

Customizing and extending Evergreen: a guide for geeks

OLA Preconference

Dan Scott
February 24, 2010

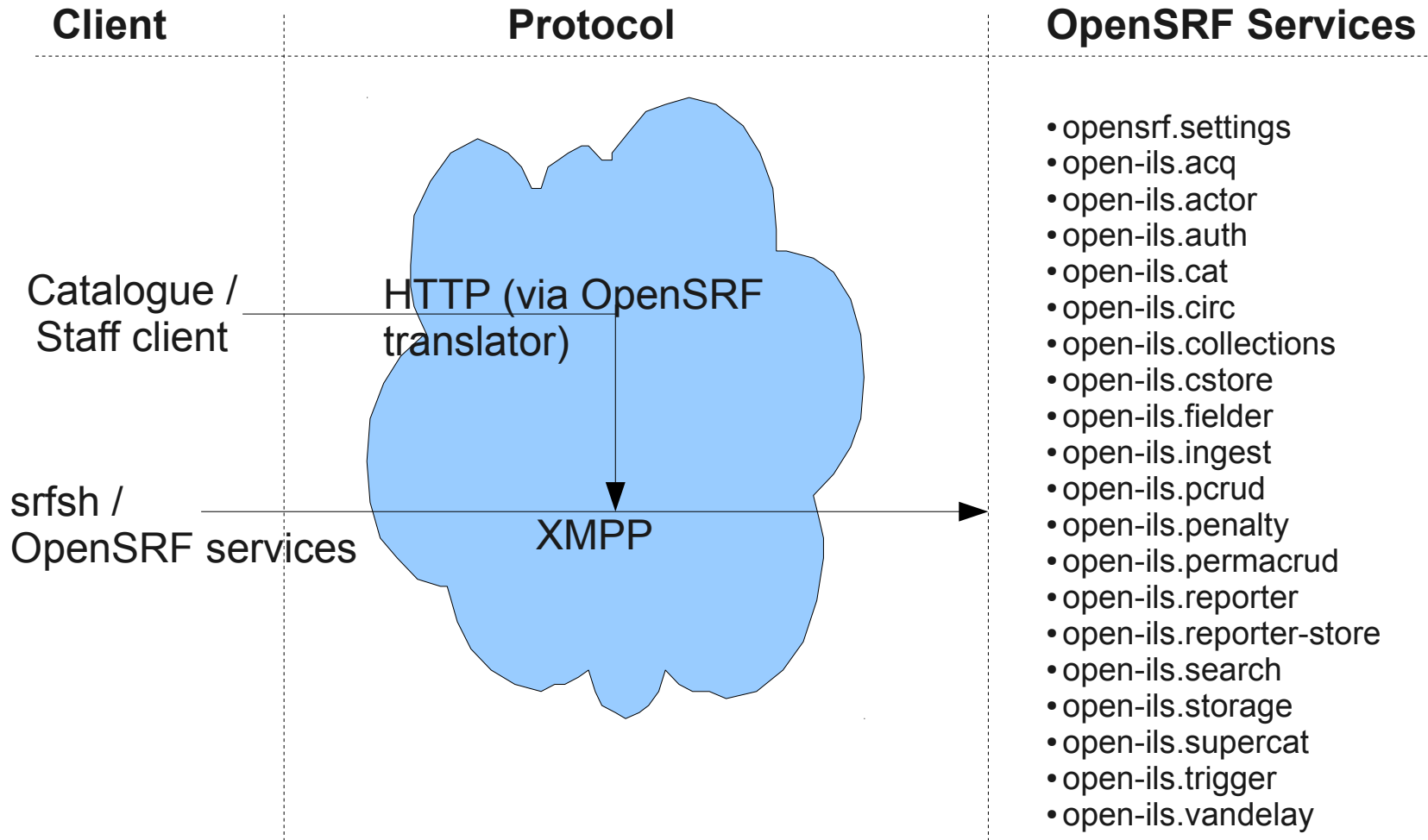
Agenda

- Let's be honest, 3 hours isn't much, but...
 - 1:00 – 1:45 – OpenSRF intro
 - 1:45 – 2:00 – Client exercises
 - 2:00 – 2:30 – OpenSRF services
 - 2:30 – 3:00 – Service exercises
 - 3:00 – 3:30 – Evergreen IDL and storage
 - 3:30 – 3:45 – Storage exercises
 - 3:45 – 4:00 – Dojo widgets

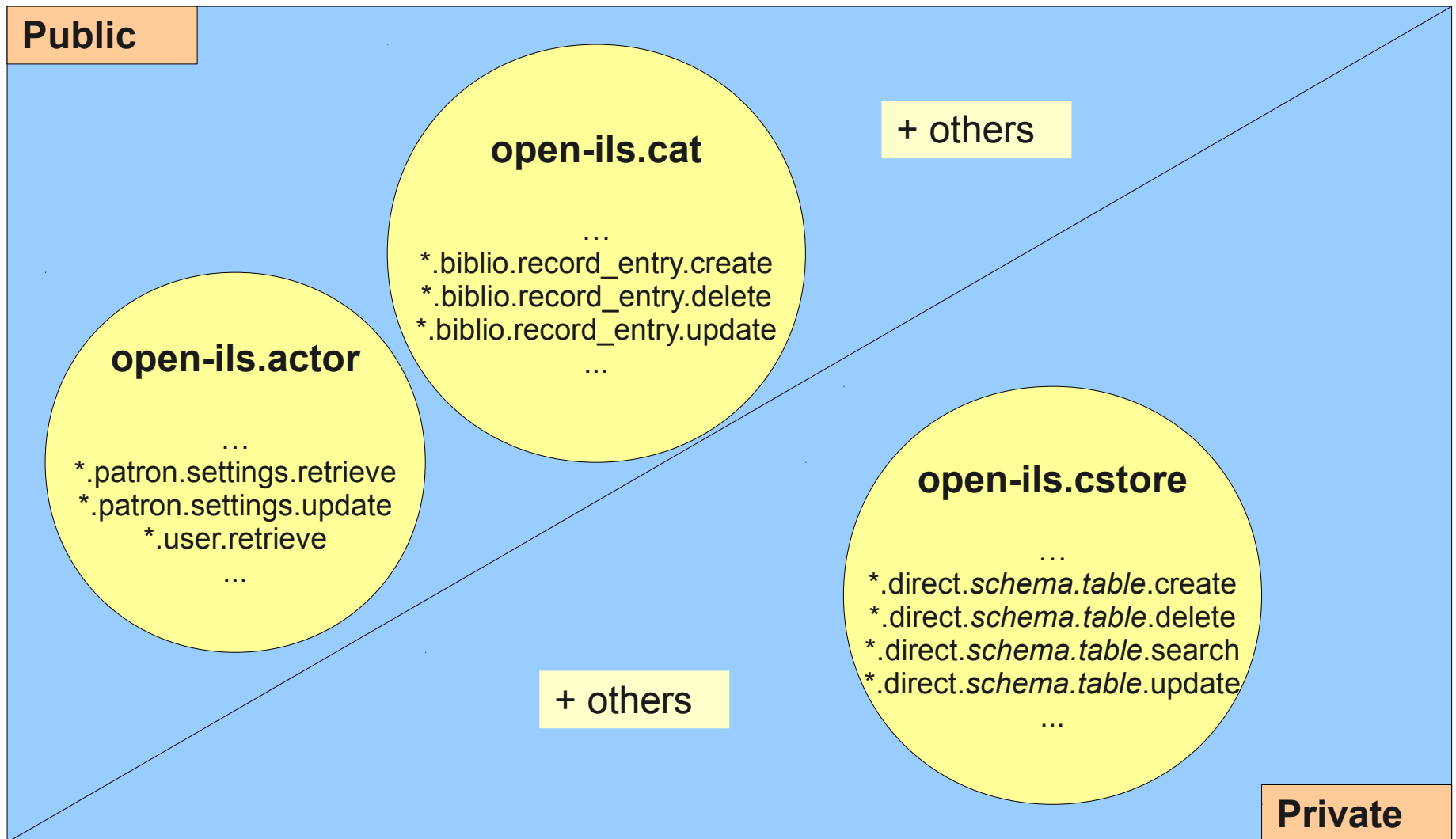
Workshop materials

- There are a number of CDs floating around with the following materials:
 - Virtual image – should run in recent versions of VMWare or VirtualBox, given 1 GB of RAM or more
 - This presentation (OpenOffice.org and PDF)
 - Developer reference material (HTML and PDF)

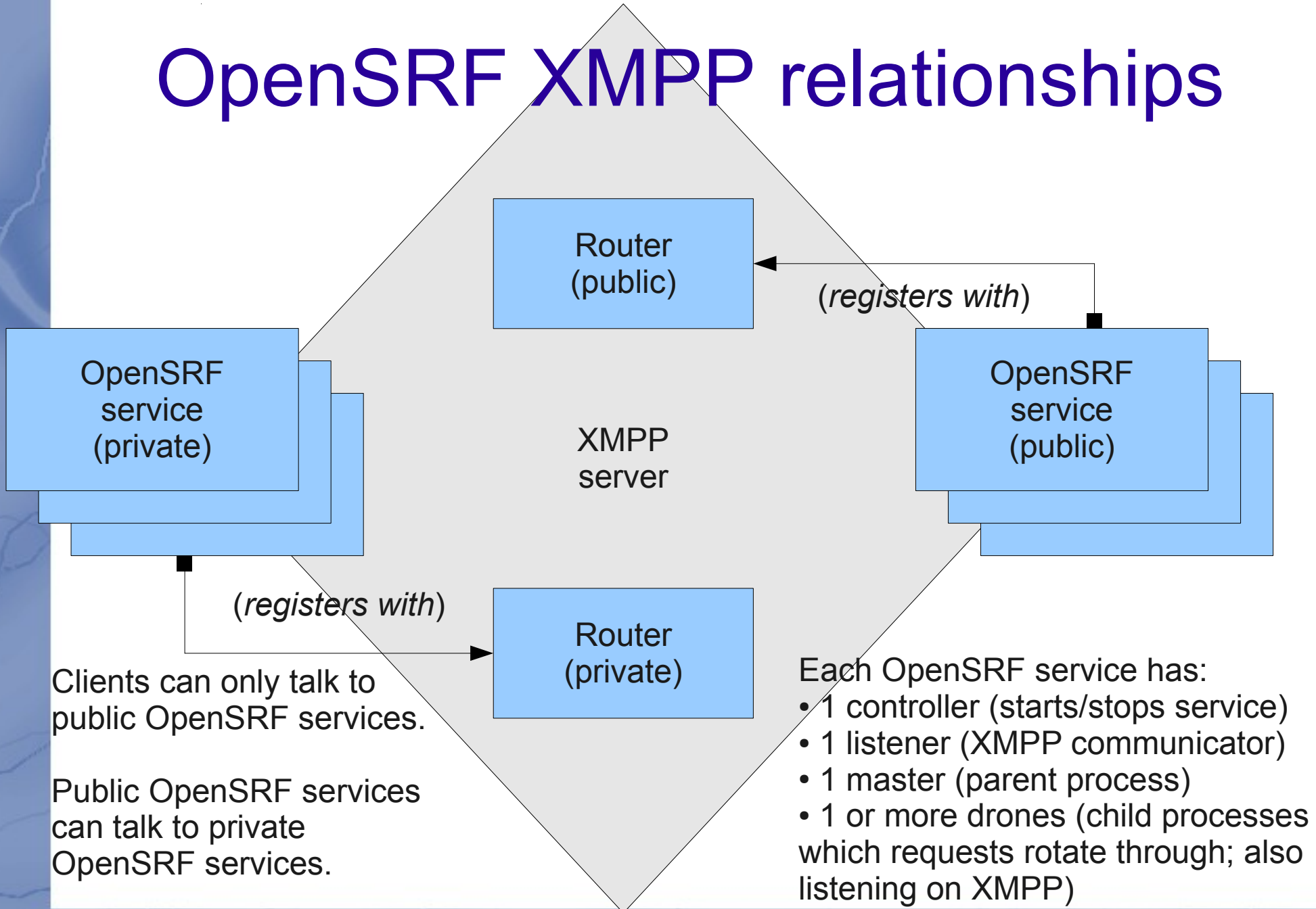
Basic Evergreen architecture



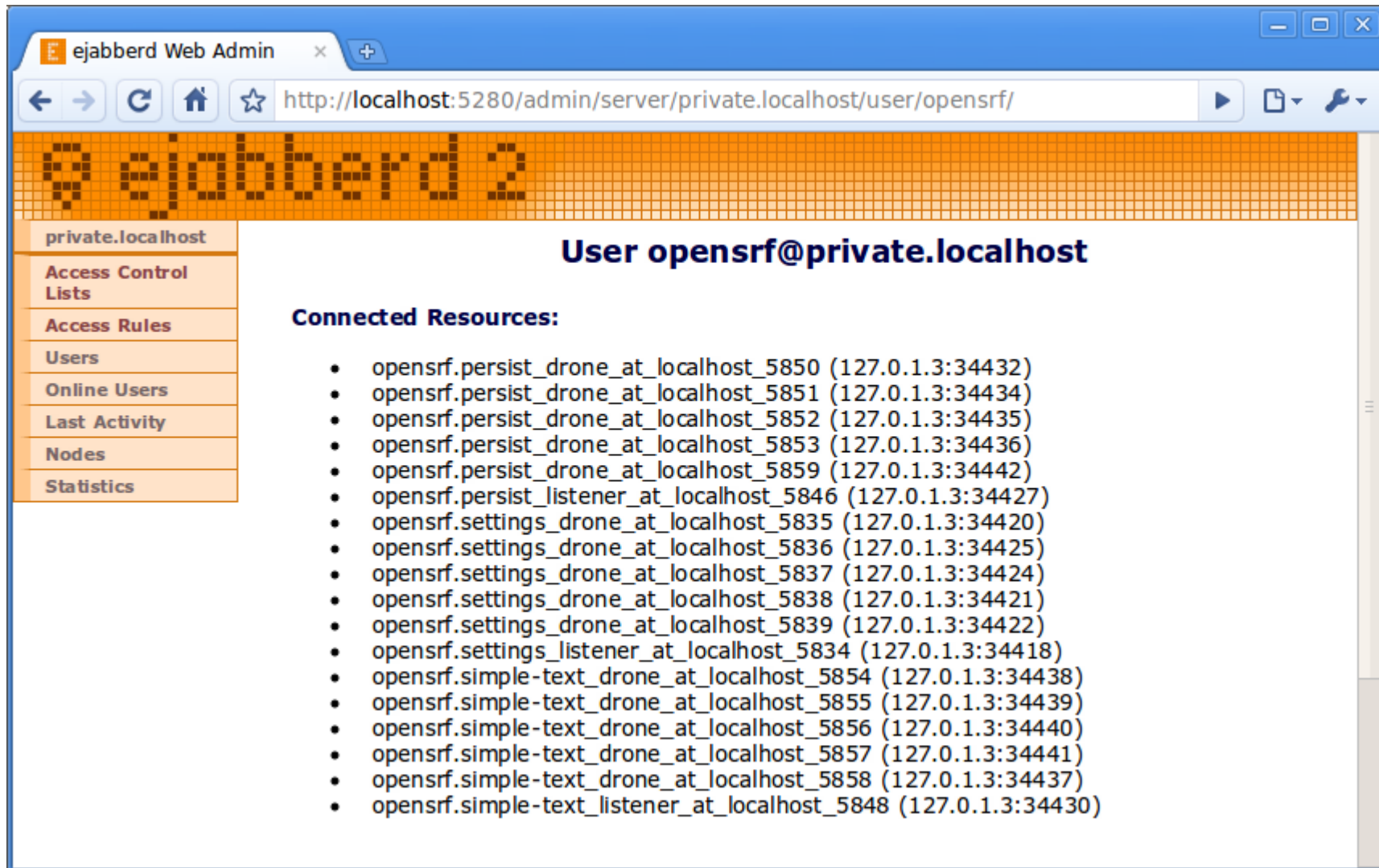
Evergreen: OpenSRF services



OpenSRF XMPP relationships



Private opensrf resources



The screenshot shows the ejabberd Web Admin interface. The browser address bar indicates the URL: `http://localhost:5280/admin/server/private.localhost/user/opensrf/`. The page title is "User opensrf@private.localhost". On the left, there is a navigation menu with the following items: "private.localhost", "Access Control Lists", "Access Rules", "Users", "Online Users", "Last Activity", "Nodes", and "Statistics". The main content area is titled "Connected Resources:" and contains a list of 20 resources, each with a unique name and IP address/ports.

private.localhost

User opensrf@private.localhost

Connected Resources:

- opensrf.persist_drone_at_localhost_5850 (127.0.1.3:34432)
- opensrf.persist_drone_at_localhost_5851 (127.0.1.3:34434)
- opensrf.persist_drone_at_localhost_5852 (127.0.1.3:34435)
- opensrf.persist_drone_at_localhost_5853 (127.0.1.3:34436)
- opensrf.persist_drone_at_localhost_5859 (127.0.1.3:34442)
- opensrf.persist_listener_at_localhost_5846 (127.0.1.3:34427)
- opensrf.settings_drone_at_localhost_5835 (127.0.1.3:34420)
- opensrf.settings_drone_at_localhost_5836 (127.0.1.3:34425)
- opensrf.settings_drone_at_localhost_5837 (127.0.1.3:34424)
- opensrf.settings_drone_at_localhost_5838 (127.0.1.3:34421)
- opensrf.settings_drone_at_localhost_5839 (127.0.1.3:34422)
- opensrf.settings_listener_at_localhost_5834 (127.0.1.3:34418)
- opensrf.simple-text_drone_at_localhost_5854 (127.0.1.3:34438)
- opensrf.simple-text_drone_at_localhost_5855 (127.0.1.3:34439)
- opensrf.simple-text_drone_at_localhost_5856 (127.0.1.3:34440)
- opensrf.simple-text_drone_at_localhost_5857 (127.0.1.3:34441)
- opensrf.simple-text_drone_at_localhost_5858 (127.0.1.3:34437)
- opensrf.simple-text_listener_at_localhost_5848 (127.0.1.3:34430)

Public router resources



The screenshot shows the ejabberd Web Admin interface in a browser window. The address bar displays the URL `http://localhost:5280/admin/server/public.localhost/user/router/`. The page title is "ejabberd 2". On the left, a sidebar menu lists navigation options: "public.localhost", "Access Control Lists", "Access Rules", "Users", "Online Users", "Last Activity", "Nodes", and "Statistics". The main content area is titled "User router@public.localhost" and displays the following information:

- Connected Resources:**
 - opensrf.simple-text (127.0.1.2:39788)
 - router (127.0.1.2:39772)
- Password:** A password field with seven dots and a "Change Password" button.
- Roster:** A "Remove User" button.

The footer of the page reads "ejabberd (c) 2002-2009 ProcessOne".

Private router resources

The screenshot shows the ejabberd Web Admin interface in a browser window. The address bar displays the URL `http://localhost:5280/admin/server/private.localhost/user/router/`. The page features a navigation sidebar on the left with the following menu items: `private.localhost`, `Access Control Lists`, `Access Rules`, `Users`, `Online Users`, `Last Activity`, `Nodes`, and `Statistics`. The main content area is titled `User router@private.localhost` and contains the following sections:

- Connected Resources:**
 - opensrf.persist (127.0.1.3:34429)
 - opensrf.settings (127.0.1.3:34423)
 - opensrf.simple-text (127.0.1.3:34433)
 - router (127.0.1.3:34416)
- Password:**
 - A password input field with masked characters (dots).
 - A `Change Password` button.
- Roster:**
 - A `Remove User` button.

The footer of the page contains the text: `ejabberd (c) 2002-2009 ProcessOne`.

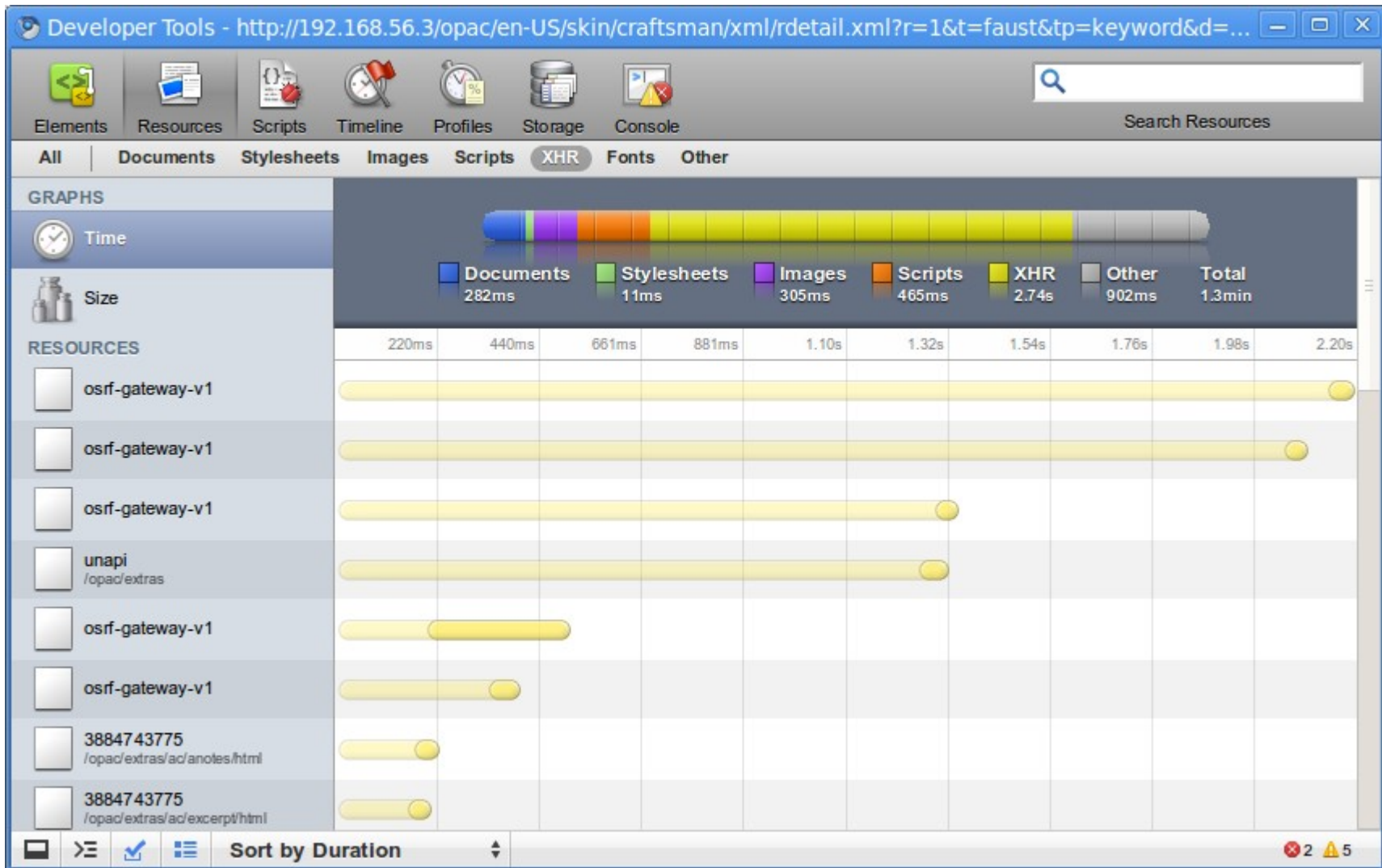
A raw OpenSRF request (XMPP)

```
<message from='router@private.localhost/opensrf.simple-text'  
  to='opensrf@private.localhost/  
    opensrf.simple-text_listener_at_localhost_6275'  
  router_from='opensrf@private.localhost/_karmic_126678.3719_6288'  
  router_to='' router_class='' router_command='' osrf_xid=''  
>  
<thread>1266781414.366573.12667814146288</thread>  
<body>  
[  
  {"__c": "osrfMessage", "__p":  
    {"threadTrace": "1", "locale": "en-US", "type": "REQUEST", "payload":  
      {"__c": "osrfMethod", "__p":  
        {"method": "opensrf.simple-text.reverse", "params": ["foobar"]}  
      }  
    }  
  }  
]  
</body>  
</message>
```

A raw OpenSRF response

```
<message from='opensrf@private.localhost/
  opensrf.simple-text_drone_at_localhost_6285'
  to='opensrf@private.localhost/_karmic_126678.3719_6288'
  router_command='' router_class='' osrf_xid=''
>
<thread>1266781414.366573.12667814146288</thread>
<body>
[
  {"__c": "osrfMessage", "__p":
    {"threadTrace": "1", "payload":
      {"__c": "osrfResult", "__p":
        {"status": "OK", "content": "raboof", "statusCode": 200}
      } , "type": "RESULT", "locale": "en-US"}
    },
  {"__c": "osrfMessage", "__p":
    {"threadTrace": "1", "payload":
      {"__c": "osrfConnectStatus", "__p":
        {"status": "Request Complete", "statusCode": 205}
      } , "type": "STATUS", "locale": "en-US"}
    }
]
</body>
</message>
```

Seeing what's going on



Making client requests

- In the OpenSRF source directory, see:
 - math_bench.pl (Perl)
 - math_client.py (Python)
 - math_client_curl.sh (via HTTP translator)
- Via srfsh:

```
srfsh# introspect opensrf.simple-text  
# lists methods
```

```
srfsh# request opensrf.simple-text opensrf.simple-text.reverse  
"foobar"  
"raboof"
```

Simple Perl client

```
#!/usr/bin/perl
use strict;
use warnings;

use OpenSRF::System;

# Bootstrap the system
OpenSRF::System->bootstrap_client('/openils/conf/opensrf_core.xml');

my $session = OpenSRF::AppSession->create("opensrf.simple-text");

my $response = $session->request(
    "opensrf.simple-text.reverse",
    "foobar"
)->gather();

$session->disconnect();

print "$response\n";
```

Review: client exercises

- Get comfortable with srfsh, cURL, and Perl
 1. List the methods for opensrf.simple-text
 2. Call each of the methods for opensrf.simple-text via srfsh
 3. Try calling several of the methods via cURL and Perl

* Sadly, Python's pyDNS requires a local DNS resolver that makes it a complete pain to deal with in a localhost environment.

Writing OpenSRF services

- Basic structure of an OpenSRF service:
 - initialize() - occurs once, for the master; good for setting up global variables
 - child_init() - invoked every time a new child process is spawned
 - *method1()*, *method2()* - implementations of OpenSRF methods
 - registration – registers the method
 - child_exit() - cleans up child resources
 - destroy() - cleans up global resources

Creating and configuring

Let's look at the recently-added-to-trunk
OpenSRF::Application::Demo::SimpleText.pm

(in /usr/local/share/perl/5.10.0/)

Then we'll look at opensrf.xml and opensrf_core.xml
to see how the service is defined

OpenSRF server exercises

1. Add another method to SimpleText.pm and test it out (remember to restart the Perl services!)
2. Create an entirely new service:
 1. Copy and paste SimpleText.pm and change its file name & package name (at a minimum)
 2. Add the appropriate entries to opensrf_core.xml and opensrf.xml
 3. Restart the Perl services and test it out

Evergreen: accessing ILS data

- There are a number of different services that offer access to Evergreen data:
 - open-ils.storage
 - open-ils.cstore (+ CStoreEditor wrapper)
 - open-ils.reporter-store
 - open-ils.permacrud
 - open-ils.pcrud
 - openils.PermaCrud (JavaScript)

Evergreen IDL

- Interface Definition Language
- IDL + fieldmapper = object-database persistence
- IDL → JavaScript: occurs via autogen.sh
- Bonus: automatic addition of *isnew* / *isdeleted* / *ischanged* setter methods as of Evergreen 1.6

IDL sample class

```
<class id="clm" controller="open-ils.cstore open-ils.pcrud"
  oils_obj:fieldmapper="config::language_map"
  oils_persist:tablename="config.language_map"
  reporter:label="Language Map" oils_persist:field_safe="true">
  <fields oils_persist:primary="code" oils_persist:sequence="">
    <field reporter:label="Language Code" name="code"
      reporter:selector="value" reporter:datatype="text"/>
    <field reporter:label="Language" name="value"
      reporter:datatype="text" oils_persist:i18n="true"/>
  </fields>
  <links/>
  <permacrud xmlns="http://open-ils.org/spec/opensrf/IDL/permacrud/v1">
    <actions>
      <create global_required="true" permission="CREATE_MARC_CODE">
      <retrieve global_required="true"
        permission="CREATE_MARC_CODE UPDATE_MARC_CODE
          DELETE_MARC_CODE">
      <update global_required="true" permission="UPDATE_MARC_CODE">
      <delete global_required="true" permission="DELETE_MARC_CODE">
    </actions>
  </permacrud>
</class>
```

CStore - open-ils.cstore

- Automatically generated object interfaces

```
open-ils.cstore.direct.config.language_map.id_list  
  {"code" { "like": "e%" } }
```

```
open-ils.cstore.direct.config.language_map.retrieve "eng"
```

```
open-ils.cstore.direct.config.language_map.search  
  {"code" : "eng"}
```

```
open-ils.cstore.direct.config.language_map.create <object>
```

```
open-ils.cstore.direct.config.language_map.update <object>
```

```
open-ils.cstore.direct.config.language_map.delete "eng"
```


CStoreEditor

- Provides a friendly Perl interface to Cstore
 - *CRUD_fieldmap*(\$obj)
 - *batch_retrieve_fieldmap*(\$id_listref)
 - *retrieve_all_fieldmap*()
 - *search_fieldmap*{search_hash}

```
require OpenLS::Utils::CStoreEditor;
my $e = OpenLS::Utils::CStoreEditor->new(xact => 1);

my $user = $e->search_actor_user(
    { family_name => { "ilike" => "green" } }
);

$e->rollback;
```

pcrud - open-ils.cstore

- Automatically generated *permission-aware* object interfaces

```
open-ils.pcrud.id_list.clm <authtoken>,  
  { "code": { "like": "e%" } }
```

```
open-ils.pcrud.retrieve.clm <authtoken>, "eng"
```

```
open-ils.pcrud.search.clm <authtoken>, { "code": "eng" }
```

```
open-ils.pcrud.create.clm <authtoken>, <object>
```

```
open-ils.pcrud.update.clm <authtoken>, <object>
```

```
open-ils.pcrud.delete.clm <authtoken>, "eng"
```

JavaScript: openils.PermaCrud

```
// limited to objects controlled by open-ils.pcrud!  
dojo.require('openils.PermaCrud');  
  
var pcrud = new openils.PermaCrud();  
  
var recs = pcrud.retrieveAll(classHint, opts);  
var rec = pcrud.retrieve(classHint, ID, opts);  
var result = pcrud.create/update/eliminate/apply(list, opts);  
var recs = pcrud.search(classHint, jsonQuery, opts);
```

Where **opts** includes:

- async: boolean
- timeout: int
- streaming: boolean
- oncomplete / onresponse: callback
- select / order_by / limit / offset: jsonQuery clauses

Data exercises

- Base examples and guides:
 - test_cstore_editor.pl: Perl (CStoreEditor)
 - bibs.tt2: JavaScript (openils.PermaCrud)
 - json_tutorial.html: JSON query guide
- Find all copies in the system and display their barcodes
- Change the call number for the book authored by “Faust” to “FAUSTCALL”

State of the art

- For the best examples from Bill Erickson, check out `/openils/var/web/templates/`
 - `./default/actor/user/register.tt2` maps to `http://localhost/eg/actor/user/register`
- Build on Template Toolkit, so the source `.tt2` files look a bit odd – but “View source” works quite nicely

AutoGrid + EditPane

Billing Types

✓	Name	Org Unit	Default Price
<input type="checkbox"/>	Overdue Materials	CONS	
<input type="checkbox"/>	Long Overdue Collection Fee	CONS	
<input type="checkbox"/>	Lost Materials	CONS	
<input type="checkbox"/>	Lost Materials Processing Fee	CONS	
<input type="checkbox"/>	System: Deposit	CONS	
<input type="checkbox"/>	System: Rental	CONS	
<input type="checkbox"/>	Damaged Item	CC	
<input type="checkbox"/>	Damaged Item Processing Fee	CC	
<input type="checkbox"/>	Notification Fee	CC	
<input type="checkbox"/>	Misc	CC	

✕

ID

Name

Org Unit ▼

Default Price

AutoFieldWidget

Age Hold Protection	<input type="text"/>
Alert Message	<input type="text"/>
Barcode	<input type="text" value="10102020"/>
Call Number/Volume	<input type="text"/>
Can Circulate	<input checked="" type="checkbox"/>
Circulating Library	<input type="text" value="BR1"/>
Circulation Modifier	<input type="text"/>
Circulation Type (MARC)	<input type="text"/>
Copy ID	<input type="text" value="1"/>
Copy Number on Volume	<input type="text"/>
Copy Status	<input type="text"/>
Copy Status Changed Time	<input type="text" value="10/28/2009"/>
Creating User	<input type="text" value="1"/>
Creation Date/Time	<input type="text" value="10/28/2009"/>
Deposit Amount	<input type="text" value="0.00"/>
Dummy ISBN	<input type="text"/>
Fine Level	<input type="text" value="2"/>
Is Deleted	<input type="checkbox"/>
Is Deposit Required	<input type="checkbox"/>
Is Holdable	<input checked="" type="checkbox"/>
Is Reference	<input type="checkbox"/>
Last Edit Date/Time	<input type="text" value="10/28/2009"/>
Last Editing User	<input type="text" value="1"/>
Loan Duration	<input type="text" value="2"/>

License

This presentation is licensed under a Creative Commons Attribution-Share Alike 2.5 Canada License.