IXP Manager Workshop

V4 Deep Dive

Barry O'Donovan & Nick Hilliard

27th Euro-IX Forum

Berlin, Germany – October 25th 2015

Introduction / House Keeping / Agenda

- What do you want to cover here?
- Suggestions / prepared stuff:
  - Current wish lists / plans
  - A deep look at IXP Manager V4
    - My toolstack for those interested
  - Long term goal – funding, sponsorship, etc.
  - Some practical scripting

Current Wish Lists / Plans

Current Wish Lists / Plans (and WIP)

- Helpdesk Integration
- L2 ACLs
- Advancing P2P functionality
- Reseller ports via p-tag
- Patch panel management
- *Intelligent* Provisioning
  - Port configuration templates and automatic configuration

L2 ACLs

- Dynamic port security *just does not work* in an IXP
  - Most of us have already moved to static L2 ACLs
- Need this functionality supported in IXP Manager:
  - Prevent driver error / PEBKAC
  - All customers to update L2ACLs -> no more 2AM phone calls
- Usual constraints exist:
  - Security
  - Switch vendor agnostic with reference implementation(s)
  - Some knobs and dials

Advancing P2P Functionality

- Currently P2P functionality is via RRDs
  - This limits feature development for anything but simply p2p graphs
- Evaluating OpenTSDB and Graphite/Carbon/Whisper
  - Whisper is like RRD in that it is a fixed size database
  - However, OpenTSDB is much more complex and is distributed
- Disk IO is an issue – but not insurmountable
- With a more database like backend, we can present more interesting details to our customers: top n peers, *burstiest* peers, presentation of the data can become a lot more user friendly, …

Reseller Ports via p-tag

- IXP Manager supports the *fan-out* port model for resellers as is
- New equipment at INEX allows the p-tag model
- This breaks a lot of things:
  - Port / traffic graphs via mrtg / snmp
  - MAC address learning for p2p graphs
  - We have two members where we have no visibility ☹
- L2ACLs + advances in P2P / sflow collection should fix all this ☺

Patch Panel Management

- Currently we manage ~25 data centre patch panels via wiki ☹
- Have hoped to implement this in IXP Manager for ~7 years ☹
- A number of false starts. Why?
  - We've been completely over thinking this with wiz-bang features
  - We just need the wiki equivlent for crying out loud!
  - No fancy graphing, no managing internal interrack panels…

Patch Panel Management

- Patch Panel Management to:
  - Record patch panel reference and media type, number of ports
  - On a per port basis, assign to a customer
  - Record the data centre assigned ID
  - Connection (and disconnection) date
  - Optionally assign to a switch port
    - Need to manage customer port moves as a process now!
  - Or optionally assign to customer co-located equipment
- No more bike-shedding this! (ourselves included!)

Intelligent Provisioning

- An often requested feature – even internally at INEX
- Create a provisioning flow for common tasks such as:
  - Joining the exchange (we still use a Word document, yuck! ☹)
  - Provisioning a port -> very manual

Intelligent Provisioning

- Provisioning a port -> very manual
  - Assign port
  - Assign patch panel port and send cross connect details
  - Configure port (quarantine)
  - Confirm cross connect is in place and record ID
  - Test port and ensure speed / duplex are matched
  - Quarantine port (check traffic, check BGP routes advertised)
  - Put port live in production peering LAN
  - Setup route collector, route server and AS112 sessions
  - Send announcement

Intelligent Provisioning

- Provisioning a port -> very manual
  - Assign port
  - Assign patch panel port and send cross connect details
  - Configure port (quarantine)
  - Confirm cross connect is in place and record ID
  - Test port and ensure speed / duplex are matched
  - Quarantine port (check traffic, check BGP routes advertised)
  - Put port live in production peering LAN
  - Setup route collector, route server and AS112 sessions
  - Send announcement

Intelligent Provisioning

- Provisioning a port -> very manual
  - Assign port and patch panel port and send cross connect details
  - Configure port (quarantine)
  - Confirm cross connect is in place and record ID
  - Test port and ensure speed / duplex are matched
  - Quarantine port (check traffic, check BGP routes advertised)
  - Put port live in production peering LAN
  - Setup route collector, route server and AS112 sessions
  - Send announcement

Intelligent Provisioning

- So, how do we design a provisioning system for every IXP?
- Carefully. Intelligently. Dynamically.
- Each step has common attributes:
  - Action pending customer or IXP
  - Test to determine if step is complete correctly / incorrectly
  - Action to take on completion?
  - Messaging
  - Advance to next step
- Create a framework using contracts to build up *a process with steps*

Intelligent Provisioning
Port Configuration Templates

- Standardised port configurations at (obviously) essential at an IXP
- INEX performed a forklift upgrade from Brocade to Extreme recently
- Used IXP Manager's database to configure all ports on new switches
  - Time efficient, no operator errors, ensured standard configs without missing elements
- Could be easily integrated into IXP Manager for copy and paste, but:
  - Would prefer it to actually: shutdown port + wipe + configure + enable again
  - Would also like it to validate RANCID/Oxidized configs for errors / ommissions
- Vendor agnostic!

IXP Manager

V4

IXP Manager V4

- Major version changes usually mean major changes
  - This is true here -> but mainly on the backend
- IXP Manager is your typical MVC stack
  - We're completely changing two elements of this:
    - The controller
    - The view

IXP Manager V4 – Doctrine ORM

- Let's talk about the model first: the database
- Layer between MySQL/MariaDB and the PHP application is Doctrine ORM
- V3 of IXP Manager changed from Doctrine 1.2 to Doctrine 2
  - Active model to ORM (Object Relational Mapping)

- An example:

```php
<?php
$c = new Entities\Customer;
$c->setName( "Big ISP Ltd" );
$c->setAutsys(64496);
// ...
D2EM::persist( $c );
```

```
$custRepo
   = D2EM->getRepository( “Entities\Customer” );

$customers = $custRepo->findAll();

foreach( $customers as $c ) {
   // do something
}
```

IXP Manager V4 – Doctrine ORM

- Stable project – around since 2006
- It's actually a number of libraries that stack together. Mainly:
  - Doctrine DBAL: Database Abstraction Layer
  - Doctrine ORM: Object Relational Mapper/ing

- http://www.doctrine-project.org/

- Key concepts:
  - Entities: represent a single database row from a given table
  - Proxies: *compiled* entities that are fully transparent to your code.
    - Allows for lazy loading, loading incomplete details, etc.
  - Repositories: handles sets of entities
    - Most of our complex queries are handled by way of proxies:

```
class Customer extends EntityRepository {
   /**
    * Utility function to provide a array of
    * all active and current customers.
    */
   public function getCurrentActive(
     $asArray =false, $trafficing =false,
     $externalOnly = false, $ixp = false )
   {
        …
   }
   …
}
```

- How did we start talking about Doctrine?
- IXP Manager is your typical MVC stack
  - We're completely changing two elements of this:
    - The controller
    - The view
- Let's look at the view next…

IXP Manager V4 – The View

- What's a *view component*? Why do we have it / need it?
  - Separates logic from presentation
  - In larger projects, UI designers don't need to be able to code to manage the frontend
  - Eradicates spaghetti code
  - Allows for templating with layouts
  - Allows for skinning
- For years, the only game in town in PHP was Smarty
  - It's godawful ☹ It stinks to high hell.
  - Okay, that's pretty harsh. It's of its' age and hasn't moved forward…

- Contenders to Smarty:
- Twig
  - modern OOP design
  - good extensibility
  - well supported and widely used
- Blade
  - the built-in defacto view for Laravel

- Problem: views add developer overhead: more syntax, libraries, functions, etc. to learn ☹

- PHP Plates
  - Native PHP templates – no new syntax to learn
  - Inspired by Twig
  - Supports layouts and inheritance
  - Easy to extend with fucntions and extensions

  - NB: Plates is a full template system, not spagetti code by another name

- In reality, IXP Manager v4 supports Smarty, Blade and Plates out of the box.

- The biggest part of MVC is C -> the controller
- Handles:
  - Routing of requests
  - Middleware
  - Input validation
  - Controllers
  - Responses

- Our new controller is Laravel. Not just a controller, a framework.

- Why change framework at all?
  - Developer apathy which leads to:
  - Stagnation of the code base
  - New or prospective developers are turned off
  - New features remain unimplemented because there are *better ways*
  - Stay modern to leverage new techniques and services

- The Laravel **framework** provides new techniques and integrations:
  - Service provider framework
  - Events
  - Queues
  - Task scheduling
  - Testing
  - Migrations
  - Package management

- We can't throw away the existing code base though.
- Over the course of V4's lifetime, we'll migrate from Zend to Laravel
  - i.e. Zend/Smarty will co-exist with Laravel/Plates for quite some time
  - This means new features can be implemented immediately using the latest technologies
  - We don't need to disappear for six months to rewrite the entire code base
- How will this be achieved?
  - An (in)elegent solution!

- Laravel is now the default framework and routes requests
- If a request hits Laravel for a route that does not exist / is not implement in Laravel:
  - It throws a 404 exception
- In app/Http/Kernel.php we catch that 404 exception
  - And spin up the Zend Framework
- Zend will then handle if possible or throw another 404
  - (handled as a page not found in Zend and presented to the user)

IXP Manager V4 – The Controller

```
try {
    return $this->sendRequestThroughRouter($request);
} catch( \Symfony\Component\HttpKernel\Exception\NotFoundHttpException $e ) {

   require_once 'Zend/Application.php';

   $application = new \Zend_Application(
        APPLICATION_ENV, APPLICATION_PATH . '/configs/application.ini'
    );

    $application->bootstrap()->run();
}
```

- Events provide a simple observer implementation
  - **You can subscribe and listen for events in packages / extentions!**
  - Events can be *fired* when something *significant* happens
  - Event listeners can queue the event for offline processing
- Laravel queues support Beanstalkd, IronMQ, Amazon SQS, Redis and synchronous (local, immediate)
- What kind of things can we do with this..?

- Physical interface changed in IXP Manager: fire **physIntChangedEvent**
  - MRTG listener can check for port or port speed change and regenerate MRTG configuration and reload the daemon
  - Billing notifications listener can check for speed change and email *accounts* for billing purposes
  - Switch configuration listener can roll out configuration change to switch (fires event)
    - Physical interface status listener can inspect interface for matching speed / duplex
  - Patch panel listener could take *some* action if a port is changed that has a connected cross connect

- VLAN interface changed in IXP Manager: fire **vlanIntChangedEvent**
  - AS112 listener can (de)configure BGP session as necessary
  - Route collector can (de) configure BGP session as necessary
  - Route servers can (de) configure BGP session as necessary
  - Other event listeners may include: regenerating Smokeping & Nagios configurations, DNS PTR entries.
  - Enabling IPv6 could additionally send email with details or start the IPv6 enable process to walk the customer through configurating sessions to route collector, servers, as112, etc.

IXP Manager V4 – Service Providers

- Central piece of Larabel's application bootstrapping
- Registers:
  - Controllers and routes
  - Event listeners
  - Middleware
- Can be used to extend IXP Manager without hacking the main codebase

IXP Manager V4 – Contracts

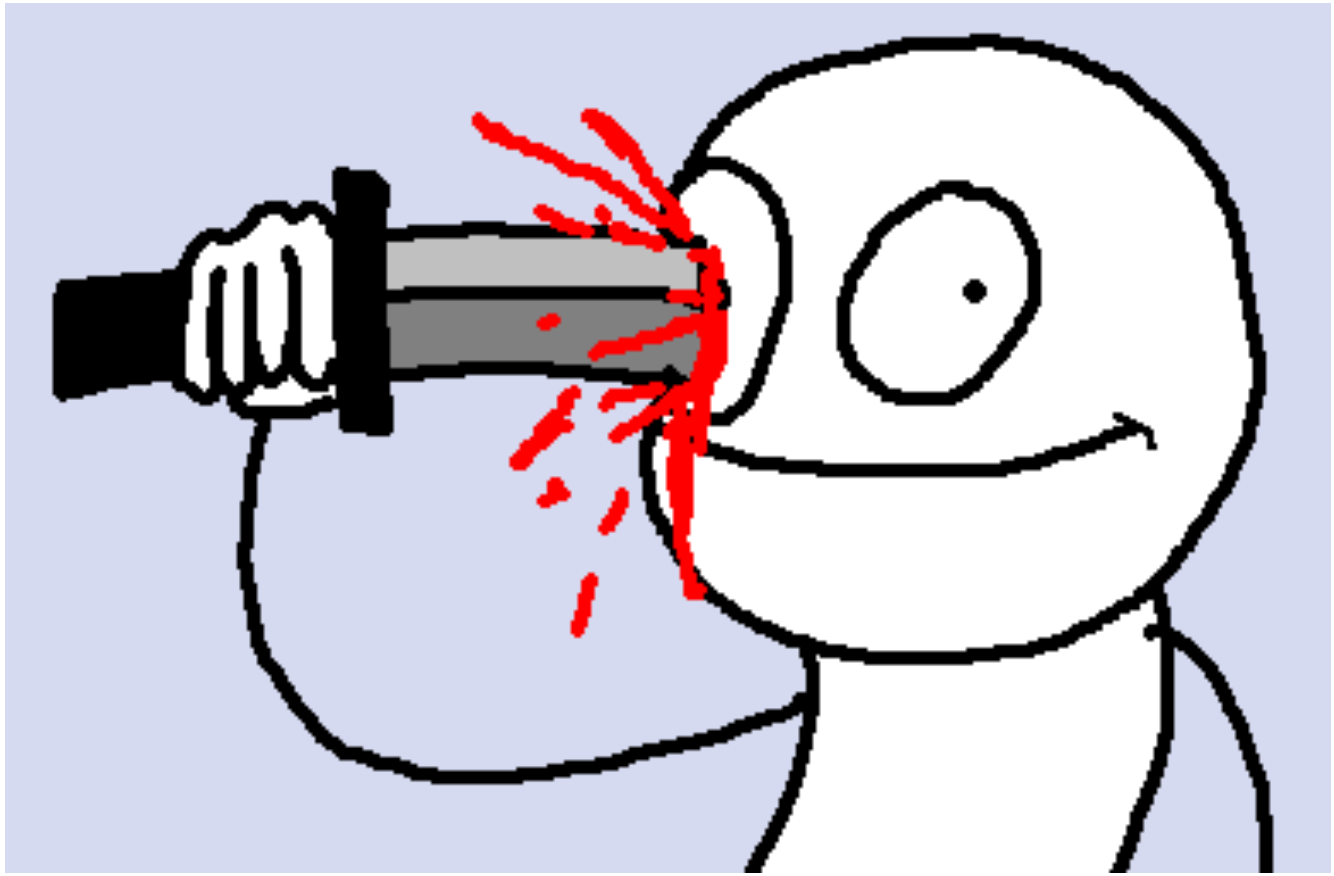
- Interfaces that define core services provided by Laravel
- Also how we will develop extensions to IXP Manager
  - Design a contract
  - Develop reference implementation(s) to that contract
- Example: Helpdesk integration
  - First a bit of history….

New Helpdesk - operations@inex.ie

- Up to 2008 - Shared IMAP Mailbox
- 2009 - Cerberus
  - Served us well but extreme feature creep
- 2013 - Realisation that we need something new
  - Helpdesk research => maximum pain

From an INEX Members' Update

New Helpdesk - operations@inex.ie



From an INEX Members' Update

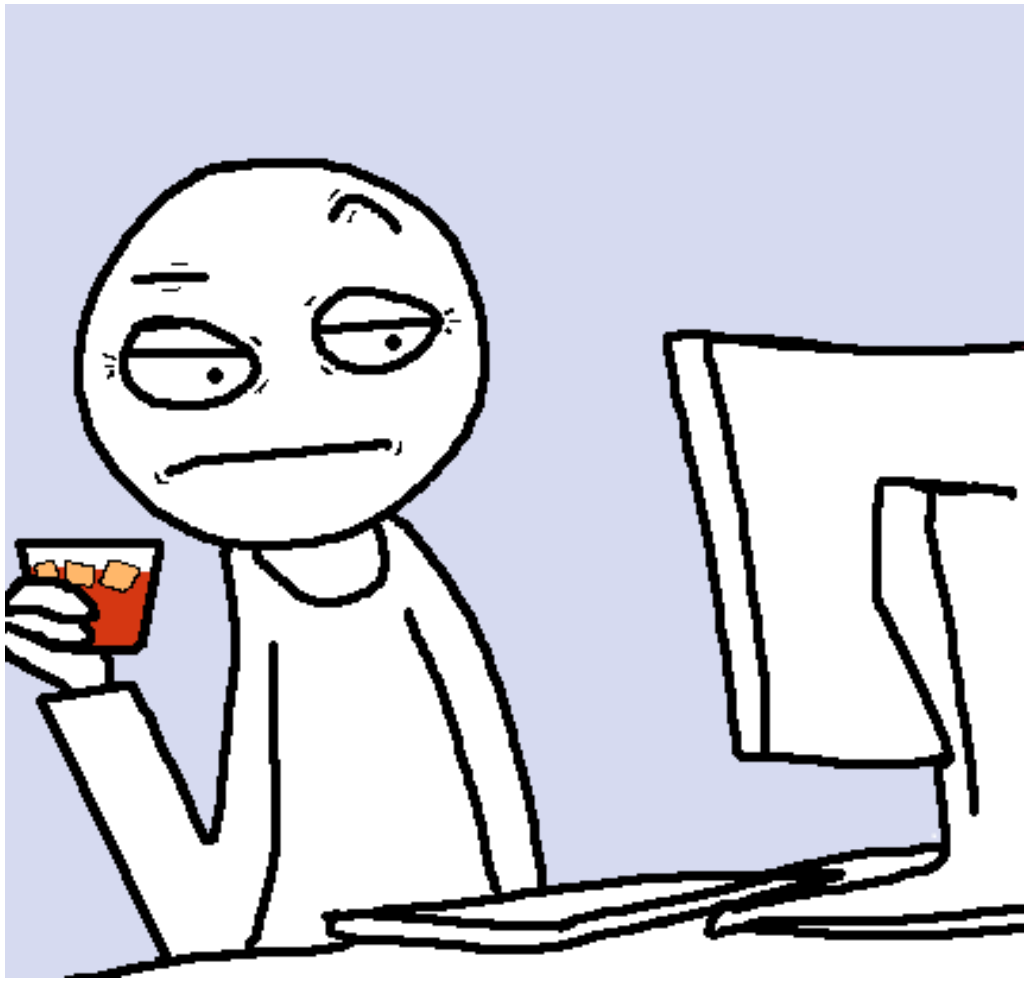
New Helpdesk - operations@inex.ie

inex internet neutral exchange

- Up to 2008 - Shared IMAP Mailbox
- 2009 - Cerberus
  - Server us well but extreme feature creep
- 2013 - Realisation that we need something new
  - Helpdesk research => maximum pain
  - Pain so great, we stuck with what we had
- 2015 - Try again, new methodology => pain killerz

From an INEX Members' Update

New Helpdesk - operations@inex.ie



From an INEX Members' Update

New Helpdesk - operations@inex.ie

- Candidates included:
  - Freshdesk, GrooveHQ, Zendesk, Kayako, Cerb5 and many more…
- Contrary to my initial preconceptions, the winner was Zendesk
  - Excellent API (essential as we needed to import old tickets)
  - Easily configurable triggers and automations
  - Supports markdown
  - Nice UI plus iOS / Android apps
  - Poor reporting (@ Zendesk Regular anyway

IXP Manager V4 – Helpdesk Integration

- Need to be able to:
  - Create customers (organisations) on the helpdesk system
  - Create users on the helpdesk system
  - Find tickets by organisation
  - Create tickets
  - Update / close tickets
- It's work in progress but most of the integration is done

- Contract: app/Contracts/Helpdesk.php
  - ticketsFindAll()
  - organisationNeedsUpdating($custLocal, $custHelpdesk)
  - organisationCreate($cust)
  - organisationUpdate($helpdeskId, $customer)
  - organisationFind($id)
  - contactNeedsUpdating(…)
  - userCreate()
  - userUpdate()
  - …

IXP Manager V4 – Helpdesk Integration

- Zendesk Reference Implementation
  - app/Services/Helpdesk/Zendesk.php
- Service Provider
  - app/Providers/HelpdeskServiceProvider.php
  - This file needs updating for new implementations
- Instantiation:
  - $helpdesk = App::make('IXP\Contracts\Helpdesk');
- Configuration:
  - config/helpdesk.php
  - Environment configuration via PHP DotEnv

- Sample .env for Zendesk:

```
HELPDESK_BACKEND=zendesk
HELPDESK_ZENDESK_SUBDOMAIN=ixp
HELPDESK_ZENDESK_TOKEN=yyy
HELPDESK_ZENDESK_EMAIL=john.doe@example.com
```

IXP Manager V4 – Statistics Backend

- IXP Manager currently only supports MRTG/log for port stats
- From earlier, we also want to support port stats via sflow / p2p
  - Would also like to support MRTG/rrd
- DE-CIX have offered a bounty for this work
- Will be implemented in the same way as the helpdesk
  - i.e. any backend could be substitued once it is implemented against the provided contract
- Three reference implementations: sflow/p2p, MRTG/log, MRTG/rrd

IXP Manager V4 – Installation Changes

- The PHP development tool chain has changed since v3
  - Git submodules no longer necessary
- PHP has a package management system called composer
  - All third party dependancies now installed via composer
  - Includes: ZF1, Laravel, Smarty, Plates, Zendesk API, Doctrine, etc.
- Frontend assets handled similarly via bower
  - Includes jquery, Bootstrap, etc.

IXP Manager V4 – Vagrant

inex
internet neutral exchange

Development environments made easy.

Create and configure lightweight, reproducible, and portable development environments.

IXP Manager V4 – Vagrant

1. Install Vagrant (http://www.vagrantup.com/)
2. Install VirtualBox (http://www.virtualbox.org/)
3. Clone IXP Manager, check out v4 and install dependancies:

```
git clone https://github.com/inex/IXP-Manager.git  ixpmanager
cd ixpmanager
git checkout v4

composer update
```

4. Start-up Vagrant: `vagrant up`
5. This will take a while – it executes bootstrap.sh which will:
   - apt-get update, upgrade and install all dependancies for IXP Manager's LAMP environment
   - Configure MySQL and phpMyAdmin
   - Install composer and bower
   - Configure and populate the IXP Manager database with sample data
   - Configure Apache and IXP Manager

6. Once it's complete, you can:

- Access IXP Manager at: http://localhost:8088/
  - Admin username and password: vagrant / vagrant1
- SSH into the virtual machine with: `vagrant ssh`
- Your ixpmanager directory is mounted under /vagrant
- MySQL is available via: mysql –u root –ppassword ixp
  - Or http://localhost:8088/phpmyadmin

- Managing your VM:
  - Shutdown cleanly by logging in and: `sudo shutdown -h now`
  - To suspend: `vagrant suspend`
  - To force shutdown: `vagrant halt`
  - To bring up: `vagrant up`
  - For status: `vagrant status`

IXP Manager V4 – Documentation

- Remember: v4 is a bridging version from ZF1 to Laravel
- As such, documentation is also a halfway house
- Existing documentation available at:
  - https://github.com/inex/IXP-Manager/wiki
- New documentation will be:
  - https://ixp-manager.readthedocs.org/en/latest/
  - Source: https://github.com/inex/ixp-manager-docs

IXP Manager V4 – My Tool Stack

- Apple OSX with Homebrew for:
  - php, bash, bgpq3, git, joe, mariadb, node, sshfs and much more
- Atom as a text editor
  - With language-php, linter (same for CSS, JS, etc) and Dash
- Vagrant (latest Ubuntu LTS)
- Git, GitHub, TravisCI
- Skipper (ORM GUI, http://www.skipper18.com/)

Quick Coding Example?

Quick Coding Example

- Let's make a new Artisan command
  - Artisan is the CLI component of Laravel

```
./artisan make:console DemoListCustomers
```

Quick Coding Example

- Now let's edit the resultant file:
  - `app/Console/Commands/DemoListCustomers.php`

- Give the command a name and description
- And let's see if it works…

Quick Coding Example

- And let's see if it works… nope ☹

- Extending the wrong class – need to use and extend:
  - use IXP\Console\Commands\Command as IXPCommand;
  - => class DemoListCustomers extends IXPCommand {
- No options / arguments required
- Need to register the command in app/Console/Kernel:
  - protected $commands = […]
- Now it works! But does nothing…

Quick Coding Example

- Let's get and list all customers:
- Complete the fire() method:

```
$customers =
    \D2EM::getRepository( 'Entities\Customer' )->getCurrentActive();

foreach( $customers as $c )
    $this->info( $c->getName() );
```

Thanks for listening!

operations@inex.ie

https://github.com/inex/IXP-Manager

Mailing list:

https://www.inex.ie/mailman/listinfo/ixpmanager