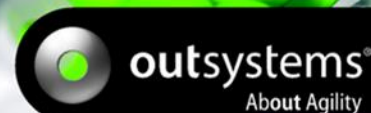


OutSystems® 10

The Detach Process for Java



This document is a step-by-step guide for extracting the source code of your applications from the Java version of OutSystems® and how to set it up to be executed and maintained independently of OutSystems, if you ever decide to detach from it.

This is a complement to the [OutSystems Platform – Standard Architecture with No Lock-in](#) technical note that gives a high-level view on how OutSystems, unlike other proprietary technologies and frameworks, generates standard, optimized, and fully documented Java source code that does not require runtime interpreters or engines.

Table of Contents

1	Detach process overview	2
2	Before you start.....	2
2.1	What will you lose?	2
2.2	Recommendations before you start the detach process	3
2.2.1	Personnel requirements	3
2.3	OutSystems Scheduler Service	3
2.4	Understand how OutSystems structures the code	3
2.5	Architecture of the Detached Application	6
3	Preparing your systems to run detached.....	6
3.1	Preparing the Application Server to deploy the Generated Code	6
3.1.1	RedHat JBoss.....	7
3.1.2	WildFly.....	8
3.2	Preparing the database to host the data of your applications.....	10
3.2.1	Using the Database that previously kept the OutSystems data	10
3.2.2	Using a brand new Database.....	10
4	Detaching an application from OutSystems	11
4.1	How to get the source code of an application	11
4.2	Configuring the Eclipse Workspace.....	13
4.2.1	Creating your base workspace	13
4.2.2	Importing the application generated code into Eclipse	13
4.3	Compiling the application.....	14
4.4	Deploying the application.....	15
5	Additional steps	15
5.1	Generating Javadoc Documentation.....	15
6	Final steps	16

1 Detach process overview

The detach process applies when you wish to completely remove OutSystems. Detaching some applications from OutSystems while other applications still use OutSystems is not a valid scenario.

To perform a complete detach, you will need to:

1. Guarantee that, before you start, you have all the requirements and followed all the recommendations from Chapter 2;
2. Prepare the Application Server and Database Server according with Chapter 3;
3. Detach, compile and deploy all individual modules as detailed in Chapter 4;
4. Test the result application(s) without OutSystems as presented in Chapter 5;

Make sure you read the full content of this document before you start in order to get a better overview of all the steps required and to better plan the process.

If at any step you run into a problem, you may need to repeat the previous steps until you get things working properly. Be careful and do not skip any instructions in the document to minimize the chance of making mistakes. In the case of not being able to proceed at any of those steps, you are free to contact OutSystems Support and we will happily help you.

2 Before you start

Do you want to evaluate the detach source code process?

For the purpose of evaluating the detach source code process, OutSystems provides a Service Studio embedded tutorial to guide you through the process. In Service Studio, in the OutSystems tab, you can find the 'No Lock-in' how-to.

The steps described in this document are not for evaluation purposes and are to be used only if you decide to stop using OutSystems.

Please refer to the [OutSystems Platform – Standard Architecture with No Lock-in](#) technical note for more information.

2.1 What will you lose?

If you choose to stop using OutSystems, you will no longer have access to all the application development, management, and operation capabilities of the platform. Also, you'll no longer benefit from the embedded change technology (ECT), performance monitoring and logging capabilities, which means you'll have to implement such functionality by resorting to other tools. You will however retain all core functionality of the applications you developed using Service Studio and Integration Studio, although any changes made to the detached source are not supported by OutSystems anymore.

After detaching the source code of your applications, all of the OutSystems tools and services will become unavailable (including Service Studio, Integration Studio, Service Center, and Lifetime), along with all of their features for visual development and composition, services repository, integration adapters automatic generation, database and application hot deployment, configuration management and versioning, packaging and staging, automatic code containment and optimization, factory access control management, performance monitoring and analytics.

Also notice that this is a one-way street. Once you detach the source code and start developing your applications using Visual Studio or Eclipse, there is no reverse engineering process to get those applications back into OutSystems.

2.2 Recommendations before you start the detach process

In order to minimize the risks of getting errors during and after the detach process, OutSystems recommends:

- Ensure that OutSystems applications are running with the expected behavior. Since OutSystems capabilities are lost, it will be more difficult to fix some problems after the detach
- Having the latest patch of your OutSystems version, in order to have all known bugs fixed of that specific version; This translates to having the latest Release of your **Major.minor.Release.revision** version installed correctly.
- Apply the detach over the same environment; Detaching Development to a Production Environment is neither a valid process nor the purpose of this document. If you want to do this, first you need to migrate the data from one environment to another and only then you can proceed with the detach.

2.2.1 Personnel requirements

Since you will lose most of the OutSystems capabilities, you must guarantee that your personnel:

- Fully understand the complete life cycle of Java applications; This includes understanding how to create, modify and deploy those application, as well as creating and managing the references between them and has a good understanding of the application server that is going to be used (JBoss, WildFly, WebLogic).
- Have access to the Database Server and have the ability to modify the existing data; During the detach process, it is necessary to change some of the OutSystems metadata. This is typically done by a DataBase Administrator (dba).

2.3 OutSystems Scheduler Service

If you are using Timers or BPT Activities in your applications, you will be able to keep this functionality even after detaching. Unlike other OutSystems services, the source code of the Scheduler Service will be provided for this purpose, just contact OutSystems by opening a Support Case and request it.

To compile the scheduler, run `ant` in the root directory of the scheduler project that was provided to you. Then, go to the `Scheduler` directory and do the following:

- Edit the `systemvars` file and check if the `JAVA_HOME` and `ANT_HOME` variables are correctly defined. If not, edit them to point to the directories containing Java and Ant, respectively.
- Execute `os.scheduler.service.sh` to start the scheduler

You can check the logs made by the scheduler in the `/Scheduler/logs` folder of the scheduler project, and in the `Scheduler_detached.log` file located in `/var/log/outsystems/`.

If you choose to re-implement these services using external tools you can later remove this dependency manually.

2.4 Understand how OutSystems structures the code

When you deploy an application module using 1-Click Publish operation, a standard JEE application is generated by the OutSystems Compiler with the output code structured as described below.

DatabaseAbstractionLayer – the Interfaces and base code responsible for allowing the Platform access and manage the data from the database.

DatabaseProviders –

iDB2DatabaseProvider – the iDB2 implementation of the DatabaseAbstractionLayer used by the platform when connected to an iDB2 database

MySQLDatabaseProvider – the MySQL implementation of the DatabaseAbstractionLayer used by the platform when connected to a MySQL database

OracleDatabaseProvider – the Oracle implementation of the DatabaseAbstractionLayer used by the platform when connected to an Oracle database.

SQLServerDatabaseProvider – the SQLServer implementation of the DatabaseAbstractionLayer used when connected to a SQLServer database.

Plugin.SAP –

Runtime - runtime code responsible for consuming SAP services

RuntimeAPI – API exposed to extend the SAP functionalities.

jcowrapper – wrapper for the SAP Java Connector (SAP JCo) libraries.

Plugin.Widgets.Runtime – the runtime code for the various custom web controls using the WidgetsRuntimeAPI

< Project Name > – the module's source code.

REST.RuntimeAPI – used to consume REST services and to extend REST behaviors using extensibility.

RETSERVICE.Runtime – the runtime code used to expose REST services.

RuntimeCommon – the basic runtime code used by all runtime libraries and modules.

RuntimePlatform – the Platform's runtime code used by the module.

RuntimeServices – code to connect to databases and OutSystems services.

RuntimeUtil – utilities used by the runtime code.

SMSInterface – the runtime code to deal with SMS.

WebWidgets – the runtime code for the various custom web controls.

WidgetsRuntimeAPI – framework to allow building custom web controls.

Inside the directory with the module's source code, the main folders are:

- **src** – the underlying core source code of your application
- **jsp** – the source code for Web Screens and Web Blocks of your application and all of the supporting Javascript files, images and style sheets.
- **proxysrc** – the proxy code for referencing other applications deployed with the platform



Figure 1: Code Structure of a generated Application.

In the src folder there is a set of packages that hold the different types of code. These are:

- **actions** - the code generated for both built-in actions and functions, and user actions and functions;
- **managedbean** – the code generated for web screens, grouped by web flows;
- **webservices** – the code generated for web services exposed by your application;
- **webreferences** – the proxy code to support the execution of external web services; **entities** – the code to manage entities;
- **structures** – the code to manage structures;
- **records** – the code to manage records defined in the development environment;
- **recordList** – the code to manage record lists;
- **exceptions** – the declaration of all user exceptions;
- **processes** – the code to manage BPT processes and process activities;
- **timers** – the code to manage timers;
- **themes** – the definitions of the web screen themes.

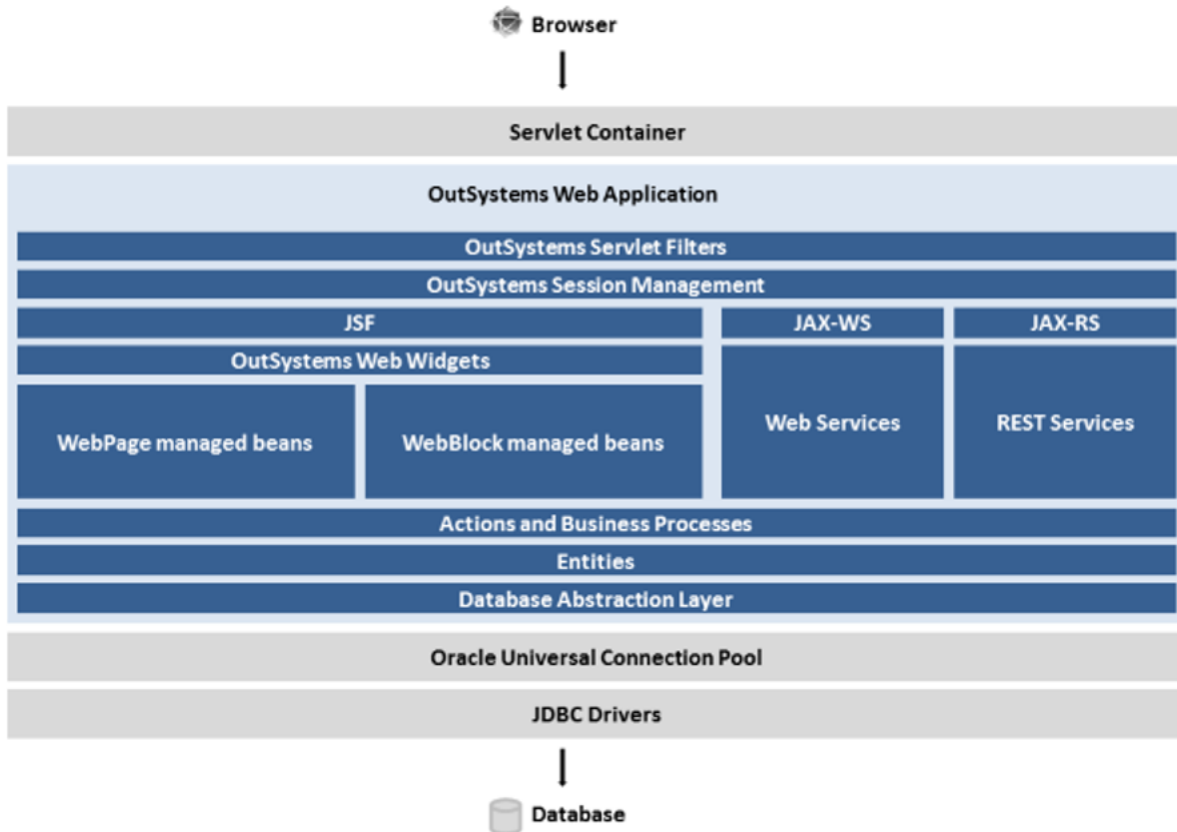
Other important files that can be found:

- **copyToApp** – external resources of the application, such as: images, Excel files;
- **WEB-INF** – the definition of both exposed and consumed web services (WSDL files), JavaServer Faces configuration file, web.xml deployment descriptor file, and additional Application Server deployment configuration files;

- **build.xml** – the ant build file used to generate the .war file that is deployed to the Application Server. This file also contains useful targets to build your application and to generate Javadoc documentation.

2.5 Architecture of the Detached Application

The architecture of an application developed with OutSystems is shown below:



Contrary to other technologies, OutSystems doesn't use a proprietary runtime engine. all applications rely only on standard Java technology:

- [JSF 1.2](#) – for the presentation layer;
- [JAX-WS](#) – for web services;
- [JAX-RS](#) – for REST services;
- [JMS](#) – queues for high-performance logging;
- [Oracle Universal Connection Pool](#) – to manage the database access;
- [JSR 88](#) – for the application deployment, using Application Server extensions for improved performance.

3 Preparing your systems to run detached

3.1 Preparing the Application Server to deploy the Generated Code

The machine that will run the detached applications must be prepared to that purpose. Making sure that all the requirements are met minimizes the chance of failure of this process.

Some web services of OutSystems communicate inside the machine through the http protocol. This requires the machine to access itself using http protocol, on port 80, in its local IP address, 127.0.0.1, most commonly known as localhost.

You must guarantee that this is correctly configured, even if you only want to use https to access your applications.

It's very important to make sure that NAT rules are defined to redirect from ports 80 and 443 to ports 8080 and 8443 respectively. To check your currently defined NAT rules you can open a command line terminal and execute the following command:

```
iptables -t nat -L
```

In case the rules are not defined they can be defined and saved running the following commands:

```
iptables -t nat -A OUTPUT -o lo -p tcp --dport 80 -j REDIRECT --to-port 8080
```

```
iptables -t nat -A OUTPUT -o lo -p tcp --dport 443 -j REDIRECT --to-port 8443
```

```
service iptables save
```

OutSystems for Java supports the following application servers:

- Red Hat JBoss
- WildFly

We will provide instructions for both application servers' configurations, some are common for both and others are application server-specific.

3.1.1 RedHat JBoss

If you are configuring the Application Server in the same machine that previously hosted OutSystems check Section 3.1.1.1. Otherwise, if a brand new Application Server is going to be used, configure it according with Section 3.1.1.2.

3.1.1.1 Using the Application Server that previously hosted OutSystems

If you are using the same Application Server that has OutSystems correctly installed, your machine already has most of the configurations needed to run the detached code.

However, it's very important to avoid problems that you keep the OutSystems folder in place, since many JBoss configurations files are actually links to files in the OutSystems folder. If you need to completely uninstall the platform from your server, it is suggested to follow the instructions for detaching into a new machine (3.1.1.2) and then return to this section.

You should stop running all OutSystems exclusive services and also stopping them from automatically starting on system boot. To get a list of the OutSystems services you can use the command:

```
service -status-all | grep outsystems
```

jboss-outsystems and jboss-outsystems-mq should **NOT** be stopped or removed from boot!

3.1.1.2 Using a brand new Application Server

You have to run JBoss with the *outsystems* configuration. To set this up, it is advisable that you use the [Checklist for OutSystems Platform Server Installation](#) as a walkthrough to install and configure your system with all the necessary tools, skipping the OutSystems installation itself. Then, copy the following files from the OutSystems server into the same folders in the target machine. Make sure you keep intact the ownership and execution attributes of the copied files.

\$JBOSS_HOME/standalone/configuration

- cacerts.truststore
- server.keystore
- standalone-outsystems-mq.xml
- standalone-outsystems.xml

\$JBOSS_HOME/bin

- standalone-outsystems.conf
- standalone-outsystems-mq.conf

- standalone-outsistemas-mq.properties
- standalone-outsistemas.properties

\$JBOSS_HOME/modules

- outsistemas

/etc/init.d

- jboss-outsistemas
- jboss-outsistemas-mq

After copying the files, execute the following commands:

- `chkconfig --add jboss-outsistemas`

This will install the outsistemas service. You can start and stop it using the following command: `service jboss-outsistemas [stop|start|restart]`

- `chkconfig --add jboss-outsistemas-mq`

This will install the message queue service. You can start and stop it using the following command: `service jboss-outsistemas-mq [stop|start|restart]`

Deploy `customHandlers.war` application, located at `/opt/outsistemas/platform` (for more information on how to deploy applications check section 4.2).

3.1.2 WildFly

If you are configuring the Application Server in the same machine that previously hosted OutSystems check Section 3.1.2.1. Otherwise, if a brand new Application Server is going to be used, configure it according with Section 3.1.2.2.

3.1.2.1 Using the Application Server that previously hosted OutSystems

If you are using the same Application Server that has OutSystems correctly installed, your machine already has most of the configurations needed to run the detached code.

To avoid problems, it is important that the OutSystems folder is not modified nor deleted, since many WildFly configurations files are actually links to files in the OutSystems folder. If you need to completely uninstall the platform from your server, it is suggested to follow the instructions for detaching into a new machine (3.1.2.2) and then return to this section.

You should stop running all OutSystems exclusive services and also stopping them from automatically starting on system boot. To get a list of the OutSystems services you can use the command:

```
service --status-all | grep outsistemas
```

wildfly-outsistemas and wildfly-outsistemas-mq should **NOT** be stopped or removed from boot!

3.1.2.2 Using a brand new Application Server

You have to run WildFly with the `outsistemas` configuration. To set this up, it is advisable that you use the [Checklist for OutSystems Platform Server Installation](#) as a walkthrough to install and configure your system with all the necessary tools, skipping the OutSystems installation itself.

When installing `jce_policy-8`, skip the instructions in the installation checklist. Instead, copy the contents of `jce_policy-8.zip` to `JAVA_HOME/lib/security` and replace the files.

After installing WildFly, you need to add a new user and user group. To do that, execute the following commands:

```
addgroup wildfly
```

```
adduser -g wildfly wildfly
```

Then, copy the following files and folders from the OutSystems server into the same folders in the target machine. Make sure you keep intact the ownership and execution attributes of the copied files and folders.

\$WILDFLY_HOME/standalone

- configuration-mq

\$WILDFLY_HOME/standalone/configuration

- cacerts.truststore
- server.keystore
- standalone-outsistemas-mq.xml
- standalone-outsistemas.xml

\$WILDFLY_HOME/bin

- standalone-outsistemas.conf
- standalone-outsistemas-mq.conf
- standalone-outsistemas-mq.properties
- standalone-outsistemas.properties

\$WILDFLY_HOME/modules

- outsistemas

/etc/init.d

- wildfly-outsistemas
- wildfly-outsistemas-mq

Give read and write permissions to the user *wildfly* and user group *wildfly* for the files and directories (for directories also give the permissions to all subdirectories and containing files):

- \$WILDFLY_HOME/standalone
- \$WILDFLY_HOME/bin/standalone-outsistemas.properties
- \$WILDFLY_HOME/bin/standalone-outsistemas-mq.properties

\$WILDFLY_HOME/modules/outsistemas contains links to files that were inside \$OUTSYSTEMS_HOME/platform, which are now broken links. To fix this you can either:

- Copy the following files and folders from the server that has OutSystems installed and place it with the exact same path in the new server:
 - \$OUTSYSTEMS_HOME/platform/lib
 - \$OUTSYSTEMS_HOME/platform/outsistemas.runtimeservices.jar
 - \$OUTSYSTEMS_HOME/platform/connectors/wildfly/module.xml
 - \$OUTSYSTEMS_HOME/platform/private.key
- Copy the same files to a new location and change all links manually
- Replace each link for the corresponding real file

Open the `standalone-outsistemas.xml` with a text editor and edit the `deployments` group to only have the applications you will deploy.

Use a text editor to make the same changes on two files: `/etc/init.d/wildfly-outsistemas` and `/etc/init.d/wildfly-outsistemas-mq`:

- Comment out the line with the include to `common.sh` (add a '#' at the beginning of the line)
- Set the variable `WILDFLY_HOME` with the path to the WildFly installation, right above "export `WILDFLY_HOME`". You can hardcode the path or execute a command to get it.
- Open the `common.sh` file, located in the Platform Server installation directory, locate the `runCommandAsUser` function, and copy it the `/etc/init.d/wildfly-outsistemas` and `/etc/init.d/wildfly-outsistemas-mq` files.

In the `$WILDFLY_HOME/bin/standalone-outsistemas.conf` file, comment out the line that includes the `common.sh` file and the three subsequent variables (`DATE`, `GCLOG`, and `JAVA_OPTS`)

Copy the contents of `/etc/.java/.systemPref/outsystems/prefs.xml` from the server with OutSystems installed to the same file in the new server.

Configure the needed services to be started on system boot by executing the following commands:

- `chkconfig --add wildfly-outsistemas`

This will install the outsystems service. You can start and stop it using the following command: `service wildfly-outsistemas [stop|start|restart]`

- `chkconfig --add wildfly-outsistemas-mq`

This will install the message queue service. You can start and stop it using the following command: `service wildfly-outsistemas-mq [stop|start|restart]`

To start wildfly server with the outsystems configurations run the command

```
./$WILDFLY_HOME/bin/standalone.sh --server-config=standalone-outsistemas.xml
```

Note: You might want to make sure that this command is always executed at system boot.

Deploy customHandlers.war application, located at `/opt/outsystems/platform` (for more information on how to deploy applications check sections 4.2, 4.3 and 4.4).

3.2 Preparing the database to host the data of your applications

Your application data will be preserved either in the same databases/schemas and database server, or by moving the databases/schemas from the current database server to a new one. In case you are keeping the same database follow the procedure of section 3.2.1, on the contrary, if you are moving the database server jump to section 3.2.2.

3.2.1 Using the Database that previously kept the OutSystems data

OutSystems encrypts sensitive configuration data such as passwords. To secure that data, an encryption key was generated when you installed OutSystems.

That key is stored in the `private.key` file under the Platform installation folder (`/opt/outsystems/platform/private.key`). To allow your application to decrypt sensitive data you need to ensure that it uses that key.

By default, your application will use the `private.key` file located in the Platform installation folder, so if you are detaching to a new server or you are completely uninstalling OutSystems from your current server, that file can be lost. For that reason, create a copy of the `private.key` file in a path where any user can write and change the configuration file accordingly:

- 1) Locate the `prefs.xml` file for JEE, following the path `/etc/.java/.systemPref/outsystems/prefs.xml` and open it with a text editor.
- 2) Find the entry with `key="OutSystems.HubEdition.SettingsKeyPath (DEFAULT)"` and set the value to be the new path to `private.key` file.

Note: While trying to access deployed applications after the detach is complete, it's possible to get error regarding database connections (not being able to access the datasource). If that happens, it might be needed to change the connection string from the encrypted value to the clear text value. You can find instructions on how to do that in the next section (3.2.2).

3.2.2 Using a brand new Database

If you will use a new database, you have to manually reconfigure your database connection strings.

To do this:

1. Locate the `prefs.xml` file for JEE, following the path `/etc/.java/.systemPref/outsystems/prefs.xml` and open it with a text editor.

2. Change the connection strings in that file. There are 7 connection strings that need to be changed:
 - `OutSystems.DB.Application.Log.ConnectionString`
 - `OutSystems.DB.Application.Runtime.ConnectionString`
 - `OutSystems.DB.Application.Session.ConnectionString`
 - `OutSystems.DB.Services.Admin.ConnectionString`
 - `OutSystems.DB.Services.Log.ConnectionString`
 - `OutSystems.DB.Services.Runtime.ConnectionString`
 - `OutSystems.DB.Services.Session.ConnectionString`
3. You will not be able to use any of these connection strings while they are encrypted, so you need to convert them to their plain text format:
 - MySQL:
`jdbc:mysql://{SERVER}/{SCHEMA}?user={USER}&password={PASSWORD}&allowMultiQueries=true&tinyInt1isBit=false&noAccessToProcedureBodies=true`
 - Oracle:
`jdbc:oracle:thin:{USER}/{PASSWORD}@://{SERVER}:{PORT}/{SERVICENAME}`

For each of those connection strings, delete the contents of the value attribute, copy the plain text format from the line above and replace the placeholders {USER}, {PASSWORD}, {SERVER}, {PORT} and {SERVICENAME} parts with the correct values for your installation.

After changing the connection strings, you will need to restart your application server.

4 Detaching an application from OutSystems

This chapter covers how to detach, compile and deploy a single module.

In order to have a system that replicates your previous factory, you must deploy all the applications published by OutSystems. This way you guarantee that all dependencies are met, further reducing the chances of failure.

Since there are some specific OutSystems applications for application and environment configuration and management, like Service Center and Lifetime, those cannot be detached.

If your OutSystems is running on the OutSystems PaaS, in order to detach the source code of your applications, you need to build first a hybrid deployment, where you install one OutSystems environment either on-premises or another public cloud.

You must follow sections 4.1 to 4.4 for a successful detach of all modules. This includes OutSystems modules that your applications most likely depend on, like RichWidgets and Users applications.

4.1 How to get the source code of an application

To obtain the source code of an application module, open the Licensing page under the Administration tab in Service Center and detach the source code as follows:

1. Click on 'eSpaces Source Code'.

outsystems[®] platform Factory Monitoring **Administration** Analytics

Production Users · Roles · Environment Configuration · Front-end Servers · Zones · Database Schemas · Database Connections · Email · Phones · Certificates · SEO URLs · Licensing

Licensing

[Upload New License](#) | [Backup License](#) | [Request a New License](#) | [Audit License](#) | [Registered End-Users](#) | [End-User Sessions](#) | [Software Units](#) | [eSpaces Source Code](#) | [Contact OutSystems Sales](#)

☐ This screen shows this Environment's Licensing information and current status. To apply a license file .lic sent to you by ...

✔ **Your license is valid for this server and all features are within licensed limits.**

Organization Name:	OutSystems SA	Activation Code:	RND . RND . RND . RND . RND . RND . RND . RND
Edition Name:	Unlimited Edition	Serial Number:	2RG-DCQ-A83-CXC-7XD-W7H-DDC-WUT
License Type:	Production	End Date:	2015-12-02

OutSystems Platform Features

☐ To narrow the Features list, type Name or part of Name and press Filter. Reset will list back all the features licensed in this ...

Name:

Feature	Limit	Usage	Next
✔ 64-bit Support Allows you to install the Platform Server in 64-bit Operating Systems.	Unlimited	-	
✔ Access Control Allows you to manage the access to your Factory Assets by enabling the configuration of differentiated privileges for Users, and the use of Roles.	Unlimited	-	

- Click on 'Detach' for the module whose source code you want to obtain.

outsystems[®] platform Factory Monitoring **Administration** Analytics

Production Users · Roles · Environment Configuration · Front-end Servers · Zones · Database Schemas · Database Connections · Email · Phones · Certificates · SEO URLs · Licensing

eSpaces Source Code

[Licensing](#)

In this page you can detach the source code of your eSpaces. This process enables you to build and deploy your eSpaces in another environment without resorting to the OutSystems Platform tools and services, ensuring No Lock-in.
To learn more about the detachment process and the No Lock-In capabilities of the OutSystems Platform, run the [No Lock-In tutorial](#) in Service Studio.

Name:

eSpace	Detach
ChartingServicesCore For Platform Version 8.9.5.26. Charting Services Core - based on FusionCharts v3. Please check the 'Chart' webblock for more information and documentation.	<input type="button" value="Detach"/>
Charts For Platform Version 8.9.5.26. Component with widgets for plotting charts.	<input type="button" value="Detach"/>

- Wait until OutSystems has finished packing the module source code and click on 'Download'.

outsystems[®] platform Factory Monitoring **Administration** Analytics

Production Users · Roles · Environment Configuration · Front-end Servers · Zones · Database Schemas · Database Connections · Email · Phones · Certificates · SEO URLs · Licensing

Source Code of ChartingServicesCore eSpace

[eSpaces Source Code](#)

☐ This screen presents sequentially the results of the operations *Compile* and *Pack* of the eSpace source. In the end, ...

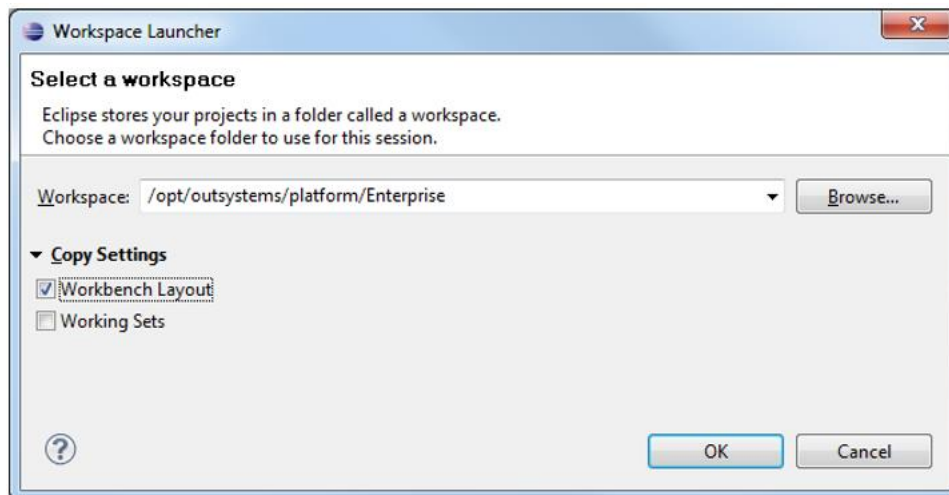
1	2
① Compiling	Compiling eSpace.
② Packing	Packing eSpace source.
Done	The eSpace source was successfully packed. Please press the Download button to download the pack.

- Save the zip file and extract it.

4.2 Configuring the Eclipse Workspace

4.2.1 Creating your base workspace

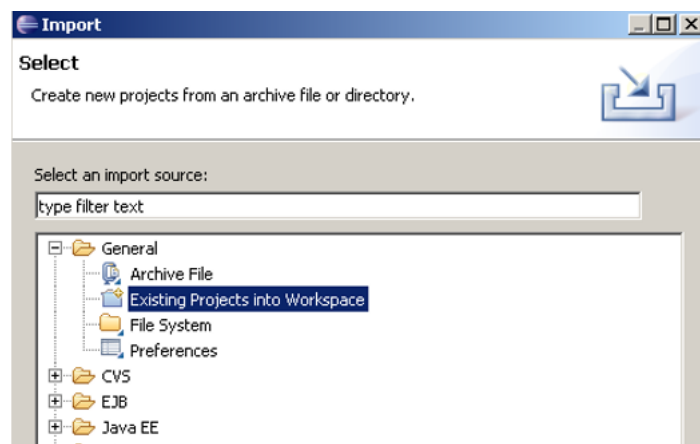
To have all the Java source code in the same Eclipse workspace, we recommend you to set up a new empty workspace: select the File->Switch Workspace->Other... option in Eclipse and choose an empty folder in a place of your choice (make sure you do not use the same folder where you saved the modules to be imported). Finally, check 'Workbench Layout'.



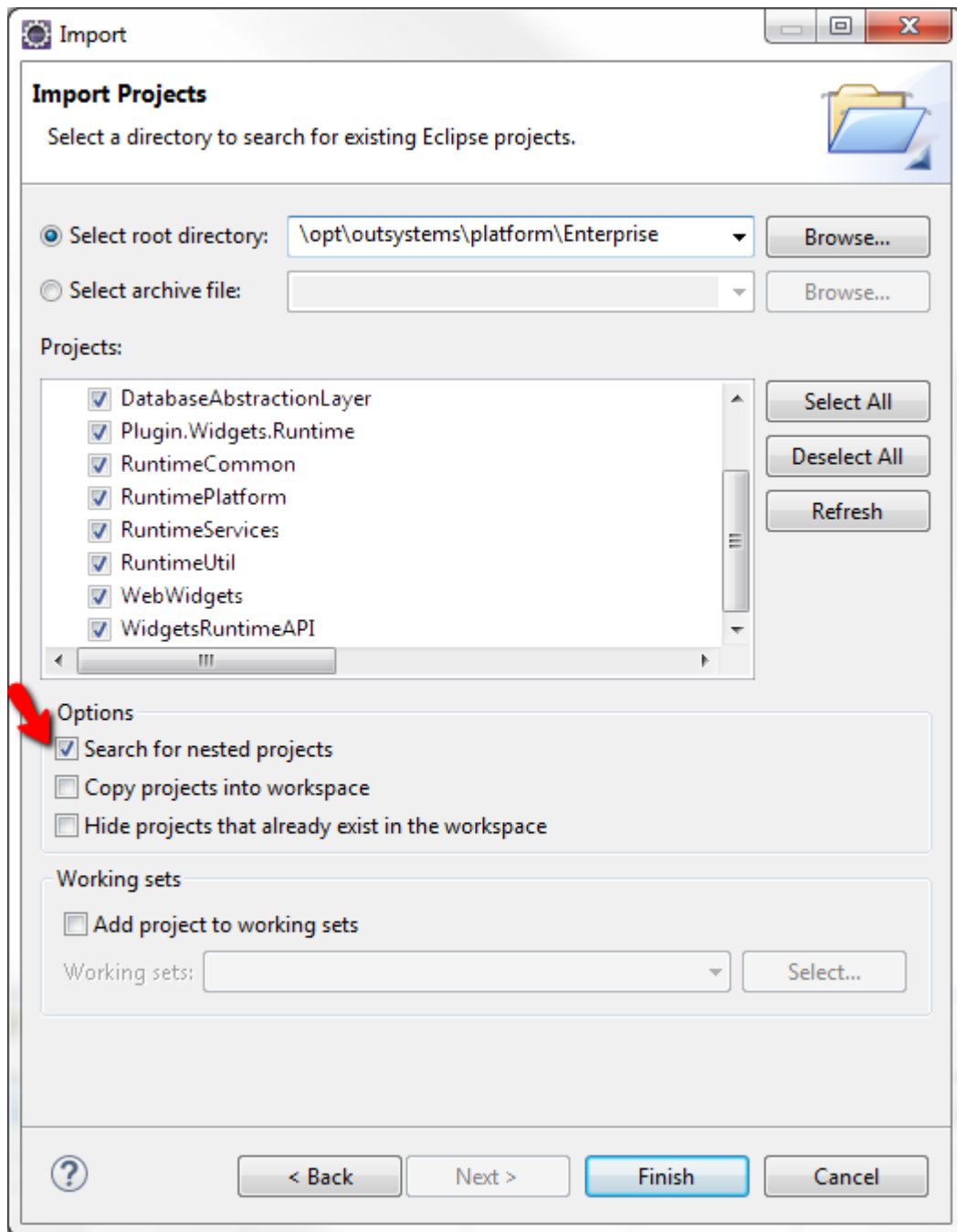
4.2.2 Importing the application generated code into Eclipse

To import the application code into your newly created workspace follow the steps below:

1. It's recommended that you unpack the generated folder to your workspace folder instead of choosing the "Copy projects to workspace" option in step 3.
2. In Eclipse's menu, select the File->Import... option and then the Existing Projects into Workspace import source inside the General folder.



3. Click on the Next button and on the Import Projects dialog select the folder to where you have transferred the generated code. Select all projects and do not forget to check the *Search into nested projects* option.



4. Click on the Finish button for the source code to be imported into your workspace. There should be no compilation errors in Eclipse. If you find any build path errors, check what is missing in the project properties.

4.3 Compiling the application

Each application contains a `build.xml` Ant file with a target for compiling it. This target will generate the application .war file at the root folder of your application. Later you will use that file to manually deploy into the application server. To execute this target run the ant command from your workspace directory:

```
ANT_HOME/ant
```

Extension modules don't generate source code. As such, source code for custom extensions is always available for detachment and reuse. Just use Integration Studio, as usual, to open and compile the source code using Eclipse.

If your application uses the OutSystems SAP plugin, then the SAP Java Connector (SAP JCo) libraries have to be manually added to the detached source code. To ensure that the application compiles successfully, you need to copy

the `libsapjco3.so` and the `sapjco3.jar` files, located in the `/opt/outsystems/platform/thirdparty/lib` folder, into the `Plugin.SAP/jcwrapper/lib` folder that came with the detached source code.

4.4 Deploying the application

The ant target does not automatically publish the application to the application server automatically, so you have to publish it manually. There are at least 3 different ways to do this (only tested with WildFly). For some of them, first you need to create a new user to access the Management Realm. You can accomplish that by running this command and following the instructions:

```
./$APPSERVER_HOME/bin/add-user.sh
```

The 3 ways to deploy an application are:

- Copy the `.war` file generated after compiling to `APPSERVER_HOME/standalone/deployments`. By default, the application server will periodically check for projects to deploy inside that folder.
- Access the application server GUI through a web browser with the URL: <http://localhost:9990/console>. When prompted to insert credentials, use the ones that you previously created for the Management Realm.
- Access the application server management console through a command line terminal. When prompted to insert credentials, use the ones that you previously created for the Management Realm. Then run the following commands:

```
./$APPSERVER_HOME/bin/jboss-cli.sh --connect
deploy {your_app_path}/{your_app_name}.war
quit
```

If an application depends on another applications, you might need to also deploy all its dependencies in order to guarantee that it keeps working correctly.

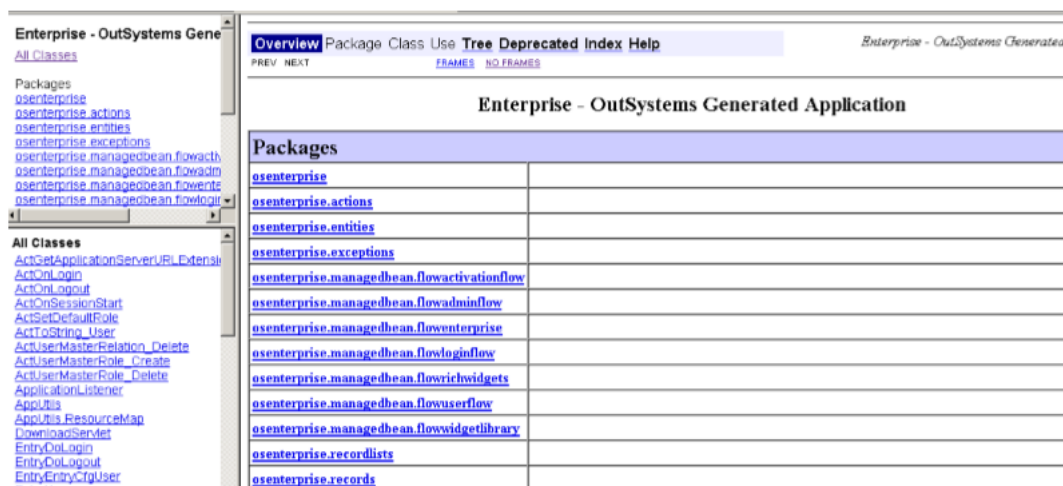
5 Additional steps

5.1 Generating Javadoc Documentation

To generate javadoc documentation all you need to do is to run ant with the `javadocs` target:

```
ant javadocs
```

This will create a `docs` subfolder with the javadoc documentation of your application.



The screenshot displays the Javadoc documentation for the 'Enterprise - OutSystems Generated Application'. The interface includes a sidebar on the left with 'All Classes' and a main area with 'Overview', 'Package', 'Class', 'Use', 'Tree', 'Deprecated', 'Index', and 'Help' tabs. The 'Overview' tab is selected, showing a table of packages.

Enterprise - OutSystems Generated Application	
Packages	
osenterprise	
osenterprise.actions	
osenterprise.entities	
osenterprise.exceptions	
osenterprise.managedbean.flowactivationflow	
osenterprise.managedbean.flowadminflow	
osenterprise.managedbean.flowenterprise	
osenterprise.managedbean.flowloginflow	
osenterprise.managedbean.flowrichwidgets	
osenterprise.managedbean.flowuserflow	
osenterprise.managedbean.flowwidgetlibrary	
osenterprise.recordlists	
osenterprise.records	

6 Final steps

When all applications have been successfully published, it is recommended that you test everything first before uninstalling OutSystems Server.

Test your application extensively to guarantee no errors are found. If any issue is found, repeat all the steps to confirm that you didn't miss any. If the error persists, contact OutSystems Support.

When you are sure everything is working correctly and no OutSystems files are present (other than the ones we asked you to keep), you have successfully performed the detach.