TECHNICAL NOTE

OutSystems® Platform

Architecture and Infrastructure Overview

The OutSystems® Platform architecture has been designed with a strong focus on performance, scalability, and high-availability. This technical note describes the OutSystems Platform major concepts, its logical and physical architectures, as well as the infrastructure you need to set it up. Most of these architecture concepts apply also to the OutSystems Platform-as-a-Service, where OutSystems takes care of the infrastructure setup and maintenance for you. For more details, read the Enterprise Cloud Tech Note.

Table of Contents

1 OutSystems Platform Architecture overview

The OutSystems Platform complements standard Web Application Server stacks with an extra set of services and repositories. The following figure presents the main components of the OutSystems Platform and the architecture of the Platform Server.

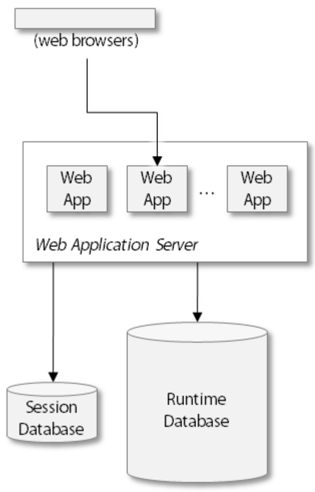


Figure 1. Typical 4-tier Web Application Architecture.

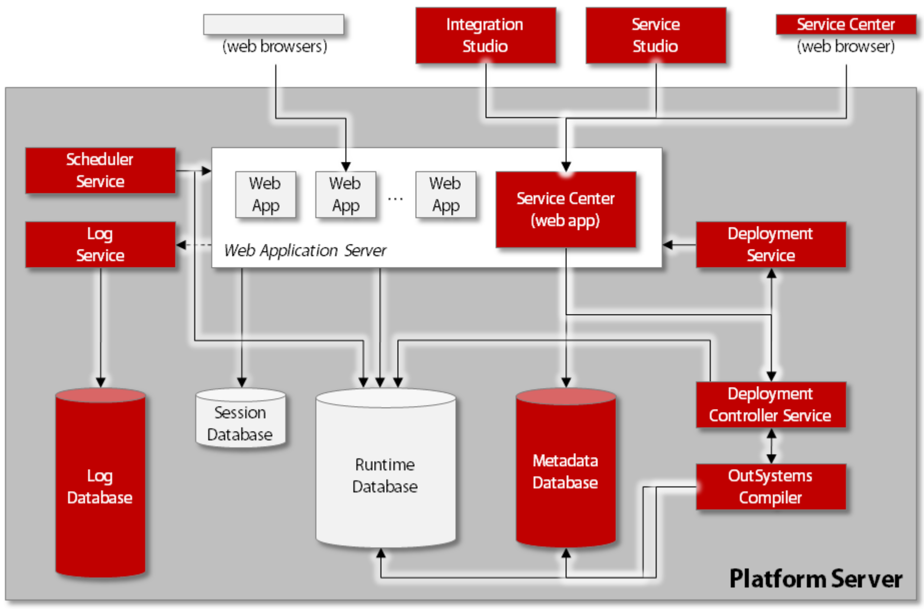


Figure 2. OutSystems Platform architecture. Service Studio and Integration Studio are desktop tools that interact with the Platform Server via Web services. The Platform Server complements a standard Web Application Server stack with extra services and repositories (in red).

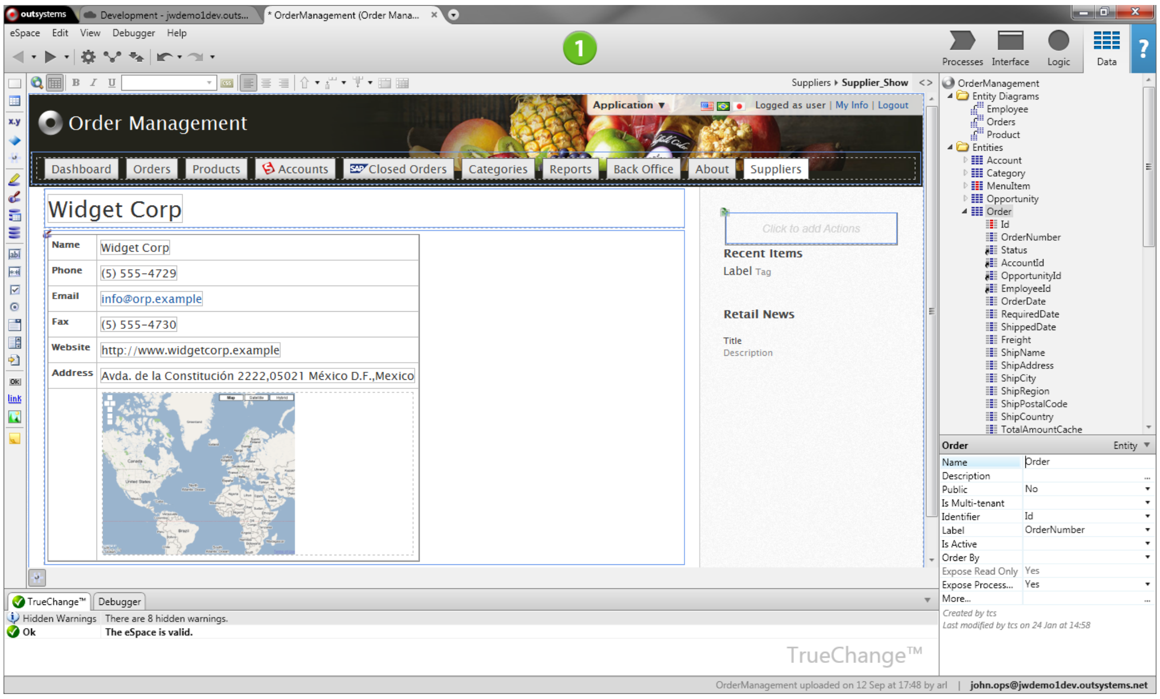
The Platform Server components take care of all the steps required to generate, build, package, and deploy native C# and Java web applications on top of a Microsoft stack, Oracle WebLogic or JBOSS. Contrary to other technologies, the OutSystems Platform doesn't use a proprietary runtime engine. All applications built with the OutSystems Platform rely upon nothing but standard .NET or Java technology.

2 Assembling and publishing an application for execution

2.1 Service Studio

Service Studio is a desktop environment targeted at business-minded developers to assemble and change web and mobile web business applications using visual models. With Service Studio business developers assemble all components necessary to completely define a web business application, using a visual drag-and-drop paradigm.

The tool also enables the modeling of User Interfaces, Business Processes, Business Logic, Databases, Integration Components, SAP BAPIs, SOAP and REST Web Services, Security Rules and Scheduling activities. Service Studio embeds a full-reference checking and self-healing engine (TrueChange™) that assures error-free, robust change across all application components.



Widget Corp

Name	Widget Corp
Phone	(5) 555-4729
Email	info@orp.example
Fax	(5) 555-4730
Website	http://www.widgetcorp.example
Address	Avda. de la Constitución 2222,05021 México D.F.,Mexico

Order	Entity
Name	Order
Description	
Public	No
Is Multi-tenant	
Identifier	Id
Label	OrderNumber
Is Active	
Order By	
Expose Read Only	Yes
Expose Process...	Yes
More...	

Created by tcs
Last modified by tcs on 24 Jan at 14:58

Hidden Warnings There are 8 hidden warnings.
Ok The eSpace is valid.

OrderManagement uploaded on 12 Sep at 17:48 by arl | john.ops@jwdemo1dev.outsystems.net

Figure 3. A screenshot of Service Studio. The OutSystems Platform's development environment.

2.2 1-Click-Publish

When the developer clicks (see Figure 4) on the 1-Click Publish button the Platform 1) validates the application model using the TrueChange™ engine and 2) creates a new version of the application in the Metadata Database. The OutSystems Compiler 3) takes the application model and all its dependencies and generates a standard .NET or JEE application. The Deployment Controller takes the generated application and all its components and dependencies and 4) through the Deployment Service deploys 5) the application onto the Application Server at the same time updating the database model of the application in the Runtime Database.

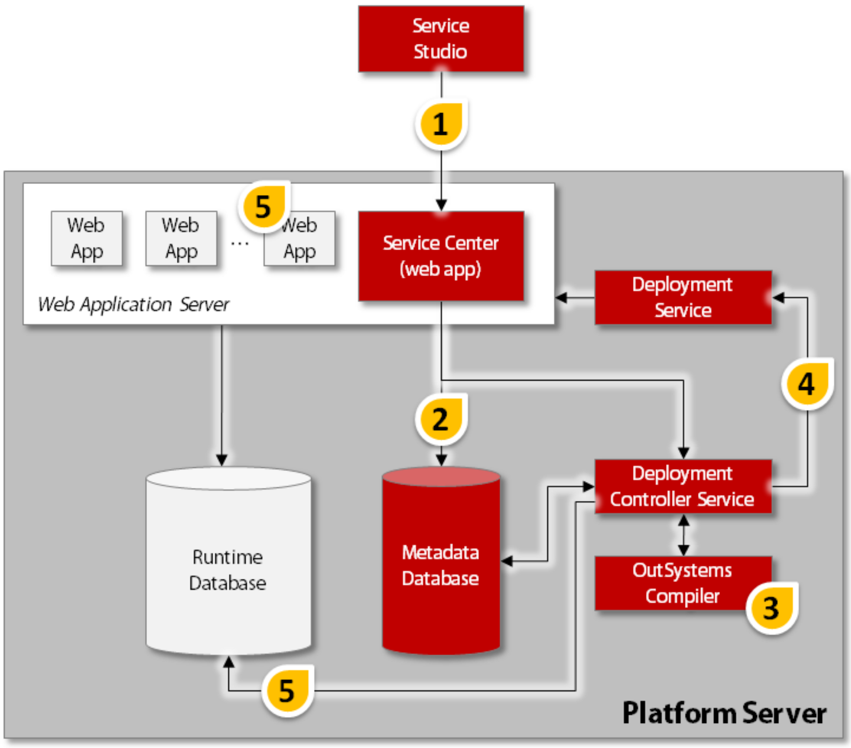


Figure 4. 1-Click-Publish mechanism in action.

2.3 Remote developer sandboxes

The OutSystems Platform promotes constant collaboration across teams of developers working together on the same application. Personal *remote sandboxes* allow developers to test their own work in a server environment (instead of a desktop environment), access a production-grade database, leverage integrations done at the server level, and not interfere with their teammates' work before publishing their changes to a centralized version repository.

When a developer decides to publish his own version of an application, he may receive notifications about what has been changed by others and what service dependencies are no longer up-to-date. With that information in hand, he can take advantage of a visual difference and merge tool that is used to rapidly combine everyone's changes. This takes advantage of the Platform's version control capabilities, meaning that the information about what has changed is obtained directly from the central version control service.

2.4 Extending the Platform and integrating with other systems

Integration Studio is a desktop environment, targeted at technical developers, used to integrate external libraries, services, and databases. Integration Studio works together with Microsoft Visual Studio to help build custom .NET components (assemblies) and with Eclipse to help build custom J2EE components (JARs) that extend the OutSystems Platform with additional functionality.

If you already have your own Java, COM, and .NET client APIs, or need to integrate with existing external databases you can use Integration Studio's discovery wizards. These will create all the required stubs of code and method signature wrappers. These wrappers can be further customized through the use of Microsoft Visual Studio or Eclipse and let you edit the implementation code.

In a .Net environment (scenario A of Figure 5.), Integration Studio 1) imports the library we want to integrate with and 2) guides the developer in creating the methods signatures that will be visible for Service Studio developers. A C# source wrapper file with the signatures is generated and 3) compiled into code. The complete package (assembly) is 4) published on the Platform and 5) stored in the Metadata Database. Any application developed in Service Studio can reuse this new component where necessary. An equivalent process is used for Java environments (scenario B).

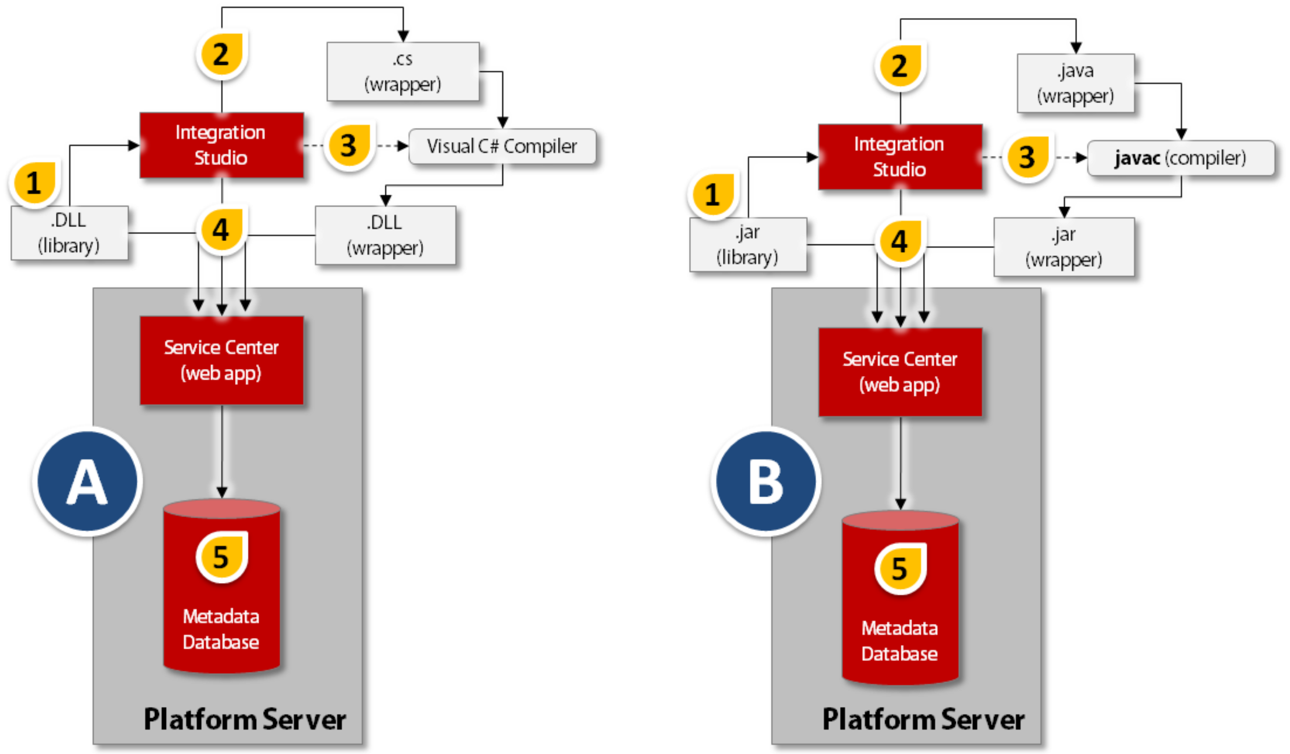


Figure 5. An example of using Integration Studio to extend the OutSystems Platform with new functionality.

3 Platform Server

The Platform Server components and services are clustered in 3 types of servers: the front-end servers, the deployment controller server and the database server.

3.1 Front-End Server

The *front-end server* is a standard Web Application Server (IIS or Java Application Server) environment complemented with 3 extra OutSystems services. A Platform Server installation can have several front-end servers. Each Front-End Server consists of the following components:

- **The Web Application Server (WAS).** Each front-end server runs an instance of the WAS. For .Net deployments the WAS is IIS. For Java deployments the WAS is WebLogic or JBOSS.

> Please refer to the OutSystems Platform System Requirements on-line reference for complete and up-to-date information on supported versions and configurations.

- **Deployment Service.** The Deployment Service works in tandem with the Deployment Controller Service and ensures that the compiled applications are installed and deployed on each Front-End Server. In a farm configuration, the upgrade of an application is assured to be consistent across all front-end servers by having the Deployment Controller Service orchestrate the deployment with all Deployment Services present at each front-end server.
- **Scheduler Service**: Manages the timely execution of scheduled jobs resulting from the compilation of *Timer* objects and business processes' activities modeled inside Service Studio.
- **Log Service**: Provides extensive, asynchronous logging services to store performance, error, and business application custom audit events generated by all running applications.

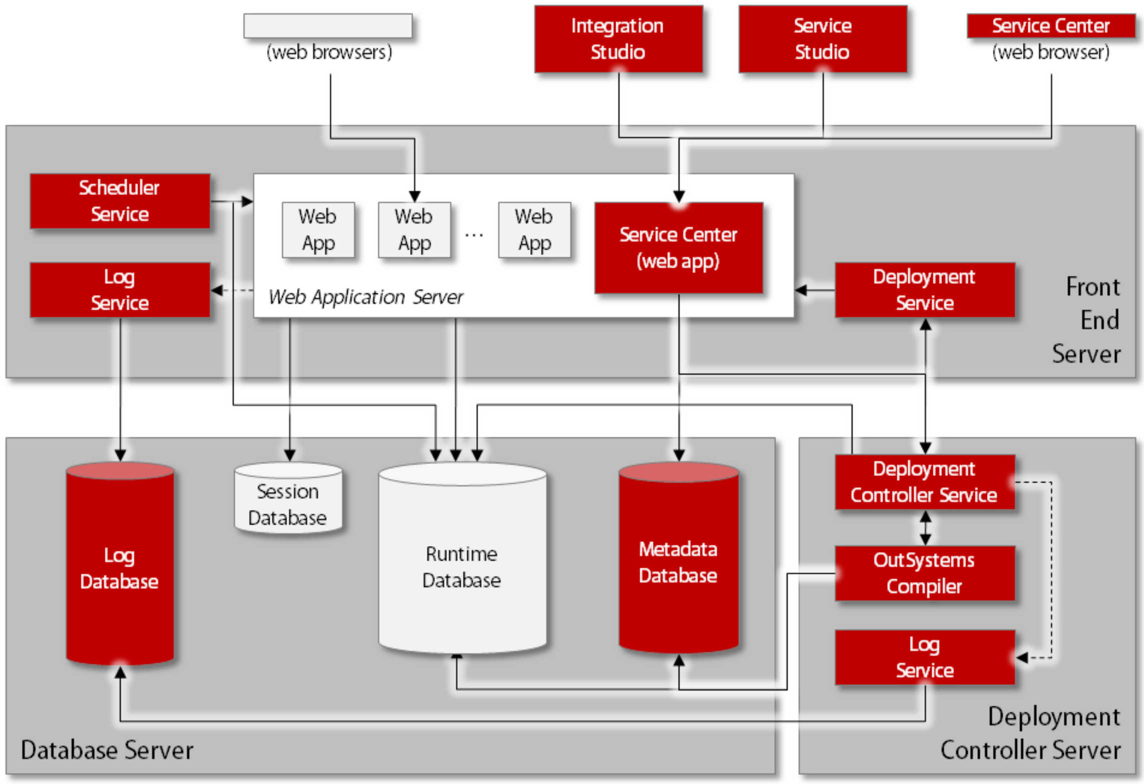


Figure 6. The Platform Server components and services are clustered in 3 types of "servers": the front-end servers, the deployment controller server and the database server. Each of these server clusters can be installed across multiple physical or virtual machines in a data center or cloud enabled environment.

3.2 Deployment Controller Server

The Deployment Controller Server is in charge of compiling the application's models, and deploying the compilation result in the Front-End Servers. There is only one Deployment Controller Server for each Platform Server installation. The Deployment Controller Server consists of the following sub-components:

- **OutSystems Compiler**: Compiles the application's models and generates the corresponding C# or Java application code. It also inspects the Metadata Database and generates the required SQL Scripts to upgrade the Runtime Database to match the new data model definitions.
- **Deployment Controller Service**: Invokes the OutSystems Compiler and database update script and then coordinates the deployment of the generated application across all front-end servers as well as updating the Runtime Database.

3.3 Database Server

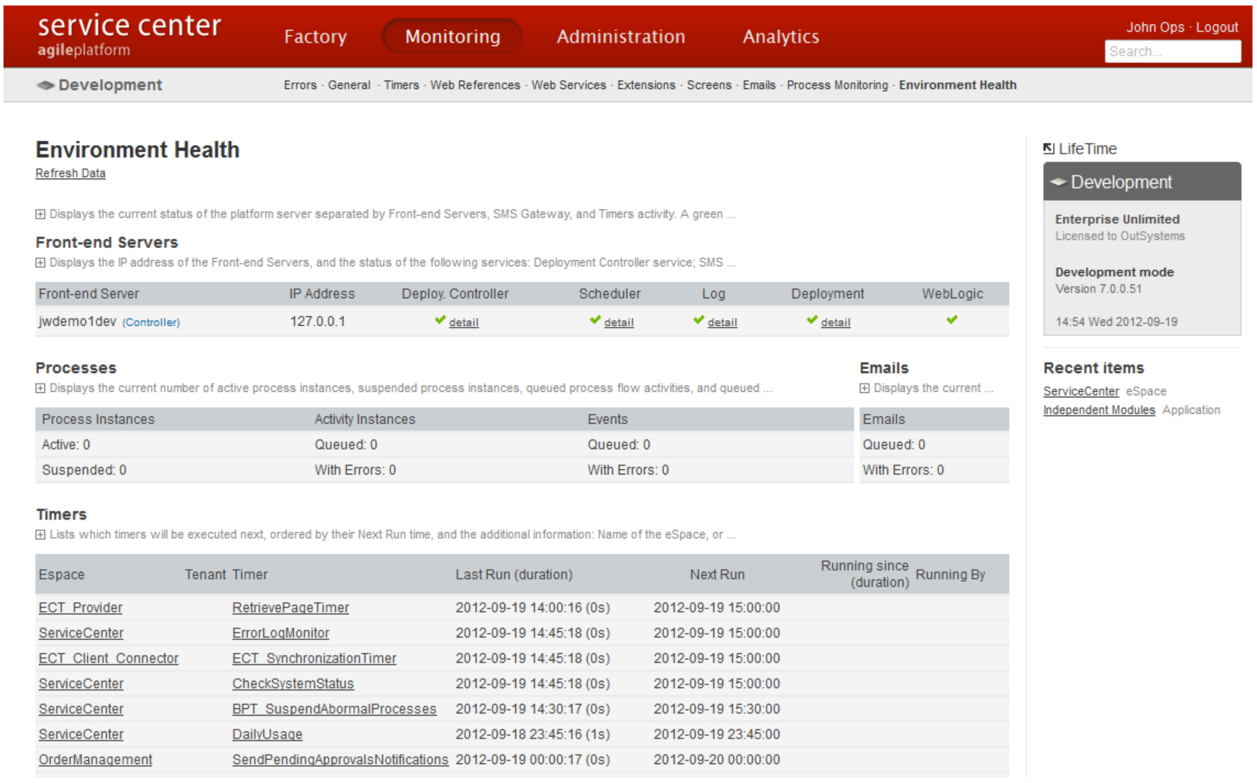
The Database Server is a Relational Database Management System (RDBMS), such as Microsoft SQL Server or Oracle database. The Platform Server groups database elements onto 4 different sets with specific purposes:

- **Runtime**: Stores all application data. This is the database used by the web applications to store and retrieve application data.
- **Metadata**: Supports the OutSystems Platform meta-model. This is where the OutSystems Platform stores application versions, platform users, security configuration policies, etc.
- **Log**: Contains the processed logs generated by the execution of all web applications supported by the Platform. These logs are created by the Log Service as it processes application events, and are the source of performance monitoring facilities presented in the Service Center administration console.
- **Session**: Keeps the user sessions' state managed by the distributed farm of Application Servers. The session database elements are configured on a dedicated database schema for improved performance and operations.

3.4 Service Center

Service Center is a web application that interacts with all the components of the Platform Server. It provides a web browser interface for system administrators, managers and operation teams to control every aspect of the OutSystems Platform. Service Center is also the application with which Service Studio and Integration Studio interact with the Platform Server via web services. Service Center is itself a web application developed with Service Studio.

Service Center provides a centralized entry point for IT operations to manage, monitor, and troubleshoot the OutSystems Platform's application portfolio. All applications and services developed with the OutSystems Platform benefit from built-in instrumentation that provides system administrators with an enterprise-grade viewport into the OutSystems Platform's health, ranging from consolidated error logging to web page and database query performance metrics.


service center
agileplatform
Factory
Monitoring
Administration
Analytics
John Ops · Logout
Search...
Development
Errors · General · Timers · Web References · Web Services · Extensions · Screens · Emails · Process Monitoring · Environment Health
LifeTime
Development

Environment Health
Refresh Data

⊞ Displays the current status of the platform server separated by Front-end Servers, SMS Gateway, and Timers activity. A green ...

Front-end Servers
⊞ Displays the IP address of the Front-end Servers, and the status of the following services: Deployment Controller service; SMS ...

Front-end Server	IP Address	Deploy. Controller	Scheduler	Log	Deployment	WebLogic
jwdemo1dev (Controller)	127.0.0.1	✓ detail	✓ detail	✓ detail	✓ detail	✓

Processes
⊞ Displays the current number of active process instances, suspended process instances, queued process flow activities, and queued ...

Process Instances	Activity Instances	Events
Active: 0	Queued: 0	Queued: 0
Suspended: 0	With Errors: 0	With Errors: 0

Emails
⊞ Displays the current ...

Emails
Queued: 0
With Errors: 0

Timers
⊞ Lists which timers will be executed next, ordered by their Next Run time, and the additional information: Name of the eSpace, or ...

Espace	Tenant	Timer	Last Run (duration)	Next Run	Running since (duration)	Running By
ECT_Provider		RetrievePageTimer	2012-09-19 14:00:16 (0s)	2012-09-19 15:00:00		
ServiceCenter		ErrorLogMonitor	2012-09-19 14:45:18 (0s)	2012-09-19 15:00:00		
ECT_Client_Connector		ECT_SynchronizationTimer	2012-09-19 14:45:18 (0s)	2012-09-19 15:00:00		
ServiceCenter		CheckSystemStatus	2012-09-19 14:45:18 (0s)	2012-09-19 15:00:00		
ServiceCenter		BPT_SuspendAbormalProcesses	2012-09-19 14:30:17 (0s)	2012-09-19 15:30:00		
ServiceCenter		DailyUsage	2012-09-18 23:45:16 (1s)	2012-09-19 23:45:00		
OrderManagement		SendPendingApprovalsNotifications	2012-09-19 00:00:17 (0s)	2012-09-20 00:00:00		

Enterprise Unlimited
Licensed to OutSystems

Development mode
Version 7.0.0.51

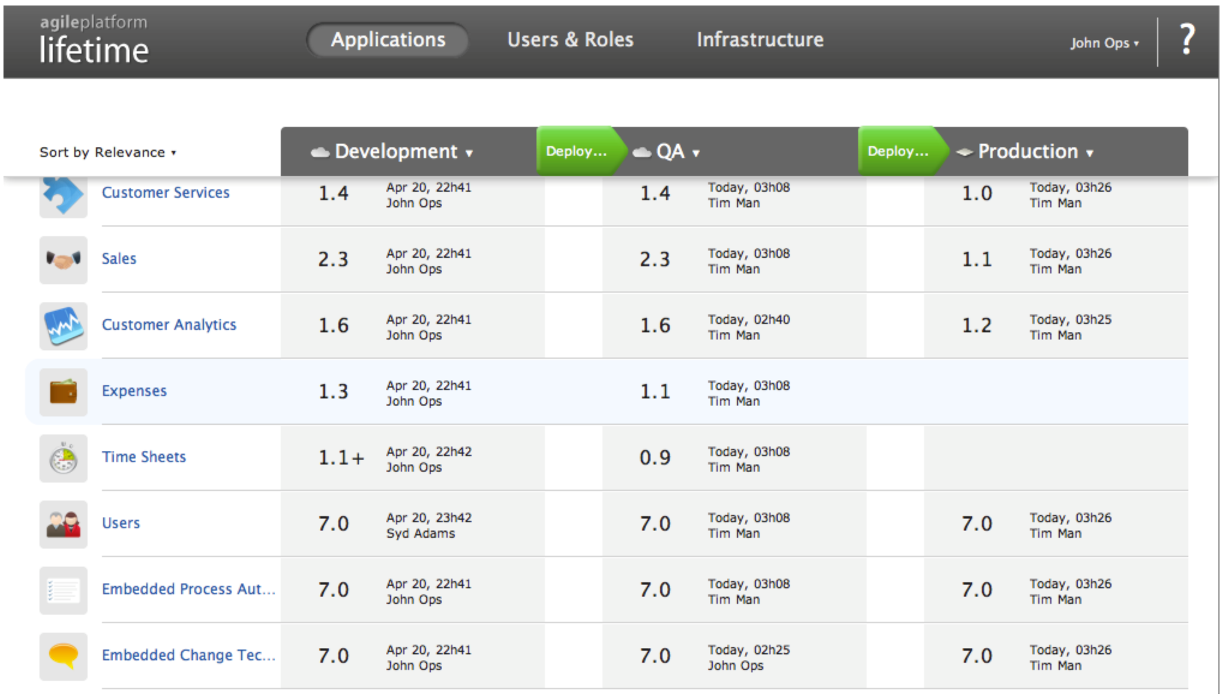
14:54 Wed 2012-09-19

Recent items
ServiceCenter eSpace
Independent Modules Application

Figure 7. A screenshot of Service Center. The OutSystems Platform's administration console.

3.5 Lifetime

Lifetime is a web application that extends Service Center capabilities to cross-environment scenarios. It provides a web browser interface for IT teams (developers and operations) to manage the application portfolio across all environments, define cross-environment security policies, and manage the complete lifecycle of applications, from development to production.



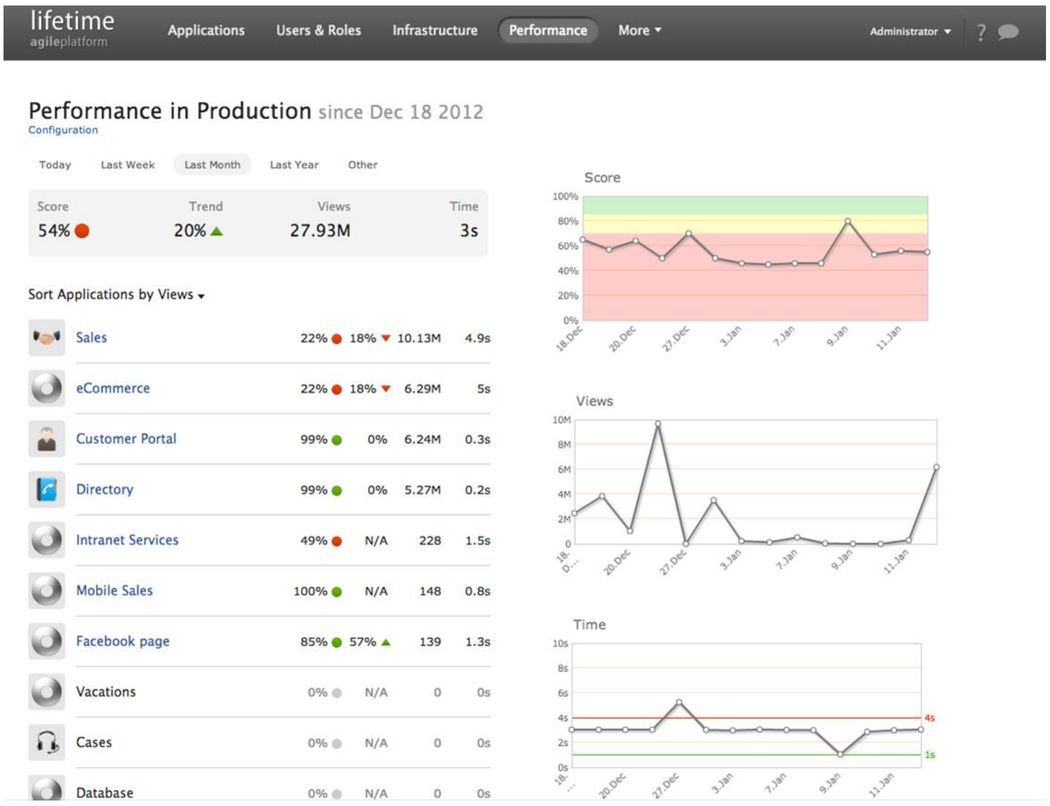
	Development ▾		QA ▾		Production ▾	
Customer Services	1.4	Apr 20, 22h41 John Ops	1.4	Today, 03h08 Tim Man	1.0	Today, 03h26 Tim Man
Sales	2.3	Apr 20, 22h41 John Ops	2.3	Today, 03h08 Tim Man	1.1	Today, 03h26 Tim Man
Customer Analytics	1.6	Apr 20, 22h41 John Ops	1.6	Today, 02h40 Tim Man	1.2	Today, 03h25 Tim Man
Expenses	1.3	Apr 20, 22h41 John Ops	1.1	Today, 03h08 Tim Man		
Time Sheets	1.1+	Apr 20, 22h42 John Ops	0.9	Today, 03h08 Tim Man		
Users	7.0	Apr 20, 23h42 Syd Adams	7.0	Today, 03h08 Tim Man	7.0	Today, 03h26 Tim Man
Embedded Process Aut...	7.0	Apr 20, 22h41 John Ops	7.0	Today, 03h08 Tim Man	7.0	Today, 03h26 Tim Man
Embedded Change Tec...	7.0	Apr 20, 22h41 John Ops	7.0	Today, 02h25 John Ops	7.0	Today, 03h26 Tim Man

Figure 8. A screenshot of Lifetime. The OutSystems Platform's cross-environment application portfolio lifecycle management console.

Lifetime interacts with each environment's Service Center via firewall-friendly web services. Lifetime is itself a web application developed with Service Studio.

3.6 User Performance Monitoring

Performance Monitoring is a LifeTime module that monitors the performance, for every application deployed, from the point of view of users. Using an industry standard indicator of user satisfaction, the Application Performance Index, the Performance Monitor measures the time from the moment a user clicks a link until the moment the page finishes loading. No more looking at system specific metrics, such as CPU or Memory usage levels, that don't provide insight into the real user experience.



Performance in Production since Dec 18 2012

Score	Trend	Views	Time
54% ●	20% ▲	27.93M	3s

Sales	22% ●	18% ▼	10.13M	4.9s
eCommerce	22% ●	18% ▼	6.29M	5s
Customer Portal	99% ●	0%	6.24M	0.3s
Directory	99% ●	0%	5.27M	0.2s
Intranet Services	49% ●	N/A	228	1.5s
Mobile Sales	100% ●	N/A	148	0.8s
Facebook page	85% ●	57% ▲	139	1.3s
Vacations	0% ●	N/A	0	0s
Cases	0% ●	N/A	0	0s
Database	0% ●	N/A	0	0s

Figure 9. A screenshot of the Performance Monitoring module.

Drill down to the screen level and assess the performance across time to understand that a simple change in the homepage of an application was having a significant impact on the experience of your users; understand if applications are degrading in performance as usage grows, or if your system is ready to handle increasing demand. And since all monitoring operations are performed asynchronously, your clients won't notice any performance impact.

3.7 Session Management

Web or Mobile Web user sessions are automatically managed by the OutSystems Platform. User sessions are stored in the Session database schema, thus allowing for multiple requests from the same user to be handled by any Front-end Server in a farm environment.

3.8 Batch/Job Processing

The OutSystems Platform supports the execution of batch jobs modeled with Timer objects in Service Studio. The Scheduler Service wakes up at scheduled times, retrieves the next pending job from the Metadata Database and starts its execution by invoking the scheduler code. Schedulers are distributed on all Front-end Servers and job load distribution is controlled via the Metadata Database.

3.9 Application Instrumentation and Log Processing

When the Platform Server compiles an application it injects performance management hooks in the generated code. The application becomes automatically instrumented and at runtime produces a continuous stream of audit events. The OutSystems Platform detaches the generation of these events from its processing so as not generate any overhead in the normal operation of the application.

As presented in the next figure, events are 1) written in a Message Queue, and 2) asynchronously retrieved by the *Log Service* (part of each Front End Server). The Log service processes the events in the Queue and 3) updates the *Log Database*. Service Center 4) leverages this information to provide a complete real-time monitoring and analytics web interface to Operations and Maintenance teams. The Log Service runs at lower priority so that it never competes with the normal processing of web requests. This architecture removes the need for developers to insert explicit instrumentation code in the application and has no performance impact on the normal operation of the running applications.

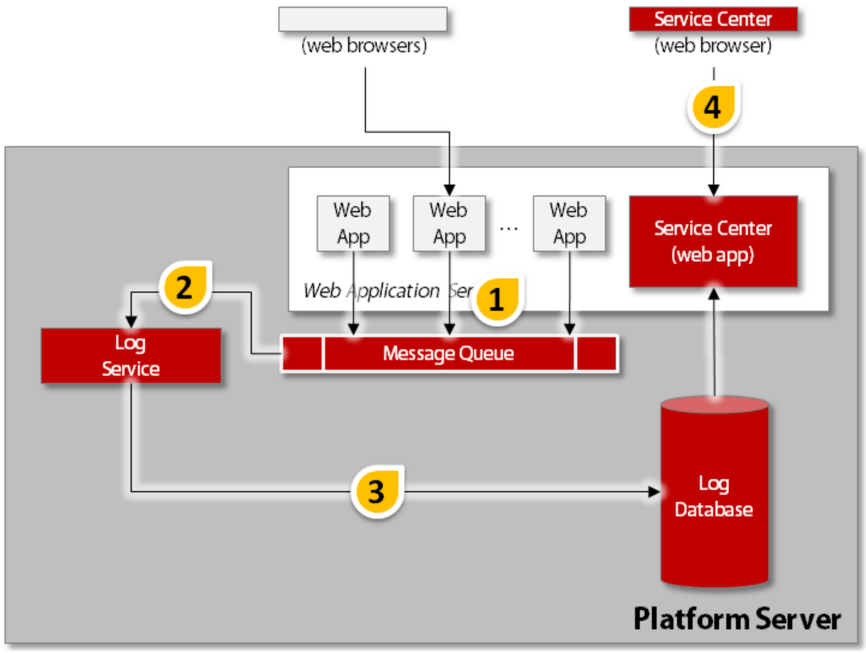


Figure 10. Web applications generate log events, which are written to a Message Queue, processed by the Log Service and stored in the Log Database for performance monitoring.

4 Physical Architecture Configurations

The Platform Server can be configured with different physical architectures to cope with different scalability, performance, reliability, and security requirements. Each installation of the OutSystems Platform is called an *environment*. A typical organization has 3 environments: development, quality (or pre-production) and production. Because of the sophisticated access control and management capabilities of the platform you can configure each environment according to each function.

4.1 Single Node configurations

Single node installations are common for non-production environments. This configuration is typically based on two machines – one for the Application Server and one for the Database Server - but using a single machine is also a possibility for smaller development teams.

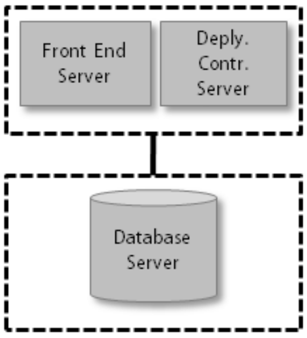


Figure 11. Typical two machine configuration adequate for Development and QA environments.

Another common setup is to have a single Database Server supporting all non-production environments using a different Database Catalog or Schema for each environment.

4.2 Development, Quality and Production environments

In enterprise scenarios the following configurations of environments are common:

- **Development Environment**. In development you create accounts for all developers and development managers. The Development environment is the one where Service Studio and Integration Studio are typically used.
- **Quality Environment**. Quality is used by Testers and Business Users to experiment with production candidates or (agile) sprint versions of the applications. They have very little scalability and redundancy requirements.
- **Production Environment**. This environment should be controlled by Operations but we advise that you setup read-only access accounts for development/maintenance teams to access analytics information on performance and application errors.

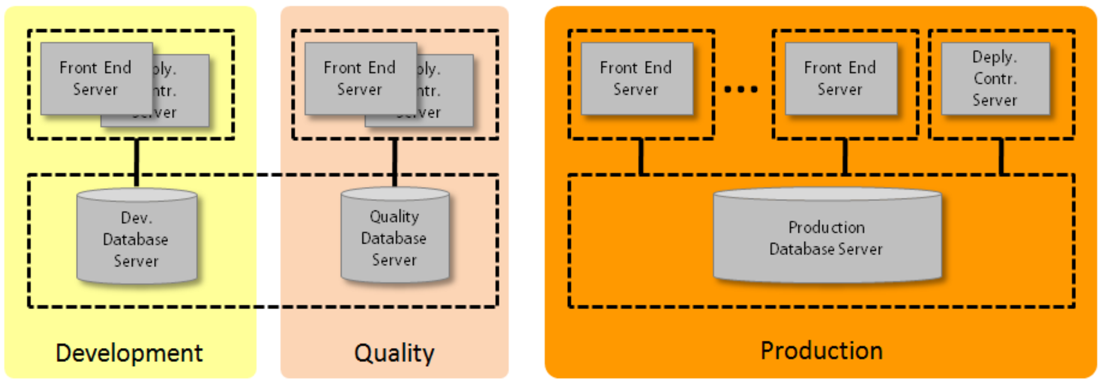


Figure 12. Typical virtual or physical machine allocation for 3 environments (in dashed lines). The development and the quality databases share the same machine resource (virtual or physical).

You can adapt these basic scenarios to your needs. Among the spectrum of OutSystems customers we can find dozens of variations.

4.3 Farm configurations

The Platform Server can be installed in a farm configuration for scalability and high availability. In these configurations a network load balancer (not included with the Platform Server) distributes web requests among the multiple front-end servers to spread request load.

To remove single point of failures, high-availability installations can also leverage the cluster mode of Microsoft SQL Server or the Oracle Real Application Cluster (RAC). Farm configuration is the most common setup for production environments.

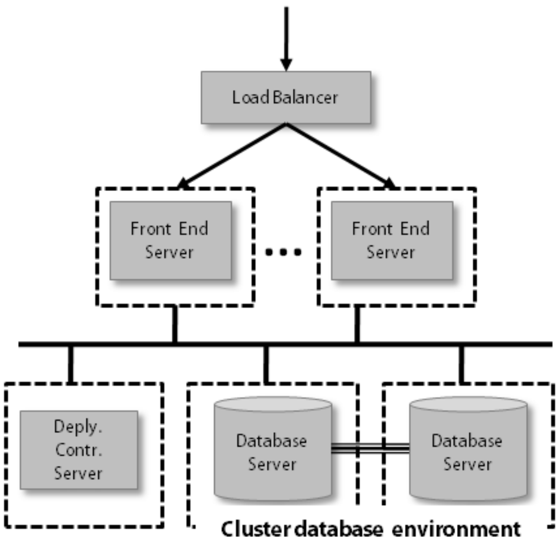


Figure 13. Typical farm configuration for high availability and scalability scenarios.

The addition of a new front-end server to a Platform Server installation is a simple process. Through Service Center an operator can configure a new farm node and have the platform automatically sync the applications on the new server with the other front-end servers.

All front-end servers on the farm will log to centralized repositories to allow for efficient monitoring, troubleshooting and accessing performance metrics in Service Center.

4.4 Selective Farm Deployment with Zones

OutSystems Platform's front-end servers can be organized in *Zones* to reflect different access control or load balancing configurations. The platform's services coordinating application deployment use these configurations to decide in which front-ends a given application should be available. *Zones* are easily configured through Service Center.

Selected deployment of applications across multi-channel infrastructures – e.g. Intranet and Extranet - can be enforced with the configuration of corresponding *Zones*. Applications that are Intranet-only are never deployed to public-facing DMZ Extranet servers. *Zones* can also be used to isolate mission-critical applications that require dedicated servers.

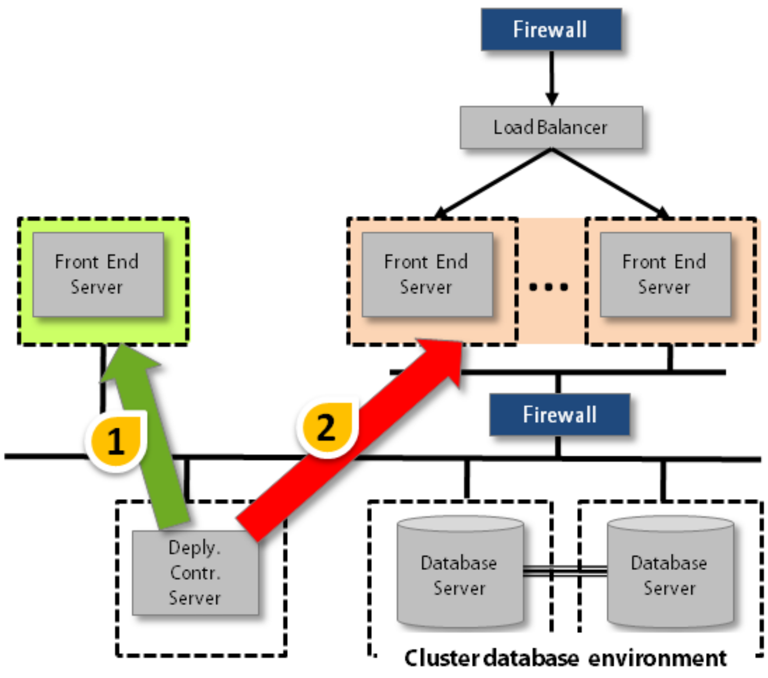


Figure 14. Typical blended scenario with 3 front-end servers organized in 2 zones. Applications associated to the Intranet Zone (in green) are 1) *never* deployed onto the front-end servers of the Extranet Zone. The Deployment Controller only publishes externally the applications that are defined as being in Extranet Zone (2).