

# Monitoring and tooling

- Query API
- Cache monitoring
- Performance sensors SaaS
- Performance samples SaaS
- Tasks SaaS
- Storefront component rendering diagnostics SaaS
- Cluster
- Application logs
  - Shop specific logging
    - ShopCodeLogDiscriminator

## Query API

Most of the rendered data in storefront and Admin is an amalgamation of complex objects, which in many cases are heavily cached to ensure good performance of the platform. In some cases when performing analysis it is required to get access to raw fresh data. Query API specifically targets this problem.

Query API is not only concerned with persistence layer but a multipurpose tool which can be extended via extension point. Out of the box the following API extensions are supported:

API	Version	Supported Nodes	Purpose	Example
SQL:Core	3.0.0+	ADM, API, SFx	SQL interface for core RDBMS	<pre>select count(*) from TSKUPRICE</pre>
HQL:Core	3.0.0+	ADM, API, SFx	Hibernate QL interface for core RDBMS	<pre>select count(s) from SkuPriceEntity s</pre>
HQL:Payment	3.0.0+	ADM	Hibernate QL interface for payment RDBMS	<pre>select p from PaymentGatewayParameterEntity p</pre>
IceCat:Product	3.3.0+ SaaS	ADM	IceCat search interface to validate product XML	<pre>72514951,72514952</pre>
FT:Product	3.0.0+	API,SFx	Lucene full text query interface	<pre>brand:toshiba name:w50</pre>

In order to perform a query over desired medium:

- Select Node on which query to be performed
- Select type of API to use and click "Add tab" button

- Enter query in required format and click "Play" button

Nodes available on cluster

DEMO ENVIRONMENT

Plugins available for given Node

Query API / (2) hql-core

EDEMO.SF0

1 SQL: Core ✕ 2 HQL: Core ✕ 3 HQL: Payment ✕ 4 IceCat: Product ✕ 5 FT: Product ✕

select count(s) from SkuPriceEntity s

Results for query select count(s) from SkuPriceEntity s:

882,

## Cache monitoring

For best performance each application node maintains local cache to reduce contingency on slow resources or speed up results of complex calculations. Cache monitoring panel allows to view all active caches on all nodes in the cluster with corresponding statistics, which include:

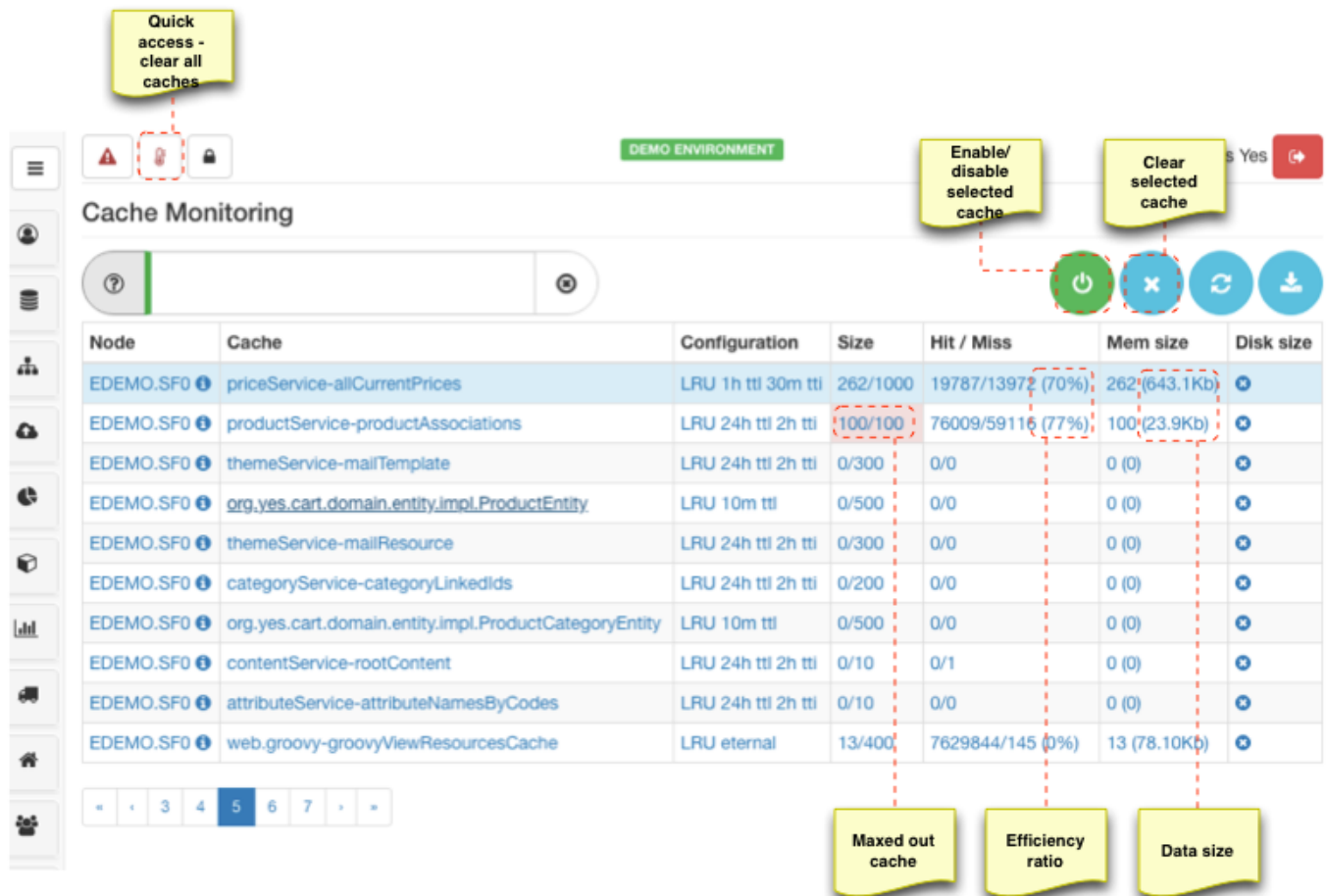
- Configuration parameters i.e. caching strategy and expiry timeframes
- Declared size and active size - essential for detecting flooded caches (performance tuning)
- Hit and Miss and efficiency ratio - essential for detecting inefficient caching (performance tuning)
- Memory size - RAM usage indicators (scalability)

All declared caches are active by default. However it is possible to enable/disable individual caches at runtime.

Caches can be cleared all at once, or individually by selecting specific cache and clicking "Evict" button

Evict all caches also features on the quick access buttons menu. This is an essential tool for development

Cache search has a number of "smart search" options to list flooded, most used and largest in size caches.



## Performance sensors SaaS

Performance sensors are fine detailed trackers or service layer API invocations.

Each sensor is Node specific and collects information on:

- Total amount of time API is used - to identify execution hot spots
- Invocation count of an API - to identify most used API (possible caching recommendation)
- Average, min and max time - to identify resource contingencies and slow API





⚙️
🔔
🔒

DEMO ENVIRONMENT

Logged in as Yes 👤

### Tasks

?
⊕

↺
⏸
▶
⚙️
🔄
📄

Node	Name / Current Schedule	Schedule Info	Original Schedule	Last Run	Previous Run	Next Run	Running
EDEMO.SF0	Reindex All Products / 0 0 5 * * ? 2099	seconds: 0 minutes: 0 hours: 5 daysOfMonth: * months: * daysOfWeek: ? lastdayOfWeek: false nearestWeekday: false NthDayOfWeek: 0 lastdayOfMonth: false years: 2099	0 0 5 * * ? 2099			2099-01-01 05:00:00 28879d 14h 43m 20s	⊙
EDEMO.SF0	Inventory Changes Product Indexing / 0 5 * * * ?	seconds: 0 minutes: 0,5,10,15,20,25,30,35,40,45,50,55 hours: * daysOfMonth: * months: * daysOfWeek: ? lastdayOfWeek: false nearestWeekday: false NthDayOfWeek: 0 lastdayOfMonth: false years: *	0 0/5 * * * ?			2019-12-07 14:15:00	○
EDEMO.SF0	product FT re-index / (manual)	product ... (0)	(manual)				○

Paused schedule

App node which executes this task

Next run time

Time left till next run

Schedule description - in this case every 5 minutes

Empty circle indicates task is not currently running

Managing task schedules can be accomplished in three distinct actions:

- Run task now - platform automatically calculates a one-off schedule to run task in 30seconds, after task has run manually it will become unscheduled and not run anymore
- Restore schedule - platform automatically will use the original schedule to from properties file
- Reschedule - user is invited to change the cron expression to instruct the platform to perform task at a different schedule

DEMO ENVIRONMENT

Logged in as Yes

Tasks / Send Mail

Mail

Selected Task

IO.JAM

Send Mail / 0 0/10 \*\*\* ?

seconds: 0  
minutes: 0,10,20,30,40,50  
hours: \*  
daysOfMonth: \*  
months: \*  
daysOfWeek: ?  
lastdayOfWeek: false  
nearestWeekday: false  
NthDayOfWeek: 0  
lastdayOfMonth: false  
years: \*

Bulk send mail ... completed, send: 0, failed: 0

0 0/10 \*\*\* ?

2019-12-07 14:10:00

2019-12-07 14:20:00 3m 19s

1

Are you sure?

Run this task now (within next 30 seconds)

Name

Send Mail

Original Schedule

0 0/10 \*\*\* ?

Current Schedule

0 0/10 \*\*\* ?

Cancel

OK

Are you sure?

Change schedule for this task

Name

Send Mail

Original Schedule

0 0/10 \*\*\* ?

Current Schedule

0 0/10 \*\*\* ?

New Schedule

0 0/10 \*\*\* ?

Cancel

OK

DEMO ENVIRONMENT

Logged in as Yes

Tasks / Send Mail

Mail

Node	Name / Current Schedule	Schedule Info	Original Schedule	Last Run	Previous Run	Next Run	Running
EDEMO.JAM	Send Mail / 0 0/10 *** ?	<div>seconds: 0 minutes: 0,10,20,30,40,50 hours: * daysOfMonth: * months: * daysOfWeek: ? lastdayOfWeek: false nearestWeekday: false NthDayOfWeek: 0 lastdayOfMonth: false years: *</div> <div>Bulk send mail ... completed, send: 0, failed: 0</div>	0 0/10 *** ?			2019-12-07 14:30:00 7m 47s	



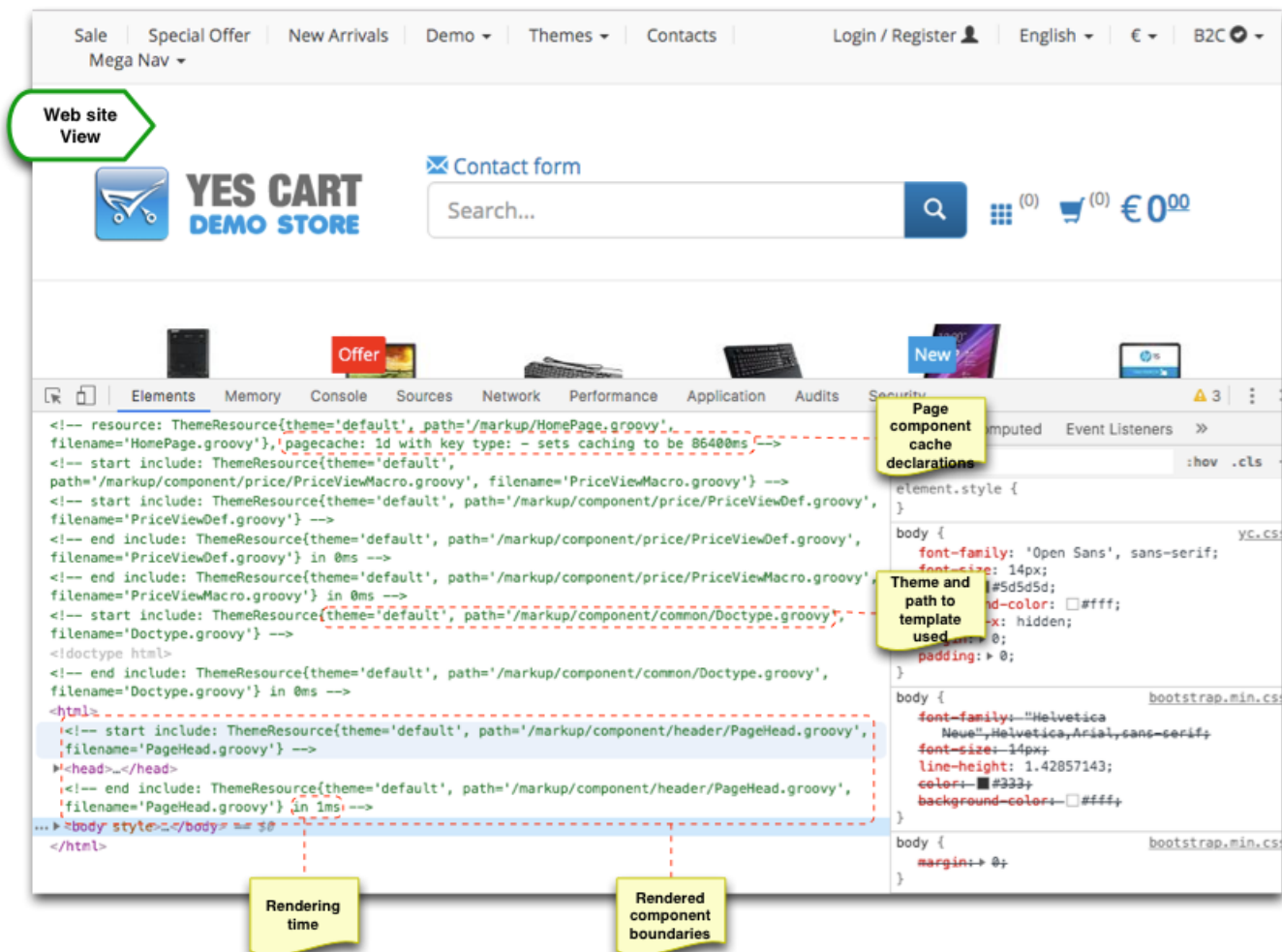
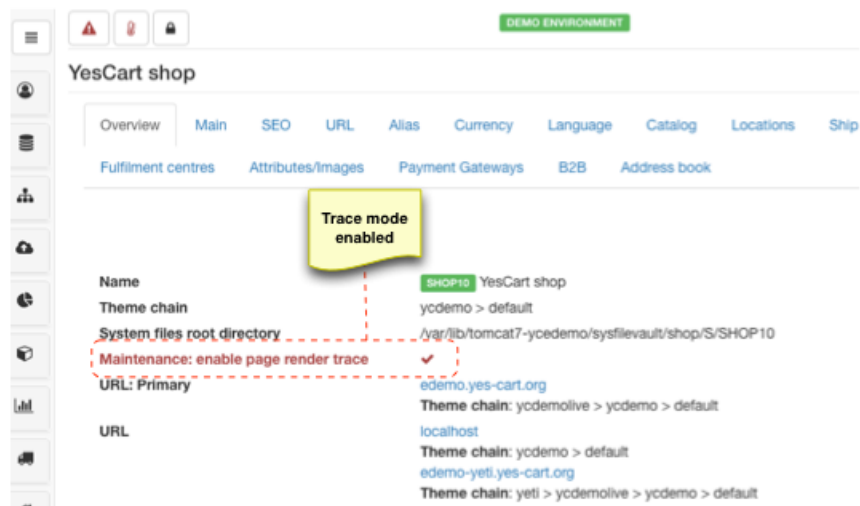
## Storefront component rendering diagnostics SaaS

Storefront themes multiplied by complexities of CMS further obscured by vast variety of data and caching sometimes results in questioning why page rendering has been done in a certain way? or why some components are not present on the page? or are present but should not be?

SFG SaaS frontend application contains a setting to produce verbose output during page rendering. In order to enable this mode set [shop](#) attribute: **Maintenance: enable page render trace**. This attribute can also be observed on the Overview tab. When enabled detailed information will be printed out during rendering including:

- Full path of the components rendered in the page
- CMS elements included in the page
- Rendering times
- Caching information





## Cluster

Cluster defines a network of applications (nodes). Cluster is either preconfigured or nodes are auto-discovered depending on the discovery module used (either WebService, REST or JGroups multicast connector).

Each node declares cluster namespace it belongs to, type (ADM, API or SFx), mode of operation and whether it uses full text search module. Along with this information cluster service can query each node to give full details of the build (i.e. all modules that are currently loaded).

Overview provided by cluster monitoring is vital in understanding platform infrastructure and composition of individual applications.

Detailed documentation on clustering is available upon request

EDEMO.JAM

core

core-module-orderstate:core-orderstate-aspects

app

jam:manager-aspects

app

jam:manager-sac

app

jam:manager-mailconfig

core

persistence-core-hibernate:dao-persistence-resources

payment

persistence-payment-hibernate:payment-persistence-resources

app

jam:jam-persistence

core

core:core-config

core

core:core-persistence

core

core:core-io

app

jam:dao-index-admin

marketing

int-module-marketing-pricerules:cache-config-ext.xml

core

core:cache-config

marketing

int-module-marketing-pricerules:dao-ext

pim

int-module-pim-icecat:dao-ext

OK

Cluster

ID	Type
EDEMO.JAM	ADM v.3.7.0.E-SNAPSHOT-rev.0895e3ed
EDEMO.SF0	SFG v.3.7.0.E-SNAPSHOT-rev.0895e3ed
EDEMO.SF1	API v.3.7.0.E-SNAPSHOT-rev.0895e3ed

Cluster namespace and App Node type

App Node mode

Indicator of App Node supporting Full text search

DEMO ENVIRONMENT

Logged in as Yes

## Application logs

All platform applications use Logback over SLF4j configuration. Capabilities of Logback a beyond the scope of this discussion and we highly recommend reviewing logback official documentation if you are not familiar with this framework.

## Shop specific logging

For multi-tenant setups it is recommended to sift logs by shop discriminator. Platform provides several flavours of ready to use implementations:

## ShopCodeLogDiscriminator

Uses current shop context to set **shopCode** variable, which produces value such as "SHOP10". This value can be used in file names to direct message of specific shop into a separate file.

```
<appender name="SHOPPAY"
class="ch.qos.logback.classic.sift.SiftingAppender">
  <!-- declare discriminator -->
  <discriminator
class="org.yes.cart.utils.log.ShopCodeLogDiscriminator"/>
  <sift>
    <!-- shopCode can be use in appender name (e.g. view in JMX) -->
    <appender name="SHOPPAY-${shopCode}"
class="ch.qos.logback.core.rolling.RollingFileAppender">
      <!-- shopCode can be use in file name -->
      <File>${catalina.base}/logs/yc-${shopCode}-pay.log</File>
      <Append>true</Append>
      <encoder>
        <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %5p %c{1}:%L -
%m%n</pattern>
      </encoder>
      <rollingPolicy
class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
        <maxIndex>10</maxIndex>
        <FileNamePattern>${catalina.base}/logs/yc-${shopCode}-p
ay.log.%i.zip</FileNamePattern>
      </rollingPolicy>
      <triggeringPolicy
class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
        <MaxFileSize>10MB</MaxFileSize>
      </triggeringPolicy>
    </appender>
  </sift>
</appender>
```

## ShopCodeAndLevelLogDiscriminator

Uses current shop context + log message level to set **shopCode** variable, which produces value such as "SHOP10-WARN". This value can be used in file names to direct message of specific level of specific shop into a separate file

```




<appender name="DEFAULT"
class="ch.qos.logback.classic.sift.SiftingAppender">
  <!-- declare discriminator -->
  <discriminator
class="org.yes.cart.utils.log.ShopCodeAndLevelLogDiscriminator"/>
  <sift>
    <!-- shopCode can be use in appender name (e.g. view in JMX) -->
    <appender name="DEFAULT-${shopCode}"
class="ch.qos.logback.core.rolling.RollingFileAppender">
      <!-- shopCode can be use in file name -->
      <File>${catalina.base}/logs/yc-${shopCode}.log</File>
      <Append>true</Append>
      <encoder>
        <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %5p %c{1}:%L -
%m%n</pattern>
      </encoder>
      <rollingPolicy
class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
        <maxIndex>10</maxIndex>
        <FileNamePattern>${catalina.base}/logs/yc-${shopCode}.1
og.%i.zip</FileNamePattern>
      </rollingPolicy>
      <triggeringPolicy
class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
        <MaxFileSize>10MB</MaxFileSize>
      </triggeringPolicy>
    </appender>
  </sift>
</appender>

```

## Out of the box logs

Log file(s)	Sift	Level	Purpose
<b>Storefront (API, SFx)</b>			
yc-\${shopCode}.log	shop code + level	INFO	DEFAULT-\${shopCode} appender (root)
yc-\${shopCode}-pay.log	shop code	DEBUG	SHOPPAY appender, payment gateway related logging, packages: <ul style="list-style-type: none"> <li>org.yes.cart.web.filter.payment</li> <li>org.yes.cart.payment.impl</li> <li>org.yes.cart.web.page.payment.callback</li> </ul>
yc-orderexport.log		INFO	ORDEREXPORT appender, order export related logging, packages: <ul style="list-style-type: none"> <li>org.yes.cart.orderexport</li> </ul>
yc-orderstate.log		INFO	ORDERSTATE appender, order transition related logging, packages: <ul style="list-style-type: none"> <li>org.yes.cart.service.order</li> </ul>

yc-mail.log		INFO	MAIL appender, email generation and send logging, packages: <ul style="list-style-type: none"> <li>org.yes.cart.service.mail.impl.MailComposerImpl</li> <li>org.yes.cart.bulkjob.mail.BulkMailProcessorImpl</li> <li>org.yes.cart.domain.message.consumer.CustomerRegistrationMessageListener</li> <li>org.yes.cart.domain.message.consumer.ManagerRegistrationMessageListener</li> <li>org.yes.cart.domain.message.consumer.StandardMessageListener</li> <li>org.yes.cart.web.aspect.ContactFormAspect</li> <li>org.yes.cart.web.aspect.NewsletterAspect</li> <li>org.yes.cart.web.aspect.RegistrationAspect</li> <li>org.yes.cart.service.domain.aspect.impl.CustomerRegistrationAspect</li> <li>org.yes.cart.service.domain.aspect.impl.ManagerRegistrationAspect</li> <li>org.yes.cart.service.domain.aspect.impl.PaymentAspect</li> <li>org.yes.cart.service.domain.aspect.impl.BaseOrderStateAspect</li> <li>org.yes.cart.service.domain.aspect.impl.OrderStateChangeListenerAspect</li> <li>org.yes.cart.orderexport.mail.EmailNotificationOrderExporterImpl</li> </ul>
yc-maildump.log		INFO	MAILDUMP appender, log full content of email that was sent
yc-audit.csv		INFO	AUDIT appender, log persistence updates audit records ⚠ Change to TRACE to enable audit logging
yc-\${shopCode}-ftq.log	shop code	INFO	FTQ-\${shopCode} appender, logs all full text queries ⚠ Change to DEBUG to enable query logging ⚠ Change to TRACE to enable query explanations logging
yc-ws.log		ERROR	WS appender, logs web services communication (WS.IN and WS.OUT) ⚠ Change to INFO to enable logging
yc-config.log		INFO	CONFIG appender, logs information on loaded modules and extension points ⚠ Change to DEBUG to enable logging
yc-\${shopCode}-sac.log	shop code	DEBUG	SAC (Security access control) appender, logs access violations
<b>Admin (ADM)</b>			
yc-\${shopCode}.log	shop code + level	INFO	DEFAULT-\${shopCode} appender (root)
yc-\${shopCode}-pay.log	shop code	DEBUG	SHOPPAY appender, payment gateway related logging, packages: <ul style="list-style-type: none"> <li>org.yes.cart.web.filter.payment</li> <li>org.yes.cart.payment.impl</li> <li>org.yes.cart.web.page.payment.callback</li> </ul>
yc-\${shopCode}-job.log	shop code + level	INFO	JOB-\${shopCode} appender, logs all tasks executions, packages: <ul style="list-style-type: none"> <li>org.yes.cart.bulkjob</li> <li>org.yes.cart.service.async.impl</li> </ul>
yc-\${shopCode}-import.log	shop code + level	INFO	BULKIMPORT-\${shopCode} appender, logs all data imports, packages: <ul style="list-style-type: none"> <li>org.yes.cart.bulkimport</li> <li>org.yes.cart.bulkjob.bulkimport</li> <li>org.yes.cart.service.async.impl</li> </ul>
yc-remote.log		INFO	REMOTE (file upload/download/move/delete operations) appender, logs all data imports, packages: <ul style="list-style-type: none"> <li>org.yes.cart.remote</li> </ul>
yc-orderexport.log		INFO	ORDEREXPORT appender, order export related logging, packages: <ul style="list-style-type: none"> <li>org.yes.cart.orderexport</li> </ul>
yc-orderstate.log		INFO	ORDERSTATE appender, order transition related logging, packages: <ul style="list-style-type: none"> <li>org.yes.cart.service.order</li> </ul>

yc-mail.log		INFO	MAIL appender, email generation and send logging, packages: <ul style="list-style-type: none"> <li>org.yes.cart.service.mail.impl.MailComposerImpl</li> <li>org.yes.cart.bulkjob.mail.BulkMailProcessorImpl</li> <li>org.yes.cart.domain.message.consumer.CustomerRegistrationMessageListener</li> <li>org.yes.cart.domain.message.consumer.ManagerRegistrationMessageListener</li> <li>org.yes.cart.domain.message.consumer.StandardMessageListener</li> <li>org.yes.cart.web.aspect.ContactFormAspect</li> <li>org.yes.cart.web.aspect.NewsletterAspect</li> <li>org.yes.cart.web.aspect.RegistrationAspect</li> <li>org.yes.cart.service.domain.aspect.impl.CustomerRegistrationAspect</li> <li>org.yes.cart.service.domain.aspect.impl.ManagerRegistrationAspect</li> <li>org.yes.cart.service.domain.aspect.impl.PaymentAspect</li> <li>org.yes.cart.service.domain.aspect.impl.BaseOrderStateAspect</li> <li>org.yes.cart.service.domain.aspect.impl.OrderStateChangeListenerAspect</li> <li>org.yes.cart.orderexport.mail.EmailNotificationOrderExporterImpl</li> </ul>
yc-maildump.log		INFO	MAILDUMP appender, log full content of email that was sent
yc-audit.csv		INFO	AUDIT appender, log persistence updates audit records  <a href="#">Change to TRACE to enable audit logging</a>
yc-ws.log		ERROR	WS appender, logs web services communication (WS.IN and WS.OUT)  <a href="#">Change to INFO to enable logging</a>
yc-config.log		INFO	CONFIG appender, logs information on loaded modules and extension points  <a href="#">Change to DEBUG to enable logging</a>
yc-\${shopCode}-sac.log	shop code	DEBUG	SAC (Security access control) appender, logs access violations
yc-security.log		INFO	SECURITY (Spring) appender, packages: <ul style="list-style-type: none"> <li>org.springframework.security</li> </ul>