DRM-X 5.0

Python + Django Custom Integration Development Guide

Complete license workflow example using Django Session and zeep

Document Information	Value
Document version	v1.0.0
Example project	DRM-X5_Python_Django_Integration_Example
Audience	Python developers, Django developers, and system integrators
Coverage	Login Session, course validation, ZJGet POST, zeep SOAP methods, and license response

> **Note** This guide is based on the verified demo code. The fixed account, fixed courses, and plain-text configuration are included only to explain the integration workflow; a production deployment must connect to the customer's own authentication and course-purchase data.

Download the sample code

Contents

1. Purpose and Scope of the Demo 3

2. System Architecture and License Workflow 3

3. Project Structure and Configuration 4

4. Login and Django Session Continuation 7

5. ZJGet POST Endpoint and Parameters 8

6. Course ID Validation Rules 10

7. zeep and the DRM-X 5.0 WSDL 10

8. DRM-X 5.0 API Method Details 11

9. Django Routes and Page Responsibilities 18

10. Local Operation and Demo Verification 19

11. Production Integration Recommendations 22

12. Troubleshooting 23

Appendix: File Responsibility Reference 24

1. Purpose and Scope of the Demo

This example shows how to integrate the DRM-X 5.0 license service with a Django website. Django receives license parameters submitted automatically by the ZJGet browser, checks the login Session and course permissions, calls the DRM-X 5.0 SOAP service through zeep, and returns the license HTML.

The demo focuses on the license workflow rather than building a complete LMS. It uses a fixed account and a fixed list of purchased courses so developers can run, debug, and understand each step quickly.

- Fixed account login: student / 123456, used to demonstrate login and Session recovery.
- Fixed purchased-course list: simulates orders, enrollments, or subscriptions in config.py.
- Course ID: submitted by ZJGet POST from the License Profile Product ID (yourproductid); it is not hard-coded in the license endpoint.
- Rights ID: both the permission update and license request use the rightsid from the same POST request.
- DRM-X configuration: stored directly in drmx5_demo/config.py; the demo does not depend on environment variables.

2. System Architecture and License Workflow

Component	Responsibility
DRM-X 5.0 License Profile	Stores the Product ID (course ID) and RightsID; DRM-X reads them during playback and passes them to ZJGet.
ZJGet browser	Opens the encrypted video and POSTs profileid, clientinfo, rightsid, yourproductid, and other parameters to the integration URL.
Django integration endpoint	Handles /licstore5, the same-origin relay, login Session, course validation, and the license response.

Component	Responsibility
zeep SOAP client	Creates a client from the WSDL and calls DRM-X 5.0 methods by parameter name.
DRM-X 5.0 service	Checks or creates users, checks revocation and lockout, updates permissions, and generates license HTML.

1. ZJGet reads the License Profile information, then POSTs to /licstore5.
2. Django first performs a same-origin POST relay so the browser includes the website Session Cookie in subsequent requests.
3. The backend validates the parameters and saves the license request in the Django Session.
4. If the user is not logged in, the login page is shown; after login, the request is restored from the Session and continues.
5. Split yourproductid on hyphens and verify that the current user has purchased at least one listed course.
6. Call CheckUserExists; if the user does not exist, call AddNewUser.
7. For an existing user, call CheckUserIsRevoked and GetUserDetails, and reject revoked or locked users.
8. Call UpdateRightWithDisableVirtualMachine with the POST rightsid.
9. Call getLicenseRemoteToTableWithVersionWithMac with the same POST rightsid.
10. Verify that the returned HTML contains license_div_drm-x5, then return it to ZJGet.

3. Project Structure and Configuration

3.1 Main Directories

```
DRM-X5_Python_Django_Integration_Example/
```

```
├─ drmx5_demo/
│  ├─ config.py        # DRM-X, demo account, and course settings
│  ├─ settings.py      # Django and Session settings
│  └─ urls.py
├─ integration/
│  ├─ views.py         # Login, course validation, and the complete DRM-X workflow
│  ├─ urls.py
│  ├─ templates/       # Login, relay, license, and error pages
├─ manage.py
└─ requirements.txt
```

The core business logic is concentrated in integration/views.py so developers can read and port it in execution order.

3.2 DRM-X 5.0 Service Configuration

```
DRMX5_ADMIN_EMAIL = "YOUR_ADMIN_EMAIL"
DRMX5_WEB_SERVICE_AUTH_STR = "YOUR_WEB_SERVICE_AUTH_STR"
DRMX5_GROUP_ID = "YOUR_GROUP_ID"
# China service
DRMX5_SERVICE_URL = "https://5.drm-x.cn/haihaisoftlicenseservice.asmx"
# International service
# DRMX5_SERVICE_URL = "https://5.drm-x.com/haihaisoftlicenseservice.asmx"
DRMX5_TIMEOUT_SECONDS = 30
```

Setting	Required	Description
DRMX5_ADMIN_EMAIL	Yes	DRM-X 5.0 administrator email corresponding to SOAP AdminEmail.
DRMX5_WEB_SERVICE_AUTH_STR	Yes	Website integration authentication string; it is sensitive and must remain on the server.
DRMX5_GROUP_ID	Yes	DRM-X user group ID used when creating users and obtaining licenses.

Setting	Required	Description
DRMX5_SERVICE_URL	Yes	Enter the .asmx service address without ?wsdl; the code appends it automatically.
DRMX5_TIMEOUT_SECONDS	No	WSDL and SOAP request timeout; the demo uses 30 seconds.

> **Service region:**
>
> The international service uses 5.drm-x.com; the China service uses 5.drm-x.cn.
>
> The configured value must end with haihaisoftlicenseservice.asmx.

3.3 Demo Account, Course, and Permission Settings

```
DEMO_USERNAME = "student"
DEMO_PASSWORD = "123456"
DEMO_EMAIL = "student@example.com"
DEMO_FULL_NAME = "Demo Student"

DEMO_PURCHASED_COURSE_IDS = ("1001",)
DEMO_COURSE_BEGIN_DATE = "2026-01-01"
DEMO_COURSE_EXPIRATION_DATE = "2036-01-01"
DEMO_BIND_NUMBER = "1"
```

Setting	Purpose
DEMO_USERNAME / DEMO_PASSWORD	Fixed login account used only to demonstrate the Session workflow.
DEMO_EMAIL / DEMO_FULL_NAME	Used when creating the DRM-X user, generating watermark text, and obtaining the license.
DEMO_PURCHASED_COURSE_IDS	Simulates the set of course IDs purchased or joined by the current user.

Setting	Purpose
DEMO_COURSE_BEGIN_DATE	Permission start date; the format must be YYYY-MM-DD.
DEMO_COURSE_EXPIRATION_DATE	Permission expiration date; the format must be YYYY-MM-DD.
DEMO_BIND_NUMBER	Number of device bindings assigned when AddNewUser creates the user.

4. Login and Django Session Continuation

Django Session stores two key objects. File-based Sessions store data in the server's .sessions directory, avoiding Cookie-size limits when clientinfo is large. A production system can use a database or cache-backed Session instead.

Session key	Stored content	Purpose
drmx5_user	username, email, full_name	Determines whether the user is logged in and provides identity information for the DRM-X user and license.
drmx5_license_request	Nine validated ZJGet parameters	Temporarily stores the request before login and restores the license workflow after login.

4.1 Why a Same-Origin POST Relay Is Needed

ZJGet may POST to the website from a different origin. With a SameSite=Lax Cookie, the first cross-site POST may not include the existing Session Cookie. When licstore5 receives the request for the first time, it returns relay.html. That page automatically POSTs to the same site again and adds the internal marker _drmx5_relay=1. The second request can then read the Session and recognize an existing login.

> **Internal parameter:** _drmx5_relay only marks the same-origin relay. It is not a DRM-X or ZJGet business parameter and is not sent to the SOAP service.

4.2 Recovery Flow When the User Is Not Logged In

1. Save the license request in drmx5_license_request.
2. Redirect to /login?resume=1.
3. The user logs in with the demo account, and Django cycle_key() rotates the Session ID.
4. After login, redirect to /licstore5?resume=1.
5. Restore the original request from the Session and continue without requiring ZJGet to POST again.

> **Session settings:**
>
> The demo uses SESSION_COOKIE_HTTPONLY=True, SESSION_COOKIE_SAMESITE='Lax', and a 30-minute lifetime. After enabling HTTPS in production, set SESSION_COOKIE_SECURE=True.

5. ZJGet POST Endpoint and Parameters

5.1 Integration URL

The license URL configured in DRM-X 5.0 should point directly to the Django backend:

```
Local test: http://localhost:8080/licstore5
Production: https://your-domain.example/licstore5
```

This endpoint accepts application/x-www-form-urlencoded POST requests. Normally, DRM-X 5.0 and ZJGet generate the parameters automatically from the current encrypted content; the web page neither needs nor should provide a manual input form.

5.2 The Nine POST Parameters

Parameter	Required by code	Source and meaning	Demo handling
profileid	Yes	Current License Profile ID.	Passed unchanged as ProfileID when obtaining the license.
clientinfo	Yes	Client license data generated by ZJGet.	Passed unchanged as ClientInfo; up to 8,192 characters are read.
rightsid	Yes	Rights ID selected in the License Profile.	Used for both the permission update and license request.
yourproductid	Yes	Product ID configured in the License Profile; this demo interprets it as a course ID.	Split on hyphens and match against purchased courses.
platform	Yes	ZJGet platform information.	Passed unchanged as Platform.
contenttype	Yes	Protected content type.	Passed unchanged as ContentType.
version	Yes	ZJGet client version.	Passed unchanged as Version.
return_url	Yes	Return address or identifier used after the license is obtained.	Displayed on the result page and used after completion; unsafe URL schemes are rejected.
mac	Yes	Device or client binding information.	Passed unchanged as Mac.

6. Course ID Validation Rules

yourproductid comes from the DRM-X 5.0 License Profile. The demo splits this value on hyphens, then selects the first course, in POST order, that appears in DEMO_PURCHASED_COURSE_IDS.

```
DEMO_PURCHASED_COURSE_IDS = ("1001", "1003")
```

POST yourproductid	Purchased courses	Result
1001	1001, 1003	Pass; selects course 1001.
1001-1002	1001, 1003	Pass; selects 1001 in POST order.
1002-1003-1001	1001, 1003	Pass; 1003 appears first and is selected.
1002-1004	1001, 1003	Reject; none of the purchased courses match.

> **Permission boundary:** If any one course in the list matches, the current license request is allowed. This does not grant the user access to the other courses in the yourproductid list. A production system must associate the current authenticated user with real order, enrollment, or subscription records.

7. zeep and the DRM-X 5.0 WSDL

The demo uses zeep to generate SOAP requests from the WSDL. get_drmx_client() loads the WSDL when a license is first requested and reuses the client through lru_cache(maxsize=1). Unlike manually constructed XML, zeep handles parameter structures and SOAPAction according to the WSDL.

Item	Rule
Configured address	DRMX5_SERVICE_URL stores the .asmx address without ?wsdl.
WSDL address	The code automatically uses DRMX5_SERVICE_URL + '?wsdl'.
Parameter passing	All methods are called with keyword arguments matching WSDL parameter names exactly, including case.
Timeout	Transport timeout and operation_timeout both use DRMX5_TIMEOUT_SECONDS.
XML settings	strict=True，xml_huge_tree=False。
Errors	zeep and connection errors are converted to DrmxError, and the endpoint returns HTTP 502.

International WSDL: https://5.drm-x.com/haihaisoftlicenseservice.asmx?wsdl
China WSDL: https://5.drm-x.cn/haihaisoftlicenseservice.asmx?wsdl

8. DRM-X 5.0 API Method Details

issue_license() executes six DRM-X methods in a fixed sequence.

Except when creating a new user, all subsequent calls use the DRM-X 5.0 username mapped from the current website user; both the permission update and license request use the same POST rightsid.

#	Method	Condition	Failure handling
1	CheckUserExists	After course validation	Stop if the result is not true/false or 1/0.
2	AddNewUser	User does not exist	Stop if the result is not 1.
3	CheckUserIsRevoked	User exists	Return 403 if revoked.

#	Method	Condition	Failure handling
4	GetUserDetails	User exists and is not revoked	Return 403 if IsLockedOut is true.
5	UpdateRightWithDisableVirtualMachine	User is allowed	Stop if the result is not 1.
6	getLicenseRemoteToTableWithVersionWithMac	Permission update succeeded	Stop if the response lacks the license marker.

8.1 CheckUserExists

Purpose: checks whether the current website user already exists in DRM-X 5.0.

Parameter	Source	Description
UserName	Current authenticated user's username	DRM-X login name and the stable user identifier for subsequent calls.
AdminEmail	config.py	DRM-X administrator email.
WebServiceAuthStr	config.py	Web Service authentication string.

A return value of true or 1 means the user exists; false or 0 means the user does not exist.

Any other value is treated as an error.

8.2 AddNewUser

When called: when CheckUserExists reports that the user does not exist. It creates a DRM-X 5.0 account corresponding to the current website user.

Parameter group	Demo source or value	Description
AdminEmail / WebServiceAuthStr / GroupID	config.py	Authenticates the request and assigns the new user to a DRM-X user group.
UserLoginName	Authenticated user's username	DRM-X login name.
UserPassword	N/A	The demo does not use DRM-X password login.
UserEmail / UserFullName	Session user information	Email and full name; the email can also be used as watermark text.
Title / Company / Address / City / Province / ZipCode / Phone / CompanyURL	N/A	Information not collected by the demo.
SecurityQuestion / SecurityAnswer	N/A	The demo does not use DRM-X password recovery.
IP	REMOTE_ADDR	Remote IP of the license request.
Money	0	Initial balance for a new user.
BindNumber	DEMO_BIND_NUMBER	Allowed number of device bindings.
IsApproved / IsLockedOut	yes / no	Approve immediately and leave unlocked.

AddNewUserResult must be 1. A newly created user is not checked again for revocation or lockout during the same request.

8.3 CheckUserIsRevoked (User License Revocation Requires the Enterprise Edition)

When called: only when the DRM-X user already existed. It checks whether the administrator has revoked that user.

Parameter	Source
AdminEmail	config.py
WebServiceAuthStr	config.py
UserLoginName	Current user's username

A return value of true, 1, or yes is treated as revoked. The backend returns HTTP 403 and refuses the license. This feature requires a DRM-X edition that supports user revocation.

8.4 GetUserDetails

When called: after an existing user passes the revocation check. The demo reads only IsLockedOut; if it is not present at the top level, it continues by reading GetUserDetailsResult.

Parameter	Source
AdminEmail	config.py
WebServiceAuthStr	config.py
UserLoginName	Current user's username

If IsLockedOut is true or 1, the backend returns HTTP 403. If the field is completely absent from the response, the service response is treated as invalid.

8.5 UpdateRightWithDisableVirtualMachine

Purpose: updates the permission identified by the POST rightsid. The demo updates this RightsID before each license is issued; a production system should decide whether every request requires an update according to its business policy.

Parameter	Demo value or source	Description
AdminEmail / WebServiceAuthStr	config.py	DRM-X service authentication.
RightsID	POST rightsid	Rights ID to update.
Description	Python Sample	Permission description.
PlayCount	-1	Unlimited play count.
BeginDate / ExpirationDate	Course date settings	Converted from YYYY-MM-DD to yyyy/MM/dd.
ExpirationAfterFirstUse	-1	Do not use a relative expiration period after first use.
RightsPrice	0	Rights price is 0.
AllowPrint / AllowClipBoard / AllowDoc	False	Disable printing, clipboard, and document permissions.
DisableVirtualMachine	True	Disable use in virtual machines.
require_admin_permission / bind_1st_screen	False / False	Do not require administrator permission and do not bind the first screen.

Watermark, Blacklist, and Intelligent Screen-Recording Prevention

Parameter	Demo value	Description
EnableWatermark / WatermarkText	True / username	Enable the primary watermark and display current user information.

Parameter	Demo value	Description
WatermarkArea	1,2,3,4,5,	Primary watermark display areas.
RandomChangeArea / RandomFrquency	True / 12	Randomly change the watermark area at the configured frequency.
EnableBlacklist / EnableWhitelist	True / True	Enable blacklist and intelligent screen-recording prevention.
ExpireTimeUnit	Day	Expiration unit: Year/Month/Day/Hour/Minute.
PreviewTime / PreviewTimeUnit / PreviewPage	3 / Day / 3	No functional effect in this demo; ignore this field.

Third-Layer Dynamic Watermark and Second-Layer Scrolling Watermark

Parameter group	Demo value	Description
enable_wm_3rd	1	Enable the third-layer dynamic watermark.
wm_3rd_random_change_angle / textalpha	1 / 90	Random angle and opacity.
wm_3rd_rows / columns	3 / 3	Watermark row and column counts.
wm_3rd_fixed_angle / textsize / textcolor	0 / 14 / 16744576	Angle, font size, and decimal color value.
wm_3rd_angle_change_interval / text	10 / user email or username watermark	Angle-change interval and watermark text.
enable_wm_2nd	0	Disable the second-layer scrolling watermark.

Parameter group	Demo value	Description
wm_2nd_text / speed / textsize / textcolor / textalpha	user watermark / 1 / 22 / 16744576 / 90	Submit these values according to the API structure even when disabled.

UpdateRightWithDisableVirtualMachineResult must be 1. These values demonstrate one policy and are not recommended as universal settings for every customer.

8.6 getLicenseRemoteToTableWithVersionWithMac

Purpose: after the permission update succeeds, generates the final license HTML from the current profile, client data, user, and device information.

Parameter	Source	Description
AdminEmail / WebServiceAuthStr	config.py	DRM-X service authentication.
ProfileID	POST profileid	Current License Profile ID.
ClientInfo	POST clientinfo	ZJGet client license data.
RightsID	POST rightsid	Uses the same RightsID as the permission update.
UserLoginName / UserFullName	Session user	DRM-X login name and full name.
GroupID	config.py	DRM-X user group ID.
Message	N/A	ASMX input/output parameter; service information is displayed on the result page.
IP	REMOTE_ADDR	Remote IP of the license request.
Platform / ContentType / Version / Mac	Corresponding POST fields	ZJGet platform, content, version, and device information.

The code reads getLicenseRemoteToTableWithVersionWithMacResult as the license HTML and also reads Message. A successful response must contain license_div_drm-x5; otherwise, the code strips HTML tags, creates an error message, and returns HTTP 502.

9. Django Routes and Page Responsibilities

Path	Method	Responsibility
/	GET	Redirects to the login page.
/login	GET / POST	Displays the fixed-account login, creates the Session, and resumes the license workflow when needed.
/logout	POST	Clears the current Session.
/api/session	GET	Returns authenticated status, user information, and whether a pending license request exists.
/licstore5	POST	ZJGet integration endpoint; receives parameters, performs the same-origin relay, and issues the license.
/licstore5?resume=1	GET	Restores the previously saved license request from the Session.

Template	Responsibility
login.html	Displays login status, the fixed-account login form, and errors.
relay.html	Automatically submits the first cross-site POST parameters to same-origin /licstore5.
license.html	Outputs the DRM-X license HTML, Message, and return-address logic.
error.html	Displays 400, 403, 502, and other errors consistently.

10. Local Operation and Demo Verification

10.1 Prerequisites

- Python 3.10 or a compatible version.
- A DRM-X 5.0/ZJGet test environment and an encrypted video.

10.2 Create the Environment and Install Dependencies

```
cd "DRM-X5_Python_Django_Integration_Example"
python -m venv .venv
.\.venv\Scripts\Activate.ps1
python -m pip install -r requirements.txt
```

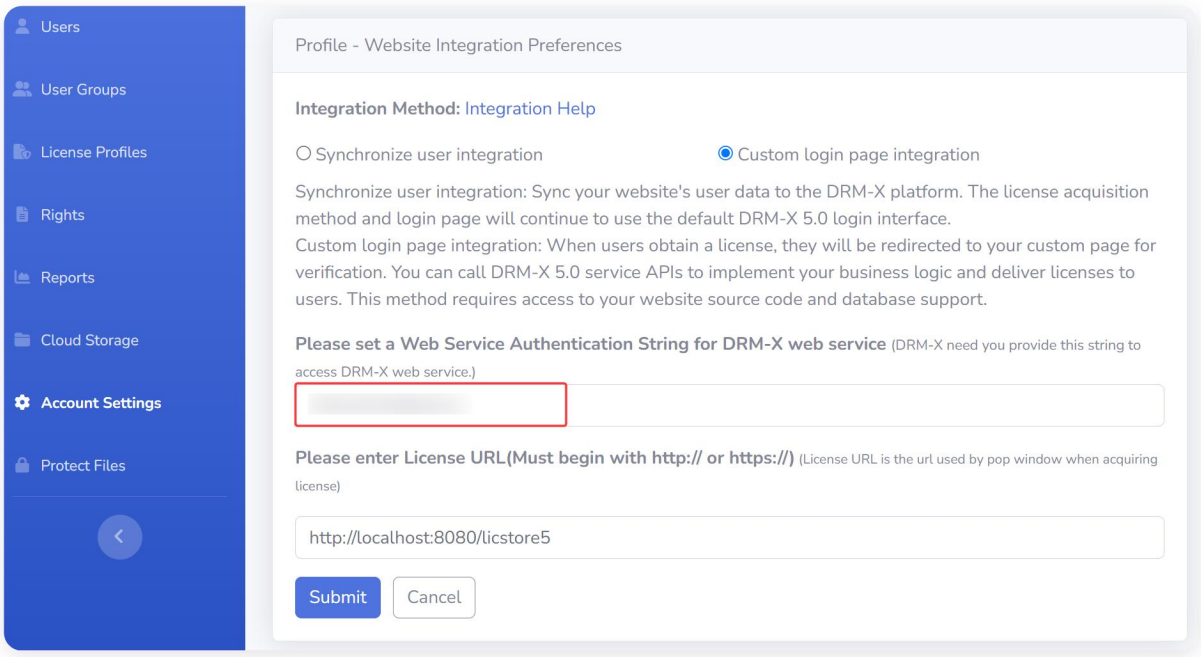
10.3 Check and Start

```
python manage.py check
python manage.py runserver 8080
```

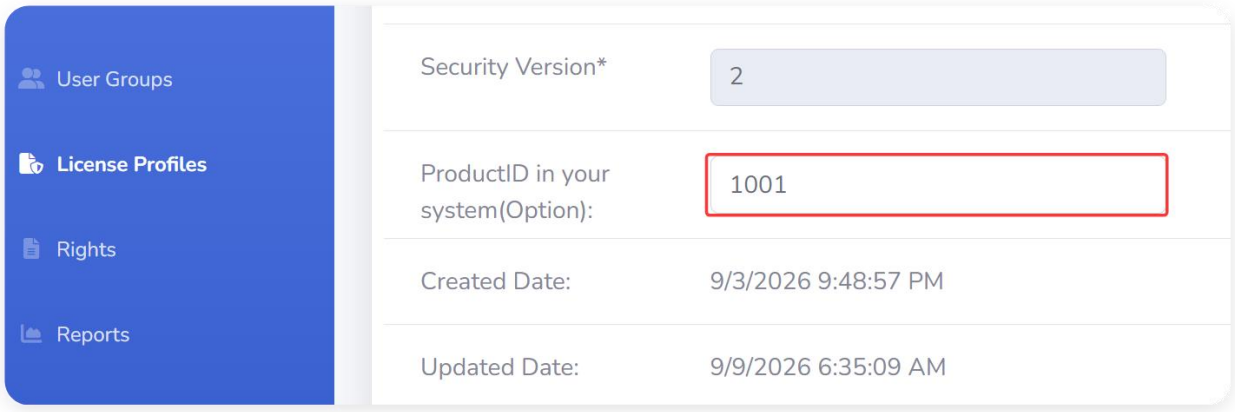
After startup, the login page is http://localhost:8080/login. The demo username is student and the password is 123456.

10.4 DRM-X 5.0 Configuration and End-to-End Verification

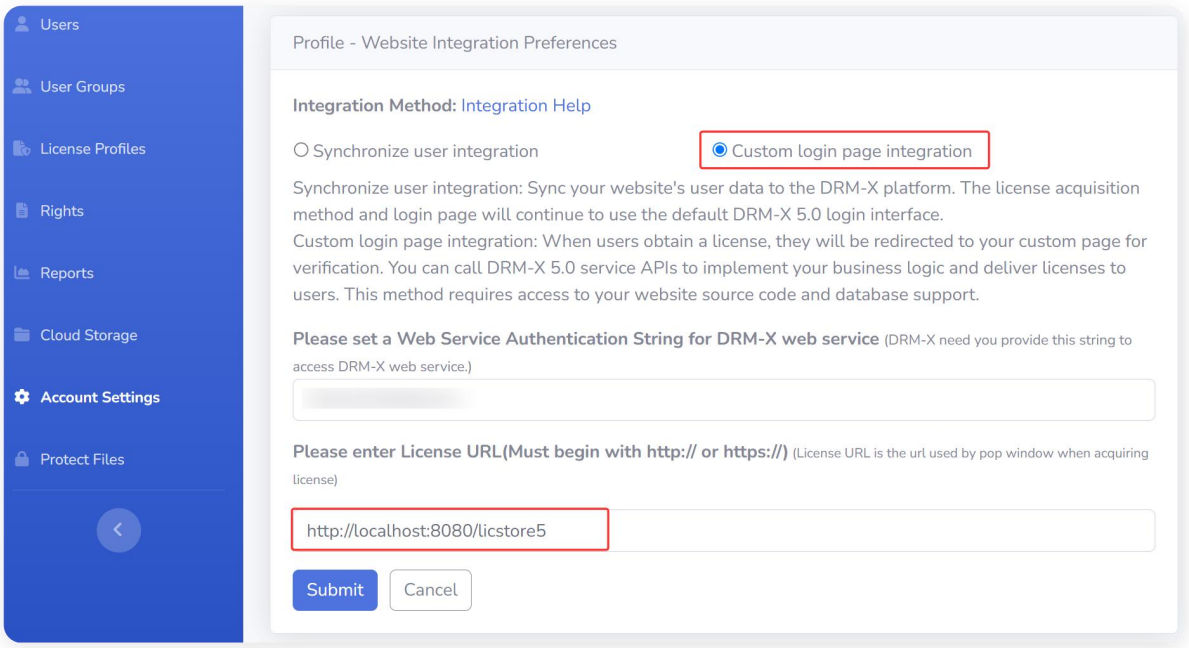
1. In config.py, enter the AdminEmail, WebServiceAuthStr, and GroupID for the demo account, and select the China or international service URL.



2. Add the demo course ID to DEMO_PURCHASED_COURSE_IDS.
3. Set the same course ID in the License Profile for the demo video, for example 1001. Multiple values can be written as 1001-1002.



4. In the DRM-X 5.0 website integration settings, select Custom Login Page Integration and set the license URL to http://localhost:8080/licstore5



5. Open the encrypted video with ZJGet. Do not browse directly to /licstore5 in a regular browser.
6. If not logged in, complete the login and confirm that the page restores the request automatically. If already logged in, the workflow should proceed directly to course and DRM-X user checks.
7. Confirm that user creation or status checks succeed, the permission is updated, and ZJGet obtains the license and plays the content.

11. Production Integration Recommendations

Demo implementation	Production recommendation
Fixed username and password	Integrate Django Authentication, SSO, or the customer's identity system, and keep username mapping stable and unique.
Fixed purchased-course list	Query real course permissions for the authenticated user from orders, enrollments, subscriptions, or the LMS database.
Plain-text config.py	Move secrets to a secret-management service or environment variables when stronger protection is required.
File-based Session	Use a database, Redis, or another shared Session backend for multi-server deployment.
Development HTTP	Production must use HTTPS, set DEBUG=False and SESSION_COOKIE_SECURE=True, and configure ALLOWED_HOSTS.
REMOTE_ADDR as IP	Behind a reverse proxy, read the forwarded client IP only after trusted-proxy settings are configured.
Update POST RightsID on every request	Decide whether to update according to play count, validity, watermark, binding, and virtual-machine policy; assess concurrency when multiple Profiles share one RightsID.
Return errors directly on a page	Keep traceable server logs, but show only necessary information to the client and avoid exposing credentials.

> **Production boundary** Course validation must prove that the current authenticated user owns at least one course in the current POST request; checking only that a course exists is insufficient. The DRM-X username mapping should also remain stable if the user changes their display name.

12. Troubleshooting

Symptom	Common cause	Resolution
Opening /licstore5 directly shows No saved license request	There is no ZJGet POST and no pending request in the Session.	Open the encrypted file with ZJGet and verify the complete workflow.
License request does not continue after login	The Cookie, Session, or relay did not take effect.	Check that .sessions is writable, and verify the browser Cookie, same-origin URL, and /login?resume=1.
ModuleNotFoundError: zeep	Dependencies are not installed or the wrong Python environment is active.	Run python -m pip install -r requirements.txt in the active virtual environment.
WSDL or connection failure	Incorrect service region, address, network, or TLS configuration.	Verify the .asmx address, 5.drm-x.com/.cn, network, and timeout; do not duplicate ?wsdl in the configuration.
HTTP 400	A required POST parameter is missing or return_url uses an unsafe scheme.	Check profileid, clientinfo, rightsid, yourproductid, and return_url.
HTTP 403	The course does not match, or the DRM-X user is revoked or locked.	Verify the purchased-course list, License Profile Product ID, and DRM-X user status.
HTTP 502	Invalid DRM-X configuration, SOAP call, or service response.	Verify credentials, GroupID, WSDL, backend logs, and the API response.
Port 8080 is already in use	Another process is listening on this port.	Stop the process or change the port and update the DRM-X integration URL accordingly.
Invalid license HTML	The service returned error text or an incomplete structure.	Check Message; successful HTML must contain license_div_drm-x5.

Appendix: File Responsibility Reference

File	Responsibility
drmx5_demo/config.py	DRM-X service configuration, demo account, course dates, purchased-course list, and binding count.
drmx5_demo/settings.py	Django middleware, templates, file-based Session, Cookie, and static-resource settings.
drmx5_demo/urls.py	Delegates root routes to integration.urls.
integration/views.py	Parameter parsing, course validation, login, Session, zeep client, and six DRM-X methods.
integration/urls.py	Defines /login, /logout, /api/session, and /licstore5.
integration/templates/integration/*.html	relay performs the same-origin POST relay; login handles authentication; license displays the license; error shows error responses.
requirements.txt	Declares Django and zeep dependencies.
README.md	Quick-start description, configuration, and run instructions.