

# Zenoss Core 4 Event Architecture

Jane Curry

[jane.curry@skills-1st.co.uk](mailto:jane.curry@skills-1st.co.uk)

based on the paper at

<http://community.zenoss.org/docs/DOC-13746> or

<http://www.skills-1st.co.uk/papers/jane/zenoss4-events/>



# Agenda

- Quick look at Zenoss 3 architecture
- Zenoss 4 architecture
  - subsystems
  - daemons
  - databases
- 10 minute comfort break (that's 600 seconds!)
- Event life cycle
  - Event generation
  - Device context
  - Event class mapping / event context / event transforms
  - Database insertion & duplicates
  - Resolution & Ageing

# In the beginning...

- Events received by various daemons
- Events **processed** by zenhub daemon
- Event life cycle
  - Event generation
  - Event class mapping
  - Event transforms
  - Resolution
  - Device context
  - Event context
  - Database insertion & deduplication
  - Ageing
- MySQL events database
  - **summary** table for active events
  - **history** table for closed events
- Database fields = Event console fields  
= Event mapping / transform fields
- Table definitions in  
**\$ZENHOME/Products/ZenEvents/db/zenevents.sql**

# In the beginning...

- Events subsystem was a severe bottleneck
- zenhub had lots of other responsibilities
- Single pipeline for processing events
- Almost no indexing on MySQL database
- Event Console updates slow
- Event deletions very slow
- Responsiveness of emails / pages / automation commands slow
- Debugging hard (zenhub.log & event.log)
- 
- Fairly easy to understand

```
CREATE TABLE IF NOT EXISTS status
```

```
(
  dedupid          varchar(255) not null,
  evid             char(25) not null,
  device           varchar(128) not null,
  component        varchar(128) default "",
  eventClass       varchar(128) default "/Unknown",
  eventKey         varchar(128) default "",
  summary          varchar(128) not null,
  message          varchar(4096) default "",
  severity         smallint default -1,
  eventState       smallint default 0,
  eventClassKey    varchar(128) default "",
  eventGroup       varchar(64) default "",
  stateChange      timestamp,
  firstTime        double,
  lastTime         double,
  count            int default 1,
  prodState        smallint default 0,
  suppid           char(36) not null,
  manager          varchar(128) not null,
  agent            varchar(64) not null,
  DeviceClass      varchar(128) default "",
  Location          varchar(128) default "",
  Systems          varchar(255) default "",
  DeviceGroups     varchar(255) default "",
  ipAddress        char(15) default "",
  facility         varchar(8) default "unknown",
  priority         smallint default -1,
  ntevid           smallint unsigned default 0,
  ownerid          varchar(32) default "",
  clearid          char(25),
  DevicePriority    smallint(6) default 3,
  eventClassMapping varchar(128) default "",
  monitor          varchar(128) default "",
  PRIMARY KEY ( dedupid ),
  Index evididx (evid),
  Index clearidx (clearid),
  Index severityidx (severity),
  Index deviceidx (device)
) ENGINE=INNODB;
```

```
"zenevents.sql" [readonly] 163 lines --0%--
```

## Zenoss 3 MySQL status table fields in events database

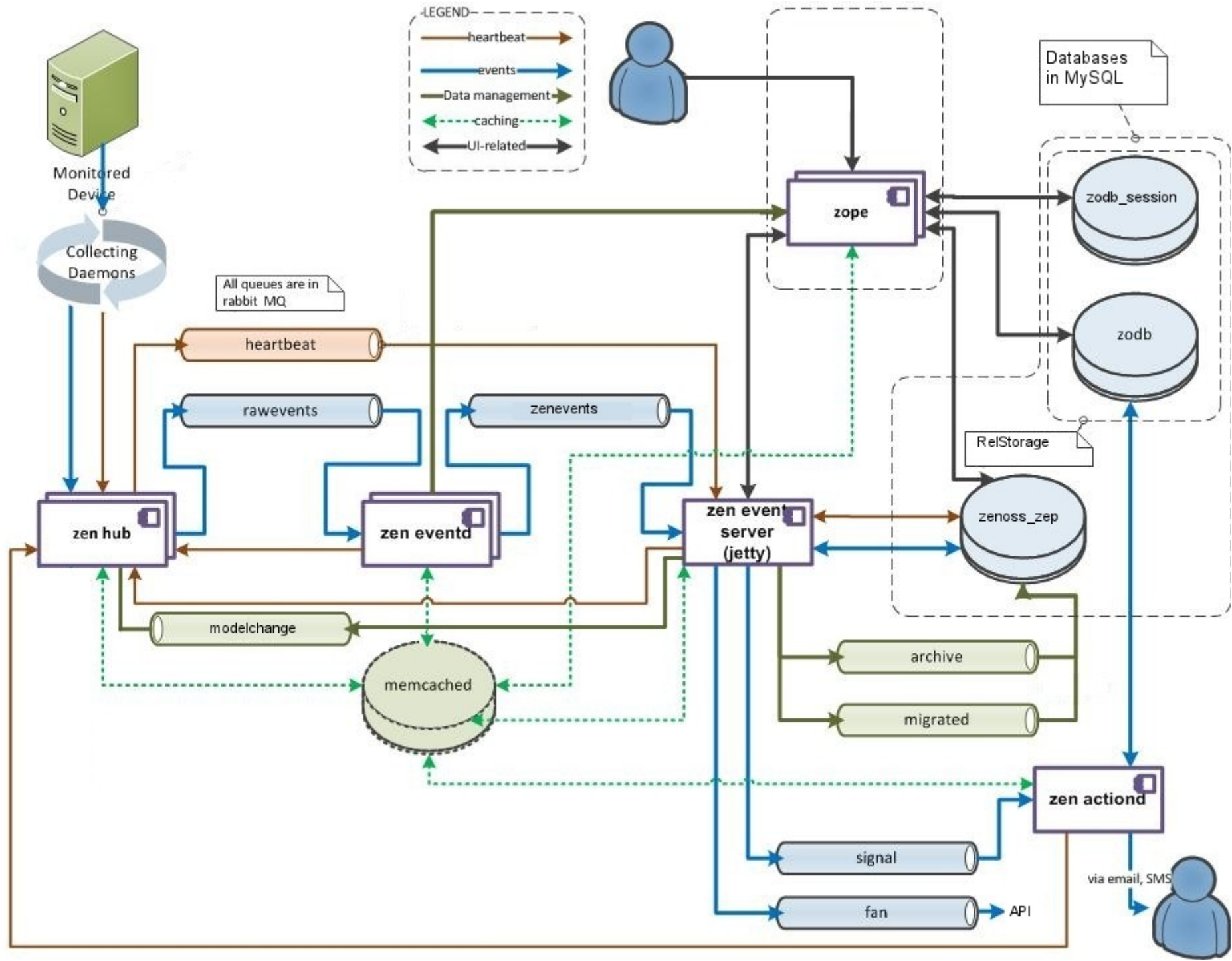
# Zenoss Core 4 Event Architecture

Collecting Daemons

- zenping
- zensyslog
- zenstatus
- zentrap
- zenmodeler
- zenperfsnmp
- zencommand
- zenprocess
- zenwin
- zeneventlog
- zenwinperf

Other key processes:  
zen jobs

GP Reich  
20121031  
J Curry  
20121207





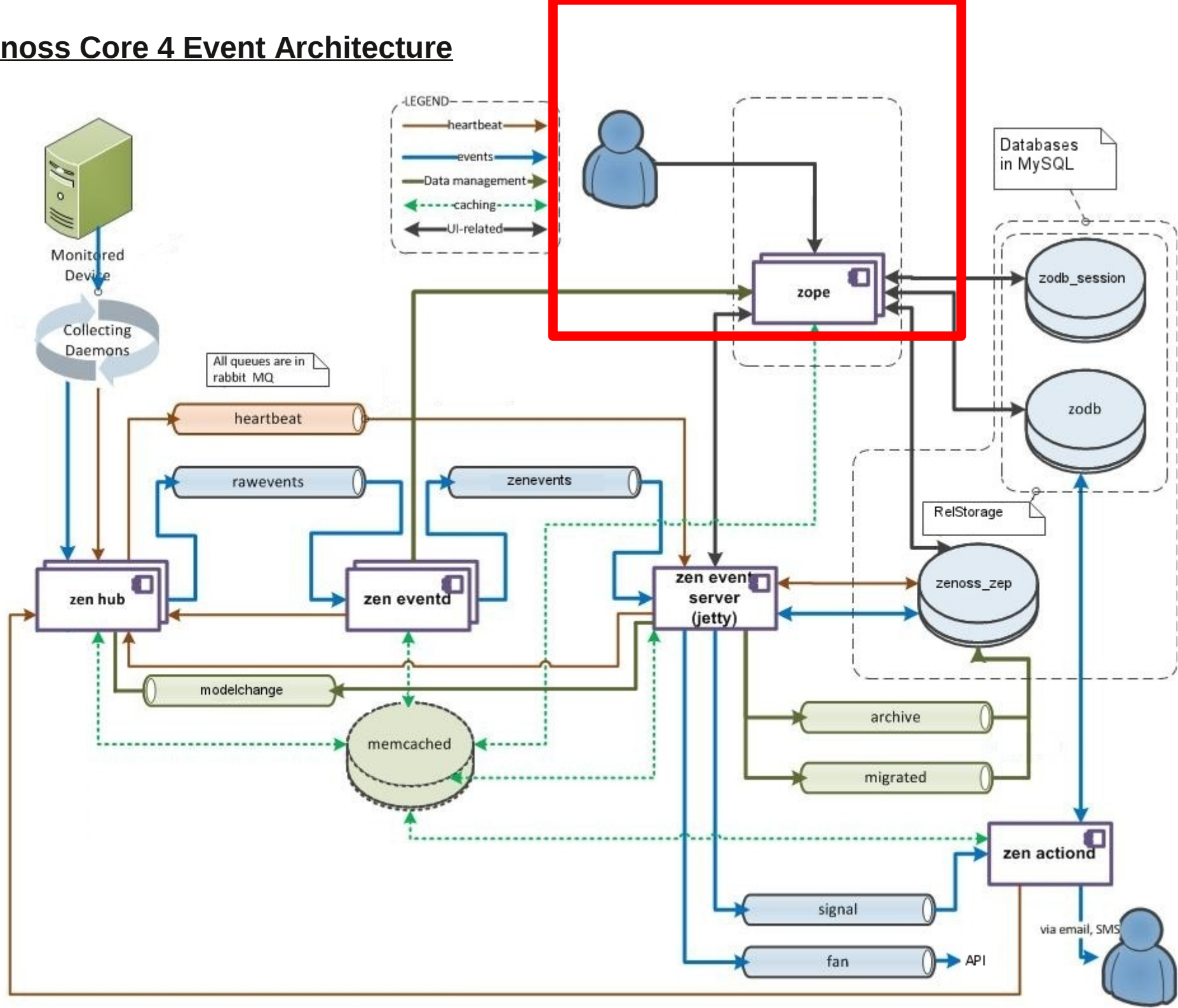
# Zenoss Core 4 Event Architecture

Collecting Daemons

zenping  
zensyslog  
zenstatus  
zentrap  
zenmodeler  
zenperfsmp  
zencommand  
zenprocess  
zenwin  
zeneventlog  
zenwinperf

Other key processes:  
zen jobs

GP Reich  
20121031  
J Curry  
20121207



# Event Console

Zenoss: Events - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Zenoss: Events

zen42.class.example.org:8080/zport/dmd/Events/evconsole

DASHBOARD EVENTS INFRASTRUCTURE REPORTS ADVANCED

Event Console Event Archive Event Classes Triggers

Select Export Configure

Status	Severity	Resource	Count	Summary	Component	Event Class
...	...		2:			
	!!	taplow-13.skills-1st.co.u...	151	10.0.0.13 is DOWN!		
✓	!!	zen31.class.example.or...	322	zen31.class.example.org is DOWN!		
	!	zen31.class.example.or...	2	SNMP agent down - no response received		
	!	zen31.class.example.or...	2	Unable to read processes on device zen31.class.examp...		/Status/Sn
	!	group-100-s1.class.exa...	2	threshold of high utilization exceeded: current value 111...	4	/Perf/Inter
	!	group-100-s1.class.exa...	9	threshold of high utilization exceeded: current value 111...	4	/Perf/Inter
	!	group-100-s1.class.exa...	4	threshold of high utilization exceeded: current value 111...	A	/Perf/Inter
	!	group-100-s1.class.exa...	3	threshold of high utilization exceeded: current value 111...	14	/Perf/Inter
	!	group-100-s2.class.exa...	3	threshold of high utilization exceeded: current value 110...	FastEth...	/Perf/Inter
	!	EC2Manager	42	No data returned for command		/Cmd/Fail
	!	group-100-s2.class.exa...	5	threshold of high utilization exceeded: current value 111...	FastEth...	/Perf/Inter
	!	zen42.class.example.or...	2	honeyd_logstart: fopen("/var/log/raddle/honeyd.log"): P...	honeyd	/Unknown
	i	zen42.class.example.or...	5	could not grab keyboard: 3	gnome-...	/Unknown
	i	win2003.class.example...	9	Logon attempt by: MICROSOFT_AUTHENTICATION_PA...	Security	/Unknown
	i	zen42.class.example.or...	2	Demoting process privileges to uid 99, gid 99	honeyd	/Unknown
	i	zen42.class.example.or...	2	listening on lo: ip and (dst host 10.191.100.4 or dst net 1...	honevd	/Unknown

Sort Ascending  
Sort Descending  
Columns

- ☐ Event ID
- ☐ Fingerprint
- ☒ Status
- ☒ Severity
- ☒ Resource
- ☒ Count
- ☒ Summary
- ☒ Component
- ☒ Event Class
- ☒ First Seen
- ☒ Last Seen
- ☒ Agent
- ☐ Production State
- ☐ Device Priority
- ☐ State Change
- ☐ Event Class Key
- ☐ Event Group
- ☐ Event Key
- ☐ Collector
- ☐ Owner
- ☐ Syslog Facility
- ☐ Syslog Priority

Refresh Actions Commands

Last Seen Agent

2012-11-27 17:17...	zenping
2012-11-27 17:17...	zenping
2012-11-27 16:38...	zenperfsnmp
2012-11-27 16:38...	zenprocess
2012-11-27 17:15...	zenperfsnmp
2012-11-27 17:15...	zenperfsnmp
2012-11-27 17:15...	zenperfsnmp
2012-11-27 17:15...	zenperfsnmp
2012-11-27 17:15...	zenperfsnmp
2012-11-27 17:15...	zenperfsnmp
2012-11-27 17:13...	zencommand
2012-11-27 17:10...	zenperfsnmp
2012-11-27 15:56...	zensyslog
2012-11-27 17:10...	zensyslog
2012-11-27 16:46...	zeneventlog
2012-11-27 15:56...	zensyslog
2012-11-27 15:56...	zensvsloa

DISPLAYING 1 - 16 of 17 ROWS

MyFooter

0 Jobs



# Event Console

- **Severity**
  - Critical
  - Error
  - Warning
  - Info
  - Debug
  - Clear
- Resource / Component
- Event Class
- Summary
- **Status**
  - New (0)
  - Acknowledged (1)
  - Suppressed (2)
  - Closed (3)
  - Cleared (4) †
  - Dropped (5) †
  - Aged (6) †
- First Seen / Last Seen
- Agent

† new in Zenoss 4



# MySQL **zenoss\_zep** database

- zenoss\_zep new with Zenoss 4
- event\_summary table
  - may contain Closed, Cleared and Aged events
- event\_archive table
  - only has Closed, Cleared and Aged events
- attributes in tables do **not** match event console
  - **Console**

	<u><b>Database</b></u>
– device	element_identifier
– component	element_sub_identifier
– eventState	status_id
– firstSeen	first_seen

```
File Edit View Search Terminal Help
mysql> show tables;
+-----+
| Tables_in_zenoss_zep |
+-----+
| agent                 |
| config                |
| daemon_heartbeat      |
| event_archive          |
| event_archive_index_queue |
| event_class            |
| event_class_key        |
| event_detail_index_config |
| event_group            |
| event_key              |
| event_summary          |
| event_summary_index_queue |
| event_time             |
| event_trigger          |
| event_trigger_signal_spool |
| event_trigger_subscription |
| index_metadata         |
| monitor               |
| schema_version        |
| v_daemon_heartbeat     |
| v_event_archive        |
| v_event_archive_index_queue |
| v_event_summary        |
| v_event_summary_index_queue |
| v_event_time           |
| v_event_trigger        |
| v_event_trigger_signal_spool |
| v_event_trigger_subscription |
| v_index_metadata       |
+-----+
29 rows in set (0.00 sec)

mysql> █
```

## Tables in zenoss\_zep

\* Lots more tables than Zenoss 3

- agent
- event\_class
- event\_class\_key
- event\_group
- event\_key
- event\_trigger
- monitor

File Edit View Search Terminal Help

mysql&gt; describe event\_summary;

Field	Type	Null	Key	Default	Extra
uuid	binary(16)	NO	PRI	NULL	
fingerprint_hash	binary(20)	NO	UNI	NULL	
fingerprint	varchar(255)	NO		NULL	
status_id	tinyint(4)	NO	MUL	NULL	
event_group_id	int(11)	YES		NULL	
event_class_id	int(11)	NO		NULL	
event_class_key_id	int(11)	YES		NULL	
event_class_mapping_uuid	binary(16)	YES		NULL	
event_key_id	int(11)	YES		NULL	
severity_id	tinyint(4)	NO	MUL	NULL	
element_uuid	binary(16)	YES	MUL	NULL	
element_type_id	tinyint(4)	YES		NULL	
element_identfier	varchar(255)	NO		NULL	
element_title	varchar(255)	YES		NULL	
element_sub_uuid	binary(16)	YES	MUL	NULL	
element_sub_type_id	tinyint(4)	YES		NULL	
element_sub_identfier	varchar(255)	YES		NULL	
element_sub_title	varchar(255)	YES		NULL	
update_time	bigint(20)	NO		NULL	
first_seen	bigint(20)	NO		NULL	
status_change	bigint(20)	NO		NULL	
last_seen	bigint(20)	NO	MUL	NULL	
event_count	int(11)	NO		NULL	
monitor_id	int(11)	YES		NULL	
agent_id	int(11)	YES		NULL	
syslog_facility	int(11)	YES		NULL	
syslog_priority	tinyint(4)	YES		NULL	
nt_event_code	int(11)	YES		NULL	
current_user_uuid	binary(16)	YES		NULL	
current_user_name	varchar(32)	YES		NULL	
clear_fingerprint_hash	binary(20)	YES	MUL	NULL	
cleared_by_event_uuid	binary(16)	YES		NULL	
summary	varchar(255)	NO			
message	varchar(4096)	NO			
details_json	mediumtext	YES		NULL	
tags_json	mediumtext	YES		NULL	
notes_json	mediumtext	YES		NULL	
audit_json	mediumtext	YES		NULL	

38 rows in set (0.01 sec)

## Fields in event\_summary

\* Most of the field names changed

- many refer to other tables eg  
agent\_id, event\_class\_id, ...

- evid -> uuid  
- dedupid -> fingerprint  
- eventState -> status\_id  
- firstSeen -> first\_seen  
- count -> event\_count  
- agent -> agent\_id  
- facility -> syslog\_facility

- device -> element\_  
- component -> element\_sub\_

- \_json fields  
details  
notes

\* However...

- summary is still summary  
- message is still message

# MySQL database.....

- Don't go there!



Here be dragons....



# Event Proxy

- Writing rules and transforms for events operate on an **Event Proxy** data structure
  - manipulate **same** field names as in Zenoss 3.x
  - translation between database field names, message queue field names and Event Console field names
- See files under  
\$ZENHOME/Products/ZenEvents/events2
- User-created event fields, including SNMP TRAP varbinds, are handled rather differently
  - event transforms may need changing

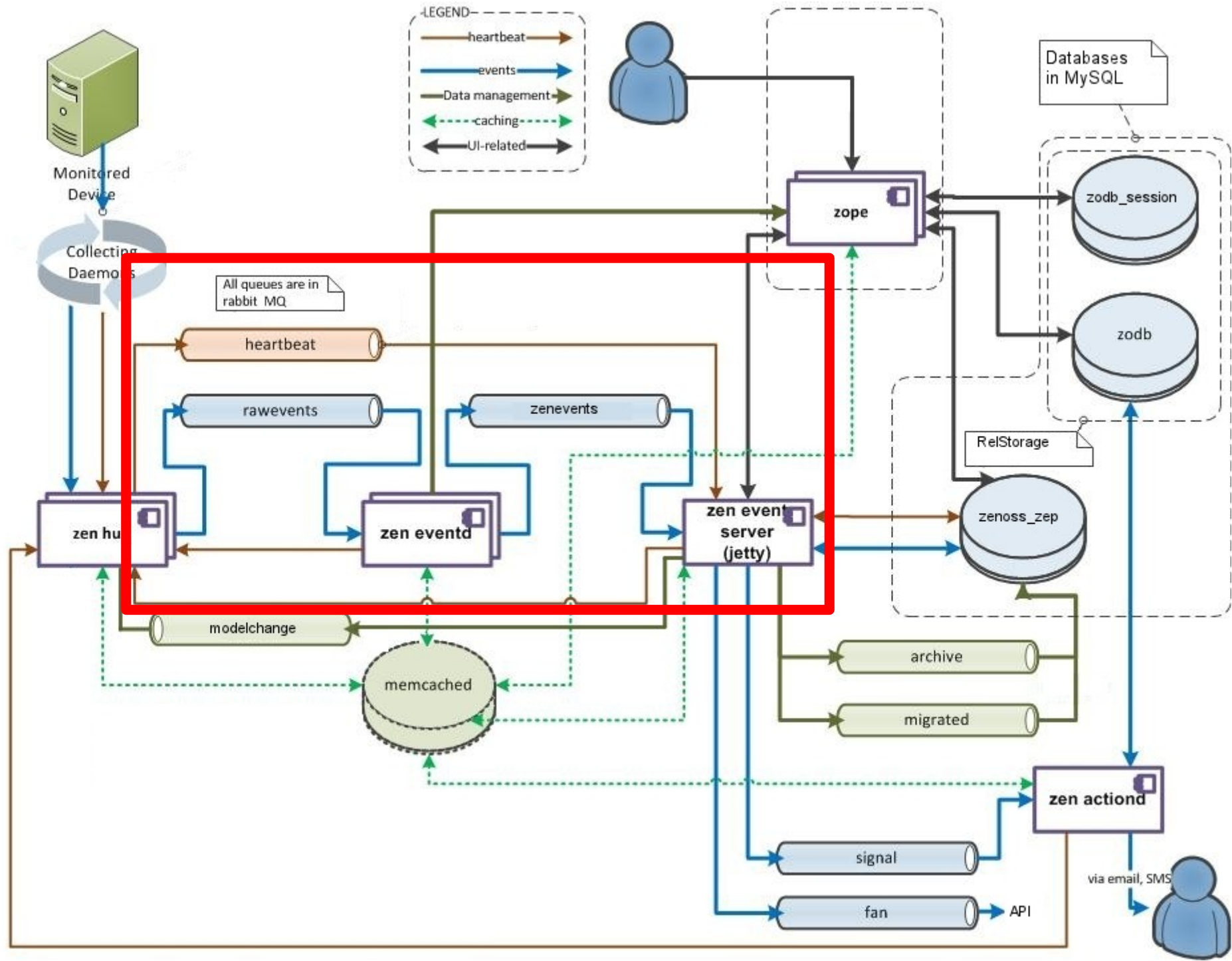
# Zenoss Core 4 Event Architecture

Collecting Daemons

- zenping
- zensyslog
- zenstatus
- zentrap
- zenmodeler
- zenperfsmp
- zencommand
- zenprocess
- zenwin
- zeneventlog
- zenwinperf

Other key processes:  
zen jobs

GP Reich  
20121031  
J Curry  
20121207



# New daemons

- zeneventserver
  - written in Java
- zeneventd
  - written in Python
- rabbitmq
  - open source Advanced Message Queueing Protocol (AMQP) package

# zeneventserver daemon

- Also known as **zep**
- Consumes data from zenevents queue
- Stores processed events in MySQL database
- Presents data to GUI via Zope web application
  - uses JSON to present data to users
- Produces data for the **signal** queue for notification actions
- Manages ageing of data from event\_summary to event\_archive table in database
- May produce data for other queues

# zeneventd daemon

- Responsible for most of the event processing
  - event classification
    - eventClassKey
    - Rule
    - Regex
  - device context
  - event context
  - transforms
  - deduplication fingerprint
- Consumes the **rawevents** queue
- Produces data for the **zenevents** queue



# rabbitmq

- Crucial to Zenoss working - many daemons depend on it
- Configured when Zenoss installed
  - VHOST = “/zenoss”
  - USER = “zenoss”
  - PASSWORD = check \$ZENHOME/etc/global.conf
- Started with *service zenoss start*
- If Zenoss server has name changed then rabbitmq needs reconfiguring
- rabbitmq commands need root privilege
- *rabbitmqctl report* - good to dump lots of data

# rabbitmq queues

```
jane@zen42:/home/jane
File Edit View Search Terminal Help

[root@zen42 jane]# rabbitmqctl -p /zenoss list_queues
Listing queues ...
celery 0
zenoss.queues.zep.signal 0
zenoss.queues.zep.modelchange 0
zenoss.queues.zep.migrated.summary 0
zenoss.queues.zep.rawevents 0
zenoss.queues.zep.heartbeats 0
zenoss.queues.zep.zenevents 0
zen42.class.example.org.celeryd.pidbox 0
zenoss.queues.zep.migrated.archive 0
...done.
[root@zen42 jane]#
```

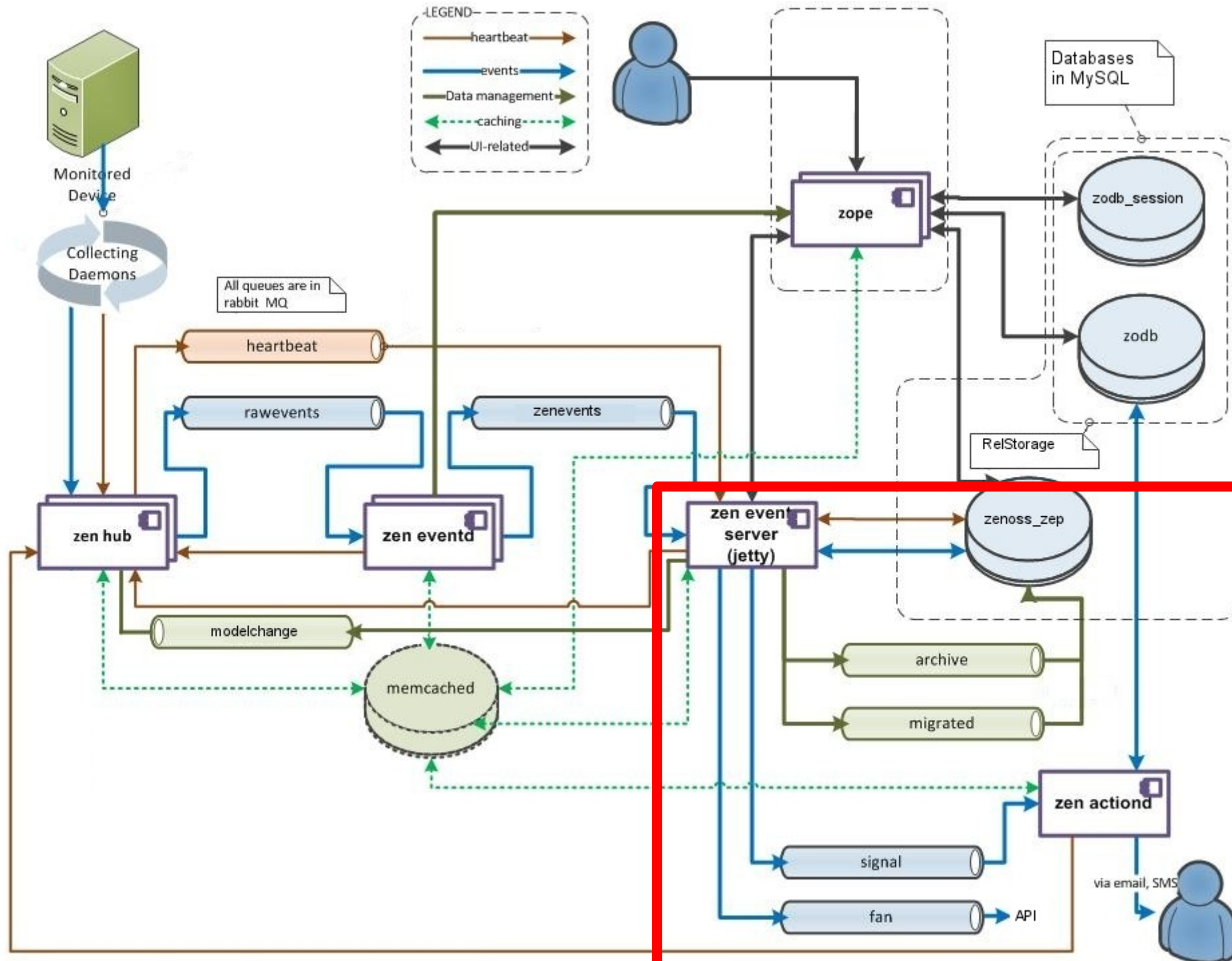
# Zenoss Core 4 Event Architecture

Collecting Daemons

zenping  
zensyslog  
zenstatus  
zentrap  
zenmodeler  
zenperfsmp  
zencommand  
zenprocess  
zenwin  
zeneventlog  
zenwinperf

Other key processes:  
zen jobs

GP Reich  
20121031  
J Curry  
20121207



# zenactiond

- Responds to **Triggers** evaluated by zeneventserver
- Provides **Notifications**
  - email / page to users
  - command scripts / SNMP TRAPs for background automation
- Existed prior to Zenoss 4 with 60 sec cycle
- Completely rewritten for Zenoss 4
- Consumer of the signal queue
  - much more responsive

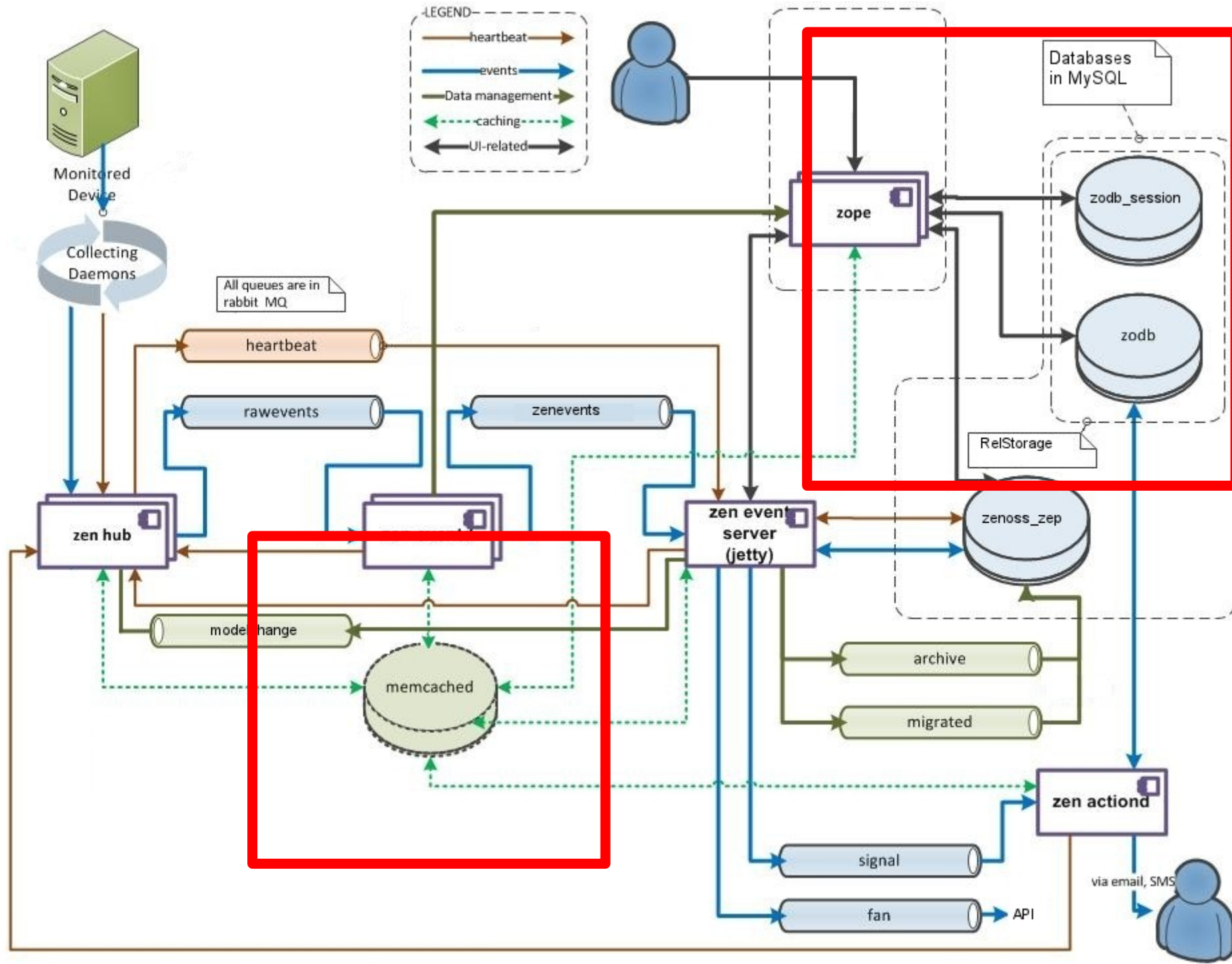
# Zenoss Core 4 Event Architecture

Collecting Daemons

- zenping
- zensyslog
- zenstatus
- zentrap
- zenmodeler
- zenperfsmp
- zencommand
- zenprocess
- zenwin
- zeneventlog
- zenwinperf

Other key processes:  
zen jobs

GP Reich  
20121031  
J Curry  
20121207





# Other Zenoss 4 changes

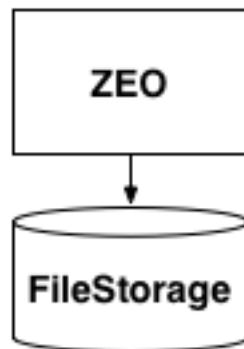
- memcached subsystem
  - shared L2 memory cache for daemons
  - configured in */etc/sysconfig/memcached* and *\$ZENHOME/etc/zope.conf* but **not** preallocated
- Zope database (zodb) now in MySQL database rather than *\$ZENHOME/var/Data.fs*
- zodb\_session database holds transient data
- RelStorage subsystem provides high-performance backend to ZODB into database
- Lucene subsystem for indexing of zodb and zenoss\_zep
- Open source packages

# Performance enhancements

Zenoss 2.x /  
Zenoss 3.x

## FileStorage & ZEO

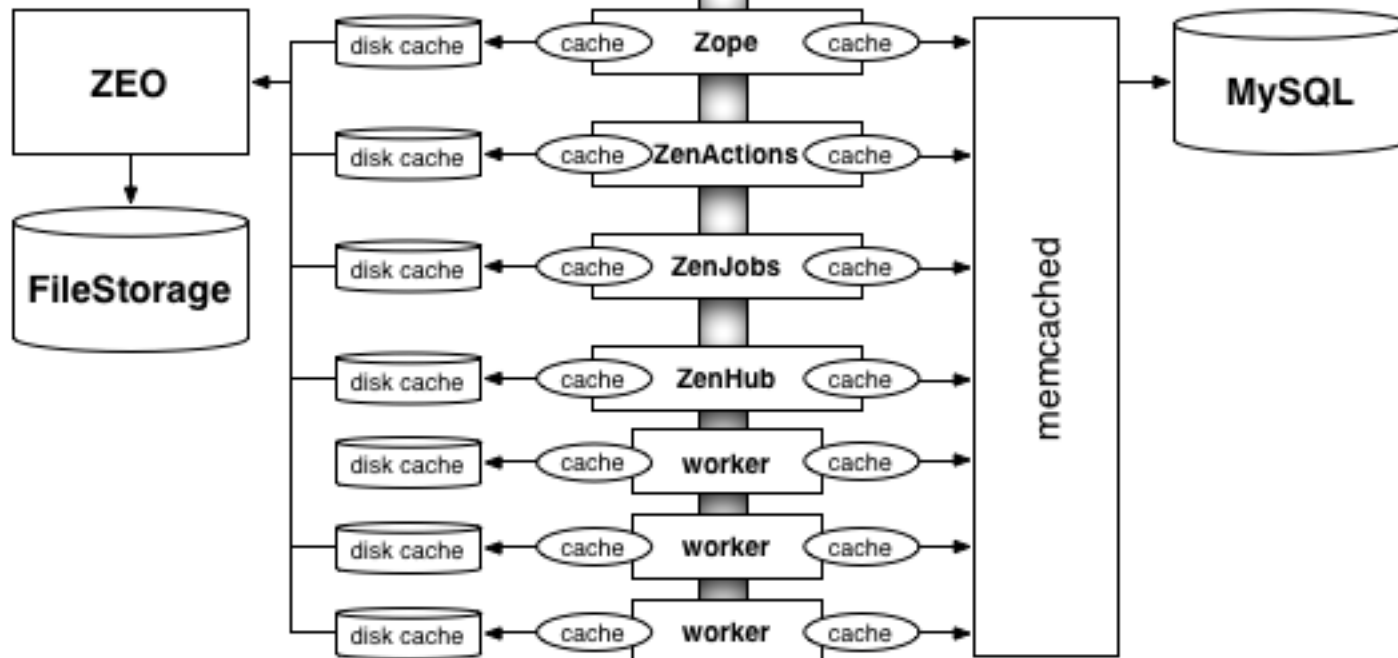
- Per-process L1 memory cache.
- Per-process L2 disk cache.
- Database layer caching provided exclusively by operating system's file cache.
- Must listen for invalidations.
- Unauthenticated connections to database.



## RelStorage & Memcached

- Per-process L1 memory cache.
- Shared L2 memory cache.
- Database layer caching provided by MySQL.
- Must poll for invalidations.
- Authenticated connections to database.

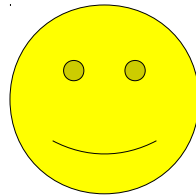
Zenoss 4.x



# Intermission....

Back in 10 minutes.....

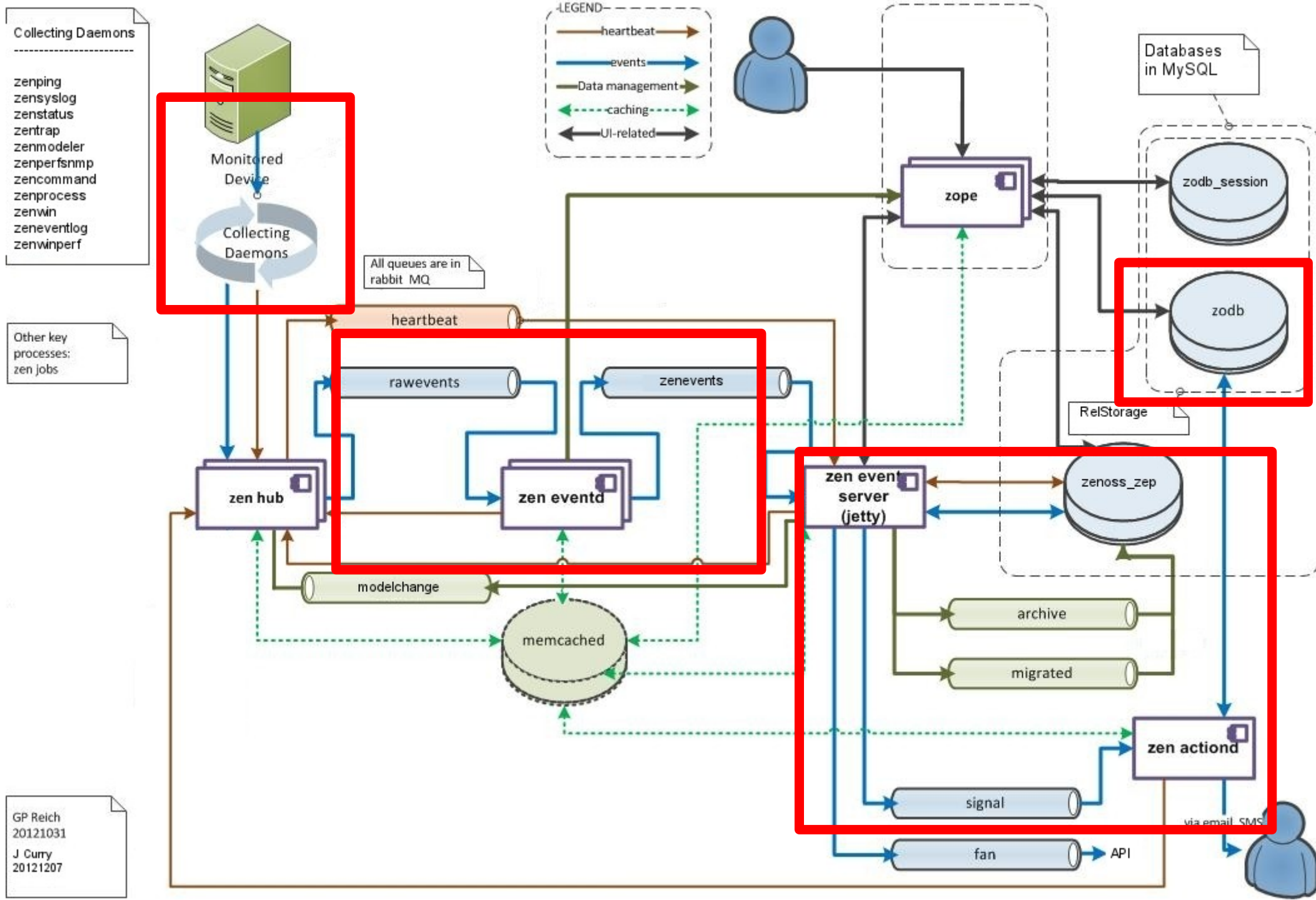
.... that's still 600 seconds



# Event life cycle

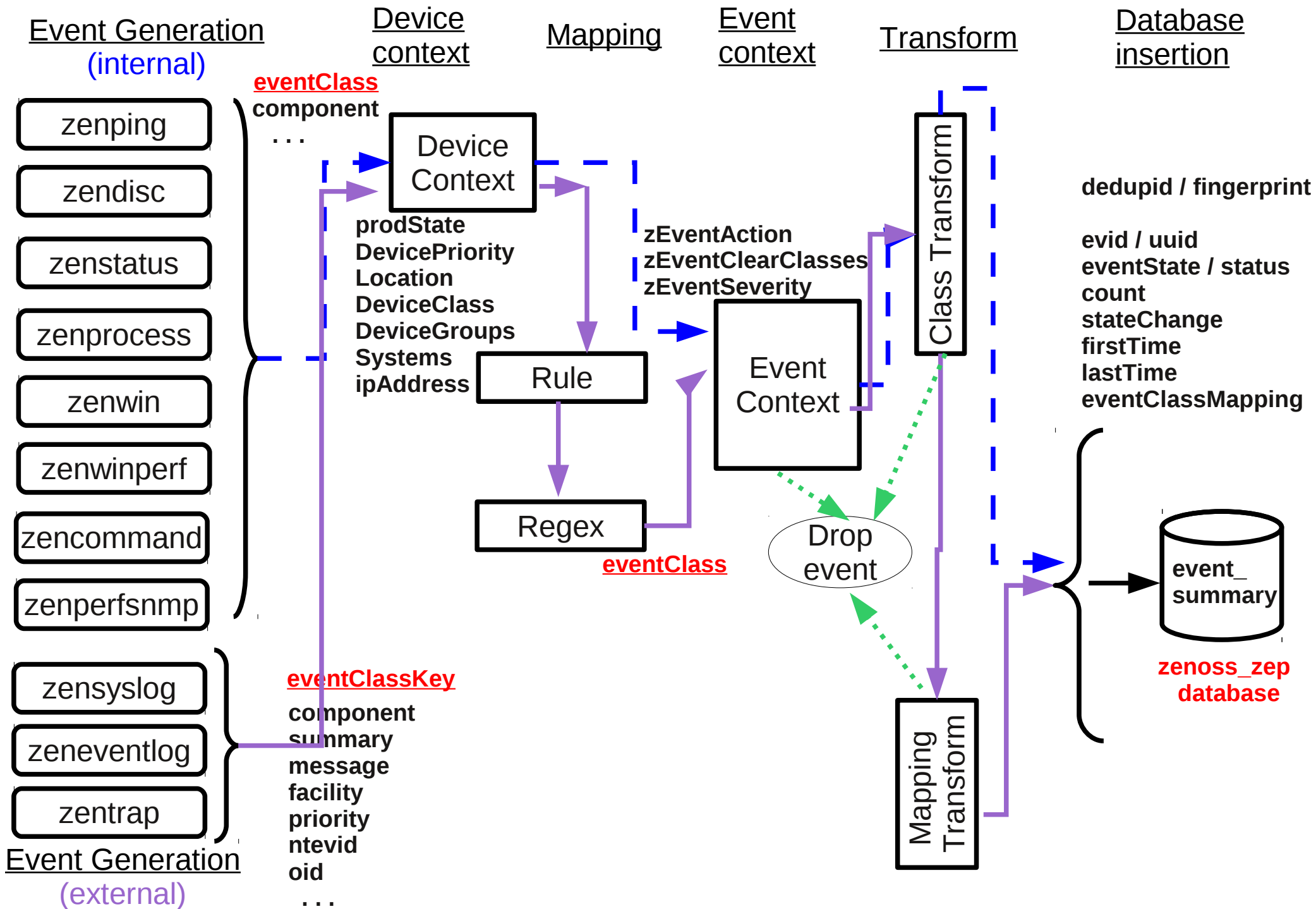
- Event generation
- Device context
- Event class mapping
- Event context
- Event transform
- Database insertion and deduplication
- Resolution
- Ageing

# Zenoss Core 4 Event Architecture

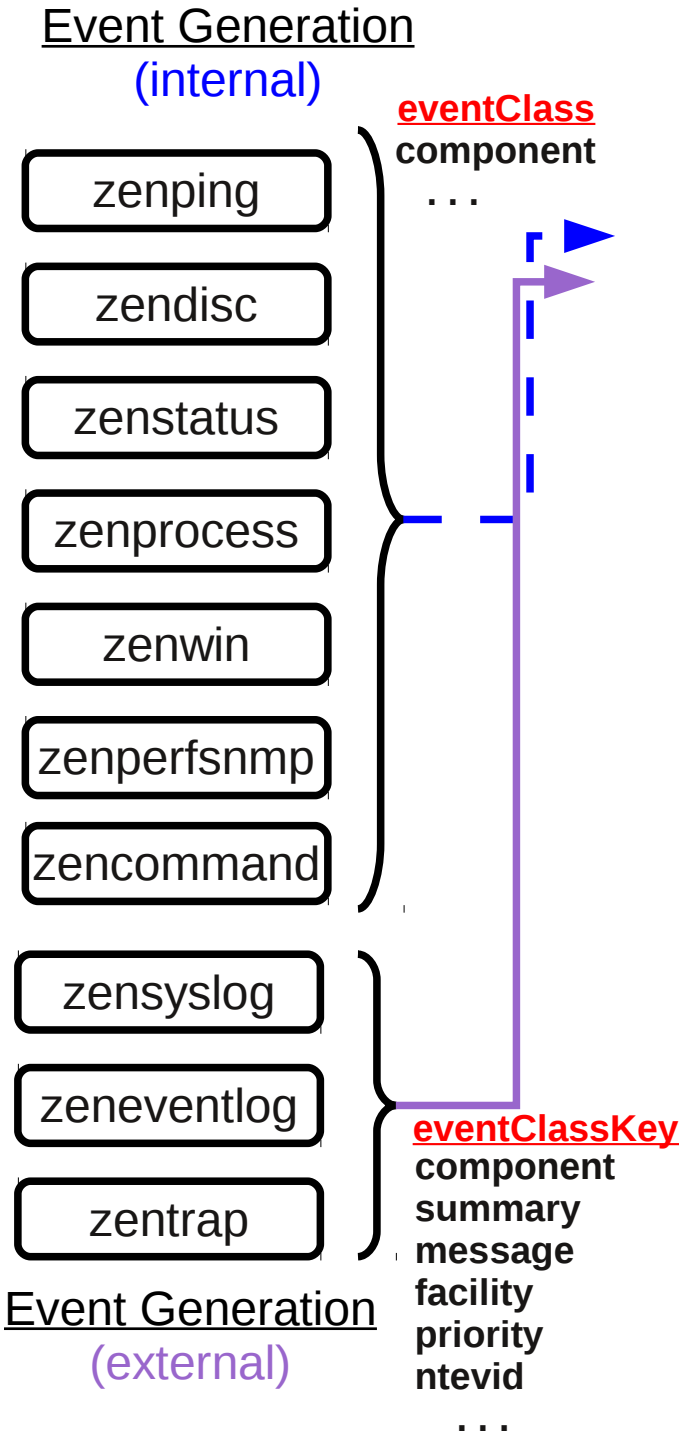




# Event Life Cycle – generation to initial database insertion



# Event Life Cycle – generation



# Event generation

- **Internal** event daemons
  - zenping
  - zendisc
  - zenstatus
  - zenprocess
  - zenwin
  - zenwinperf
  - zencommand
  - zenperfsnmp
  - other daemons introduced by ZenPacks
- **External** event daemons
  - zensyslog
  - zeneventlog
  - zentrap

# Event Life Cycle – generation to device context

Event Generation  
(internal)

zenping

zendisc

zenstatus

zenprocess

zenwin

zenperfsnmp

zencommand

zensyslog

zeneventlog

zentrap

Device  
context

eventClass

component

...

Device  
Context

prodState  
Location  
DeviceClass  
DeviceGroups  
Systems  
ipAddress

eventClassKey

component

summary

message

facility

priority

ntheid

...

Event Generation  
(external)



# Device context

- Event fields pertaining to the device that generated the event
  - prodState
  - Location
  - DeviceClass
  - DeviceGroups
  - Systems
  - ipAddress
- This configuration data is retrieved from the Zope Object Database (ZODB)

# Event Life Cycle – generation to event class mapping

Event Generation  
(internal)

zenping

zendisc

zenstatus

zenprocess

zenwin

zenperfsnmp

zencommand

zensyslog

zeneventlog

zentrap

eventClass  
component

...

Device  
context

Device  
Context

prodState  
Location  
DeviceClass  
DeviceGroups  
Systems  
ipAddress

Mapping

Rule

Regex

eventClass

eventClassKey

component  
summary  
message  
facility  
priority  
ntevd

...

Event Generation  
(external)

# Event class mapping

- Used to map externally generated events to Zenoss event format
- Incoming event typically has **eventClassKey** field
  - Several source events may have same eventClassKey
- Target is to determine an **eventClass** field
- Python **Rule** tests any available field of event
  - If Rule exists, it must be satisfied
- Python **Regex** parses summary field of event
  - Regex must be satisfied if Rule doesn't exist
  - User-defined fields can be created if Regex matches
  - Regex need not be satisfied if Rule does exist
- Event mappings have sequence numbers



# Event class mapping

The screenshot shows the Zenoss Core interface with the 'EVENTS' tab selected. The breadcrumb path is 'Events > Skills > linetest'. A summary bar shows 8 critical events (red), 0 warning events (yellow), 0 info events (blue), and 0 debug events (grey), with a total event count of 8. The 'EventClassInst' configuration for 'linetest' is highlighted with a red box. It shows a sequence of 10, a rule 'evt.component=='linetest' and device.snmpContact == 'Jane Curry', a regex 'test line (?P<line\_num>\d+)', and an example message 'This is test line 1'. Below this, a Python script for event transformation is visible.

**Zenoss CORE** DASHBOARD **EVENTS** INFRASTRUCTURE REPORTS ADVANCED  jane SIGN OUT ?

Event Console Event Archive **Event Classes** Triggers Page Tips

Events > Skills > linetest

Events 8 0 0 0 Total Event Count 8

**EventClassInst**

Event Class Key	linetest
Sequence	10
Rule	evt.component=='linetest' and device.snmpContact == 'Jane Curry'
Regex	test line (?P<line_num>\d+)
Example	This is test line 1

```
#import pdb; pdb.set_trace()
evt.myLine num = evt.line_num
evt.myDevId = device.id
evt.mySnmpSysLoc = device.snmpLocation
evt.mySnmpSysContact = device.snmpContact
evt.mySnmpStatus = device.getSnmpStatusString()
evt.summary = "Problem is %s on device %s. Please call %s" % (evt.summary, evt.myDevId, evt.mySnmpSysContact)
#evt.myEventState = evt.eventState
```

# Event Life Cycle – generation to event context

Event Generation  
(internal)

zenping

zendisc

zenstatus

zenprocess

zenwin

zenperfsnmp

zencommand

zensyslog

zeneventlog

zentrap

eventClass  
component  
...

Device  
context

Device  
Context

prodState  
Location  
DeviceClass  
DeviceGroups  
Systems  
ipAddress

Mapping

Rule

Regex

eventClass

Event  
context

zEventAction  
zEventClearClasses  
zEventSeverity

Event  
Context

Drop  
event

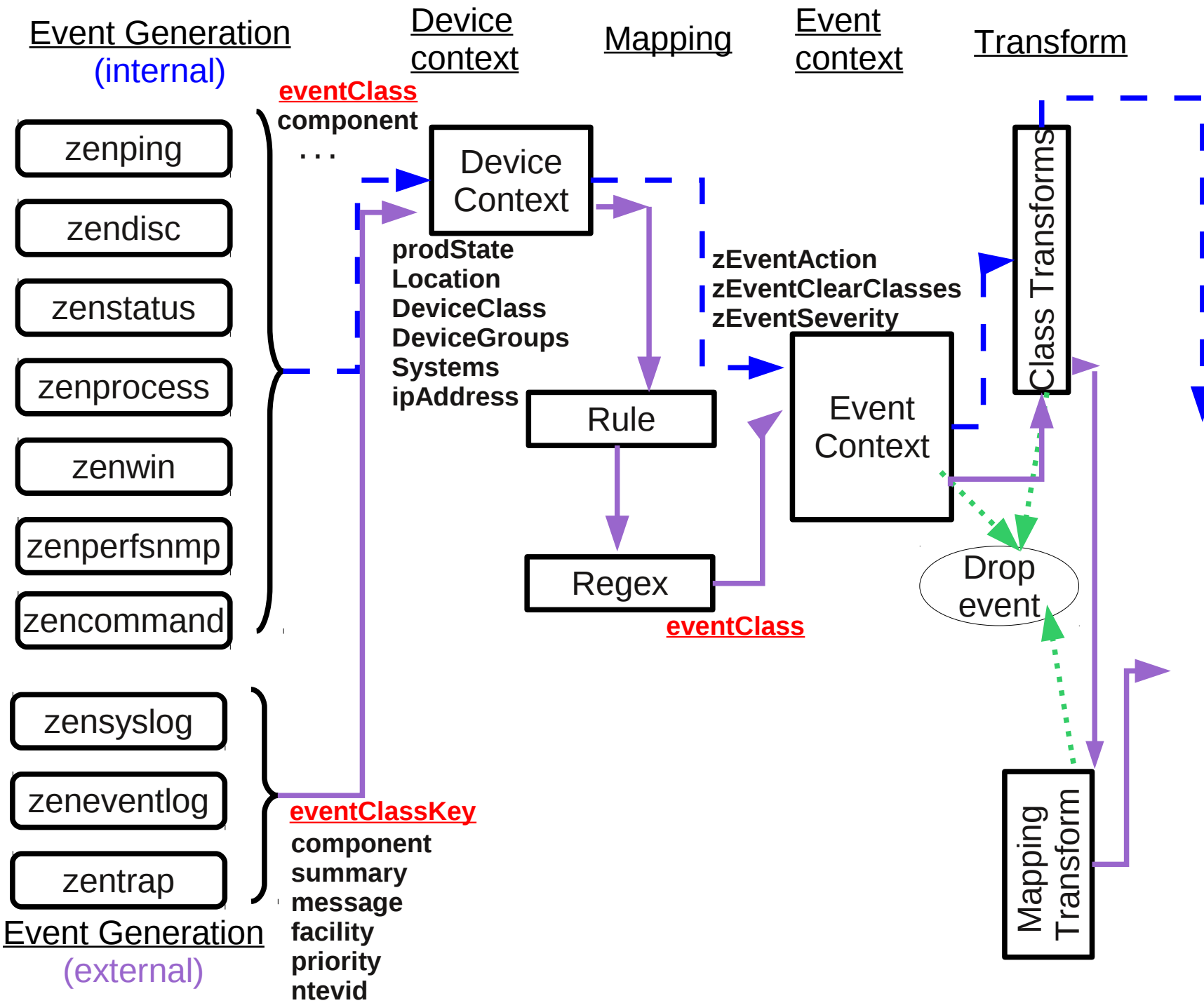
eventClassKey  
component  
summary  
message  
facility  
priority  
ntevId  
...

Event Generation  
(external)

# Event context

- Defined by 3 event zProperties:
  - zEventAction                      status | history | drop
  - zEventClearClasses
  - zEventSeverity
- Event context is applied **after** event mapping but **before** event transforms
- Thus, zEventAction event context may specify *history* but an event transform could override by setting evt.\_action to the value *status*
  - these values reflect old database schema
    - status maps to eventState = New
    - history maps to eventState = Closed
    - both are stored in the event\_summary table

# Event Life Cycle – generation to transform



# Event transforms

- Transforms can be applied to an **event class mapping** or to an **event class**
- Python statement(s) to modify:
  - Any available fields of the event
  - Any available property of the device
- Transform can create user-defined event fields
- Transform only applied if Rule / Regex satisfied
- Since 2.4 cascading event class transforms apply event class transform(s) *then* mapping transform

# Event class mapping transform

The screenshot displays the Zenoss Core web interface. The top navigation bar includes 'Zenoss CORE', 'DASHBOARD', 'EVENTS', 'INFRASTRUCTURE', 'REPORTS', and 'ADVANCED'. A user profile for 'jane' and a 'SIGN OUT' link are on the right. Below the navigation bar, the 'Event Classes' tab is selected, showing a breadcrumb 'Events > Skills > linetest'. A summary bar indicates 8 events with various status icons. The main configuration area for 'EventClassInst' includes fields for 'Event Class Key' (linetest), 'Sequence' (10), and 'Rule' (a logical expression). The 'Regex' field contains 'test line (?P<line\_num>\d+)'. The 'Example' field shows 'This is test line 1'. The 'Transform' field, highlighted with a red box, contains a Python script for event processing.

**Zenoss CORE** DASHBOARD **EVENTS** INFRASTRUCTURE REPORTS ADVANCED jane SIGN OUT

Event Console Event Archive **Event Classes** Triggers Page Tips

Events > Skills > linetest

Events 8 0 0 0 Total Event Count 8

**EventClassInst**

Event Class Key linetest

Sequence 10

Rule

evt.component=='linetest' and device.snmpContact == 'Jane Curry'

Regex

test line (?P<line\_num>\d+)

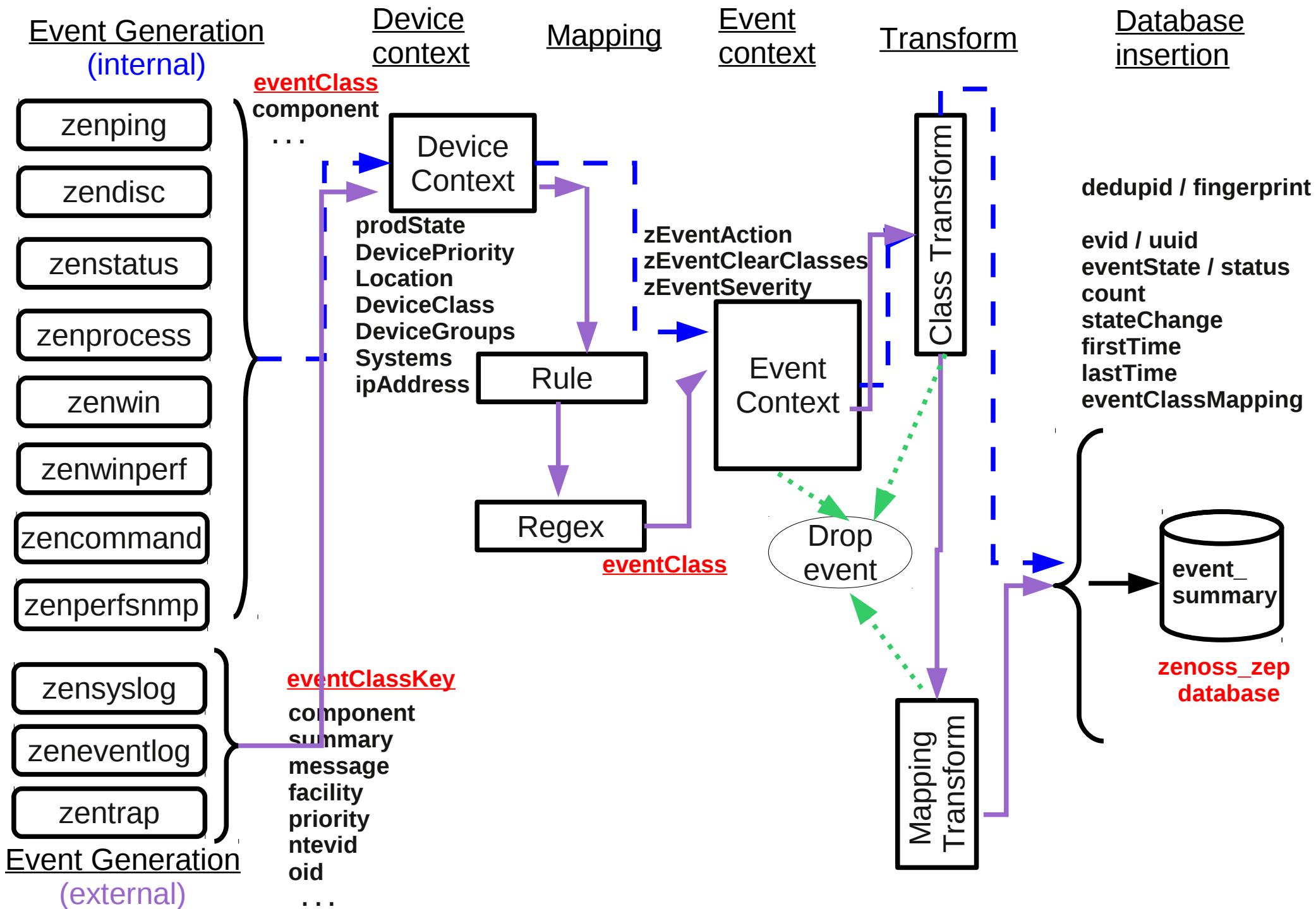
Example

This is test line 1

Transform

```
#import pdb; pdb.set_trace()
evt.myLine num = evt.line_num
evt.myDevId = device.id
evt.mySnmpSysLoc = device.snmpLocation
evt.mySnmpSysContact = device.snmpContact
evt.mySnmpStatus = device.getSnmpStatusString()
evt.summary = "Problem is %s on device %s. Please call %s" % (evt.summary, evt.myDevId, evt.mySnmpSysContact)
#evt.myEventState = evt.eventState
```

# Event Life Cycle – generation to database insertion





# Database insertions

- All events inserted into **event\_summary** table
- Some fields only assigned at insertion time:
  - \* count      \* stateChange    \*dedupid    \*eventState
  - \* firstTime   \* lastTime   \* evid    \* eventClassMapping
- Automatic duplicate detection based on:
  - \* device                      \* component          \* eventClass
  - \* severity      \* eventKey              \* summary<sup>†</sup>
- dedupid made by concatenating above fields
- In Zenoss 4, this is known as the **fingerprint**
- Created by zeneventd; applied by zeneventserver
- Old event updated with details of new event

# Resolution

- User closes event (eventState = Closed)
- Event context zEventAction = history / drop
- Transform sets evt\_action to history / drop
- Clearing events will automatically clear all **similar** events, based on:
  - \* component UUID      \* eventClass      \*eventKey
    - if componentUUID exists; otherwise:
  - \* eventClass      \* device      \* component      \* eventKey
- zEventClearClasses zProperty on clearing events, **also** clears similar defined events ( based on stated class, plus same conditions)
- eventState will be Cleared

# Ageing

- Event Manager provides for:
  - Events with severity < Error get eventState = Aged after 4 hours (configurable)
  - Event archive threshold default of 3 days after which Closed, Cleared and Aged events moved from event\_summary to event\_archive table
  - Delete Archived Events Older Than (days) really deletes data
- Deleting data now much faster - daily partitions
- Python script ZenDeleteHistory from Zenoss 3 does not exist or have equivalent function

# Conclusions

- Understanding the architecture is vital
- Understanding the fields of an event is crucial
- Understanding the attributes of devices is important
- Knowledge of Python is required to write transforms
- Good knowledge of SNMP and syslog helps
- Working knowledge of SQL syntax helps
- 3-day Zenoss Event Management Workshop

<http://www.skills-1st.co.uk/products/courses/>

# Questions

