DSpace 10.0

Complete Installation Manual

Step-by-Step Guide for Ubuntu Linux 22.04 / 24.04

Prepared by Sukhdev Singh
June 2026

Based on Official DSpace 10.0 Documentation (wiki.lyrasis.org)

Section 1: Overview of DSpace 10.0

DSpace 10.0 is a major release of the DSpace platform, officially released in May/June 2026. It is the first release that begins the formal merger of DSpace and DSpace-CRIS into a single unified platform. The merger will complete in DSpace 11.0.

1.1 Architecture

DSpace 10 consists of two separate applications that must both be installed:

- Backend (Server API): A Spring Boot Java application. It provides the REST API, OAI-PMH, SWORD, and RDF interfaces. It has no user interface on its own.
- Frontend (User Interface): An Angular 20 application running on Node.js. It provides the browser-based interface for users. It cannot function without the backend.

NOTE	Install the Backend FIRST. The Frontend cannot function without a running Backend.

1.2 Key Changes from DSpace 9.x

- JDK 21 is now required (upgraded from JDK 17). JDK 17 is no longer supported.
- Frontend upgraded to Angular 20 (from Angular 17).
- DSpace-CRIS merger begins: new features ported including Edit in Submission mode, Shared Workspaces, Custom URLs, Audit Trails, Nested Metadata, and Metadata-level Security.
- New Dataset entity type for research data repositories.
- DOI handling unified: dc.identifier.doi for DSpace-generated DOIs; CrossRef DOI registration now supported.
- Six new Indian language interfaces: Gujarati, Malayalam, Marathi, Odia, Tamil, Telugu.
- Runnable JAR option for backend: Tomcat is now embedded — no separate Tomcat install required if using this approach.

Section 2: System Requirements

2.1 Backend Requirements

Component	Requirement	Notes
Operating System	Ubuntu 22.04 or 24.04 LTS (recommended)	Windows also supported but Linux is standard for production
Java (JDK)	JDK 21 (OpenJDK or Oracle JDK)	JDK 17 is NO LONGER supported. JDK 11-20 not supported.
Apache Maven	3.9.x or above	Required to build the DSpace backend from source
Apache Ant	1.10.x or later	Required to deploy the built installer
PostgreSQL	14.x, 15.x, 16.x or 17.x	pgcrypto extension NOT required for PostgreSQL 13+
Apache Solr	9.x (9.8+ recommended)	Solr 8.x is end-of-life. Port 8983 must be available.
Tomcat (optional)	10.1.x	Only needed for traditional deployment. Not needed for Runnable JAR.
RAM	Minimum 4 GB, 8 GB recommended	Solr and Tomcat both consume significant memory
Disk space	Minimum 20 GB free	More needed depending on repository size

2.2 Frontend Requirements

Component	Requirement	Notes
Node.js	v20.19+, v22.x or v24.x (LTS recommended)	Angular 20 requires Node v20.19.0 minimum
npm	Bundled with Node.js	Used to install frontend dependencies and build the UI
PM2	Latest stable (optional but recommended)	Node.js process manager for production deployments
RAM	2 GB minimum for building	Set NODE_OPTIONS=--max-old-space-size=4096 if build fails

NOTE	All commands in this manual assume you are logged in as a non-root user with sudo privileges on Ubuntu Linux. Replace placeholder values like [dspace], [dspace-source], [dspace-angular] with your actual paths.

Section 3: Pre-Installation — System Preparation

3.1 Update the System

Always start with a fully updated system.

```
sudo apt update && sudo apt upgrade -y
```

3.2 Install Java JDK 21

DSpace 10 requires JDK 21. Install Eclipse Adoptium Temurin 21 (OpenJDK 21 also works).

```
sudo apt install -y wget apt-transport-https gnupg
wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | sudo
gpg --dearmor -o /etc/apt/trusted.gpg.d/adoptium.gpg
echo "deb https://packages.adoptium.net/artifactory/deb $(awk -F=
'/^VERSION_CODENAME/{print$2}' /etc/os-release) main" | sudo tee
/etc/apt/sources.list.d/adoptium.list
sudo apt update
sudo apt install -y temurin-21-jdk
```

Verify installation:

```
java -version
# Expected output: openjdk version "21.x.x" ...
```

Set JAVA_HOME. Find your exact JDK folder first:

```
ls /usr/lib/jvm/
# Note the exact folder name shown, e.g. temurin-21-amd64
```

WARN	The folder name shown by 'ls /usr/lib/jvm/' is your actual JDK folder. Replace 'temurin-21-amd64' below with YOUR exact folder name if it differs.

```
echo 'export JAVA_HOME=/usr/lib/jvm/temurin-21-amd64' >> ~/.bashrc
echo 'export PATH=$JAVA_HOME/bin:$PATH' >> ~/.bashrc
source ~/.bashrc
echo $JAVA_HOME
# Should print: /usr/lib/jvm/temurin-21-amd64
```

3.3 Install Apache Maven

```
sudo apt install -y maven
mvn -version
# Expected: Apache Maven 3.9.x or above
```

NOTE	If your Ubuntu package manager provides Maven 3.6.x, download Maven 3.9.x manually from https://maven.apache.org/download.html and set M2_HOME accordingly.

3.4 Install Apache Ant

```
sudo apt install -y ant
ant -version
# Expected: Apache Ant version 1.10.x
```

3.5 Install PostgreSQL

Install PostgreSQL 16 (recommended). Ubuntu 22.04+ also supports 14, 15, or 17.

```
sudo apt install -y postgresql postgresql-contrib
sudo systemctl enable postgresql
sudo systemctl start postgresql
psql --version
```

Configure PostgreSQL to allow TCP/IP connections. Edit postgresql.conf:

```
sudo nano /etc/postgresql/16/main/postgresql.conf
# Uncomment or verify this line exists:
# listen_addresses = 'localhost'
```

Edit pg_hba.conf to add DSpace authentication:

```
sudo nano /etc/postgresql/16/main/pg_hba.conf
# Add this line BEFORE any lines matching 'all' databases:
# host    dspace    dspace    127.0.0.1/32    md5
```

```
sudo systemctl restart postgresql
```

3.6 Create DSpace PostgreSQL User and Database

```
sudo -u postgres createuser --no-superuser --pwprompt dspace
# Enter a password when prompted. Remember this password — you will need it.

sudo -u postgres createdb --owner=dspace --encoding=UNICODE dspace

# Verify connection:
psql -U dspace -W -h localhost -d dspace
# Enter your password. You should see the dspace=> prompt. Type \q to exit.
```

3.7 Install Apache Solr 9.x

Download and install Apache Solr 9.8 (or latest 9.x):

```
cd /opt
sudo wget https://archive.apache.org/dist/solr/solr/9.8.1/solr-9.8.1.tgz
sudo tar -xzf solr-9.8.1.tgz
sudo mv solr-9.8.1 /opt/solr
```

Create solr.in.sh with required DSpace settings for Solr 9.8+:

```
sudo nano /opt/solr/bin/solr.in.sh
# Add or verify these two lines at the end of the file:
# SOLR_MODULES=analysis-extras
# SOLR_OPTS="-Dsolr.config.lib.enabled=true"
```

WARN	For Solr 9.8 and later, BOTH lines — SOLR_MODULES=analysis-extras AND -Dsolr.config.lib.enabled=true — are required. Missing either will cause DSpace search to fail.

Create a dedicated solr system user and set ownership:

```
sudo useradd -r -s /bin/false solr
sudo chown -R solr:solr /opt/solr
```

Create a systemd service for Solr:

```
sudo nano /etc/systemd/system/solr.service
```

Paste the following content into the file:

```
[Unit]
Description=Apache Solr
After=network.target

[Service]
Type=forking
User=solr
Group=solr
Environment=SOLR_INCLUDE=/opt/solr/bin/solr.in.sh
ExecStart=/opt/solr/bin/solr start
ExecStop=/opt/solr/bin/solr stop
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

```
sudo systemctl daemon-reload
sudo systemctl enable solr
sudo systemctl start solr
# Verify Solr is running at http://localhost:8983/solr
curl -s http://localhost:8983/solr/admin/info/system | grep solr-spec-version
```

3.8 Install Apache Tomcat 10.1 (Traditional Deployment)

NOTE	Tomcat is OPTIONAL if you use the Runnable JAR approach (Section 5.4). If you prefer the simpler Runnable JAR method, skip this section and go directly to Section 4.

```
sudo apt install -y tomcat10
sudo systemctl enable tomcat10
```

Set Tomcat memory and UTF-8 encoding. Create setenv.sh:

```
sudo nano /usr/share/tomcat10/bin/setenv.sh
# Add this line:
# JAVA_OPTS="-Xmx2048M -Xms512M -Dfile.encoding=UTF-8"
```

Edit server.xml to enable URIEncoding:

```
sudo nano /etc/tomcat10/server.xml
# Find the <Connector port="8080" ...> element and add URIEncoding="UTF-8"
# It should look like:
# <Connector port="8080" protocol="HTTP/1.1"
#            connectionTimeout="20000"
#            URIEncoding="UTF-8"
#            redirectPort="8443" />
```

3.9 Create the DSpace System User

```
sudo useradd -m dspace
sudo passwd dspace
# Set a password for the dspace user
```

Section 4: Download and Configure DSpace 10.0 Backend

4.1 Download DSpace 10.0 Source

```
cd /home/dspace
sudo -u dspace wget https://github.com/DSpace/DSpace/archive/refs/tags/dspace-10.0.tar.gz
sudo -u dspace tar -xzf dspace-10.0.tar.gz
sudo -u dspace mv DSpace-dspace-10.0 dspace-source
ls /home/dspace/dspace-source
# You should see: dspace/, dspace-api/, dspace-server-webapp/, pom.xml etc.
```

4.2 Create the DSpace Installation Directory

```
sudo mkdir /dspace
sudo chown dspace:dspace /dspace
ls -la / | grep dspace
# Should show: dspace dspace  /dspace
```

4.3 Create local.cfg — Core Configuration File

The local.cfg file controls all key settings for your DSpace installation. Copy the example file and edit it:

```
sudo -u dspace cp /home/dspace/dspace-source/dspace/config/local.cfg.EXAMPLE /home/dspace/dspace-source/dspace/config/local.cfg
sudo -u dspace nano /home/dspace/dspace-source/dspace/config/local.cfg
```

Set the following key values inside local.cfg (minimum required for initial installation):

```
# === REQUIRED SETTINGS ===

# Path to DSpace installation directory
dspace.dir = /dspace

# Public URL of the DSpace backend (REST API)
# Do NOT end with a slash
dspace.server.url = http://localhost:8080/server

# Public URL of the DSpace frontend (User Interface)
# Do NOT end with a slash
dspace.ui.url = http://localhost:4000

# Human-readable name of your DSpace site
dspace.name = My Institutional Repository

# URL of the Solr server
solr.server = http://localhost:8983/solr

# Database settings
db.url = jdbc:postgresql://localhost:5432/dspace
db.driver = org.postgresql.Driver
db.dialect = org.dspace.storage.rdbms.hibernate.postgres.DSpacePostgresDialect
db.username = dspace
db.password = YOUR_DB_PASSWORD_HERE
```

```
db.schema = public

# Email settings (optional at first install, configure later for production)
# mail.server = smtp.yourserver.com
# mail.from.address = dspace@yourinstitution.edu
# feedback.recipient = dspace-admin@yourinstitution.edu
# mail.admin = dspace-admin@yourinstitution.edu
```

WARN	Replace YOUR_DB_PASSWORD_HERE with the actual password you set in Section 3.6. Never leave this as a placeholder in a production system.

Section 5: Build and Install the Backend

STEP 1

Build the Installation Package with Maven

This compiles the entire DSpace backend. Takes 10-20 minutes on first run.

```
cd /home/dspace/dspace-source
sudo -u dspace mvn package
# Wait for BUILD SUCCESS message
# If the build is interrupted, resume with:
# sudo -u dspace mvn package -rf :dspace-installer
```

NOTE If Maven fails with a network timeout, check your internet connection. If behind a proxy, configure ~/.m2/settings.xml with your proxy settings. If heap error occurs, run: export MAVEN_OPTS=-Xmx1024m

STEP 2

Install DSpace to /dspace using Ant

Deploys the built package to your installation directory.

```
cd /home/dspace/dspace-source/dspace/target/dspace-installer
sudo -u dspace ant fresh_install
# Wait for BUILD SUCCESSFUL message
# Verify installation:
ls /dspace
# Should show: bin/ config/ etc/ lib/ log/ solr/ webapps/
```

STEP 3

Initialize the Database

Runs all database migrations to create the DSpace schema.

```
cd /dspace/bin
sudo -u dspace ./dspace database migrate
# Verify the database is fully initialized:
sudo -u dspace ./dspace database info
# All migrations should show state: Success
```

NOTE If any migrations show state 'Failed' or 'Pending', check /dspace/log/dspace.log for ERROR messages before proceeding.

STEP 4

Copy Solr Cores to Solr

DSpace installation creates pre-configured Solr cores that must be copied to your Solr instance.

```
# Copy all DSpace Solr cores to Solr's configsets directory
sudo cp -R /dspace/solr/* /opt/solr/server/solr/configsets/

# Set correct ownership
```

```
sudo chown -R solr:solr /opt/solr/server/solr/configsets/

# Restart Solr to load the new cores
sudo systemctl restart solr

# Verify cores are loaded — open browser or use curl:
curl -s 'http://localhost:8983/solr/search/select' | head -50
# Should return an empty JSON result, NOT an error
```

WARN	DSpace has 6 Solr cores: search, statistics, oai, authority, suggestion, and qaevent. All must be copied. The cp -R command above copies all of them in one step.

5.4 Deploy the Backend — Two Options

Choose ONE of the following deployment methods. Option B (Runnable JAR) is simpler and does not require Tomcat.

Option A: Traditional Tomcat Deployment

```
# Create a context file to tell Tomcat where to find DSpace
sudo mkdir -p /etc/tomcat10/Catalina/localhost
sudo nano /etc/tomcat10/Catalina/localhost/server.xml
# Paste this content:
# <?xml version='1.0'?>
# <Context docBase="/dspace/webapps/server"/>

# Give Tomcat read/write access to /dspace
sudo nano /lib/systemd/system/tomcat10.service
# Add this line in the [Service] section:
# ReadWritePaths=/dspace

sudo systemctl daemon-reload
sudo systemctl start tomcat10
sudo systemctl enable tomcat10

# Verify backend is running:
curl http://localhost:8080/server/api
# Should return JSON with _links section
```

Option B: Runnable JAR (Recommended — No Tomcat Required)

NOTE	This is the simpler option for new installations. Tomcat is embedded inside the JAR file.

```
# Run the backend as dspace user
sudo -u dspace java -jar /dspace/webapps/server-boot.jar &

# To run it as a background service, create a systemd unit:
sudo nano /etc/systemd/system/dspace-backend.service
```

Paste the following into the service file:

```
[Unit]
Description=DSpace 10 Backend (REST API)
```

```
After=network.target postgresql.service solr.service

[Service]
User=dspace
ExecStart=/usr/bin/java -jar /dspace/webapps/server-boot.jar
SuccessExitStatus=143
Restart=on-failure
RestartSec=10

[Install]
WantedBy=multi-user.target
```

```
sudo systemctl daemon-reload
sudo systemctl enable dspace-backend
sudo systemctl start dspace-backend

# Check status:
sudo systemctl status dspace-backend

# Verify backend is running:
curl http://localhost:8080/server/api
# Should return JSON with _links section
```

STEP 5

Create Administrator Account

Creates the initial DSpace administrator from the command line.

```
cd /dspace/bin
sudo -u dspace ./dspace create-administrator
# Follow the prompts to set:
#   Email address
#   First name
#   Last name
#   Password
#   Confirm password
```

Section 6: Install the Frontend (User Interface)

6.1 Install Node.js

Install Node.js v22.x LTS (recommended for DSpace 10 / Angular 20):

```
curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
sudo apt install -y nodejs
node --version
# Expected: v22.x.x
npm --version
```

6.2 Install PM2 (Production Process Manager)

```
sudo npm install --global pm2
pm2 --version
```

6.3 Download DSpace Angular (Frontend) Source

```
cd /home/dspace
sudo -u dspace wget https://github.com/DSpace/dspace-
angular/archive/refs/tags/dspace-10.0.tar.gz -O dspace-angular-10.0.tar.gz
sudo -u dspace tar -xzf dspace-angular-10.0.tar.gz
sudo -u dspace mv dspace-angular-dspace-10.0 dspace-angular
```

6.4 Install Frontend Dependencies

```
cd /home/dspace/dspace-angular
sudo -u dspace npm install
# This downloads all required packages. Takes 5-10 minutes.
```

NOTE	If npm install fails with peer dependency errors, try: sudo -u dspace npm install --legacy-peer-deps

6.5 Configure the Frontend

Create the production configuration file:

```
sudo -u dspace mkdir -p /home/dspace/dspace-angular/config
sudo -u dspace nano /home/dspace/dspace-angular/config/config.prod.yml
```

Paste the following configuration (adjust hostnames for your server):

```
# UI server settings — where Node.js will listen
ui:
  ssl: false
  host: localhost
```

```
    port: 4000
    nameSpace: /

  # REST API settings — must match dspace.server.url in local.cfg
  rest:
    ssl: false
    host: localhost
    port: 8080
    nameSpace: /server
```

NOTE	For production with a real domain and HTTPS, change ssl: true, set the host to your domain, and port to 443. See Section 7 for HTTPS setup.

6.6 Build the Frontend for Production

```
cd /home/dspace/dspace-angular

# Set Node.js memory to 4GB to prevent heap out-of-memory errors
export NODE_OPTIONS=--max-old-space-size=4096

sudo -u dspace npm run build:prod
# This takes 15-30 minutes. Wait for the 'Build complete' message.

# Verify the build output exists:
ls /home/dspace/dspace-angular/dist/
# Should show: browser/  server/
```

WARN	If the build fails with 'JavaScript heap out of memory', increase the value: export NODE_OPTIONS=--max-old-space-size=8192

NOTE	If build fails with 'Cannot find module @popperjs/core', run: sudo -u dspace npm install @popperjs/core — then retry the build.

6.7 Create Deployment Directory

```
sudo mkdir -p /opt/dspace-ui
sudo chown dspace:dspace /opt/dspace-ui

# Copy the compiled dist folder
sudo -u dspace cp -r /home/dspace/dspace-angular/dist /opt/dspace-ui/

# Copy the config folder
sudo -u dspace mkdir -p /opt/dspace-ui/config
sudo -u dspace cp /home/dspace/dspace-angular/config/config.prod.yml
/opt/dspace-ui/config/

# Verify structure:
ls /opt/dspace-ui/
# Should show: dist/  config/
```

6.8 Start the Frontend with PM2

Create a PM2 configuration file:

```
sudo -u dspace nano /opt/dspace-ui/dspace-ui.json
```

Paste the following:

```
{
    "apps": [
        {
           "name": "dspace-ui",
           "cwd": "/opt/dspace-ui",
           "script": "dist/server/main.js",
           "instances": "max",
           "exec_mode": "cluster",
           "env": {
              "NODE_ENV": "production"
           }
        }
    ]
}
```

```
cd /opt/dspace-ui
pm2 start dspace-ui.json

# Check status:
pm2 status

# Check logs:
pm2 logs dspace-ui

# Save PM2 process list so it restarts on reboot:
pm2 save
pm2 startup
# Run the command that pm2 startup outputs
```

Verify the frontend is running:

```
curl http://localhost:4000
# Should return HTML content
```

Section 7: HTTPS Setup with Nginx (Production)

WARN Running DSpace on HTTP is only acceptable for local development. For any production installation accessible from outside your machine, HTTPS is required. Logins will not work without HTTPS.

7.1 Install Nginx

```
sudo apt install -y nginx
sudo systemctl enable nginx
sudo systemctl start nginx
```

7.2 Obtain SSL Certificate (Let's Encrypt — Free)

```
sudo apt install -y certbot python3-certbot-nginx
sudo certbot --nginx -d yourdomain.edu
# Follow the prompts. Certbot will automatically configure your SSL
certificate.
```

NOTE Replace yourdomain.edu with your actual domain name. Your server must be publicly accessible with that domain for Let's Encrypt to work.

7.3 Configure Nginx as Reverse Proxy

```
sudo nano /etc/nginx/sites-available/dspace
```

Paste the following Nginx configuration:

```
# Redirect HTTP to HTTPS
server {
    listen 80;
    server_name yourdomain.edu;
    return 301 https://$host$request_uri;
}

# HTTPS server
server {
    listen 443 ssl;
    server_name yourdomain.edu;

    ssl_certificate /etc/letsencrypt/live/yourdomain.edu/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/yourdomain.edu/privkey.pem;

    # Backend: proxy /server to DSpace REST API on port 8080
    location /server {
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header X-Forwarded-Host $host;
        proxy_pass http://localhost:8080/server;
    }

    # Frontend: proxy all other requests to PM2 on port 4000
```

```
    location / {
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header X-Forwarded-Host $host;
        proxy_pass http://localhost:4000/;
    }
}
```

```
sudo ln -s /etc/nginx/sites-available/dspace /etc/nginx/sites-enabled/
sudo nginx -t
# Should say: configuration file test is successful
sudo systemctl reload nginx
```

7.4 Update local.cfg for HTTPS URLs

```
sudo -u dspace nano /home/dspace/dspace-source/dspace/config/local.cfg
# Update these two lines to use HTTPS:
# dspace.server.url = https://yourdomain.edu/server
# dspace.ui.url = https://yourdomain.edu
```

Update the frontend config.prod.yml as well:

```
sudo -u dspace nano /opt/dspace-ui/config/config.prod.yml
# Update to:
# ui:
#   ssl: false
#   host: localhost
#   port: 4000
#   nameSpace: /
# rest:
#   ssl: true
#   host: yourdomain.edu
#   port: 443
#   nameSpace: /server
```

Restart the backend and frontend:

```
sudo systemctl restart dspace-backend
pm2 restart dspace-ui
```

Section 8: Post-Installation Tasks

8.1 Verify Full Installation

```
# 1. Check backend REST API
curl https://yourdomain.edu/server/api
# Should return JSON with _links

# 2. Check OAI-PMH endpoint
curl 'https://yourdomain.edu/server/oai/request?verb=Identify'

# 3. Open browser and navigate to:
#    https://yourdomain.edu
#    You should see the DSpace homepage

# 4. Log in with the administrator account you created in Section 5
```

8.2 Set Up Scheduled Tasks (Required for Full Functionality)

Several DSpace features require cron jobs to run regularly. Set them up as the dspace user:

```
sudo -u dspace crontab -e
```

Add the following cron entries:

```
# Generate thumbnails for newly submitted items (every 5 minutes)
*/5 * * * * /dspace/bin/dspace filter-media > /dev/null 2>&1

# Send subscription emails daily at 8am
0 8 * * * /dspace/bin/dspace subscription-send -f D > /dev/null 2>&1

# Re-index Solr statistics (nightly at 1am)
0 1 * * * /dspace/bin/dspace solr-reindex > /dev/null 2>&1

# Generate Google Scholar sitemaps (weekly, Sunday at 2am)
0 2 * * 0 /dspace/bin/dspace generate-sitemaps > /dev/null 2>&1

# Clean up old sessions / temporary data (nightly at 3am)
0 3 * * * /dspace/bin/dspace cleanup > /dev/null 2>&1
```

8.3 Index Content into Solr

```
cd /dspace/bin
sudo -u dspace ./dspace index-discovery
# Initial indexing may take a few minutes
```

8.4 Enable Geographic Location Statistics (Optional)

Download the free DB-IP city database for location-based statistics:

```
cd /dspace/config
```

```
sudo -u dspace wget https://download.db-ip.com/free/dbip-city-lite-$(date +%Y-
%m).mmdb.gz
sudo -u dspace gunzip dbip-city-lite-*.mmdb.gz
sudo -u dspace mv dbip-city-lite-*.mmdb /dspace/config/

# Add to local.cfg:
# usage-statistics.dbfile = /dspace/config/dbip-city-lite-YYYY-MM.mmdb
```

Section 9: Troubleshooting Common Issues

9.1 Backend Will Not Start

```
# Check systemd logs:
sudo journalctl -u dspace-backend -f

# Check DSpace log:
tail -100 /dspace/log/dspace.log

# Verify Solr is running:
sudo systemctl status solr
curl http://localhost:8983/solr/search/select
```

9.2 Solr Error: 'Expected mime type application/octet-stream but got text/html'

This means Solr cores are not properly installed or Solr is not running.

```
# Restart Solr:
sudo systemctl restart solr

# Re-copy cores if missing:
sudo cp -R /dspace/solr/* /opt/solr/server/solr/configsets/
sudo chown -R solr:solr /opt/solr/server/solr/configsets/
sudo systemctl restart solr

# Test each core:
curl http://localhost:8983/solr/search/select
curl http://localhost:8983/solr/statistics/select
curl http://localhost:8983/solr/oai/select
```

9.3 Frontend Shows 'No _links section found' Error

```
# Test backend connection from UI machine:
curl https://yourdomain.edu/server/api

# Verify dspace.ui.url in local.cfg exactly matches your frontend URL
# Verify dspace.server.url exactly matches your backend URL
# Both must use the same protocol (http or https) and NO trailing slash
```

9.4 Login Returns '403 Invalid CSRF Token'

This almost always means the backend is not running HTTPS in a production setup.

```
# Verify backend URL in local.cfg uses https:
grep dspace.server.url /home/dspace/dspace-source/dspace/config/local.cfg

# Verify Nginx is sending X-Forwarded-Proto header
# Check Nginx config includes: proxy_set_header X-Forwarded-Proto https;
```

9.5 Frontend Build — JavaScript Heap Out of Memory

```
export NODE_OPTIONS=--max-old-space-size=8192
sudo -u dspace npm run build:prod
```

9.6 Maven Build Fails — Network Timeout

```
# Resume a failed Maven build from the point of failure:
cd /home/dspace/dspace-source
sudo -u dspace mvn package -rf :dspace-installer
```

9.7 Database Connection Error During 'ant fresh_install'

```
# Test PostgreSQL connection directly:
psql -U dspace -W -h localhost -d dspace
# Enter password. If this fails, check pg_hba.conf and postgresql.conf.

# Verify PostgreSQL is running:
sudo systemctl status postgresql
```

Section 10: Quick Reference — Service Commands

Use these commands to manage DSpace services on a daily basis.

10.1 Start / Stop / Restart All Services

```
# PostgreSQL
sudo systemctl start postgresql
sudo systemctl stop postgresql
sudo systemctl restart postgresql

# Solr
sudo systemctl start solr
sudo systemctl stop solr
sudo systemctl restart solr

# DSpace Backend
sudo systemctl start dspace-backend
sudo systemctl stop dspace-backend
sudo systemctl restart dspace-backend

# DSpace Frontend (PM2)
pm2 start dspace-ui.json
pm2 stop dspace-ui
pm2 restart dspace-ui

# Nginx
sudo systemctl start nginx
sudo systemctl stop nginx
sudo systemctl reload nginx
```

10.2 DSpace Admin Commands

```
# All dspace commands must be run from /dspace/bin as the dspace user
cd /dspace/bin

# Index content into Solr search
sudo -u dspace ./dspace index-discovery

# Index content incrementally (updates only)
sudo -u dspace ./dspace index-discovery -b

# Generate thumbnails
sudo -u dspace ./dspace filter-media

# Database info
sudo -u dspace ./dspace database info

# Clean up orphaned/temporary data
sudo -u dspace ./dspace cleanup

# Generate sitemaps
sudo -u dspace ./dspace generate-sitemaps

# Check version
sudo -u dspace ./dspace version
```

10.3 Log File Locations

```
# DSpace backend log
tail -f /dspace/log/dspace.log

# Solr log
tail -f /opt/solr/server/logs/solr.log

# Nginx access and error logs
tail -f /var/log/nginx/access.log
tail -f /var/log/nginx/error.log

# PM2 frontend logs
pm2 logs dspace-ui
```

DSpace 10.0 — Official Documentation
https://wiki.lyrasis.org/display/DSDOC10x/
For community support: https://groups.google.com/g/dspace-tech
Compiled by AI4LIB | ai4lib.in | June 2026