pazpar2.

Each command is described in sub sections to follow.

info

Returns version and statistics about the Pazpar2 instance.

init

Initializes a session. Returns session ID to be used in subsequent requests. If a server ID is given in the Pazpar2 server section, then that is included in the session ID as suffix after a period (.).

If the init command is performed as a HTTP GET request, service and settings from local files are used. The service parameter may choose a particular local service.

If the init command is performed as a HTTP POST request and the content-type is text/xml, then the content is XML parsed and treated as service for the session. The root element should be service. Refer to description of the service format. The posting of a service appeared in Pazpar2 version 1.2.1.

Example:

```
search.pz2?command=init
```

Response:

```
<init>
 <status>OK</status>
 <session>2044502273</session>
</init>
```

The init command may take a number of setting parameters, similar to the 'settings' command described below. These settings are immediately applied to the new session. Other parameters for init are:

clear If this is defined and the value is non-zero, the session will not use the predefined databases in the configuration; only those specified in the settings parameters (per session databases).

service If this is defined it specifies a service ID. Makes the session use the service with this ID. If this setting is omitted, the session will use the unnamed service in the Pazpar2 configuration.

ping

Keeps a session alive. An idle session will time out after one minute. The ping command can be used to keep the session alive, absent other activity. It is suggested that any browser client have a simple alarm handler which sends a ping every 50 seconds or so, once a session has been initialized.

Example:

```
search.pz?command=ping&session=2044502273
```

Response:

```
<ping>
  <status>OK</status>
</ping>
```

settings

The settings command applies session-specific settings to one or more databases. A typical function of this is to enable access to restricted resources for registered users, or to set a user- or library-specific username/-password to use against a target.

Each setting parameter has the form name[target]=value, where name is the name of the setting (e.g. pz:authentication), target is a target ID, or possibly a wildcard, and value is the desired value for the setting.

Because the settings command manipulates potentially sensitive information, it is possible to configure Pazpar2 to only allow access to this command from a trusted site -- usually from server-side scripting, which in turn is responsible for authenticating the user, and possibly determining which resources that they have access to, etc.

Note
As a shortcut, it is also possible to override settings directly in the init command.

If the settings command is performed as HTTP POST and the content-type is text/xml, then the content is XML parsed and treated as settings - with a format identical to local settings files. The posting of settings appeared in Pazpar2 version 1.2.1.

Example:

```
search.pz?command=settings&session=2044502273&pz:allow[search.com:210/db1 ↩
   ]=1
```

Response:

```
<settings>
  <status>OK</status>
</settings>
```

search

Launches a search. Parameters:

session Session ID

query CCL query

filter Limits the search to a given set of targets specified by the filter. The filter consists of a comma-separated list of *setting+operator+args* pairs all of which must be satisfied (matched) for Pazpar2 to include the target. The *setting* is a Pazpar2 setting (such as `pz:id`). The *operator* is either = (string match) or ~ (substring match). The *args* is a list of values separated by `|`. If either of these values match, the key-value pair is matched.

For Pazpar2 1.13.0 the filter can be prefixed with vertical bar (`|`) as first character. In this case, if any of the key-value pairs matches, Pazpar2 includes the target.

limit Narrows the search by one or more fields (typically facets). The limit is sequence of one or more *name=args* pairs separated by comma. The *args* is a list of values separated by vertical bar (`|`). The meaning of `|` is alternative (i.e. OR). A value that contains a comma (`,`), a vertical bar (`|`) or backslash itself must be preceded by backslash (`\`). The pz:limitmap configuration item defines how the searches are mapped to a database.

startrecs Specifies the first record to retrieve from each target. The first record in a result set for a target is numbered 0, next record is numbered 1. By default startrecs is 0.

maxrecs Specifies the maximum number of records to retrieve from each target. The default value is 100. This setting has same meaning as per-target setting pz:maxrecs . If pz:maxrecs is set, it takes precedence over argument maxrecs.

sort Specifies sort criteria. The argument is a comma-separated list (no whitespace allowed) of sort fields, with the highest-priority field first. A sort field may be followed by a colon followed by the number '0' (decreasing) or '1' (increasing). Default sort order is decreasing. Sort field names can be any field name designated as a sort field in the pazpar2.cfg file, or the special names 'relevance', 'retrieval' and 'position'.

Sort type 'position' sorts by position/offset for each database. Sort type 'retrieval' sorts by position of retrieval (first record retrieved is 1, second record is 2, etc.).

If not specified here or as sort-default in pazpar2.cfg, Pazpar2 will default to the built-in 'relevance' ranking.

Having sort criteria at search is important for targets that supportnative sorting in order to get best results. Pazpar2 will trigger a new search if search criteria changes from Pazpar2 to target-based sorting or vice versa.

mergekey Sets mergekey for this search and rest of session, or until another mergekey is given for show/search. The mergekey value is a comma-separated list with one or more names as they appear in the service description equivalent to `mergekey="optional"` inside a metadata element. If the empty string is given for mergekey it is disabled and rest of session will use the default mergekey from service or stylesheet.

This facility, "dynamic mergekey", appeared in Pazpar2 version 1.6.31.

rank Sets rank method for this search and rest of session, or until another rank is given for show/search. The rank value is a comma-separated list of field=value pairs. The format is the same as rank for a metadata element. If the empty string is given for rank, it is disabled and rest of session will use the default rank from metadata or stylesheet.

This facility, "dynamic ranking", appeared in Pazpar2 version 1.6.31.

Example:

```
search.pz2?session=2044502273&command=search&query=computer+science
```

Response:

```
<search>
  <status>OK</status>
</search>
```

stat

Provides status information about an ongoing search. Parameters:

session Session ID

Example:

```
search.pz2?session=2044502273&command=stat
```

Output

```
<stat>
  <activeclients>26</activeclients>
  <hits>84919</hits>            -- Total hitcount
  <records>742</records>        -- Total number of records fetched in last ↩
     query
  <clients>45</clients>         -- Total number of associated clients
  <unconnected>0</unconnected>  -- Number of disconnected clients
  <connecting>0</connecting>    -- Number of clients in connecting state
  <working>26</working>         -- ... working (searching, presenting, etc ↩
     .)
  <idle>2</idle>                -- ... idle (not doing anything)
  <failed>0</failed>            -- ... Connection failed
  <error>13</error>             -- ... Error was produced somewhere
</stat>
```

show

Shows records retrieved. Parameters:

session Session ID

start First record to show - 0-indexed.

num Number of records to show. If omitted, 20 is used.

block If block is set to 1, the command will hang until there are records ready to display. Use this to show first records quickly without requiring rapid polling.

If block is set to `preferred`, the command will wait until records have been received from all databases with preferred setting

sort Specifies sort criteria. The argument is a comma-separated list (no whitespace allowed) of sort fields, with the highest-priority field first. A sort field may be followed by a colon followed by the number '0' (decreasing) or '1' (increasing). Default sort order is decreasing. Sort field names can be any field name designated as a sort field in the pazpar2.cfg file, or the special names 'relevance', 'retrieval' and 'position'.

Sort type 'position' sorts by position/offset for each database. Sort type 'retrieval' sorts by position of retrieval (first record retrieved is 1, second record is 2, etc.).

If not specified here or as sort-default in pazpar2.cfg, then Pazpar2 will default to the built-in 'relevance' ranking.

Having sort criteria at search is important for targets that supports native sorting in order to get best results. Pazpar2 will trigger a new search if search criteria changes from pazpar2-based to target-based sorting.

For targets where pz:sortmap is defined, a sort operation will be executed (which will possibly include extending the search).

mergekey Sets mergekey for this show, and for the rest of the session, or until another mergekey is given for show/search. The mergekey value is a comma-separated list with one or more names as they appear in the service description equivalent to `mergekey="optional"` inside a metadata element. If the empty string is given for mergekey, it is disabled and rest of session will use the default mergekey from service or stylesheet.

This facility, "dynamic mergekey", appeared in Pazpar2 version 1.6.31.

rank Sets rank method for this show, and for the rest of the session, or until another rank is given for show/search. The rank value is a comma-separated list of field=value pairs. The format is the same as rank for a metadata element. If the empty string is given for rank, it is disabled and rest of session will use the default rank from metadata or stylesheet.

This facility, "dynamic ranking", appeared in Pazpar2 version 1.6.31.

snippets If specified and set to 1, then data will include snippets marked with <match> tags. Otherwise snippets will not be included.

This facility, "snippets", appeared in Pazpar2 version 1.6.32.

version If specified and set to 2, enables Pazpar2 to return approximation on hits and counts when doing record filtering using the limit parameter on search and a limitmap with a value of "local:".

This facility, "version", appeared in Pazpar2 version 1.6.13.

Example:

```
search.pz2?session=2044502273&command=show&start=0&num=2&sort=title:1
```

Output:

```
<show>
  <status>OK</status>
  <activeclients>3</activeclients>     -- How many clients are still  ↩
     working
  <merged>6</merged>                   -- Number of merged records
  <total>7</total>                     -- Total of all hitcounts
  <start>0</start>                     -- The start number you requested
  <num>2</num>                         -- Number of records retrieved
  <hit>
    <md-title>How to program a computer, by Jack Collins</md-title>
    <count>2</count>                   -- Number of merged records
    <recid>6</recid>                   -- Record ID for this record
  </hit>
  <hit>
    <md-title>
  Computer processing of dynamic images from an Anger scintillation camera  ↩
     :
  the proceedings of a workshop /
    </md-title>
    <recid>2</recid>
  </hit>
</show>
```

record

Retrieves a detailed record. Unlike the show command, this command returns metadata records before merging takes place. Parameters:

session Session ID

id record ID as provided by the show command.

offset This optional parameter is an integer which, when given, makes Pazpar2 return the original record for a specific target. The record set from first target is numbered 0, second record set is numbered 1, etc. The nativesyntax setting, as usual, is used to determine how to create XML from the original record - unless parameter `binary` is given in which the record is fetched as "raw" from ZOOM C (raw, original record).

When offset/checksum is not given, the Pazpar2 metadata for the record is returned and with metadata for each target's data specified in a "location" list.

checksum This optional parameter is a string which, when given, makes Pazpar2 return the original record for a specific target. The checksum is returned as attribute 'checksum' in element 'location' for show command and record command (when checksum and offset is NOT given). The nativesyntax setting, as usual, is used to determine how to create XML from the original record - unless parameter `binary` is given in which the record is fetched as "raw" from ZOOM C (raw, original record).

When offset/checksum is not given, the Pazpar2 metadata for the record is returned and with metadata for each targets' data specified in a 'location' list.

nativesyntax This optional parameter can be used to override pz:nativesyntax as given for the target. This allow an alternative nativesyntax to be used for original records (see parameteroffset above).

syntax This optional parameter is the record syntax used for raw transfer (i.e. when offset is specified). If syntax is not given, but offset is used, the value of pz:requestsyntax is used.

esn This optional parameter is the element set name used for retrieval of a raw record (i.e. when offset is specified). If esn is not given, but offset is used, the value of pz:elements is used.

binary This optional parameter enables "binary" response for retrieval of a original record (i.e. when offset is specified). For binary response, the record by default is fetched from ZOOM C using the "raw" option or by parameter nativesyntax if given.

snippets If specified and set to 1, then data will include snippets marked with <match> tags. Otherwise snippets will not be included.

This facility, "snippets", appeared in Pazpar2 version 1.6.32.

Example:

```
search.pz2?session=605047297&command=record&id=3
```

Example output:

```
<record>
  <md-title>
  The Puget Sound Region : a portfolio of thematic computer maps /
  </md-title>
  <md-date>1974</md-date>
  <md-author>Mairs, John W.</md-author>
  <md-subject>Cartography</md-subject>
</record>
```

stop

Makes Pazpar2 stop further search and retrieval for busy databases.

termlist

Retrieves term list(s). Parameters:

session Session ID

name comma-separated list of termlist names. If omitted, all termlists are returned.

num maximum number of entries to return - default is 15.

version If specified and set to 2, enables Pazpar2 to return approximation on hits and counts when doing record filtering using the limit parameter on search and a limitmap with a value of "local:".

This facility, "version", appeared in Pazpar2 version 1.6.13.

Example:

```
search.pz2?session=2044502273&command=termlist&name=author,subject
```

Output:

```
<termlist>
  <activeclients>3</activeclients>
  <list name="author">
    <term>
      <name>Donald Knuth</name>
      <frequency>10</frequency>
    </term>
    <term>
      <name>Robert Pirsig</name>
      <frequency>2</frequency>
    </term>
  </list>
  <list name="subject">
    <term>
      <name>Computer programming</name>
      <frequency>10</frequency>
    </term>
  </list>
</termlist>
```

For the special termlist name "xtargets", results are returned about the targets which have returned the most hits. The 'term' subtree has additional elements, specifically a state and diagnostic field. In the example below, a target ID is returned in place of 'name'. This may or may not change later.

Example

```
<term>
  <name>library2.mcmaster.ca</name>
  <frequency>11734</frequency>          -- Number of hits
  <state>Client_Idle</state>            -- See the description of 'bytarget' ↩
      below
  <diagnostic>0</diagnostic>            -- Z39.50 diagnostic codes
</term>
```

bytarget

Returns information about the status of each active client. Parameters:

session Session ID

version If specified and set to 2, enables Pazpar2 to return approximation on hits and counts when doing record filtering using the limit parameter on search and a limitmap with a value of "local:".

This facility, "version", appeared in Pazpar2 version 1.6.13.

Example:

```
search.pz2?session=605047297&command=bytarget&id=3
```

Example output:

```
<bytarget>
  <status>OK</status>
  <target>
    <id>lx2.loc.gov:210/LCDB_MARC8</id>
    <name>Library of Congress</name>
    <hits>10000</hits>
    <diagnostic>0</diagnostic>
    <records>20</records>
    <filtered>0</filtered>
    <state>Client_Working</state>
    <query_type>pqf</query_type>
    <query_data>@attr 1=1016 birds</query_data>
  </target>
  <!-- ... more target nodes below as necessary -->
</bytarget>
```

The following client states are defined: Client_Connecting, Client_Idle, Client_Working, Client_Error, Client_Failed, Client_Disconnected.

service

Returns service definition (XML). Parameters:

session Session ID

The service command appeared in Pazpar2 version 1.6.32

SEE ALSO

Pazpar2: pazpar2(8)

Pazpar2 Configuration: pazpar2_conf(5)

4.3 Pazpar2 conf

pazpar2_conf — Pazpar2 Configuration

Synopsis

```
pazpar2.conf
```

DESCRIPTION

The Pazpar2 configuration file, together with any referenced XSLT files, govern Pazpar2's behavior as a client, and control the normalization and extraction of data elements from incoming result records, for the purposes of merging, sorting, facet analysis, and display.

The file is specified using the option -f on the Pazpar2 command line. There is not presently a way to reload the configuration file without restarting Pazpar2, although this will most likely be added some time in the future.

FORMAT

The configuration file is XML-structured. It must be well-formed XML. All elements specific to Pazpar2 should belong to the namespace `http://www.indexdata.com/pazpar2/1.0` (this is assumed in the following examples). The root element is named "`pazpar2`". Under the root element are a number of elements which group categories of information. The categories are described below.

threads

This section is optional and is supported for Pazpar2 version 1.3.1 and later. It is identified by element "`threads`" which may include one attribute "`number`" which specifies the number of worker-threads that the Pazpar2 instance is to use. A value of 0 (zero) disables worker-threads (all work is carried out in main thread).

sockets

This section is optional and is supported for Pazpar2 version 1.13.0 and later . It is identified by element "`sockets`" which may include one attribute "`max`" which specifies the maximum number of sockets to be used by Pazpar2.

file

This configuration takes one attribute `path` which specifies a path to search for local files, such as XSLTs and settings. The path is a colon separated list of directories. Its default value is "`.`" which is equivalent to the location of the main configuration file (where indeed the file element is given).

server

This section governs overall behavior of a server endpoint. It is identified by the element "server" which takes an optional attribute, "id", which identifies this particular Pazpar2 server. Any string value for "id" may be given.

The data elements are described below. From Pazpar2 version 1.2 this is a repeatable element.

listen Configures the webservice -- this controls how you can connect to Pazpar2 from your browser or server-side code. The attributes 'host' and 'port' control the binding of the server. The 'host' attribute can be used to bind the server to a secondary IP address of your system, enabling you to run Pazpar2 on port 80 alongside a conventional web server. You can override this setting on the command line using the option -h.

proxy If this item is given, Pazpar2 will forward all incoming HTTP requests that do not contain the filename 'search.pz2' to the host and port specified using the 'host' and 'port' attributes. The 'myurl' attribute is required, and should provide the base URL of the server. Generally, the HTTP URL for the host specified in the 'listen' parameter. This functionality is crucial if you wish to use Pazpar2 in conjunction with browser-based code (JS, Flash, applets, etc.) which operates in a security sandbox. Such code can only connect to the same server from which the enclosing HTML page originated. Pazpar2's proxy functionality enables you to host all of the main pages (plus images, CSS, etc.) of your application on a conventional webserver, while efficiently processing webservice requests for metasearch status, results, etc.

icu_chain Specifies character set normalization for relevancy / sorting / mergekey and facets - for the server. These definitions serve as default for services that don't have these given. For the meaning of these settings refer to the icu_chain element inside service.

relevance / sort / mergekey / facet Obsolete. Use element icu_chain instead.

settings Specifies target settings for the server. These settings serve as default for all services which don't have these given. The settings element requires one attribute 'src' which specifies a settings file or a directory. If a directory is given, all files with suffix `.xml` are read from this directory. Refer to the section called "TARGET SETTINGS" for more information.

service This nested element controls the behavior of Pazpar2 with respect to your data model. In Pazpar2, incoming records are normalized, using XSLT, into an internal representation. The 'service' section controls the further processing and extraction of data from the internal representation, primarily through the 'metadata' sub-element.

Pazpar2 version 1.2 and later allows multiple service elements. Multiple services must be given a unique ID by specifying attribute `id`. A single service may be unnamed (service ID omitted). The service ID is referred to in the init webservice command's `service` parameter.

metadata One of these elements is required for every data element in the internal representation of the record (see Section 3.2). It governs subsequent processing as pertains to sorting, relevance ranking, merging, and display of data elements. It supports the following attributes:

name This is the name of the data element. It is matched against the 'type' attribute of the 'metadata' element in the normalized record. A warning is produced if metadata elements with an unknown name are found in the normalized record. This name is also used to represent data elements in the records returned by the webservice API, and to name sort lists and browse facets.

type The type of data element. This value governs any normalization or special processing that might take place on an element. Possible values are 'generic' (basic string), 'year' (a range is computed if multiple years are found in the record). Note: This list is likely to increase in the future.

brief If this is set to 'yes', then the data element is included in brief records in the webservice API. Note that this only makes sense for metadata elements that are merged (see below). The default value is 'no'.

sortkey Specifies that this data element is to be used for sorting. The possible values are 'numeric' (numeric value), 'skiparticle' (string; skip common, leading articles), and 'no' (no sorting). The default value is 'no'.
When 'skiparticle' is used, some common articles from the English and German languages are ignored. At present the list is: 'the', 'den', 'der', 'die', 'des', 'an', 'a'.

rank Specifies that this element is to be used to help rank records against the user's query (when ranking is requested). The value is of the form

```
M [F N]
```

where M is an integer, used as a weight against the basic TF*IDF score. A value of 1 is the base, higher values give additional weight to elements of this type. The default is '0', which excludes this element from the rank calculation.
F is a CCL field and N is the multiplier for terms that matches those parts of the CCL field in search. The F+N combo allows the system to use a different multiplier for a certain field. For example, a rank value of "`1 au 3`" gives a multiplier of 3 for all terms part of the au (author) terms, and 1 for everything else.
For Pazpar2 1.6.13 and later, the rank may also be defined "per-document", by the normalization stylesheet.
The per field rank was introduced in Pazpar2 1.6.15. Earlier releases only allowed a rank value M (simple integer).
See Section 3.7 for more about ranking.

termlist Specifies that this element is to be used as a termlist, or browse facet. Values are tabulated from incoming records, and a highscore of values (with their associated frequency) is made available to the client through the webservice API. The possible values are 'yes' and 'no' (default).

merge This governs whether, and how elements are extracted from individual records and merged into cluster records. The possible values are: 'unique' (include all unique elements), 'longest' (include only the longest element (strlen)), 'range' (calculate a range of values across all matching records), 'all' (include all elements), or 'no' (don't merge; this is the default);
Pazpar2 1.6.24 also offers a new value for merge, 'first', which is like 'all' but only takes all from first database that returns the particular metadata field.

mergekey If set to '`required`', the value of this metadata element is appended to the resulting mergekey if the metadata is present in a record instance. If the metadata element is not present, then a unique mergekey will be generated instead.
If set to '`optional`', the value of this metadata element is appended to the resulting mergekey if the metadata is present in a record instance. If the metadata is not present, it will be empty.
If set to '`no`' or the mergekey attribute is omitted, the metadata will not be used in the creation of a mergekey.

facetrule Specifies the ICU rule set to be used for normalizing facets. If facetrule is omitted from metadata, the rule set 'facet' is used.

limitcluster Allow a limit on merged metadata. The value of this attribute is the name of actual

metadata content to be used for matching (most often same name as metadata name).

> **Note**
> Requires Pazpar2 1.6.23 or later.

limitmap Specifies a default limitmap for this field. This is to avoid mass configuring of targets. However it is important to review/do this on a per-target basis, since it is usually target-specific. See limitmap for format.

facetmap Specifies a default facetmap for this field. This is to avoid mass configuring of targets. However it is important to review/do this on a per-target basis, since it is usually target-specific. See facetmap for format.

icurule Specifies the ICU rule set to be used for normalizing metadata text. The "display" part of the rule is kept in the returned metadata record (record+show commands), the end result - normalized text - is used for performing within-cluster merge (unique, longest, etc.). If the icurule is omitted, type generic (text) is converted as follows: any of the characters " , / . : ([" are chopped of prefix and suffix of text content *unless* it includes the characters " : / / " (i.e. a URL).

> **Note**
> Requires Pazpar2 1.9.0 or later.

setting This attribute allows you to make use of static database settings in the processing of records. Three possible values are allowed. 'no' is the default and doesn't do anything. 'postproc' copies the value of a setting with the same name into the output of the normalization stylesheet(s). 'parameter' makes the value of a setting with the same name available as a parameter to the normalization stylesheet, so you can further process the value inside of the stylesheet, or use the value to decide how to deal with other data values.
The purpose of using settings in this way, can either be to control the behavior of normalization stylesheets in a database-dependent way, or to easily make database-dependent values available to display-logic in your user interface, without having to implement complicated interactions between the user interface and your configuration system.

xslt Defines a XSLT stylesheet. The `xslt` element takes exactly one attribute `id` which names the stylesheet. This can be referred to in target settings pz:xslt.

The content of the xslt element is the embedded stylesheet XML

icu_chain Specifies a named ICU rule set. The icu_chain element must include attribute 'id' which specifies the identifier (name) for the ICU rule set. Pazpar2 uses the particular rule sets for particular purposes. Rule set 'relevance' is used to normalize terms for relevance ranking. Rule set 'sort' is used to normalize terms for sorting. Rule set 'mergekey' is used to normalize terms for making a mergekey. Rule set 'facet' is normally used to normalize facet terms, unless facetrule is given for a metadata field.

The icu_chain element must also include a 'locale' attribute which must be set to one of the locale strings defined in ICU. The child elements listed below can be in any order, except the 'index' element which logically belongs to the end of the list. The stated tokenization, transformation and charmapping instructions are performed in order from top to bottom.

casemap The attribute 'rule' defines the direction of the per-character casemapping. Allowed values are "l" (lower), "u" (upper), "t" (title).

transform Normalization and transformation of tokens follows the rules defined in the 'rule' attribute. For possible values we refer to the extensive ICU documentation found at the ICU transformation home page. Set filtering principles are explained at the ICU set and filtering page.

tokenize Tokenization is the only rule in the ICU chain which splits one token into multiple tokens. The 'rule' attribute may have the following values: "s" (sentence), "l" (line-break), "w" (word), and "c" (character), with the latter probably not being very useful in a pruning Pazpar2 installation.

From Pazpar2 version 1.1 the ICU wrapper from YAZ is used. Refer to the yaz-icu utility for more information.

relevance Specifies the ICU rule set used for relevance ranking. The child element of 'relevance' must be 'icu_chain' and the 'id' attribute of the icu_chain is ignored. This definition is obsolete and should be replaced by the equivalent construct:

```
<icu_chain id="relevance" locale="en">..<icu_chain>
```

sort Specifies the ICU rule set used for sorting. The child element of 'sort' must be 'icu_chain' and the 'id' attribute of the icu_chain is ignored. This definition is obsolete and should be replaced by the equivalent construct:

```
<icu_chain id="sort" locale="en">..<icu_chain>
```

mergekey Specifies ICU tokenization and transformation rules for tokens that are used in Pazpar2's mergekey. The child element of 'mergekey' must be 'icu_chain' and the 'id' attribute of the icu_chain is ignored. This definition is obsolete and should be replaced by the equivalent construct:

```
<icu_chain id="mergekey" locale="en">..<icu_chain>
```

facet Specifies ICU tokenization and transformation rules for tokens that are used in Pazpar2's facets. The child element of 'facet' must be 'icu_chain' and the 'id' attribute of the icu_chain is ignored. This definition is obsolete and should be replaced by the equivalent construct:

```
<icu_chain id="facet" locale="en">..<icu_chain>
```

ccldirective Customizes the CCL parsing (interpretation of query parameter in search). The name and value of the CCL directive is given by attributes 'name' and 'value' respectively. Refer to possible list of names in the YAZ manual .

rank Customizes the ranking (relevance) algorithm. Also known as rank tweaks. The rank element accepts the following attributes - all being optional:

cluster Attribute 'cluster' is a boolean that controls whether Pazpar2 should boost ranking for merged records. Is 'yes' by default. A value of 'no' will make Pazpar2 average ranking of each record in a cluster.

debug Attribute 'debug' is a boolean that controls whether Pazpar2 should include details about ranking for each document in the show command's response. Enable by using value "yes", disable by using value "no" (default).

follow Attribute 'follow' is a floating point number greater than or equal to 0. A positive number will boost weight for terms that occur close to each other (proximity, distance). A value of

1, will double the weight if two terms are in proximity distance of 1 (next to each other). The default value of 'follow' is 0 (order will not affect weight).

lead Attribute 'lead' is a floating point number. It controls if term weight should be reduced by position from start in a metadata field. A positive value of 'lead' will reduce weight as it appears further away from the lead of the field. Default value is 0 (no reduction of weight by position).

length Attribute 'length' determines how/if term weight should be divided by length of metadata field. A value of "linear" will divide by length. A value of "log" will divide by log2(length). A value of "none" will leave term weight as is (no division). Default value is "linear".

Refer to Section 3.7 to see how these tweaks are used in computation of score.

Customization of ranking algorithm was introduced with Pazpar2 1.6.18. The semantics of some of the fields changed in versions up to 1.6.22.

sort-default Specifies the default sort criteria (default 'relevance'), which previously was hard-coded as default criteria in search. This is a fix/work-around to avoid re-searching when using target-based sorting. In order for this to work efficiently, the search must also have the sort criteria parameter; otherwise pazpar2 will do re-searching on search criteria changes, if changed between search and show command.

This configuration was added in Pazpar2 1.6.20.

settings Specifies target settings for this service. Refer to the section called "TARGET SETTINGS".

timeout Specifies timeout parameters for this service. The `timeout` element supports the following attributes: `session`, `z3950_operation`, `z3950_session` which specifies 'session timeout', 'Z39.50 operation timeout', 'Z39.50 session timeout' respectively. The Z39.50 operation timeout is the time Pazpar2 will wait for an active Z39.50/SRU operation before it gives up (times out). The Z39.50 session time out is the time Pazpar2 will keep the session alive for an idle session (no operation).

The following is recommended but not required: z3950_operation (30) < session (60) < z3950_session (180) . The default values are given in parentheses.

The Z39.50 operation timeout may be set per database. Refer to pz:timeout.

EXAMPLE

Below is a working example configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<pazpar2 xmlns="http://www.indexdata.com/pazpar2/1.0">
 <threads number="10"/>
 <file path=".:/usr/share/pazpar2/xsl"/>
 <server>
  <listen port="9004"/>
  <service>
   <rank debug="yes"/>
   <metadata name="title" brief="yes" sortkey="skiparticle"
             merge="longest" rank="6"/>
```

```
   <metadata name="isbn" merge="unique"/>
   <metadata name="date" brief="yes" sortkey="numeric"
             type="year" merge="range" termlist="yes"/>
   <metadata name="author" brief="yes" termlist="yes"
             merge="longest" rank="2"/>
   <metadata name="subject" merge="unique" termlist="yes" rank="3" limitmap ↩
      ="local:"/>
   <metadata name="url" merge="unique"/>
   <icu_chain id="relevance" locale="el">
    <transform rule="[:Control:] Any-Remove"/>
    <tokenize rule="l"/>
    <transform rule="[[:WhiteSpace:][:Punctuation:]] Remove"/>
    <casemap rule="l"/>
   </icu_chain>
   <settings src="mysettings"/>
   <timeout session="60"/>
  </service>
 </server>
</pazpar2>
```

INCLUDE FACILITY

The XML configuration may be partitioned into multiple files by using the `include` element which takes a single attribute, `src`. The `src` attribute is regular Shell-like glob pattern. For example,

```
   <include src="/etc/pazpar2/conf.d/*.xml"/>
```

The include facility requires Pazpar2 version 1.2.

TARGET SETTINGS

Pazpar2 features a cunning scheme by which you can associate various kinds of attributes, or settings with search targets. This can be done through XML files which are read at startup; each file can associate one or more settings with one or more targets. The file format is generic in nature, designed to support a wide range of application requirements. The settings can be purely technical things, like, how to perform a title search against a given target, or it can associate arbitrary name=value pairs with groups of targets -- for instance, if you would like to place all commercial full-text bases in one group for selection purposes, or you would like to control what targets are accessible to users by default. Per-database settings values can even be used to drive sorting, facet/termlist generation, or end-user interface display logic.

During startup, Pazpar2 will recursively read a specified directory (can be identified in the pazpar2.cfg file or on the command line), and process any settings files found therein.

Clients of the Pazpar2 webservice interface can selectively override settings for individual targets within the scope of one session. This can be used in conjunction with an external authentication system to determine which resources are to be accessible to which users. Pazpar2 itself has no notion of end-users, and so can be used in conjunction with any type of authentication system. Similarly, the authentication tokens submitted

to access-controlled search targets can similarly be overridden, to allow use of Pazpar2 in a consortial or multi-library environment, where different end-users may need to be represented to some search targets in different ways. This, again, can be managed using an external database or other lookup mechanism. Setting overrides can be performed either using the init or the settings webservice command.

In fact, every setting that applies to a database (except pz:id, which can only be used for filtering targets to use for a search) can be overridden on a per-session basis. This allows the client to override specific CCL fields for searching, etc., to meet the needs of a session or user.

Finally, as an extreme case of this, the webservice client can introduce entirely new targets, on the fly, as part of the init or settings command. This is useful if you desire to manage information about your search targets in a separate application such as a database. You do not need any static settings file whatsoever to run Pazpar2 -- as long as the webservice client is prepared to supply the necessary information at the beginning of every session.

Note

The following discussion of practical issues related to session and settings management are cast in terms of a user interface based on Ajax/Javascript technology. It would apply equally well to many other kinds of browser-based logic.

Typically, a Javascript client is not allowed to directly alter the parameters of a session. There are two reasons for this. One has to do with access to information; typically, information about a user will be stored in a system on the server side, or it will be accessible in some way from the server. However, since the Javascript client cannot be entirely trusted (some hostile agent might in fact 'pretend' to be a regular WS client), it is more robust to control session settings from scripting that you run as part of your webserver. Typically, this can be handled during the session initialization, as follows:

Step 1: The Javascript client loads, and asks the webserver for a new Pazpar2 session ID. This can be done using a Javascript call, for instance. Note that it is possible to submit Ajax HTTPXmlRequest calls either to Pazpar2 or to the webserver that Pazpar2 is proxying for. Refer to Pazpar2 protocol(7).

Step 2: Code on the webserver authenticates the user, by database lookup, LDAP access, NCIP, etc., and determines which resources the user has access to, and any user-specific parameters that are to be applied during this session.

Step 3: The webserver initializes a new Pazpar2 settings, and sets user-specific parameters as necessary, using the init webservice command. A new session ID is returned.

Step 4: The webserver returns this session ID to the Javascript client, which then uses the session ID to submit searches, show results, etc.

Step 5: When the Javascript client ceases to use the session, Pazpar2 destroys any session-specific information.

SETTINGS FILE FORMAT

Each file contains a root element named <settings>. It may contain one or more <set> elements. The settings and set elements may contain the following attributes. Attributes in the set node override those in the setting root element. Each set node must specify (directly, or inherited from the parent node) at least a target, name, and value.

target This specifies the search target to which this setting should be applied. Targets are identified by their Z39.50 URL, generally including the host, port, and database name, (e.g. `z3950.indexdata.com:210`. Two wildcard forms are accepted: * (asterisk) matches all known targets; `z3950.indexdata.com:210` matches all known databases on the given host.

A precedence system determines what happens if there are overlapping values for the same setting name for the same target. A setting for a specific target name overrides a setting which specifies target using a wildcard. This makes it easy to set defaults for all targets, and then override them for specific targets or hosts. If there are multiple overlapping settings with the same name and target value, the 'precedence' attribute determines what happens.

For Pazpar2 1.6.4 or later, the target ID may be user-defined, in which case, the actual host, port, etc., is given by the setting pz:url.

name The name of the setting. This can be anything you like. However, Pazpar2 reserves a number of setting names for specific purposes, all starting with 'pz:', and it is a good idea to avoid that prefix if you make up your own setting names. See below for a list of reserved variables.

value The value of the setting. Generally, this can be anything you want -- however, some of the reserved settings may expect specific kinds of values.

precedence This should be an integer. If not provided, the default value is 0. If two (or more) settings have the same content for target and name, the precedence value determines the outcome. If both settings have the same precedence value, they are both applied to the target(s). If one has a higher value, then the value of that setting is applied, and the other one is ignored.

By setting defaults for target, name, or value in the root settings node, you can use the settings files in many different ways. For instance, you can use a single file to set defaults for many different settings, like search fields, retrieval syntaxes, etc. You can have one file per server, which groups settings for that server or target. You could also have one file which associates a number of targets with a given setting, for instance, to associate many databases with a given category or class that makes sense within your application.

The following examples illustrate uses of the settings system to associate settings with targets to meet different requirements.

The example below associates a set of default values that can be used across many targets. Note the wildcard for targets. This associates the given settings with all targets for which no other information is provided.

```
<settings target="*">

<!-- This file introduces default settings for pazpar2 -->

<!-- mapping for unqualified search -->
<set name="pz:cclmap:term" value="u=1016 t=l,r s=al"/>

<!-- field-specific mappings -->
<set name="pz:cclmap:ti" value="u=4 s=al"/>
<set name="pz:cclmap:su" value="u=21 s=al"/>
<set name="pz:cclmap:isbn" value="u=7"/>
<set name="pz:cclmap:issn" value="u=8"/>
<set name="pz:cclmap:date" value="u=30 r=r"/>
```

```
    <set name="pz:limitmap:title" value="rpn:@attr 1=4 @attr 6=3"/>
    <set name="pz:limitmap:date" value="ccl:date"/>

    <!-- Retrieval settings -->

    <set name="pz:requestsyntax" value="marc21"/>
    <set name="pz:elements" value="F"/>

    <!-- Query encoding -->
    <set name="pz:queryencoding" value="iso-8859-1"/>

    <!-- Result normalization settings -->

    <set name="pz:nativesyntax" value="iso2709"/>
    <set name="pz:xslt" value="marc21.xsl"/>

    </settings>
```

The next example shows certain settings overridden for one target, one which returns XML records containing Dublin Core elements, and which furthermore requires a username/password.

```
    <settings target="funkytarget.com:210/db1">
    <set name="pz:requestsyntax" value="xml"/>
    <set name="pz:nativesyntax" value="xml"/>
    <set name="pz:xslt" value="../etc/dublincore.xsl"/>

    <set name="pz:authentication" value="myuser/password"/>
    </settings>
```

The following example associates a specific name/value combination with a number of targets. The targets below are access-restricted, and can only be used by users with special credentials.

```
    <settings name="pz:allow" value="0">
    <set target="funkytarget.com:210/*"/>
    <set target="commercial.com:2100/expensiveDb"/>
    </settings>
```

RESERVED SETTING NAMES

The following setting names are reserved by Pazpar2 to control the behavior of the client function.

pz:allow Allows or denies access to the resources it is applied to. Possible values are ’0’ and ’1’. The default is ’1’ (allow access to this resource).

pz:apdulog If the ’pz:apdulog’ setting is defined and has other value than 0, then Z39.50 APDUs are written to the log.

pz:authentication Sets an authentication string for a given database. For Z39.50, this is carried as part of the Initialize Request. In order to carry the information in the "open" elements, separate username and password with a slash (In Z39.50 it is a VisibleString). In order to carry the information in the idPass elements, separate username term, password term and, optionally, a group term with a single blank. If three terms are given, the order is *user, group, password*. If only two terms are given, the order is *user, password*.

For HTTP based protocols, such as SRU and Apache Solr, the authentication string includes a username term and, optionally, a password term. Each term is separated by a single blank. The authentication information is passed either by HTTP basic authentication or via URL parameters. The mode of operation is determined by `pz:authentication_mode` setting.

pz:authentication_mode Determines how authentication is carried in HTTP based protocols. Value may be "`basic`" or "`url`".

pz:block_timeout (Not yet implemented). Specifies the time for which a block should be released anyway.

pz:cclmap:xxx This establishes a CCL field definition or other setting, for the purpose of mapping end-user queries. XXX is the field or setting name, and the value of the setting provides parameters (e.g. parameters to send to the server, etc.). Please consult the YAZ manual for a full overview of the many capabilities of the powerful and flexible CCL parser.

Note that it is easy to establish a set of default parameters, and then override them individually for a given target.

pz:elements The element set name to be used when retrieving records from a server.

pz:extendrecs If a show command goes to the boundary of a result set for a database - depends on sorting - and pz:extendrecs is set to a positive value. then Pazpar2 wait for show to fetch pz:extendrecs more records. This setting is best used if a database does native sorting, because the result set otherwise may be completely re-sorted during extended fetch. The default value of pz:extendrecs is 0 (no extended fetch).

Warning
The pz:extendrecs setting appeared in Pazpar2 version 1.6.26. But the behavior changed with the release of Pazpar2 1.6.29.

pz:facetmap:*name* Specifies that for field *name*, the target supports (native) facets. The value is the name of the field on the target.

pz:facetmap:split:*name* Like pz:facetmap, but makes Pazpar2 inspect the term value consisting of two items separated by colon. First item is the raw ID to be sent to database if limitmap on the field *name* is used. The second item is the display term.

This facility was added in Pazpar2 version 1.11.0.

pz:id This setting can't be 'set' -- it contains the ID (normally ZURL) for a given target, and is useful for filtering -- specifically when you want to select one or more specific targets in the search command.

pz:limitmap:*name* Specifies attributes for limiting a search to a field - using the limit parameter for search. It can be used to filter locally or remotely (search in a target). In some cases the mapping of a field to a value is identical to an existing cclmap field; in other cases the field must be specified in a different way - for example to match a complete field (rather than parts of a subfield).

The value of limitmap may have one of three forms: referral to an existing CCL field, a raw PQF string or a local limit. Leading string determines type; either `ccl:` for CCL field, `rpn:` for PQF/RPN, or `local:` for filtering in Pazpar2. The local filtering may be followed by a field a metadata field (default is to use the name of the limitmap itself).

For Pazpar2 version 1.6.23 and later the limitmap may include multiple specifications, separated by `,` (comma). For example: `ccl:title,local:ltitle,rpn:@attr 1=4`.

Note

The limitmap facility is supported for Pazpar2 version 1.6.0. Local filtering is supported in Pazpar2 1.6.6.

---

pz:maxrecs Controls the maximum number of records to be retrieved from a server. The default is 100.

pz:memcached If set and non-empty, libMemcached will configured and enabled for the target. The value of this setting is same as the ZOOM option `memcached`, which in turn is the configuration string passed to the `memcached` function of libMemcached.

This setting is honored in Pazpar2 1.6.39 or later. Pazpar2 must be using YAZ version 5.0.13 or later.

pz:redis If set and non-empty, redis will be configured and enabled for the target. The value of this setting is exactly as the redis option for ZOOM C of YAZ. Refer to the YAZ manual.

This setting is honored in Pazpar2 1.6.43 or later. Pazpar2 must be using YAZ version 5.2.0 or later.

pz:nativesyntax Specifies how Pazpar2 should map retrieved records to XML. Currently supported values are `xml`, `iso2709` and `txml`.

The value `iso2709` makes Pazpar2 convert retrieved MARC records to MARCXML. In order to convert to XML, the exact character set of the MARC must be known (if not, the resulting XML is probably not well-formed). The character set may be specified by adding: `;`*`charset`* to `iso2709`. If omitted, a charset of MARC-8 is assumed. This is correct for most MARC21/USMARC records.

The value `txml` is like `iso2709` except that records are converted to TurboMARC instead of MARCXML.

The value `xml` is used if Pazpar2 retrieves records that are already XML (no conversion takes place).

pz:negotiation_charset Sets character set for Z39.50 negotiation. Most targets do not support this, and some will even close connection if set (crash on server side or similar). If set, you probably want to set it to `UTF-8`.

pz:piggyback Piggybacking enables the server to retrieve records from the server as part of the search response in Z39.50. Almost all servers support this (or fail it gracefully), but a few servers will produce undesirable results. Set to '1' to enable piggybacking, '0' to disable it. Default is 1 (piggybacking enabled).

pz:pqf_prefix Allows you to specify an arbitrary PQF query language substring. The provided string is prefixed to the user's query after it has been normalized to PQF internally in pazpar2. This allows you to attach complex 'filters' to queries for a given target, sometimes necessary to select sub-catalogs in union catalog systems, etc.

pz:pqf_strftime Allows you to extend a query with dates and operators. The provided string allows certain substitutions and serves as a format string. The special two character sequence '%%' gets converted to the original query. Other characters leading with the percent sign are conversions supported by strftime. All other characters are copied verbatim. For example, the string `@and @attr 1=30 @attr 2=3 %Y %%` would search for current year combined with the original PQF (%%).

This setting can also be used as more general alternative to pz:pqf_prefix -- a way of embedding the submitted query anywhere in the string rather than appending it to prefix. For example, if it is desired to omit all records satisfying the query `@attr 1=pica.bib 0007` then this subquery can be combined with the submitted query as the second argument of `@andnot` by using the pz:pqf_strftime value `@not %% @attr 1=pica.bib 0007`

pz:preferred Specifies that a target is preferred, e.g. possible local, faster target. Using block=preferred on show command will wait for all these targets to return records before releasing the block. If no target is preferred, the block=preferred will be identical to block=1, which release when one target has returned records.

pz:present_chunk Controls the chunk size in present requests. Pazpar2 will make (maxrecs / chunk) request(s). The default is 20.

pz:queryencoding The encoding of the search terms that a target accepts. Most targets do not honor UTF-8 in which case this needs to be specified. Each term in a query will be converted if this setting is given.

pz:recordfilter Specifies a filter which allows Pazpar2 to only include records that meet a certain criteria in a result. Unmatched records will be ignored. The filter takes the form name, name~value, or name=value, which will include only records with metadata element (name) that has the substring (~value) given, or matches exactly (=value). If value is omitted all records with the named metadata element present will be included.

pz:requestsyntax This specifies the record syntax to use when requesting records from a given server. The value can be a symbolic name like marc21 or xml, or it can be a Z39.50-style dot-separated OID.

pz:sort Specifies sort criteria to be applied to the result set. Only works for targets which support the sort service.

pz:sortmap:*`field`* Specifies native sorting for a target where *`field`* is a sort criterion (see command show). The value has two components separated by a colon: strategy and native-field. Strategy is one of `z3950`, `type7`, `cql`, `sru11`, or `embed`. The second component, native-field, is the field that is recognized by the target.

Note
Only supported for Pazpar2 1.6.4 and later.

---

pz:sru This setting enables SRU/Solr support. It has four possible settings. 'get', enables SRU access through GET requests. 'post' enables SRU/POST support, less commonly supported, but useful if very large requests are to be submitted. 'soap' enables the SRW (SRU over SOAP) variation of the protocol.

A value of 'solr' enables Solr client support. This is supported for Pazpar version 1.5.0 and later.

pz:sru_version This allows SRU version to be specified. If unset Pazpar2 will use the default of YAZ (currently 1.2). Should be set to 1.1 or 1.2. For Solr, the current supported/tested version is 1.4 and 3.x.

pz:termlist_term_count Specifies number of facet terms to be requested from the target. The default is unspecified e.g. server-decided. Also see pz:facetmap.

pz:termlist_term_factor Specifies whether to use a factor for pazpar2 generated facets (1) or not (0). When mixing locally generated (by the downloaded (pz:maxrecs) samples) facet with native (target-generated) facets, the later will dominate the facet list since they are generated based on the complete result set. By scaling up the facet count using the ratio between total hit count and the sample size, the total facet count can be approximated and thus better compared with native facets. This is not enabled by default.

pz:timeout Specifies timeout for operation (e.g. search, and fetch) for a database. This overrides the z3650_operation timeout that is given for a service. See timeout.

Note

The timeout facility is supported for Pazpar2 version 1.8.4 and later.

---

pz:url Specifies URL for the target and overrides the target ID.

Note

`pz:url` is only recognized for Pazpar2 1.6.4 and later.

---

pz:xslt A comma-separated list of stylesheet names that specifies how to convert incoming records to the internal representation.

For each name, the embedded stylesheets (XSL) that comes with the service definition are consulted first, and take precedence over external files; see xslt of service definition). If the name does not match an embedded stylesheet, it is considered a filename.

The suffix of each file specifies the kind of tranformation. Suffix "`.xsl`" makes an XSL transform. Suffix "`.mmap`" will use the MMAP transform (described below).

The special value "`auto`" will use a file which is the pz:requestsyntax's value followed by `'.xsl'`.

When mapping MARC records, XSLT can be bypassed for increased performance with the alternate "MARC map" format. Provide the path of a file with extension ".mmap" containing on each line:

```
<field> <subfield> <metadata element>
```

For example:

```
245 a title
500 $ description
      773 * citation
```

To map the field value, specify a subfield of '$'. To store a concatenation of all subfields, specify a subfield of '*'.

pz:zproxy The 'pz:zproxy' setting has the value syntax 'host.internet.adress:port'. It is used to tunnel Z39.50 requests through the named Z39.50 proxy.

SEE ALSO

pazpar2(8) yaz-icu(1) pazpar2_protocol(7)

4.4 Pazpar2_play

pazpar2_play — Play recorded HTTP log against pazpar2

Synopsis

`pazpar2_play` [`-v` *`level`*] [file] [server-addr]

DESCRIPTION

pazpar2_play is a utility that plays recorded HTTP activity against pazpar2. Pazpar2 may record activity to a file with option -R. This utility allows you to play the activity again against pazpar2. Pazpar2 must use predictable sessions in order for this to work. This means that either Pazpar2 should be executed with option -R or with debug mode (-X).

OPTIONS

`-v` *`level`* Sets log level (YAZ log level system).

EXAMPLES

Typical usage. First step is to enable recording in pazpar2. Something like:

```
pazpar2 -f config.xml -R /tmp/recording.log
```

For the RPM version of Pazpar2, add/modify `/etc/sysconfig/pazpar2` and modify OPTIONS. For the Debian version of Pazpar2, add/modify `/etc/defaults/pazpar2` and modify OPTIONS.

At some point we want to run recording, perhaps against another Pazpar2 instance for analysis. We start Pazpar2 with '-R -' to ensure that sessions are numbered in the same way as initial recording.

```
pazpar2 -f config.xml -R -
```

We can now run the player against it:

```
pazpar2_play /tmp/recording.log localhost:9004
```

SEE ALSO

Pazpar2 daemon: pazpar2(8)

Appendix A

License

Appendix B

GNU General Public License

B.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software - to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and
2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

B.2 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

B.2.1 Section 0

This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

B.2.2 Section 1

You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

B.2.3 Section 2

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that

you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: If the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

B.2.4 Section 3

You may copy and distribute the Program (or a work based on it, under Section 2 in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

B.2.5 Section 4

You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

B.2.6 Section 5

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

B.2.7 Section 6

Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

B.2.8 Section 7

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

B.2.9 Section 8

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

B.2.10 Section 9

The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

B.2.11 Section 10

If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

B.2.12 NO WARRANTY Section 11

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

B.2.13 Section 12

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING

ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

B.3 How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type "show w". This is free software, and you are welcome to redistribute it under certain conditions; type "show c" for details.

The hypothetical commands "show w" and "show c" should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than "show w" and "show c"; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program "Gnomovision" (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.