Configuration Location

- UI instance level: `Control Panel` → `Configuration` → `Instance settings` → `Authentication` → `Facebook`
- UI system level: `Control Panel` → `Configuration` → `System Settings` → `Foundation` → `Facebook`
- Configuration file: `{liferay-home}/osgi/configs/com.liferay.portal.security.sso.facebook.connect.configuration.FacebookConnectConfiguration.cfg`

The configuration contains following fields:

- `Enabled`: Check this box to enable Facebook Connect SSO authentication.
- `Verified account required`: Check this box to allow logins by Facebook users who have gone through the Facebook email verification process to prove that they can access the inbox associated with the email address they provided when registering for a Facebook account.
- `Application ID`: Can only be set at the platform instance level. Enter the ID of your registered Facebook application.
- `Application Secret`: Can only be set at the portal instance level. Enter the secret of your registered Facebook application.
- `Graph URL`: The base URL of the Facebook graph API. You will only need to change this if Facebook changes their graph API, otherwise the default URL will suffice.
- `OAuth Authorization URL`: Facebook's OAuth authorization URL. You will only need to change this if Facebook change their OAuth authorisation endpoint. This URL will be decorated with dynamic data and linked to from the Liferay login portlet.
- `OAuth Token URL`: Facebook's OAuth access token URL. Liferay will use this URL to exchange a request token for an access token.
- `OAuth Redirect URL`: This is the URL that the User will be directed back to once a OAuth request token has been generated. The URL points to a Liferay service which will exchange the request token for

the access token which is required in order for Liferay to make successful calls to the Facebook Graph API. You should only ever need to change this URL if requests to your Liferay instance need to go via a fronting webserver such as Apache that does URL rewriting.

6.7.6 OpenID Single Sign On Authentication

OpenID SSO authentication was introduced in Liferay. As a decentralized authentication protocol, it leaves the User to decide which iDP (Identity provider) the User wishes to use to log into the Liferay Portal.

Managing OpenID SSO authentication

Configuration can be applied at two levels:

- System (all portal instances)
- Portal instance

Configuration location:

- UI instance level : `Control Panel` → `Configuration` → `Instance settings` → `Authentication` → `OpenID`
- UI system level : `Control Panel` → `Configuration` → `System Settings` → `Foundation` → `OpenID`
- Configuration file: `{liferay-home}/osgi/configs/com.liferay.portal.security.sso.openid.configuration.OpenIdConfiguration.cfg`
- Only the configuration field is enabled.

6.7.7 OpenSSO Single Sign On Authentication

At present, if OpenSSO SSO authentication is enabled, this will override all other SSO implementations and take full control over the Liferay Portal login and logout functionality. This limitation has been logged as issue LPS-52940.

In order for OpenSSO to work, all applications that require SSO must be in the same web domain, because the OpenSSO relies on cookie sharing between applications.

Managing OpenSSO SSO authentication

Configuration can be applied at two levels:

- System (all portal instances)
- Portal instance

Configuration location:

- UI instance level : `Control Panel` → `Configuration` → `Instance settings` → `Authentication` → `OpenSSO`
- UI system level : `Control Panel` → `Configuration` → `System Settings` → `Foundation` → `OpenSSO`
- Configuration file: `{liferay-home}/osgi/configs/com.liferay.portal.security.sso.opensso.configuration. OpenSSOConfiguration.cfg`

The configuration contains the following fields:

- `enabled` if enabled
- `importFromLDAP`: If true then users authenticated from OpenSSO that do not exist in the portal are imported from LDAP. LDAP must be enabled.
- `loginURL`: The URL to the login page of the OpenSSO server
- `logoutURL`: The URL to the logout page of the OpenSSO server
- `serviceURL`: The URL where OpenSSO can be accessed to use the authentication web services
- `screenNameAttr`: The name of the attribute on the OpenSSO representing the user's screen name
- `emailAddressAttr`: The name of the attribute on the OpenSSO representing the user's email address
- `firstNameAttr`: The name of the attribute on the OpenSSO representing the user's first name
- `lastNameAttr`: The name of the attribute on the OpenSSO representing the user's last name

6.7.8 NTLM Single Sign On Authentication

At present, if NTLM SSO authentication is enabled, this will override all other SSO implementations and take full control over the Liferay Portal login and logout functionality. This limitation has been logged as issue LPS-52940.

Additionally, in order to use NTLM SSO, your portal instance authentication type must be set to screen name.

6.7.9 Managing NTLM SSO authentication

Configuration can be applied at two levels:

- System (all portal instances)
- Portal instance

Configuration location:

- UI instance level : `Control Panel` → `Configuration` → `Instance settings` → `Authentication` → `NTLM`
- UI system level : `Control Panel` → `Configuration` → `System Settings` → `Foundation` → `NTLM`
- Configuration file: `{liferay-home}/osgi/configs/com.liferay.portal.security.sso.ntlm.configuration. NtlmConfiguration.cfg`

The configuration contains following fields:

- `enabled` if enabled
- `domainController` IP address of your domain controller
- `domain` The domain / workgroup name
- `serviceAccount` You need to create a service account for NTLM. This account will be a computer account, not a User account
- `servicePassword` Enter the password for the service account
- `negotiateFlags` Only available at system level. Set according to the client's requested capabilities and the server's ServerCapabilities. See the following link: `http://msdn.microsoft.com/en-us/library/cc717152%28v=PROT.10%29.aspx`

6.7.10 Google authentication

Managing OpenSSO SSO authentication

Configuration can be applied at two levels:

- System (all portal instances)
- Portal instance

Configuration location:

- UI instance level : `Control Panel` → `Configuration` → `Instance settings` → `Authentication` → `Google`
- UI system level : `Control Panel` → `Configuration` → `System Settings` → `Foundation` → `Google`
- Configuration file: `{liferay-home}/osgi/configs/com.liferay.portal.security.sso.google.configuration.GoogleAuthorizationConfiguration.cfg`

The configuration contains following fields:

- `enabled` if enabled
- `clientId` The client ID provided by Google
- `clientSecret` The client secret provided by Google

6.8 Hardening Liferay

6.8.1 Principles

Hardening is usually the process of securing a system by reducing its vulnerability. Reducing available avenues of attack typically includes the removal of unnecessary software, usernames, or logins, and disabling or removing unnecessary services.

Related links:
`https://en.wikipedia.org/wiki/Hardening_computing`
`https://www.techopedia.com/definition/24833/hardening`

6.8.2 Layers of hardening

1. Network
2. Server / Operating System
3. Application

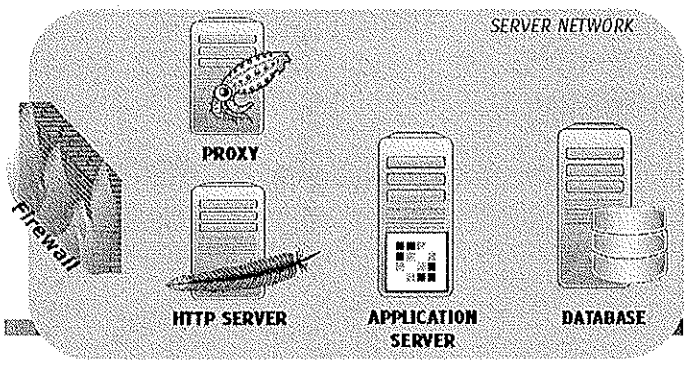
Network layer

Usually Liferay Portal's operating environment has been spread out to multiple servers.
Design your network to be safe:

- Protect network by firewall to control access to various servers
- Remove or disable unnecessary communication between servers
- Liferay should be at least behind an HTTP server to give an additional level of protection.
- Use a dedicated proxy server to connect Liferay to remote services at internet (pass-through).

(Figure 6.23, page 246)

Figure 6.23:



Server / OS layer

- Physical servers should stay in a locked room and have strictly limited user access.
- Server administration (Unix, Linux)
  - Administrators should use their own personal User account to administer.
  - No root level access; use SUDO only
  - Audit administrators
- Disable (remove) unwanted services.
- Block unnecessary ports with firewall.
- All applications and services should run with their respective operating system account.
- Always make sure host OS is up-to-date and install the latest security fixes.
- Analyze server to detect potential leaks (e.g. open ports, services, User accounts).

Application server layer

- Application Server should run with their respective operating system account (Do not run as root).
- Configure your web server to run daemon as a separate user.
  - ch root the application server installation.
- Use log rotation to prevent the disk from being full.
- Allow only http servers to connect to the application server.
- Configure protocol per need: HTTP, HTTPS, mixed mode.
- Hardening HTTP server (Apache): `https://www.liferay.com/web/olaf.kock/blog/-/blogs/securing-liferay-chapter -3-port-issues-and-http-https`
- Strengthen database server:
  - Limit access to servers.

  - Create dedicated account for Liferay operates only one schema (user).

6.8.3 Application - Liferay layer

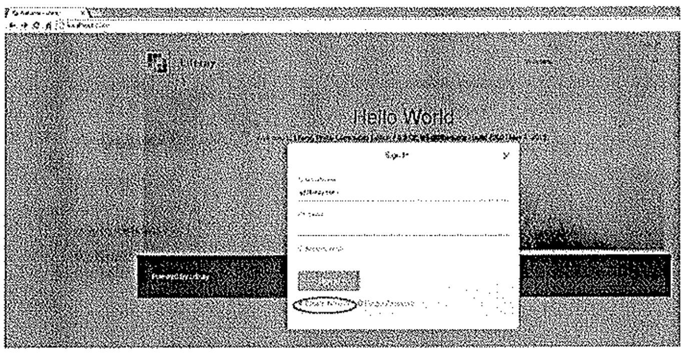
Account creation

- Disable account creation if registration is not required

```
#portal.properties
company.security.strangers=false open.id.auth.enabled=false
```

(Figure 6.24, page 248)

Figure 6.24:



- Verify necessary settings.
  - What are the default Roles / Site memberships / Groups / Organization for the new User?
  - Check at Control Panel → Instance settings → Configuration.

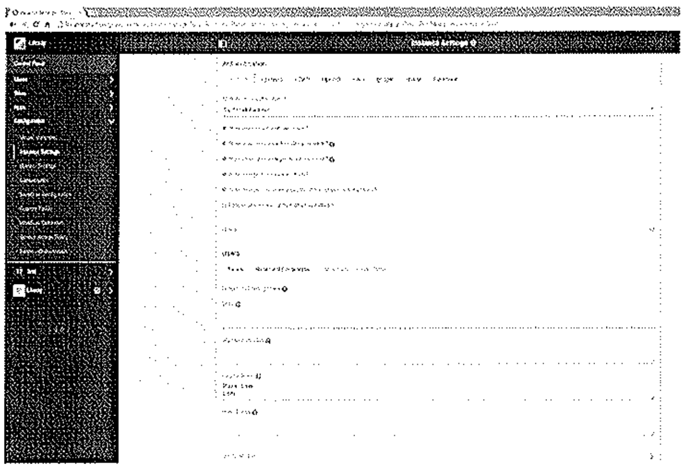
(Figure 6.25, page 249)

User Personal Pages

- **Disable** user's personal Site (public and private pages) if not needed.

```
#portal.properties

layout.user.public.layouts.enabled=false
```

Figure 6.25:



```
layout.user.public.layouts.power.user.required=false
layout.user.public.layouts.auto.create=false
layout.user.private.layouts.enabled=false
layout.user.private.layouts.auto.create=false
layout.user.private.layouts.power.user.required=false
```

Default administrator settings

- **Change** the default administrative account.

```
# portal.properites default values

default.admin.password=test default.admin.screen.name=test
default.admin.email.address.prefix=test
```

Balancing Security and Performance

- Liferay offers various encryption algorithms:

  ```
  #passwords.encryption.algorithm.legacy=BCRYPT
  #passwords.encryption.algorithm.legacy=MD5
  #passwords.encryption.algorithm.legacy=SHA
  ...
  ```

- **Change** encryption algorithms per need. Stronger encryption will provide better security but will cost performance (i.e. login times).

Auto-login and servlet filters

- **Disable** non-used auto-login functionality and servlet filters.

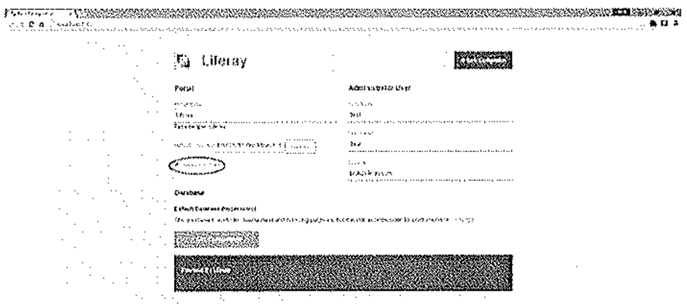
```
#portal.properties, changed default "true" value

com.liferay.portal.servlet.filters.autologin.AutoLoginFilter=false
com.liferay.portal.servlet.filters.sso.cas.CASFilter=false
com.liferay.portal.servlet.filters.sso.ntlm.NtlmFilter=false
com.liferay.portal.servlet.filters.sso.ntlm.NtlmPostFilter=false
com.liferay.portal.servlet.filters.sso.opensso.OpenSSOFilter=false
com.liferay.portal.sharepoint.SharepointFilter=false

#Remove any attached auto login hook
auto.login.hooks=
auto.login.ignore.hosts=
auto.login.ignore.paths=
```

Sample data

- Liferay DXP contains demo data for testing purposes.
    - Do not install test data for production.

(Figure 6.26, page 250)

Figure 6.26:



Unused portlets

Non-used OSGi portlets can be removed from Liferay:

- **Delete** correspondig OSGI bundle (JAR) from `LIFERAY_HOME/osgi/modules`.
- **Backup** before deleting.
- *Some OSGI modules (including porlets) are essential for minimal portal operation and cannot be deleted.*

Securing Liferay Services

Securing Liferay Services is done through 4 different layers:

- IP Permission Layer
  - *Check request originates* is white listed in Liferay server's portal properties file.
- Authorization/Verification Layer (Browser Only)
  - Verifies the authentication token to check if it returns a valid user
- User Permission Layer
  - Check to see if the User has the rights to perform the service
- Security Access Profile Layer (introduced in Liferay DXP)
  - Whitelist of services/functionality a User has access to

Web service request must pass all 4 layers of security.

Whitelisting Services

Whitelist services or specific methods:

```
com.liferay.portal.service.UserService
com.liferay.portlet.documentlibrary.service.DLAppService#get*
```

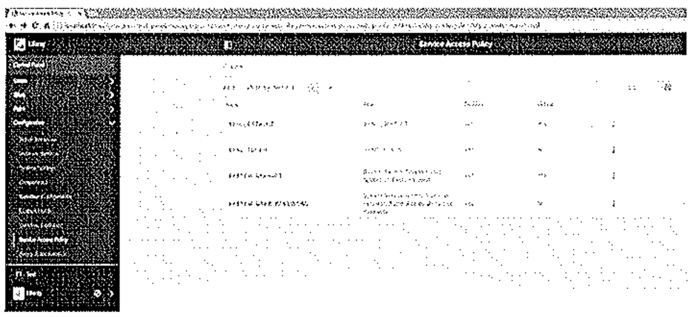
Service Access Policy

Liferay contains four service access policies that are enabled by default:

- SYNC_DEFAULT: Applies to every Liferay Sync request, including unauthenticated Sync requests.
- SYNC_TOKEN: Policy applies to Sync requests which are accompanied by an authentication token.
- SYSTEM_DEFAULT: Applies to every request, including unauthenticated requests.
- SYSTEM_USER_PASSWORD: Requests whenever User authentication took place using a password.

(Figure 6.27, page 252)

Figure 6.27:



`https://dev.liferay.com/discover/deployment/-/knowledge_base/7-0/service-access-policies`

Custom development directives

- OWASP 10
- Use built-in frameworks (e.g. Alloy UI) to avoid XSS.
- Use Liferay API to escape content.
  - `HtmlUtil.escape()`
  - AlloyUI elements are protected by its nature.
  - For Liferay tags, make sure that `escapeModel=true`
- Use Liferay's permission framework.

Installing the Latest Fixes

The Liferay Support Team regulary creates and releases fixes to handle various portal issues. These are shipped in zipped pacakges which are usually referred to as Fix Packs. Fix pack typically contain multiple fixes in one package. Customers can download fix packs from Liferay's website or receive them directly from support.

Fix packs are usually installed with the **Patching Tool** which is included in every Liferay bundle.

Benefits of patching tool

- Dedicated and continously updated tool for patching
- Auto-detection of working environment
- Easy to install or revert fix packs
- Show occasional patch (fix-pack) collisions
- Provides two levels of patching:
  - In case of binary patching, the deployed portal application is patched (default mode).
  - In case of source patching, it is possible to debug the modified source files.

Make sure to learn about the Patching Tool and install the latest Fix Packs when working on a project.

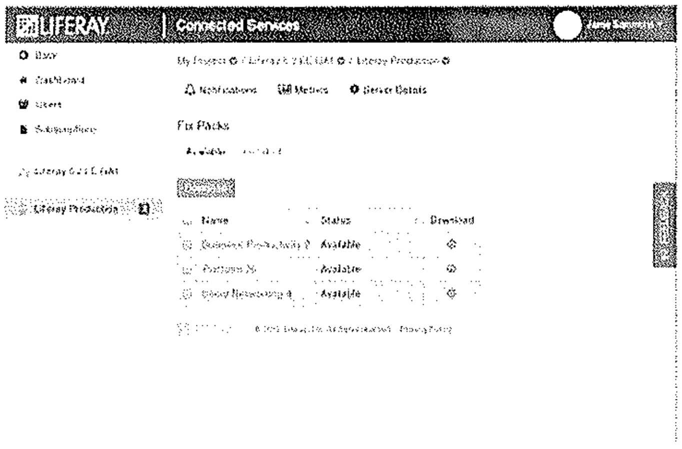
More on patching: `https://www.liferay.com/group/customer/products/portal/patching`

Liferay Connected Services

Liferay Connected Services is a new online platform that offers a set of tools and services such as:

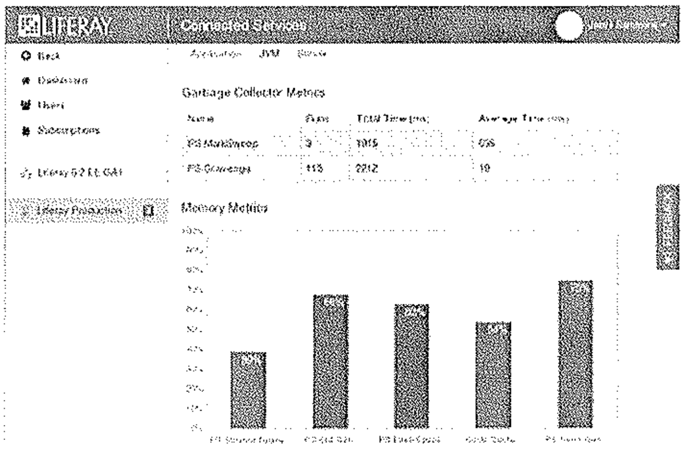
- Fix pack management
  - Easily manage and install fix packs across several servers and clusters. *(Figure 6.28, page 254)*
- Metrics:

Figure 6.28:



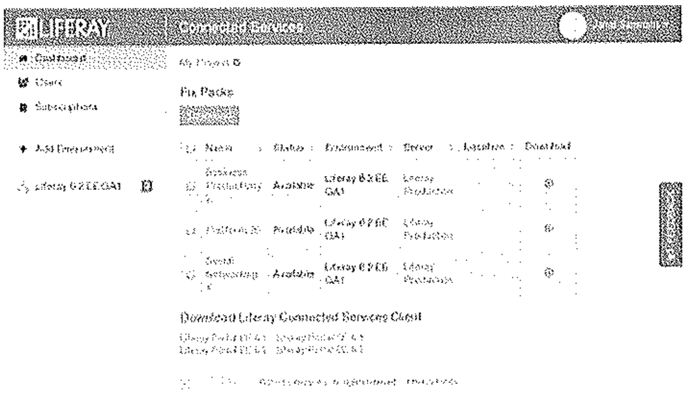
- Various metrics in simple and clean format: page visits, portlet load times, memory metrics *(Figure 6.29, page 254)*

Figure 6.29:



- Online Dashboard and Alerts
  - Access all the information about your different environments from a single point of access.
  - Dedicated alerts section where you will be notified when there are new fix packs available, servers are down, or there are configuration issues. *(Figure 6.30, page 255)*

Figure 6.30:



LIFERAY Connected Services

Dashboard
Users
Subscriptions
Add Environment
Liferay 6.2 EE GA1

My Project

Fix Packs

Name	Status	Environment	Server	Location	Download
Business Productivity	Available	Liferay 6.2 EE GA1	Liferay Production		
Platform [illegible]	Available	Liferay 6.2 EE GA1	Liferay Production		
Social Networking	Available	Liferay 6.2 EE GA1	Liferay Production		

Download Liferay Connected Services Client

[illegible]

Get more information here:
`https://www.liferay.com/products/liferay-connected-services`

6.9 Disaster Recovery

- Once your Site is running, you should set up proper backup procedures in case of catastrophe.
- Liferay is not much different from any other application running in your application server, and fits in well with most, if not all, sound backup procedures.
- There are three specific components of Liferay that need to be backed up on a regular basis:
  - Source code
  - Database
  - Default Data
  - Configuration Files
  - Apps & Modules
  - Search Indexes
  - File Repository

6.9.1 Source Code

Source Code: If you have extended Liferay or developed custom plugins, these should be stored in a source control repository:

- Git
- Subversion
- CVS
- Mercurial

This source code repository should be backed up on a regular basis.

6.9.2 Database

- Liferay's database stores most of the information that Liferay needs to run, as well as all of the User accounts and data for the portal.
- It is the most important component that needs to be backed up.
- If your database vendor supports it, you can back it up live. Otherwise, you can do a dump of the database to a text file and then back up the exported file.
- Backup can be done using the tools provided by your database vendor.
  - mysqldump (MySQL)
  - pg_dump utility (PostgreSQL)
  - Oracle data pump, exp/imp utility

6.9.3 Configuration files

- It's always important to back up all Liferay configuration files.
  - `portal-ext.properties` file(s) from Liferay Home and other folders

6.9.4 Default data

- Liferay stores configuration files, search indexes, and cache information in a folder called `/data` in Liferay Home by default.
- If you're using the *File System store* or the *Advanced File System store*, the Documents and Media repository is also stored here by default.

  It's always important to back up your `/data` folder.

6.9.5 Applications and modules

- The files that comprise Liferay's OSGi runtime are stored in a folder called `/osgi` in Liferay Home.
- This folder contains:
  - JAR files for all of the apps and modules that you've deployed to Liferay.
  - Other required JAR files, configuration files, and log files

  It's also important to back up your `/osgi` folder.

6.9.6 Search indexes and File Repository

- Search indexes must be backed up individually:
  - If embedded. Elasticsearch is configured for a different storage location than `/data`.
  - If dedicated. Elasticsearch server is used.
  - If replaced to Solr server
- File Repository must be saved regularly when using:
  - Shared file system (e.g SAN or NAS)
  - DBStore (*handled during database backup*)

6.9.7 Java EE Server or Web Container

- **If possible, back up your entire application server and Liferay Home folder.**

- Allows you to revert to a working instance quickly in the event of an issue

6.9.8 Test your backup system

- It's important to test your backup system.
- You should only be confident that a backup really is a backup if you've demonstrated that you're able to use it to restore a fully-functional system on a new computer.
- Too often backups are dutifully taken, only to discover too late that vital parts are missing.
- This discovery is typically made in disaster situations; you should make sure that such discoveries happen during testing.

6.10 Upgrading Liferay

6.10.1 Overview

- Any upgrade process can be split in two main parts:
  - Database upgrade
  - Custom developments upgrade
- This section will be focused on in upgrading the database.
- We'll also talk about differences between an upgrade to Liferay 7 and previous versions of LIferay.

6.10.2 Before performing the upgrade

- As with previous upgrades, we need to ensure a proper backup is in place in case of failure. The backup should include at least:
  - Database backup
  - Document Library backup
- You will also need to check *portal-legacy-6.2.properties* in case you need to maintain a certain previous behavior.

- We also recommend you disable search indexing during this process. To achieve that, you should add a file called *com.liferay.portal.search.configuration. IndexWriterHelperConfiguration.cfg* into your *osgi/configs/* folder with the following content:

```
indexReadOnly=true
```

- Once you have upgraded your portal, remove that property or set it to false and reindex all objects from the Control Panel.

6.10.3 Running the upgrade tool

- Starting up the server to upgrade the portal is no longer supported. The following exception is thrown if you try to do that:

```
   [MainServlet:237] java.lang.RuntimeException: You must first
upgrade to Liferay Portal 7000
```

- The new way to run the upgrade is using a standalone tool provided by Liferay:

```
java -jar com.liferay.upgrade.tool.jar
```

- The upgrade requires two files to be configured before it can run, *portal-ext.properties* and *server.properties*.
- You can also set the log file, where you can print the upgrade output and your own JVM options.

6.10.4 Running the upgrade tool

- The data upgrade is now broken up into two parts:
  - **Core upgrade**: One upgrade process for the core similar to what you have seen in the past.
  - **OSGI modules upgrade**: every module has its own upgrade process runnable separately. These processes are based on a new framework that we will explain later.
- The upgrade tool is configured to upgrade both automatically by default.

6.10.5 Upgrading modules individually

- You can configure the portal to only upgrade the core, and not the modules, by adding a file called *com.liferay.portal.upgrade.internal. configuration.ReleaseManagerConfiguration.cfg* in the *osgi/configs/* folder with the following content:

```
autoUpgrade=false
```

- Once you have upgraded the portal successfully you can run the upgrade for every module individually using the *Gogo Console*.

6.10.6 Gogo console to check module upgrades

- To run the upgrades for the modules or check them, you can use the Gogo shell. The upgrade tool will automatically connect to Gogo Shell after you complete the core upgrade. You can also connect to the shell by executing:

```
telnet localhost 11311
```

- Once you are connected, you can type the following available commands in the upgrade namespace:
  - upgrade:list
  - upgrade:execute
  - verify:list
  - verify:execute

6.10.7 Gogo console: upgrade:list command

- By typing *upgrade:list*, the console will show you the modules you can upgrade since all their upgrade dependencies are covered.
- If you miss a certain module, it is because we need to upgrade its dependencies first. You could enter the command *scr:info {upgrade_qualified_class_name* to check which dependencies are unsatisfied. For example:

```
scr:info com.liferay.journal.upgrade.JournalServiceUpgrade
```

- You can also type *upgrade:list {module_name}* to know the status of the upgrade and upgrade steps for a certain module.

6.10.8 Gogo console: upgrade:list command

- To understand how this works, it can be useful to see an example; if you execute that command for the bookmarks service module, you will get this:

```
   g! upgrade:list com.liferay.bookmarks.service

Registered upgrade processes for com.liferay.bookmarks.service 0.0.1
{fromSchemaVersionString=0.0.0, toSchemaVersionString=1.0.0,
upgradeStep=com.liferay.portal.spring.extender.internal.context.
ModuleApplicationContextExtender$
ModuleApplicationContextExtension$1@71cd739f}
{fromSchemaVersionString=0.0.1, toSchemaVersionString=0.0.2-step-1,
upgradeStep=com.liferay.bookmarks.
upgrade.v1_0_0.UpgradeKernelPackage@67620617}
{fromSchemaVersionString=0.0.2, toSchemaVersionString=1.0.0-step-1,
upgradeStep=com.liferay.bookmarks.upgrade.
v1_0_0.UpgradeLastPublishDate@547e0507}
{fromSchemaVersionString=0.0.2-step-1, toSchemaVersionString=0.0.2,
 upgradeStep=com.liferay.bookmarks.upgrade.
 v1_0_0.UpgradePortletId@1d7c39e7}
{fromSchemaVersionString=1.0.0-step-1, toSchemaVersionString=1.0.0,
 upgradeStep=com.liferay.bookmarks.upgrade.
 v1_0_0.UpgradePortletSettings@46bd7bd0}
```

6.10.9 Gogo console: upgrade:list command

- Taking into account the previous example, we can conclude:
  - The current module status is 0.0.1 (shown at the end of the first line) and it has five upgrade steps.
  - There is an available upgrade process from 0.0.1 to 1.0.0 by executing the following classes divided into ordered steps:
    1. From 0.0.1 to 0.0.2-step-1: *com.liferay.bookmarks.upgrade. v1_0_0.UpgradeKernelPackage.*
    2. From 0.0.2-step-1 to 0.0.2: *com.liferay.bookmarks.upgrade. v1_0_0.UpgradePortletId.*
    3. From 0.0.2 to 1.0.0-step-1: *com.liferay.bookmarks.upgrade. v1_0_0.UpgradeLastPublishDate.*

4. From 1.0.0-step-1 to 1.0.0: *com.liferay.bookmarks.upgrade.v1_0_0.UpgradePortletSettings*.

   – There is also another step, from 0.0.0 to 1.0.0 but that one is used if you come from an empty database (not for upgraded ones).

6.10.10 Gogo console: upgrade:execute command

- By typing *upgrade:execute {module_name}*, you will upgrade a certain module.
- If there is an error during the process, you will be able to restart the process from the latest step executed successfully instead of executing the whole process again. This can be really useful to diagnose issues faster.
- You could check the current status of an upgrade executing *upgrade:list {module_name}*.

6.10.11 New upgrade framework for modules

- As we said before, Liferay now implements a new upgrade framework for OSGi modules. It is important to know and understand previous commands.
- Every module has its own upgrade process; each individual process is split into several steps.
- Every step has its own status. There a few reserved statuses:
  - **0.0.0**: Status for modules whose data has not been created yet. This is the status before installing Liferay for the first time.
  - **0.0.1**: Status for modules that are not upgraded. After upgrading the core, this is the status for modules included at the core in 6.2 (they have been converted into OSGi modules).
  - **1.0.0**: Status for modules upgraded properly.
- It is important to take into account that OSGi modules that were plugins in 6.2 follow different rules regarding statuses. Check `upgrade:list` for those modules to know the final expected status.

- You can take a look at how defined the steps are for an upgrade process in a module by checking the class `BookmarksServiceUpgrade`:

```
java
@Component(immediate = true, service =
UpgradeStepRegistrator.class)
public class BookmarksServiceUpgrade implements
UpgradeStepRegistrator {
@Override
public void register(Registry registry) {
    registry.register(
        "com.liferay.bookmarks.service", "0.0.1", "0.0.2",
      new UpgradeKernelPackage(), new UpgradePortletId());

    registry.register(
        "com.liferay.bookmarks.service", "0.0.2", "1.0.0",
         new UpgradeLastPublishDate(),
         new UpgradePortletSettings(_settingsFactory));
}
```

- For the whole example: `https://github.com/liferay/liferay-portal/blob/master/modules/apps/collaboration/bookmarks/bookmarks-service/src/main/java/com/liferay/bookmarks/upgrade/BookmarksServiceUpgrade.java`

6.10.12 Verifiers

- As with previous versions of Liferay, there are some processes called verifiers that check and, in some cases, modify, data.
- The core contains its own verify processes, but we can also find verifiers in some modules. To obtain the list of available verifiers, you can run `verify:list` at the Gogo console.
- You can run them at any time (it's no longer necessary to restart your server) by typing verify: execute `verify_qualified_name}`. You can also select the input for the log before executing it.
- Liferay also provides a new framework to create your own verifiers easily. You can add them into your OSGi module by defining them and declaring this:

Moving instances

- Liferay includes a new offline tool to allow you to move instances (*companies*) to independent databases.
- This tools generates SQL files to achieve this for all supported databases.

6.10.16 Upgrade from Liferay 6.1

- Altough you will be able to upgrade to Liferay 7 in one step, if you come from Liferay 6.1 you will also need to take into account the upgrade to 6.2 instructions: `https://dev.liferay.com/discover/deployment/-/knowledge_base/6-2/upgrading-liferay`