Pattern 1: Pod Restarts / CrashLoopBackOff Troubleshooting Guide

---

Issue/Error

The **Pod** is stuck in `CrashLoopBackOff` or is **restarting repeatedly**. This state is caused by either the application crashing immediately on startup (Application Failure) or Kubernetes forcefully restarting it due to a failed health check (Probe Failure).

Analysis Steps: Application & Configuration Checks

These steps help identify issues *within* the application or its environment that cause a crash:

1. **Examine Pod Details & Events:**

Run:
Bash
kubectl describe pod <pod-name>

    - **Check the `Reason` and `Events` section** for clues like `OOMKilled`, `Error`, or `CrashLoopBackOff`.
2. **Tail Application Logs:**

Retrieve logs to find the immediate cause of the crash (e.g., exceptions during startup):
Bash
kubectl logs -f <pod-name>

    - **Look for configuration exceptions** or connectivity errors.
3. **Verify Configuration & Secrets:**
    - Check the **ConfigMap** and **Secrets** to ensure all required environment variables, configuration values, and credentials (e.g., DB URLs) are present and correct.

Analysis Steps: Liveness / Readiness Probe Checks

These steps help identify misconfigurations in the Kubernetes probes that trigger unnecessary restarts or prevent traffic flow:

1. **Check Probe Configuration:**
    - Review the **Liveness** and **Readiness** sections outputted by the `kubectl describe pod` command.
    - **Liveness Failure:** Kubernetes is repeatedly **restarting the container** because the health endpoint is unreachable or returning an error.
    - **Readiness Failure:** The pod is **never marked** `Ready` and receives no traffic.
2. **Validate Probe Endpoints:**
    - Manually verify the **probe path** (e.g., `/actuator/health`) and **port** used in the probe definition are correct and accessible *inside* the container.

Root Causes & Resolutions

Category	Root Causes	Resolution
Application Crash	**Invalid/missing ConfigMap or Secret values.**	Correct the missing or incorrect configuration.
	Wrong DB/service URL or credentials.	Update connectivity details in the configuration.
	OOMKilled (Memory limit breached).	Increase the resources.limits.memory in the Pod spec.
	Missing files (e.g., certificates/keystores).	Ensure all required files are mounted correctly via volumes.

Probe Failure	**Incorrect probe path or port.**	**Update probe path** to the correct health endpoint (e.g., `/health/ready`).
	Application slow to start (initialDelaySeconds too low).	**Increase `initialDelaySeconds`** to allow the application to fully initialize.
	Application slow to respond (timeoutSeconds too low).	**Increase `timeoutSeconds`** to prevent premature probe failure.

Pattern 2: Login Screen Not Launching (502 / 503 / 504 Errors)

This guide separates troubleshooting actions into **Internal Validation** (actions the support team can perform) and **External Validation** (checks required by the team managing Ingress/Load Balancer).

🛠️ Phase 1: Internal Validation (Support Team Action)

Objective: Confirm that the ARX application pods are running and successfully communicating internally, bypassing the external Ingress/Load Balancer layer.

Step 1: Pod Health and Connectivity Check

Validation Command	Location	Purpose	Expected Success (HTTP Status)
`kubectl get pods -l app=arx-sso` / `arx-idam`	Control Plane/Node	Check for Pod Status: **`Running`** and Ready: **`1/1`**.	N/A (Pod Status Check)
`curl -I http://<SSO_Service_Name>:<<Port>>/arx-sso/oauth/dologin`	**From ARX-IDAM pod**	Confirm **ARX-SSO Service** is alive and responding via its internal service name.	**200/302**
`curl -I http://<IDAM_Service_Name>:8080/arx-idam/user/v1/configuration/application`	**From ARX-SSO pod**	Confirm **ARX-IDAM Service** is alive and responding via its internal service name.	**200**

kubectl logs <arx-sso-pod-name>	Control Plane/Node	Check for application errors, startup failures, or dependency connection timeouts.	N/A (Log Review)

🛑If Internal Checks FAIL:

- **Action:** Continue debugging the application and Kubernetes resources (e.g., check configuration files, pod descriptions, service selectors, resource limits). The issue is internal to the application layer.

✅ If Internal Checks SUCCEED:

- **Action:** The application is confirmed healthy. The issue lies with the external access chain (Ingress/Load Balancer). **Escalate to the respective DevOps/Infrastructure team** with the confirmed internal validation results. Proceed to Phase 2 for suggested checks.

🚦 Phase 2: External Validation (Escalation/DevOps Required Validation)

If the application is confirmed to be healthy internally, the 502/503/504 error requires investigation of the Ingress Controller or Load Balancer.

The following checks should be suggested to the team responsible for managing the Ingress and Load Balancer:

Error Type	Required Validation Checks for DevOps/Infrastructure Team

502 Bad Gateway / 503 Service Unavailable	**1. Ingress Controller Logs: Review Ingress Controller logs (e.g., NGINX/ALB Ingress) for errors indicating connection failures, upstream connection refusals, or rejected requests to the ARX SSO Service endpoint. 2. Ingress/Service Mapping Validation: Check the Ingress resource (`kubectl describe ing <arx-ingress-name>`) and the Service definition (`kubectl describe svc arx-sso`). Ensure the Service Name and Service Port in the Ingress rule precisely match the Kubernetes Service definition. (Primary check for 502/503) 3. Service Endpoints (503): Verify that the ARX SSO Service has healthy endpoints by confirming the Endpoints list for `arx-sso` contains active Pod IPs.**
504 Gateway Timeout	**1. Ingress/Load Balancer Timeout Review: Review and confirm the backend connection timeout settings on both the external Load Balancer (if applicable) and the Ingress Controller (via annotations, e.g., `proxy-read-timeout`). Suggest increasing these values if the application startup or initial processing is confirmed to be slow. 2. External Network Check: Validate firewall rules or Security Groups are not abruptly terminating long-running connections between the Load Balancer and the Ingress Controller/Node.**

Pattern 3 : Login Page Display / Styling Issues

Issue/Error

The login page loads but appears **unstyled**, has a **broken layout**, or shows **CSS errors** (e.g., plain HTML text, missing background, missing images).

Analysis Steps

1. **Configuration Check:**

    - Validate the `DNS_URL_SSO` value in the ARX SSO configuration files (like `application.properties` or environment variables) is set to the **correct external URL** used by end-users.
2. **Client-Side Debugging:**
    - Instruct the user to open the **Browser Developer Tools (Network Tab)** (F12) and refresh the page. Look for requests to static resources (like files ending in `.css` or `.js`) that show a **failed status** (**404**, **Blocked**, or **Refused**).
    - Check the **Console Tab** for errors related to **Content Security Policy (CSP)** or **Mixed Content** (loading HTTP resources on an HTTPS page).

Root Cause

- **Blocked Resources:** Firewall, network policies, or security groups are preventing CSS/JS files from being downloaded by the client browser.
- **Configuration Error:** The `DNS_URL_SSO` value is misconfigured, causing the application to generate incorrect resource paths (URLs) in the HTML output.

Resolution

1. **Correct Configuration:** Update and confirm the `DNS_URL_SSO` (`https://arxauth.intellectproduct.com`) value is correct. **Redeploy** the service if necessary for the changes to take effect.
2. **Resolve Blocking:** If resources are blocked (per the Network tab analysis), **escalate to the appropriate team** (DevOps/Network) to **check and adjust firewall/network rules** to allow traffic to the static resource paths.
3. **Client Refresh:** Instruct the user to **clear their browser cache and perform a hard-refresh** (e.g., Ctrl+F5 or Cmd+Shift+R) after changes are implemented.

Pattern 4 : Login Issue : Broken or Delayed Scripts

This guide helps fix problems where code (scripts) on the page is changed or loads too slowly, causing website features to break.

Issue/Symptom

You see code (scripts) on the website with an **unusual label** (like type="rocket-loader-auto") or the code loads **much later than it should**, which stops website features from working correctly.

Analysis Steps

1. **Check the Code:** Look at the page's source code (using your browser's Developer Tools).
    - **Look for strange tags:** Inspect the main script tags (<script>). If you see the type label changed to something like **type="...-text/javascript"** (for example, type="rocket-loader-auto"), this indicates interference.
    - **Check Loading Time:** Note if the broken script is loading at the very end of the page, which suggests it's being delayed.
2. **Confirm Cloudflare:** Check if the website uses **Cloudflare** as a service provider.

Root Cause

- **Cloudflare's "Rocket Loader" Feature is ON:** This is a performance feature that tries to make your site faster by changing the script tags and delaying when the code runs. This change often breaks complex or time-sensitive application code.

Resolution – Disable Rocket Loader (DevOps Action Required)

This configuration change must be performed by the DevOps/Infrastructure Team as it requires access to the Cloudflare administrative portal.

Option 1: Cloudflare Dashboard (Global Disable)

1. The team logs into the **Cloudflare account** and selects the affected website.
2. They navigate to **Speed** $\rightarrow$ **Optimization**.
3. They find the setting called **Rocket Loader** and switch it **Off**.

Option 2: Page Rule (Fixes only certain pages)

1. In Cloudflare, the team goes to **Rules** $\rightarrow$ **Page Rules**.
2. They **create a rule** for the specific page URL that is having issues.
3. They set the action for that rule to **Rocket Loader: Off**.

Final Check

After the DevOps team confirms the change is saved in Cloudflare, refresh the website page:

- **Confirm:** The script tags should look normal again (without the modified type label).

- **Confirm:** The website features should load and run in the correct order.

Pattern 5: ARXWeb Blank Page After Deployment

Issue/Error

The **ARXWeb UI loads a completely blank page** after a seemingly successful deployment.

Analysis Steps

1. **Check Configuration Flags:** Review the following application configuration files for incorrect or conflicting flag settings:
    - Validate the **`ENCRYPT_SERVER_CALLS`** flag in `Arxwebsystempreferences.properties`. This flag should be **N** for non ssl environment.
    - Validate the **`CHECK_CLIENT_IP_FOR_SESSION_VALIDATION`** flag in `Arxwebsecurityconfig.properties`. This flag should be set to **N.**
2. **Review Application Logs:** Check logs across the integrated services for potential exceptions that occurred during startup or initial requests:
    - Examine **ARX-MS logs**.
    - Examine **ARX-SSO logs**.
    - Examine **ARXWeb logs**.

Root Cause

Common causes are often related to security misconfiguration (flags preventing communication) or stale session data preventing the application from initializing the UI state.

Resolution

1. **Configuration Correction:** If the flags validated in the Analysis Step are found to be incorrect, **correct the flag values** and redeploy the ARXWeb application.

2. **Clear Stale Data & Restart:** If no clear configuration or logging errors are found, the issue may be caused by old, invalid session/token data. Perform the following database actions and restart:

Execute the database commands to clear session and token tables:
SQL
TRUNCATE TABLE TB_ARM_SESSION;

TRUNCATE TABLE IMS_USER_OAUTH_ACCESSTOKEN;

    - **Restart the ARXWeb pod** after truncating the tables.

Pattern 6: SSL Handshake Failure / 404 on Token Endpoint

Issue/Error

You encounter an **SSLHandshakeException** or a **404 Not Found** error when calling the /token endpoint, while the initial authorization step, /authorize, works correctly.

Analysis Steps

1. **Identify the Call Type:** Confirm the two endpoints behave differently:
    - /authorize is a **browser redirect** (client → Load Balancer → ARX-SSO service).
    - /token is a **server-to-server call** (APPICATION service → ARX-SSO service).
2. **Check Internal Network Path:** Verify the network path and protocols used for the server-to-server call to the /token endpoint. Look for:
    - Attempts to use HTTPS internally (which may fail if certificates aren't available).
    - Whether the Load Balancer/Ingress is routing this *internal* traffic incorrectly.

Root Cause

- The **Load Balancer or Ingress Controller** is configured to handle external client traffic but **does not permit or correctly route internal server-to-server backend calls** when routing by URL, especially if it attempts to enforce SSL on the internal call.

Resolution

The fix involves **separating the configuration** for the two different types of traffic (browser vs. server) to bypass the load balancer for the internal call:

1. **Use Two Separate Configurations:** Configure the ARX application to use different URLs for the two endpoints:
    - **/authorize endpoint:** Continue using the **External Load Balancer URL** (the HTTPS URL exposed to the client).
    - **/token endpoint:** Change the configuration to use the **Internal Service Name** (e.g., http://arx-sso:8080/arx-sso/oauth/token) or the **IP-based HTTP URL** for direct server-to-server communication.
2. **Validation:** After applying the configuration change, ensure the ARX services are restarted to load the new settings, and test the full OAuth flow.

Pattern 7: "Change Entity" on Dashboard Not Working

Issue/Error

The **"Change Entity" feature on the dashboard fails** when ARX services are deployed across different hosts (e.g., in an OpenShift environment) instead of sharing the same host with only differing context roots.

Analysis Steps

1. **Check Environment:** Confirm that ARX services (ARX-LAUNCHER, ARX-SSO, etc.) are deployed on **different hostnames**.
2. **Browser Console Check:** Use browser Developer Tools (Console tab) while attempting the "Change Entity" action. Look specifically for **Content Security Policy (CSP) violation errors** that mention blocking a connection to the SSO host.

Root Cause / Possible Cause

- **Content Security Policy (CSP) Restriction:** The application's security configuration (CSP) is **not updated** to explicitly allow the ARX-LAUNCHER application to make

connections (`connect-src`) to the specific, separate hostname of the ARX-SSO service for the current environment.
- **Browser Blocking:** The browser adheres to the strict CSP defined in the application, blocking cross-domain AJAX requests required for the entity change functionality.

Resolution (Configuration Update)

This resolution requires a database update and service restart.

1. **Identify SSO Host:** Determine the **full, correct SSO host URL** for your environment (e.g., `http://arxsso-test.intellectproduct.com`).

Update Database: Execute the following SQL query against the database that holds the ARX configuration to update the `Content-Security-Policy` header in the `tb_arm_app_config` table:

SQL

UPDATE tb_arm_app_config

SET cnfg_value = '{

"Content-Security-Policy":"default-src "self" "unsafe-eval" "unsafe-inline"; style-src "self" "unsafe-inline"; connect-src "self" http://**<SSO_HOST_URL>**; object-src "none"; child-src "self"; frame-ancestors "self"; block-all-mixed-content",

"X-XSS-Protection":"0",

"X-Content-Type-Options":"nosniff",

"X-Frame-Options":"SAMEORIGIN",

...

"Clear-Site-Data":"cache, cookies, storage"

}'

WHERE cnfg_name = 'Security_Headers';

2. ***Ensure you replace `http://<SSO_HOST_URL>` with the correct URL identified in Step 1.***
3. **Restart Services:** Save and commit the database change, then **restart the relevant ARX services** to load the new configuration.

4. **Final Validation: Refresh the dashboard** and test the "Change Entity" functionality. Users may need to **clear their browser cache** for the new CSP rules to be fully enforced.

Pattern 8 : ARXWeb Deployment Issues (JAR Conflict)

This guide addresses specific deployment failures characterized by NoSuchMethodError, cryptographic provider authentication issues, and IllegalStateException.

1. NoSuchMethodError: Base64.encodeBase64String

Issue	**NoSuchMethodError: Base64.encodeBase64String**
Root Cause	Conflict between different versions of the **Apache Commons Codec** library, where an older version is being loaded.
Resolution	**Remove the conflicting JAR file:** Locate and delete commons-codec_1.3.0.v201101211617.jar from the deployment path (e.g., the application server's library folder or the deployment artifact).

---

2. JCE cannot authenticate the provider BC

Issue	**JCE cannot authenticate the provider BC**
Root Cause	Conflict with or incorrect declaration of the Bouncy Castle security provider. The runtime environment is attempting to load two versions or cannot verify the integrity of the provided JAR.
Resolution	**Remove and Modularize:** Locate and **remove** the external JAR file bcprov-ext-jdk15on-150.jar. Subsequently, **declare it as a module** within the

	application server's configuration (e.g., JBoss/Wildfly modules) to ensure proper, non-conflicting loading.

---

3. IllegalStateException: Timer already cancelled

Issue	**IllegalStateException: Timer already cancelled**
Root Cause	Often relates to internal timing or thread management errors within the application server environment (e.g., JBoss/Wildfly), potentially caused by restricted temporary file access.
Resolution	**Escalate to IT/Infrastructure Team** with the following suggestion: Grant **execute permissions** (chmod +x) to the /tmp directory (or the application server's specific temporary directory, e.g., /tmp in JBoss/Wildfly) where the application is running.

Pattern 9 : Web Admin application not loading with 404 error

🛑 Issue Description

When users launch the **web application** from the **ARX launcher** dashboard, the web application is not loaded and returns a 404 error.
This behavior breaks the user to use arx web application and users cannot use any features of user journey with the ARX application.

Possible cause

This issue is likely to be observed when there is an SSL offloading, the process of transferring the heavy, CPU-intensive task of encrypting and decrypting SSL/TLS traffic from web servers to a separate device, like a load balancer or a cloud-based proxy.

🔎 Root Cause Analysis (RCA)

The issue has been traced to an incorrect redirection url with respect to the protocol(http/https) mismatch configured for WEB_URL.

WEB_URL (Primary Root Cause)

Finding:
Internal URLs are currently being redirected to HTTP instead of HTTPS, since SSL termination is happening at the ingress level.

Impact:
Due to the incorrect protocol, the url becomes inaccessible and thereby the user is not able to launch the web application and getting 404 - Not found.

✅ Mandatory Permanent Solution (Application Teams Action)

This needs necessary changes to externalize the configuration, allowing the full URL to be specified for internal redirect resources.

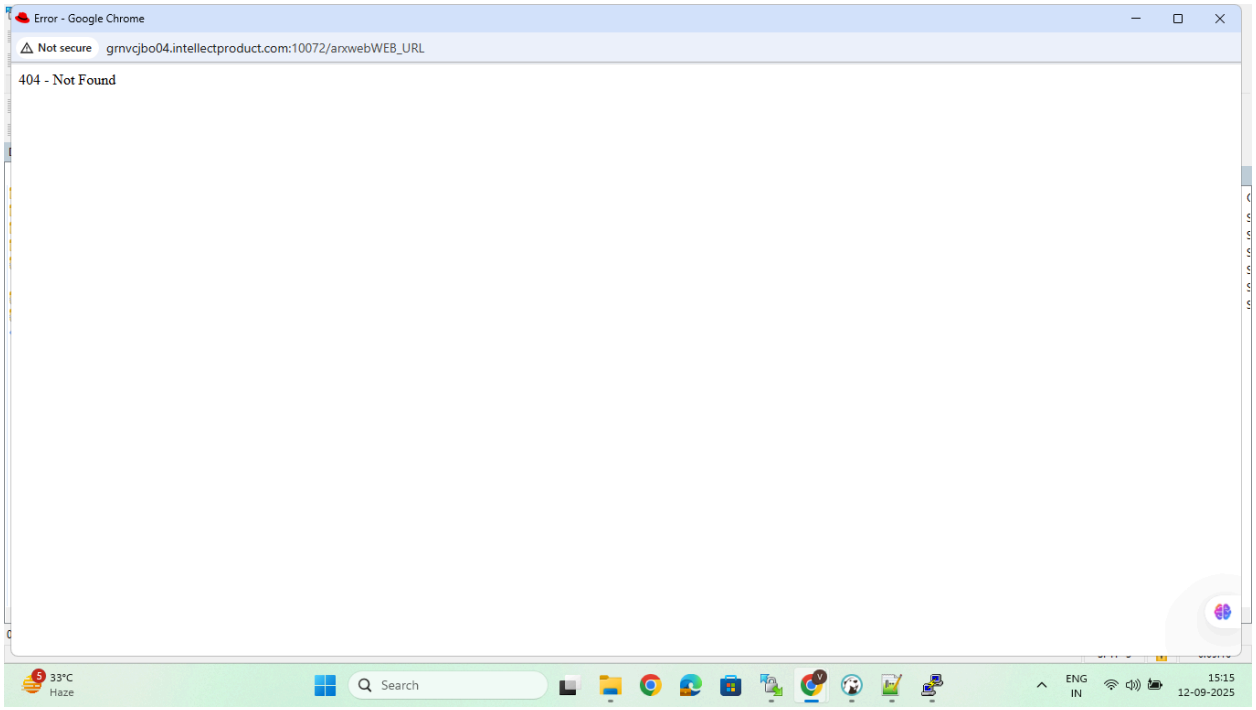
Step	Action	Rationale (Intended Outcome)
1	**Configuration changes:** WEB_URL=http://<<DNS>>/arxweb/home (Please modify the DNS and protocol based on the environment) **Sample url:** WEB_URL= http://grnkvcjbots0432.intellectproduct.com:10072/arxweb/home	This allows ARX to redirect to the configured externalized WEB_URL and load the arx web application without any issues.

Need to add the WEB_URL key in the below properties file to externalize the value against HOME_PAGE_URL which will be loaded from Project Environment details.

The actual value of WEB_URL will be configured in console system properties for monolithic servers and can be placed in yaml files for cloud servers.

arxwebapp/src/main/resources/Arxwebsecurityconfig.properties
HOME_PAGE_URL=${env.WEB_URL}

Sample error screen while launching web application:



Pattern 10: ARX SSO login screen not loaded properly (styles not loaded) below ARX 20

🛑 Issue Description

Users are not able to load/view the ARX SSO login screen properly. This is due to a failure in loading the CSS which is blocked due to the restrictions in the content security policy. This is applicable for the ARX product versions below 20.

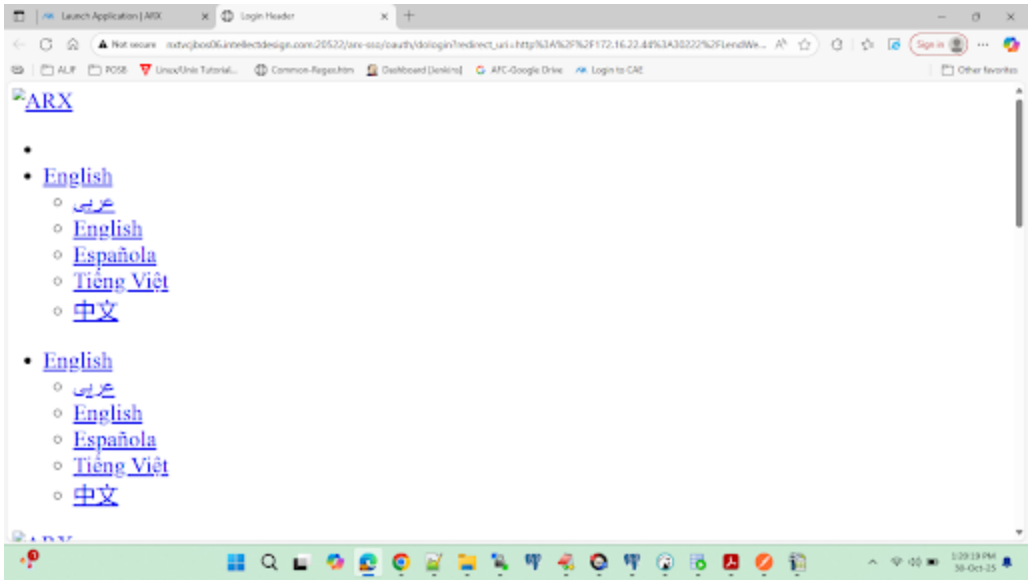
🔎 Root Cause Analysis (RCA)

This loading issue can be attributed to two primary, independent causes:

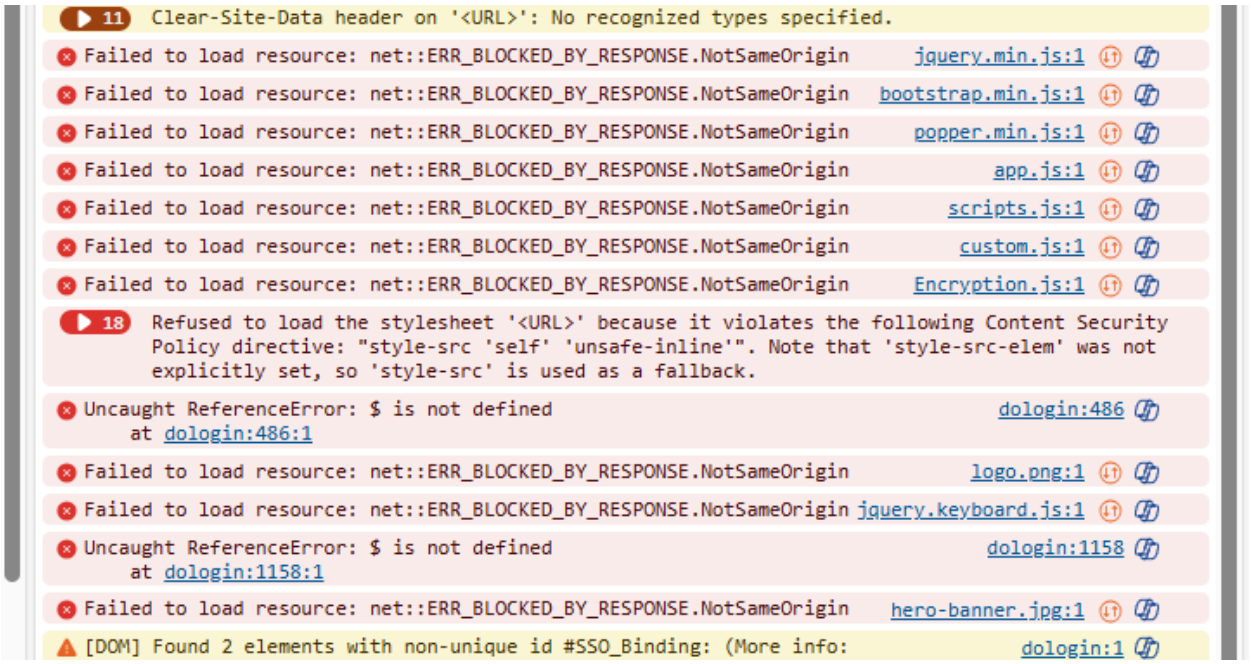
1. Missing host name in Content security policy (Secondary/Environmental)

- **Finding:** The affected application's CSP (Content-Security-Policy) does not have the host or IP address included as a safe and trusted source.
- **Impact:** ARX application will consider the request as unsafe and originated from an untrusted source and thereby styles and content were not loaded properly.

Sample ARX login screen during this issue:


Launch Application | ARX
Login Header
Not secure
ALIF
POSB
Linux/Unix Tutorial...
Common-Regex.htm
Dashboard [Jenkins]
AFC-Google Drive
Login to CAE
Other favorites
ARX
English
عربى
English
Española
Tiếng Việt
中文
English
عربى
English
Española
Tiếng Việt
中文

Sample errors from console:



✅ Mandatory Permanent Solution

The following changes are **mandatory** for ARX applications and must be implemented to resolve the content loading issue.

Step	Action	Rationale (Intended Outcome)
2.	Update the Content-Security-Policy in Security_Headers using the below query. Query is given as sample to include the hostname in unsafe-iline and for loading styles, this can be modified with the existing Content security policy to include the hostname as trusted source and rest all contents can be kept as is. **UPDATE tb_arm_app_config SET CNFG_VALUE = '{"Content-Security-Policy":"default-src "self"; connect-src "self"; style-src "self"**	Enables ARX to trust the IP address/hostname as trusted source and allows to load the style and contents properly.

	"unsafe-inline" http://localhost.intellectproduct.com; style-src-elem "self" http://localhost.intellectproduct.com; object-src "none"; child-src "self"; frame-src "self"; script-src "self"; frame-ancestors "self"; img-src "self"; base-uri "self"; block-all-mixed-content","X-Content-Type-Options":"nosniff","X-Frame-Options":"SAMEORIGIN"}' WHERE CNFG_NAME = 'Security_Headers'; **COMMIT;**	

🛠️ Additional Technical Checks & Temporary Workaround

Category	Action	Note
Console checks	Inspect application elements console of ARX login screen.	Use browser DevTools (Console tab) to verify if there are any errors for css.

Pattern11 : Liquibase execution errors in incremental

🛑 Issue Description

Users are getting the exceptions/errors while executing the incremental liquibase scripts with ValidationFailedException in the log trace.

🔎 Root Cause Analysis (RCA)

The exception occurs due to the below primary cause:

1. Changes in any scripts/files in the existing changeset which was executed already

- **Finding:** If there are any changes in any of the script or addition/deletion of any of the files in the changeset which was already executed in the specific environment.
- **Impact:** Whenever any incremental liquibase scripts are executed, it will validate the MD5SUM value stored in the DATABASECHANGELOG table along with the actual scripts and folders present in the changeset. If there are any changes/mismatches between these values, incremental liquibase execution will return ValidationFailedException.

Checksum Generation

- **What it hashes:** For each changeset, Liquibase computes an MD5 hash based on key attributes and content, including:
    - The changeset's id, author, and changes (e.g., SQL statements, table alterations).
    - Excludes metadata like comments or whitespace that don't affect execution.
- **When it's generated:** The hash is calculated dynamically each time Liquibase processes the changelog. It's not a direct hash of the file/folder itself but of the parsed changeset structure.
- **Storage:** Upon successful execution, the checksum is stored in the DATABASECHANGELOG table (a Liquibase-managed table in your database) alongside the changeset's metadata.

How Liquibase Checksums Work

Liquibase uses MD5 checksums to ensure the integrity of changesets (individual migration units within a changelog). This prevents accidental or malicious modifications to already-executed changesets, which could lead to inconsistent database states. The process involves generating, storing, and validating these checksums against the parsed content of changeset definitions in your changelog files.

This validation applies to individual changesets, not entire folders—Liquibase doesn't checksum whole directories. Instead, it ensures each changeset's integrity by comparing its parsed representation against the database record.

✅ Mandatory Permanent Solution

The following is **mandatory** for ARX incremental liquibase execution to resolve the issue permanently.

Step	Action	Rationale (Intended Outcome)

1.	**Revert the changes:** Previously executed changeset should not be modified and kept as is, any additional changes should be added as a new changeset and changelog.yml should be modified accordingly.	This prevents the md5sum check validation error and allows the user to complete the incremental liquibase execution successfully provided no other syntax or sql errors.

🛠️ Additional Technical Checks & Temporary Workaround

Category	Action	Note
Revert changes	Revert the changes done in existing changeset and make sure it is in sync with the previously executed changeset folders and scripts	Ensures the incremental liquibase script execution is completed without any errors in validation failure for existing changeset.
Update MD5SUM value	Update the **MD5SUM** in the **DATABASECHANGELOG** table with the new value as returned in the error log. This method can be followed for specific situations where we are unable to revert the previous changesets and need modifications in the same to resolve any issues.	Ensures the incremental liquibase script execution is completed without any errors in validation failure for existing changeset.

Sample error logs from liquibase execution:

[ERROR] Error setting up or running Liquibase:
[ERROR] liquibase.exception.ValidationFailedException: Validation Failed:
[ERROR] 2 changesets check sum
[ERROR] changelog.yml::ARX1.7::ARX was: 9:7e70ab0838423c6d63f2d178f3117d33 but is now: 9:57690c0d64f70869bb2a93a3cadea4f9
[ERROR] changelog.yml::ARX1.12::ARX was: 9:9cdd690ae46477e63ee34c06a22cacc2 but is now: 9:77c44d0a1fc80956a41df6294dfe9227

In the above scenario for changeset ARX1.7 and ARX1.12 we have faced a mismatch in md5sum value as there are some changes available in the respective changeset compared to the previously executed changeset folders.

As per permanent solution, we need to revert the changes and sync the changeset as the previous one.

As per additional temporary workaround, please update the md5sum value in DATABASECHANGELOG table using below query with the new value from logs.

Sample query:

```
Update DATABASECHANGELOG set MD5SUM='57690c0d64f70869bb2a93a3cadea4f9' where ID='ARX1.7' and MD5SUM='9:7e70ab0838423c6d63f2d178f3117d33';
commit;
```

Pattern 12: Setting SameSite and Secure Attributes for ARX Cookies

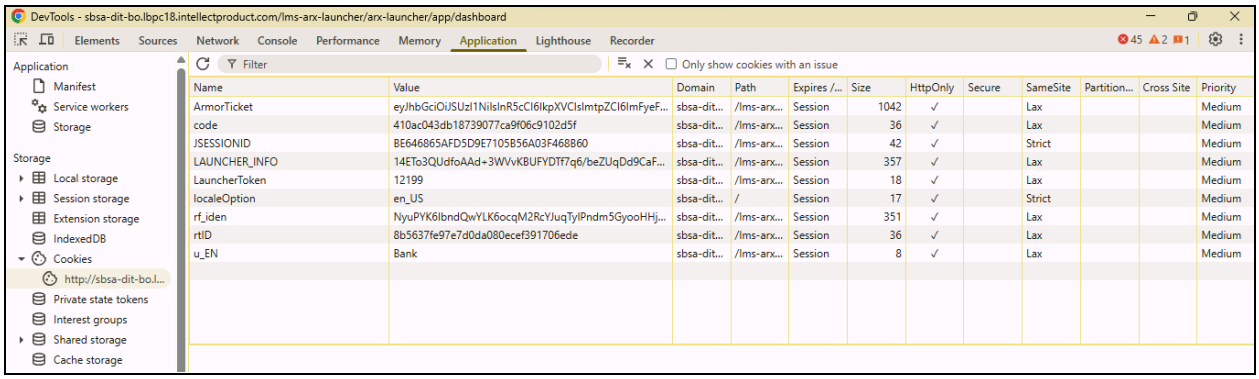
Description

When sharing an application for **Vulnerability Assessment and Penetration Testing (VAPT)**, a common security issue reported is the need to enable the **SameSite** value as **Strict** and to enable the **Secure** cookie attribute for ARX Cookies. This configuration helps mitigate risks like Cross-Site Request Forgery (CSRF) and ensures cookies are only transmitted over secure (HTTPS) connections.

Current Cookie Configuration

The attached screenshot from the browser's Developer Tools shows the current configuration of the cookies. As highlighted in the image, many cookies, including key ARX cookies, are missing the recommended security attributes:

- The **SameSite** column for most cookies is set to **Lax** (or is effectively missing the Strict attribute).
- The **Secure** column is **empty**, indicating the secure attribute is not set.



Solution

To resolve the reported security vulnerability, the cookie attributes must be updated in the application's configuration database. This requires executing specific SQL update queries.

The solution involves setting the **SameSite** attribute to **Strict** and enabling the **Secure** flag for relevant cookies.

Configuration Name	New Value	Purpose	SQL Query
SameSite_Key	**Strict**	Enforces that cookies are only sent with requests originating from the site that set the cookie, helping to prevent CSRF.	Update tb_arm_app_config set cnfg_value ='Strict' where cnfg_name ='SameSite_Key';
TICKET_SECURE_FLAG	**TRUE**	Ensures the cookie is only sent to the server over an encrypted	Update tb_arm_app_config set cnfg_value ='TRUE' where cnfg_name ='TICKET_SECURE_FLAG';

		HTTPS connection.	

Note :

These changes require the application to be restarted for the new configuration values to be loaded and applied to the ARX cookies.

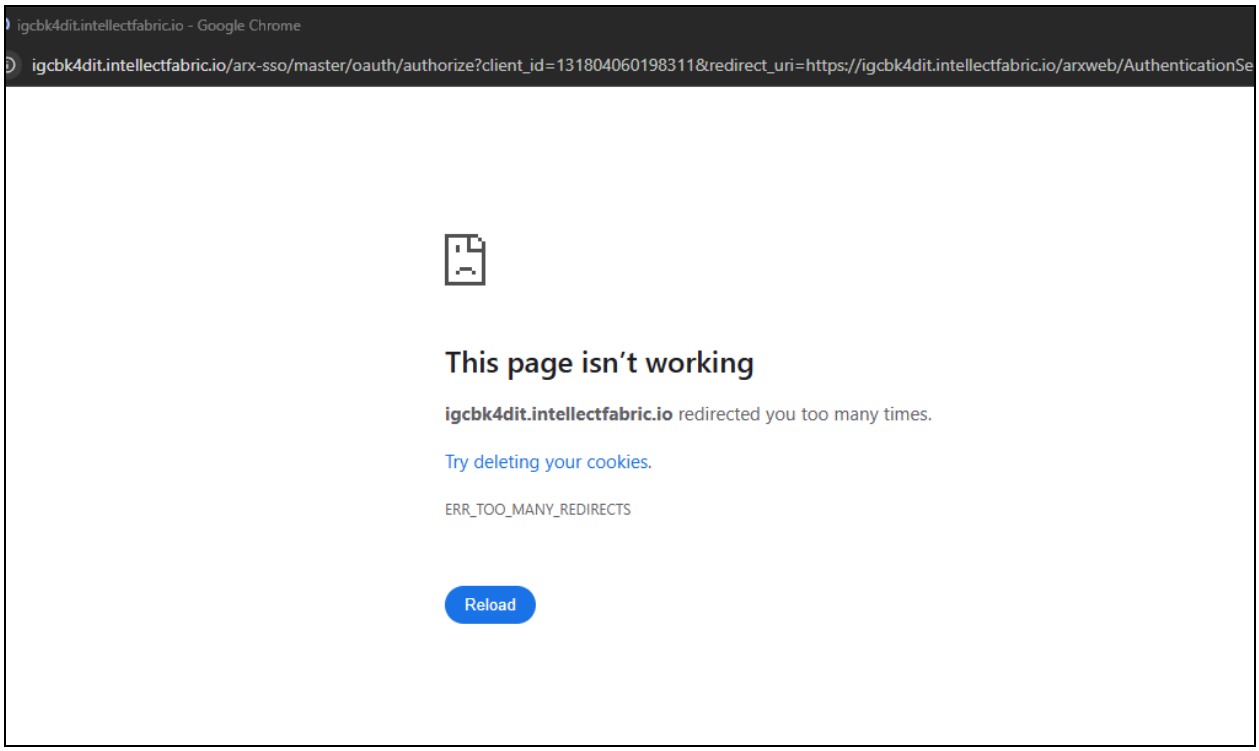
Pattern 13: Arxweb Application Access Issue: ERR_TOO_MANY_REDIRECTS

Description

When attempting to access the application, the system enters an infinite redirection loop, resulting in a blank page error stating **"This page isn't working"** with the message: **"[application URL] redirected you too many times."** The underlying browser error is typically **ERR_TOO_MANY_REDIRECTS**.

Observed Issue

The attached screenshot illustrates the error page displayed when the client attempts to access the application. The URL indicates a failed attempt at the OAuthcode endpoint , often involving a continuous loop between the application (e.g., igcbk4dit.intellectfabric.io/arx-sso/...).



Root Cause Analysis

The primary root cause for this redirection loop issue is an **incorrect configuration** of the client credentials for the arxweb application. Specifically, the client application cannot successfully authenticate the application because the shared secrets or IDs do not match.

- **The issue comes when the client ID and client secret of arxweb are maintained incorrectly.**

Solution and Remediation Steps

The solution involves verifying and correcting the client credentials within the arxweb application's yaml file.

1. **Verify Key Values:** Check and confirm that the values for the following two keys are **correctly set** and **match** the configuration registered on your Identity Provider (SSO) server:

Key Name in arxweb YAML	Description	Action

CLIENTID_WEB	The unique identifier for the arxweb application registered with the SSO server.	**Verify** the value is accurate.
CLIENTSECRET_WEB	The shared secret key used by arxweb to securely authenticate with the SSO server.	**Verify** the value is accurate.

Crucially, ensure there are no leading or trailing spaces

2. **Check Client ID and Secret Stored in the Database**

   To check the client Id and Secret in the Database , use the below query :

   Query - Select * from ims_oauth_registration;

   Note : The client secret stored in the ims_oauth_registration table is in **encrypted** format.

   To get the unencrypted, actual client secret value, you must use the ARXUtility to decrypt the secret value retrieved from the table.

Pattern 14: Report Generation Failure and Performance Degradation

Issue Description

Users are unable to generate the **Report** within the **ARX application**.
When triggered, the report generation process remains in a **pending state indefinitely** and never completes successfully.

This issue typically arises due to one or more of the following factors:

- Database performance degradation
- Data inconsistency within the ARX schema
- Incorrect connection configuration (e.g., ARMCONNPOOL JNDI issues)

- Insufficient memory allocation to the ARX Web Pod

Error Details (Observed Case)

Error Message:

Caused by: java.lang.RuntimeException: Error parsing XPath /qMapConfig/transactionManager/dataSource/end)'.
Cause: com.ibatis.sqlmap.client.SqlMapException: There was an error configuring JndiDataSource TransactionPool.
Cause: javax.naming.NameNotFoundException: ARMCONNPOOL - service jboss.naming.context.java.ARMCONNPOOL

Potential Root Cause (from error trace):
The issue occurs when the **ARMCONNPOOL JNDI resource** is not properly defined or configured in the **database connection definition**.
As a result, the application fails to establish a valid datasource connection due to missing or incorrect JNDI mapping in the ARX schema.

🔎 Root Cause Analysis (RCA)

S.No	Root Cause	Impact	Resolution Summary
1	Invalid data entries in `Report view in arx schema`	Causes query execution failure	Validate and correct data in the ARX schema
2	High query cost due to large dataset	Report remains in pending state indefinitely	Apply filters and optimize SQL query
3	Insufficient memory allocation to ARX Web Pod	Pod restarts during heavy report processing	Allocate ≥ 4 GB memory for large user bases
4	Misconfigured JNDI connection (`ARMCONNPOOL`)	Application fails to connect to datasource	Correct JNDI mapping in `CONNECTION_DEFINITION` table and restart ARX server

5	Unoptimized SQL execution plan	Report generation extremely slow	Review execution plan and add indexes to large tables if required

---

✅ Permanent Resolution Steps

Step 1: Validate Canvas (CT) Logs

- Review Canvas (CT) logs for runtime exceptions related to:
    - SQL errors
    - Data source configuration
    - Transaction pool initialization
- Pay special attention to logs referencing **ARMCONNPOOL** or JNDI binding failures.

Step 2: Verify Database View Integrity
Execute the following query in SQL Developer:

```
SELECT * FROM <<REPORT VIEW>>;
```

- **Validation:**
    - If execution fails → inspect and fix invalid or corrupt data in the ARX schema.
    - If execution succeeds → proceed to performance and follow step 4.

Step 3: Validate and Correct ARMCONNPOOL JNDI Configuration

If logs contain errors related to **ARMCONNPOOL**, perform the following:

Validate current configuration

```
SELECT * FROM CONNECTION_DEFINITION WHERE CONNECTION_ID =
'ARMCONNPOOL';
```

1. **Update JNDI mapping based on application server:**

For JBoss:

```
 UPDATE CONNECTION_DEFINITION
SET CONNECTION_UNDIE = 'java:jboss/datasources/ARMCONNPOOL'
WHERE CONNECTION_ID = 'ARMCONNPOOL';
```

For WebLogic / WebSphere / Cloud:

```
 UPDATE CONNECTION_DEFINITION
SET CONNECTION_UNDIE = 'java:comp/env/jdbc/ARMCONNPOOL'
WHERE CONNECTION_ID = 'ARMCONNPOOL';
```

2. **Restart the ARX server** after updating configurations.

Step 4: Analyze Query Performance (Explain Plan)

- Use **Explain Plan** in SQL Developer to check the execution cost of the `report` query.
- If the **cost > 30,000**, optimize the query by:
    - Applying filters (e.g., date range, user group)
    - Reducing dataset size
    - Adding necessary indexes on frequently accessed columns

---

Step 5: Monitor ARX Web Pod Stability

- Check if the ARX Web Pod restarts during or after report generation.
- Frequent restarts indicate **memory exhaustion** or **resource misconfiguration**.

Step 6: Verify and Tune Pod Memory Allocation

- Review current resource configuration for the ARX Web Pod.
- If resource exhaustion persists:
    - Increase memory allocation to **at least 4 GB** for user **data bases > 100,000.**
    - Ensure adequate **CPU and heap memory** resources.
      For your reference :- CPU 300M
- Restart the Pod and validate report generation post-tuning.

🧩 Resolution Summary

- Validate and correct invalid data in the ARX schema.
- Optimize the SQL query and apply filters to minimize execution time.
- Correct **ARMCONNPOOL JNDI** configuration in `CONNECTION_DEFINITION`.
- Increase ARX Web Pod memory allocation to at least **4 GB**.
- Restart ARX services after configuration changes and validate the fix.

Pattern 15: Add User API – Token Deletion Issue

Issue Description:
When invoking the **Add User API**, the authentication token is unexpectedly deleted from the **TB_ARM_SESSION** table.

Exception in logs:

java.lang.Exception: Exception while validating JWT
at
com.intellectdesign.arx.common.dao.ARXCommonServicesRDBMSImpl.getTokenDetails(ARXCommonServicesRDBMSImpl.java:5377) ~[!/:3.5.3]

at jdk.internal.reflect.GeneratedMethodAccessor58.invoke(Unknown Source) ~[?:?]

Root Cause Analysis (RCA):

The issue occurs when there is a time mismatch between the application server and the database server.

If the system time and database time are not synchronized, session tokens can appear expired or invalid, causing them to be removed from the `TB_ARM_SESSION` table.

Solution:

1. Verify Time Synchronization:

Run the following query in the database to check timestamps:

Query:

SELECT SYSDATE, SYSTIMESTAMP, CURRENT_TIMESTAMP FROM dual;

Ensure both times (DB and server) are synchronized.

When we click on the **SYSTIMESTAMP** column value, a pop-up appears showing the exact time. We can compare it with the **CURRENT_TIMESTAMP** value to check whether both columns are correctly aligned.

2. Post-Validation:

Once time synchronization is confirmed, follow these steps:

-> Re-execute the Add User API after syncing the system and DB times.

-> Validate Token Persistence: Ensure the session token remains valid, and no deletion occurs from the **TB_ARM_SESSION** table.

Pattern 16: 500 Internal Server Error When Launching ARX Web Application

Issue Description:
Users encounter a 500 Internal Server Error when attempting to launch the ARX Web Application using the SYSADMIN credentials. The issue occurs immediately after clicking the web app link, and the application fails to load.

Exception in Logs:
2025-11-17 13:17:55,175 | FATAL | [http-nio-8080-exec-2] | com.intellectdesign.arx.servlets.login.ARXAuthenticationProvider:extractUserCredentials(283) | java.lang.Exception: **Unable to retrieve Access Token due to Invalid ClientId or ClientSecret or OAuthCode**

Poterntial Root Cause:

The issue occurs due to incorrect or mismatched configuration values ARX web. Specifically, inconsistencies in authentication-related configurations—such as `CLIENTID_WEB`, `CLIENTSECRET_WEB`, or certificate keys—can prevent successful application startup and result in a 500 error.

Solution:

1. Validate Configuration Values:
Review the ARX Web Application ConfigMaps and ensure the following keys contain correct and consistent.

WEB App Config Maps Keys	Description	Action
`ARX_CERT_KEY`	Unique identifier/key used to authenticate the *arxweb* app with the ARX system.	**Verify** the value is accurate.
`ARX_RSA_CERT`	RSA certificate or certificate identifier for *arxweb* registered with ARX.	**Verify** the value is accurate.

CLIENTID_WEB	Client ID of the *arxweb* application registered with ARX	Verify the Client ID is correct.
CLIENTSECRET_WEB	Client Secret associated with the above Client ID for authentication.	**Verify** the value is accurate.
DB_DRIVERCLASS	Defines the JDBC driver class used for DB connectivity. **oracle.jdbc.OracleDriver**	Ensure the Driver Class name is accurate.
TXN_JNDI	JNDI name of the application server's Transaction Manager. Required for managing JDBC transactions. Example: **`java:comp/UserTransaction`.**	Ensure the Tnx JNDI name is accurate.
JNDI_NAME	JNDI name for the database DataSource. **`java:comp/env/jdbc/ARMCONNPOOL`.**	Ensure the JNDI name is accurate.

Crucially, ensure there are no leading or trailing spaces

2. Verify OAuth Registration Details:

To check the client Id and Secret in the Database , use the below query :

Query - Select * from ims_oauth_registration;

Note : The client secret stored in the ims_oauth_registration table is in **encrypted** format.

To get the unencrypted, actual client secret value, you must use the ARXUtility to decrypt the secret value retrieved from the table.

3. Restart and Validate

After correcting any mismatched or incorrect values:

1. Restart the ARX pod or service.
2. Reattempt launching the ARX web application.
3. Confirm the application loads successfully without a 500 error.

Pattern 17 : ARX 24 arxweb context conflict issue

🛑 Issue Description

When same arxweb context is used for multiple ARX versions which are installed in the same **namespace** causing conflict in the context rendering the arxweb inaccessible

🔎 Root Cause Analysis (RCA)

The issue has been traced to multiple ARX versions installed in the same **namespace** causing conflict in the arxweb context rendering the application inaccessible.

✅ Mandatory Permanent Solution (Application Teams Action)

All integrated application teams must ensure that unique arxweb context is maintained in helm charts throughout all the modules of ARX in each version to avoid context conflicts in arxweb. Update the APPLICATION_MASTER table with required arxweb context in ARX 24.

Example:

```
UPDATE APPLICATION_MASTER SET DOMAIN_URL = 'arxweb24' WHERE APPLICATION_ID IN ('ARX', 'master');

UPDATE APPLICATION_MASTER SET DOMAIN_URL = 'arxwebtenant1' WHERE APPLICATION_ID IN ('ARX', 'tenant1');
```

Pattern 18 : JWT Creation Error – Application Logout Issue

🛑 Issue Description

Integrated applications are being redirected to their **logout pages immediately upon launch**. In the arx-idam logs, the most frequent error observed is:**"Error while creating JWT"**

This prevents users from maintaining valid sessions across integrated applications, resulting in forced logout behavior right after login.

🔎 Potential Root Cause

The issue has been traced to **invalid or missing token expiry configurations** in the **IMS_OAUTH_REGISTRATION** table. Specifically:

1. **Null Expiry Values for Tokens (Primary Root Cause)**
   Finding:
   The `ACCESS_TOKEN_EXPIRY`, `ID_TOKEN_EXPIRY`, and `REFRESH_TOKEN_EXPIRY` columns in the `IMS_OAUTH_REGISTRATION` table contain **null values** for affected applications.

   SQL Query to check the values:

   SELECT ACCESS_TOKEN_EXPIRY,ID_ACCESS_EXPIRY,REFRESH_TOKEN_EXPIRY FROM IMS_OAUTH_REGISTRATION WHERE APP_NAME ='<APP_NAME>';

   Impact:
   When ARX attempts to generate or validate JWTs, it fails to compute expiry timestamps, leading to the error *"Error while creating JWT"*.
   As a result, the authentication process fails, causing an immediate redirection to the application's logout page.

2. **Incorrect Script Execution During Registration (Secondary Cause) Finding:**
   Application teams often execute registration scripts where these expiry values are **not provided (set as null)**.
   Impact:
   Missing expiry configurations prevent ARX from constructing valid tokens for session continuity, breaking integration between ARX and client applications.

✅ Mandatory Permanent Solution (Application Teams Action)

All integrated application teams must ensure the **IMS_OAUTH_REGISTRATION** table is populated with valid token expiry values.

Step	Action	Rationale (Intended Outcome)

1	**Update Expiry Values:** Set valid expiry times (in minutes) for access, ID, and refresh tokens in the `IMS_OAUTH_REGISTRATION` table.	Ensures ARX can successfully create and validate JWTs for session management.
2	**Validate Registration Scripts:** Verify that all application integration scripts include non-null expiry parameters.	Prevents recurrence of null expiry configurations during future deployments.

Example:

```
UPDATE IMS_OAUTH_REGISTRATION
   SET ACCESS_TOKEN_EXPIRY = 15,
       ID_ACCESS_EXPIRY = 3600,
       REFRESH_TOKEN_EXPIRY = 4320
 WHERE APP_NAME = '<APP_NAME>';
```

Pattern 19: Integrated Applications Not Closing on ARX Logout

🛑 Issue Description

When users log out from **ARX**, the **integrated applications remain open** and are not automatically closed.
This behavior breaks the expected **Single Logout (SLO)** functionality, where logging out of ARX should automatically close all connected application windows.

🔎 Potential Root Cause

The issue has been traced to **incompatible browser response headers** configured on the integrated application side.

Cross-Origin-Opener-Policy Header Enabled (Primary Root Cause)

Finding:
The affected applications have the `Cross-Origin-Opener-Policy` header set in their HTTP response (e.g., `Cross-Origin-Opener-Policy: same-origin`).

Impact:
The problem originates from the `Cross-Origin-Opener-Policy` (COOP) response header that is configured in the integrated application's HTTP responses. This header is part of a modern browser security model that enforces "cross-origin isolation" to mitigate side-channel attacks (like Spectre) and prevent unintended data sharing between different origins (domains).

🛠️ Mandatory Technical Checks

Category	Action	Note
Header Configuration	Inspect response headers of integrated applications requests in the F12 DevTools Network Tab	**1. Open Integrated App:** Launch the integrated application through ARX. **2. Open Developer Tools:** Press **F12** → go to **Network** tab. **3. Select any Request:** Click the request. **5. Inspect Response Headers:** Under *Headers* → *Response Headers*, look for: `Cross-Origin-Opener-Policy` **6. Check Header Values:** If this header is configured `then` the application is isolated and ARX cannot close the window.

✅ Mandatory Permanent Solution (Application Teams Action)

All application teams must review and modify their response header configurations.

Step	Action	Rationale (Intended Outcome)
1	**Remove the `Cross-Origin-Opener-Policy` header** from the integrated application's HTTP response.	Allows ARX (parent window) to communicate with and close the child application windows during logout.

Pattern 20 : Dropdown Extra Attribute Values Not Loading in Add User Form

🛑 Issue Description

In the **Add User** form, the dropdown fields corresponding to **extra attributes** are not displaying any values.
This prevents users from selecting required attribute options (e.g., roles, departments, or custom configuration values), blocking completion of the user creation process.

🔎 Root Cause Analysis (RCA)

The issue arises due to **missing or inaccessible data** in the source tables used for populating dropdowns.

1. **Missing Data in IMS_EXTR_ATTR_VAL_DTL (Primary Root Cause)**
   Finding:
   The `IMS_EXTR_ATTR_VAL_DTL` table, which stores the valid value sets for dropdown attributes, does not contain records for the corresponding attribute IDs.
   Impact:
   When the application queries this table during the Add User form load, it receives an empty result set, leading to blank dropdowns.

2. **Missing Synonyms or Access Privileges (Secondary Root Cause)**
   Finding:
   In some environments, dropdown values are fetched from the application's own schema instead of ARX's schema.
   However, **synonyms** or **grants** to access these schema objects are missing.
   Impact:
   Without proper synonyms or privileges, ARX cannot query the referenced tables, causing dropdown value retrieval to fail.

🛠️ Mandatory Technical Checks

Category	Action	Note
Data Validation	Execute the below query to verify the presence of dropdown data: `SELECT * FROM IMS_EXTR_ATTR_VAL_DTL WHERE EXTR_ATTR_ID= '<Attribute_ID>';`	If the query returns no rows, the table is missing required values.
Synonym Check	Run: `SELECT * FROM ALL_SYNONYMS WHERE SYNONYM_NAME LIKE '%<TABLE_NAME>%';`	Confirms whether the necessary synonyms exist for cross-schema references.
Access Validation	Ensure the ARX DB user has access to the source tables: `SELECT * FROM ALL_TAB_PRIVS WHERE GRANTEE = 'ARX_SCHEMA';`	Missing grants can block data fetching for dropdowns.

✅ Mandatory Permanent Solution (Application Teams Action)

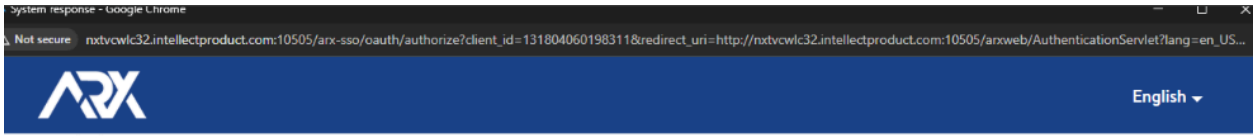
Application teams must ensure that all extra attribute value tables are properly populated and accessible to ARX.

Step	Action	Rationale (Intended Outcome)
2	**Create Required Synonyms** for application-specific tables used by ARX to fetch attribute values.	Allows ARX to query tables across schemas without direct cross-schema references.
3	**Grant Required Access Privileges** (e.g., `SELECT`) to the ARX database user for all dependent tables.	Prevents “table or view does not exist” errors and ensures consistent data access.

Pattern 21: Invalid Request, Redirect URL Error in ARXWeb Screen

Issue Description

While logging into the application and launching the ARXWeb module, the issue **"Invalid Request Redirect URL"** intermittently appears on the ARXWeb screen.This issue prevents the user from successfully launching the ARXWeb application after authentication. It occurs inconsistently, making it harder to trace without analyzing the backend configuration.



Invalid Request-RedirectURL

🔎 Root Cause Analysis (RCA)

Redirect URL mismatch

- **The ARXWeb application uses OAuth-based authentication, where the redirect URL (the callback URL after successful authentication) must exactly match the URL registered in the backend configuration tables.**
- **The system validates the redirect URL during the OAuth handshake. If even a small deviation (such as a missing slash, difference in case, or additional parameter) exists between the registered URL and the request URL, the validation fails — leading to the "Invalid Request Redirect URL" error.**

Inconsistency between configuration tables

- **The URLs maintained in the following tables were not consistent:**
  - `IMS_OAUTH_REGISTRATION`
  - `TB_ARM_WEB_APP`

- **The redirect_url in `IMS_OAUTH_REGISTRATION` and the web_app_url (or equivalent field) in `TB_ARM_WEB_APP` should be identical, including:**
    - **Exact spelling**
    - **Case sensitivity (e.g., `https://ARXWeb.intellect.com` ≠ `https://arxweb.intellect.com`)**
    - **Trailing slashes or spaces**

Environment-based differences

- **Sometimes, this issue appears intermittently due to:**
    - **Application load balancing (requests routed to different nodes with slightly different configurations)**
    - **Partial configuration updates during migration or deployment**
    - **Cached URLs in the authentication layer**

✅ Mandatory Permanent Solution (Application Teams Action)

1 Verify and align redirect URLs in both tables

- Execute queries to validate that both tables contain the exact same redirect URL:

  select * from IMS_OAUTH_REGISTRATION;

  select * from TB_ARM_WEB_APP;

2 Update inconsistent entries

Ensure:

- No extra spaces before/after the URL
- Correct use of HTTP/HTTPS
- Consistent upper/lower case

3.Restart or refresh the affected services

- Once the URLs are aligned, restart or refresh the related services (if applicable) to clear any cached entries.

4.Implement a configuration validation check

- As a preventive measure, add a **pre-deployment validation script** or **configuration consistency check** to verify URL consistency across these tables before each deployment or migration.

🛠️ Additional Technical Checks & Temporary Workaround

1.Check logs

- Review application or server logs to confirm which URL is being sent and which one is being validated.

2. Clear browser and application cache

- Cached tokens or sessions in the browser can sometimes retain older redirect URLs.
- Clear cookies/cache or use incognito mode to validate behavior.

3.Check for load balancer / proxy rewrite rules

- Ensure that the reverse proxy or load balancer is not changing the case, path, or protocol (HTTP ↔ HTTPS) of the redirect URL.If the issue is related to the proxy or load balancer, please coordinate with the Infra or DevOps team for further support.

4. Temporary Workaround

- As an immediate workaround, correct the redirect URL directly in the `IMS_OAUTH_REGISTRATION` table to match the one currently being used in the browser request.

  UPDATE IMS_OAUTH_REGISTRATION SET REDIRECT_URL ='http://iappsun05.intellectdesign.com:10071/ICUSTODY/MenuController?Event=GetMenu&ApplicationName=ICUSTODY' WHERE APP_NAME = 'ICUSTODY';

Pattern 22 : Logout Endpoint Failure Due to CSP Restrictions

🛑 Issue Description

When users log out from the **Launcher**, the logout endpoint call fails and does not complete successfully.
In **F12 Developer Tools**, a **CSP (Content Security Policy) error** appears, showing that the request is blocked due to cross-domain restrictions.

This issue occurs mainly in **cloud environments** where each service (Launcher, SSO, ARX) operates on a **different domain**.

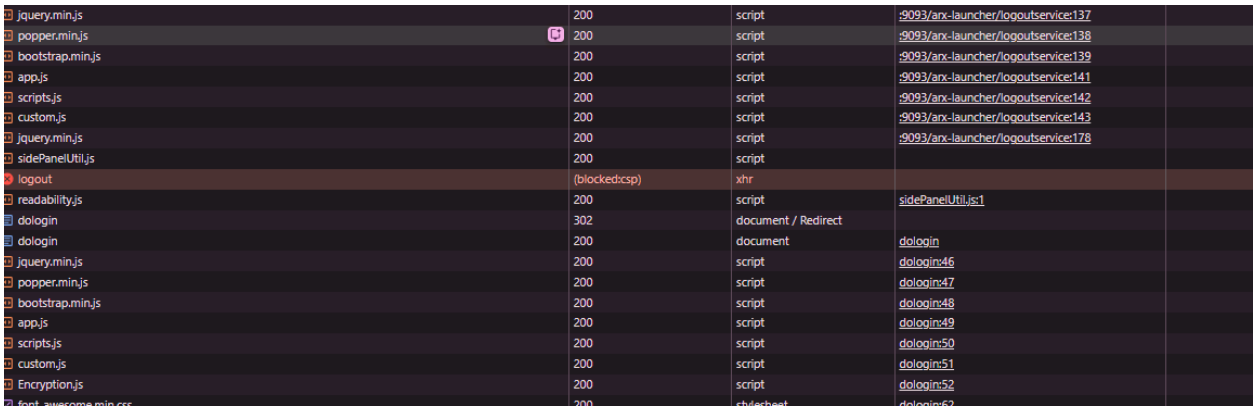
🔎 Root Cause Analysis (RCA)

The failure is caused by restrictive browser security enforced through the **Content Security Policy (CSP)** headers configured in the ARX application.

Primary Root Cause — Missing Domains in `connect-src` CSP Directive

Finding:
The `connect-src` directive in the security headers does **not** include the SSO domain.
Example of incorrect header:

```
Content-Security-Policy: connect-src 'self';
```



Impact:
Because Launcher and SSO run on **different domains**, browsers block cross-domain network calls that are not explicitly allowed.
This results in:

- Logout API request being **blocked**
- CSP error in Console
- Logout flow **not completing**, leaving the SSO session active

Browsers strictly enforce CSP rules for:

- fetch / XHR calls
- token / logout calls
- AJAX requests
- background API calls

✅ Mandatory Permanent Solution (Configuration Update Required)

All deployments must update the **security headers** to allow cross-domain communication between Launcher and SSO.

Step-by-Step Fix

Step	Action	Rationale (Intended Outcome)
1	Update `connect-src` in **TB_ARM_APP_CONFIG_TABLE** under *security-headers*.	Ensures browser is allowed to call SSO logout API.

2	Add both **Launcher** and **SSO** domain URLs to the directive.	Allows required cross-domain communication.
3	Restart or redeploy the service.	Applies the updated CSP configuration.
4	Retest logout flow in DevTools.	Confirms that CSP errors are resolved.

✔ Correct Configuration Example

```
Content-Security-Policy: connect-src 'self'
https://launcher.domain.com https://sso.domain.com;
```

📌 Sample Query for Updating TB_ARM_APP_CONFIG TABLE

Use the following sample query to update CSP headers:

```
UPDATE TB_ARM_APP_CONFIG SET CONFIG_VALUE = '.... connect-src ''self''
https://launcher.domain.com https://sso.domain.com;  ....' WHERE
CNFG_NAME = 'security-headers';
```

📌 Applicability

This CSP configuration update is **mandatory and applicable for all versions up to ARX 24.1**.
(From ARX 24.2 onward, SSO and Launcher is deployed as same service)

Pattern 23 : Introspection API Not Triggered – Session Timeout on ARX

🛑 Issue Description

Users are seeing **inactive session timeout popups** on ARX even while actively working.
This happens because the **Introspection API** is not being triggered by the application.

As a result:

- The backend does **not** update `last_xfer_time` in the session table.
- The system assumes no activity, even though the user is active.
- ARX displays the **session timeout popup** prematurely.

This issue is observed across environments where the Introspection API call is expected to be triggered periodically.

🔎 Root Cause Analysis (RCA)

Primary Root Cause — Introspection API Not Invoked During Active Session

Finding:
The application's front-end or middleware is **not calling the Introspection API** at regular intervals.

Impact:
Since the backend does not receive token-validation or activity signals:

- `last_xfer_time` remains unchanged
- SELECT LAST_XFER_TIME FROM TB_ARM_SESSION WHERE USER_ID = <<USERID>>;
- ARX incorrectly marks the session as **inactive**
- Active users are shown timeout popups and forced to re-login

✅ Mandatory Permanent Solution (Application Logic Update Required)

The application layer must invoke the **Introspection API** at regular intervals or on user interactions to ensure session continuity.

Application teams needs to check the Introspection API is working properly or not

Sample DB Query

```
SELECT last_xfer_time
FROM TB_ARM_SESSION
WHERE user_id = '<USER_ID>';
```

Pattern 24: Duplicate Entries in IMS_REQUEST Table During Modify User API

🛑 Issue Description

During execution of the **Modify User API**, duplicate entries are being inserted into the **IMS_REQUEST** table.
This occurs when the **default branch** parameter is **not passed** in the request payload.

Instead of using the configured **database sequence**, the system calculates the request ID using:

```
SELECT MAX(ID) + 1 FROM IMS_REQ;
```

This flawed logic results in **non-unique request IDs** under concurrent API calls, leading to inconsistent data and intermittent failures.

🔎 Root Cause Analysis (RCA)

Primary Root Cause — Request ID Generated Using MAX(ID)+1 Instead of Database Sequence

Finding:
When the default branch is missing, the backend executes a query similar to:

```
SELECT MAX(ID) + 1 FROM IMS_REQ;
```

instead of using the correct sequence:

```
IMS_REQ_SEQ.NEXTVAL
```

✅ Mandatory Permanent Solution (Backend Update Required)

The backend logic must be updated to ensure the request ID is **always generated using the database sequence**, regardless of which parameters are passed.

Step-by-Step Fix

Step	Action	Rationale (Intended Outcome)
1	Replace MAX(ID)+1 logic with sequence: `IMS_REQ_SEQ.NEXTVAL`.	Ensures a unique ID is generated for every request.

✔ Specific Code Change Required

Update the SQL returned by:

```
getUpdateUserHistoryExcludeBranchSql()
```

✅ Fixed Version

Fixed is provided in the latest ARX24.2 version.

Pattern 25: Session Expiry Resolution

🛑 Issue Description

Users are experiencing **session expiry** (timeout issues) across several applications. This is due to a failure in the ARX system to maintain active user sessions, leading to premature token invalidation.

🔎 Root Cause Analysis (RCA)

The session expiry can be attributed to two primary, independent causes:

1. Missing OpenID Scope in Authorization Request (Primary)

- **Finding:** The affected applications are **not including the required openid scope** in the initial authorization endpoint call.
- **Impact:** Without the openid scope, ARX (the Identity Provider) cannot properly manage the user's session lifecycle. Crucially, the subsequent **Introspection API** call, which is designed to inform ARX that the user is actively using the application (i.e., not idle), is either failing or not providing the necessary context for ARX to extend the session. This results in the user token being marked as inactive prematurely.

2. Time Synchronization and Configuration Drift (Secondary/Environmental)

- **Finding:** There may be discrepancies between the **Database (DB) time** and the **Server time** where ARX services are running. Additionally, system-level timeout configurations are set too low.
- **Impact:** Time differences can lead to tokens being calculated as expired even if they are still valid, or session renewal checks failing due to clock skew. Low RETRYTIME configurations exacerbate the timeout issue, cutting off sessions aggressively.

✅ Mandatory Permanent Solution (Application Teams Action)

The following flow is **mandatory** for ARX session management and must be implemented by all application teams to resolve the timeout issue permanently.

Step	Action	Rationale (Intended Outcome)
1.	**Send openid Scope:** All applications must send the scope as **openid** in the **authorization endpoint** call.	Enables ARX to initiate and manage the full OpenID Connect session lifecycle.

2.	**Call Introspection API:** Call the Introspection API for token validation.	This actively **intimates ARX that the user is currently using the application** (not idle).
3.	**Handle Inactive Status:** If the Introspection API returns an **inactive** status, immediately call the **refresh token endpoint**.	Ensures seamless session renewal and prevents user logouts during active use.

🛠️ Additional Technical Checks & Temporary Workaround

Category	Action	Note
Environment Check	**Verify DB and Server Time**	Ensure clock synchronization between the ARX server and the database.
Restart	**Sequential ARX Service Restart**	Restart arxidam, arxsso, arxlauncher, and arxweb once and monitor behavior.
Timezone	**Set UTC Timezone**	If data center location changes or time synchronization is unfeasible, set the timezone to **UTC** for both the DB and Pods.
Temporary Fix	**Increase Timeout Configurations**	To maintain business continuity while the permanent solution is implemented, apply these updates:

	update tb_arm_app_config set cnfg_value =3600 where cnfg_name ='RETRYTIME_CLIENT';	**Note:** This is a **temporary workaround** and must be reverted once the permanent fix is confirmed.
	update tb_arm_app_config set cnfg_value =3600 where cnfg_name ='RETRYTIME_SERVER';	

Pattern 26: ARM-004215: Error while resetting password

Issue Description

While performing the resetUserPwd operation, the following error is encountered:

ARM-004215: Error while resetting password: java.lang.NumberFormatException

Additionally, the response includes:

"responseCode":4213, "responseMsg":"Error while reading App.Config"

This error prevents the system from successfully resetting the user's password and is observed during password reset requests in the ARM (Access Rights Management) module.

🔎 Root Cause Analysis (RCA)

1. Missing configuration entry in the IMS_PSWD_CONFIG table

During the password reset process, the application reads configuration parameters (like password policy, length, complexity, expiry days, etc.) from the IMS_PSWD_CONFIG table.

Each user type or identifier must have a corresponding configuration record in this table.

In this case, the entry for the respective IDENTIFIER_ID (linked to the user type being reset) is missing, leading to an invalid or null value being read by the application.

2.Null or empty value conversion to number

The application attempts to convert configuration values (like minimum length or expiry days) into numeric types (e.g., Integer.parseInt() or Long.parseLong() in Java).

Since the configuration entry is missing, the system retrieves a null or blank string — resulting in:

java.lang.NumberFormatException: For input string: ""

Hence, the password reset operation fails during configuration reading.

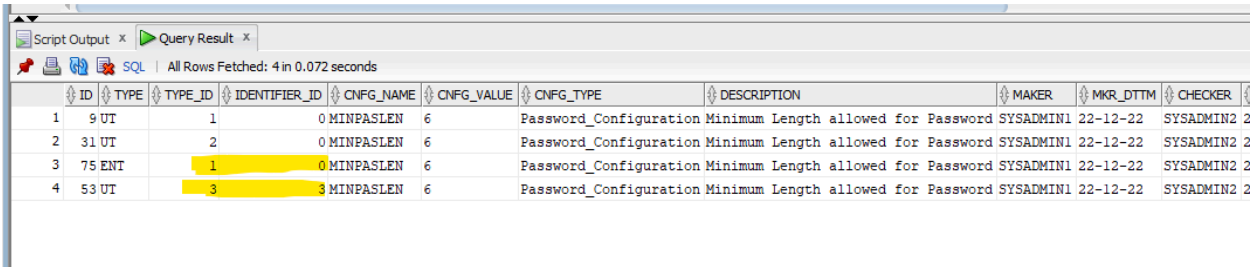
3.App Config Read Failure

The secondary message "Error while reading App.Config" is a direct symptom of the missing configuration.

The IMS_PSWD_CONFIG table is considered part of App.Config, and the absence of a valid record for the corresponding identifier triggers the exception.

The USER_IDENTIFIER_ID is used for most corporate and retail users to maintain a unique ID for each user group. It is also used to apply the correct password policy configuration based on the corresponding IDENTIFIER_ID.

In the table below, your TYPE_ID and IDENTIFIER_ID entry must be present other get above query.


Script Output | Query Result

SQL | All Rows Fetched: 4 in 0.072 seconds

	ID	TYPE	TYPE_ID	IDENTIFIER_ID	CNFG_NAME	CNFG_VALUE	CNFG_TYPE	DESCRIPTION	MAKER	MKR_DTTM	CHECKER	
1	9	UT	1	0	MINPASLEN	6	Password_Configuration	Minimum Length allowed for Password	SYSADMIN1	22-12-22	SYSADMIN2	2
2	31	UT	2	0	MINPASLEN	6	Password_Configuration	Minimum Length allowed for Password	SYSADMIN1	22-12-22	SYSADMIN2	2
3	75	ENT	1	0	MINPASLEN	6	Password_Configuration	Minimum Length allowed for Password	SYSADMIN1	22-12-22	SYSADMIN2	2
4	53	UT	3	3	MINPASLEN	6	Password_Configuration	Minimum Length allowed for Password	SYSADMIN1	22-12-22	SYSADMIN2	2

To check USER_IDENTIFIER_ID select * from TB_ARM_USER_IDENTIFIER;

```
6  select * from TB_ARM_USER_IDENTIFIER;
```

Script Output | Query Result

SQL | All Rows Fetched: 4 in 0.034 seconds

	ID	IDENTIFIER_NAME	FK_USER_TYPE_ID	IDENTIFIER_ALIAS
1	0	All	1	All
2	1	BCACBGCORP	1	Functional User
3	2	BCAADMIN00	1	Administrator
4	3	Retail	3	Retail

✅ Mandatory Permanent Solution (Application Teams Action)

1.Insert missing configuration

E.g

Insert into IMS_PSWD_CONFIG (ID,TYPE,TYPE_ID,IDENTIFIER_ID,CNFG_NAME,CNFG_VALUE,CNFG_TYPE,DESCRIPTION,MAKER,MKR_DTTM,CHECKER,CHKR_DTTM) values (9,'UT',1,0,'MINPASLEN','6','Password_Configuration','Minimum Length allowed for Password','SYSADMIN1',to_date('22-12-22','DD-MM-RR'),'SYSADMIN2',to_date('22-12-22','DD-MM-RR'));

2.Cross-check with working environment

select * from IMS_PSWD_CONFIG;

Compare the configuration entries with a working environment (like UAT or SIT) to ensure all expected keys are maintained.

🛠️ Additional Technical Checks & Temporary Workaround

1. Check logs for detailed stack trace.
2. Verify consistency of configuration data types.

Pattern 27 : JWT Bearer Assertion Validation and Resolution

🛑 Issue Description

Applications are failing to establish an initial authentication session with the ARX. This is specifically due to the rejection of the JWT Bearer Assertion when presented to the ARX Token Endpoint (e.g., `/oauth2/token`).

Analysis Steps

1. Token Endpoint Interaction Check

Instruct the application team to capture the complete **HTTP POST request** made to the ARX Token Endpoint (`/oauth2/token`).

- **Verify URL and Method:** Confirm the request is a `POST` to the correct token endpoint URL.
- **Check Request Body:** Inspect the `application/x-www-form-urlencoded` body to ensure the following parameters are present and correctly formatted:
  - `grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer`
  - `assertion=<jwt>` (The assertion is the base64-encoded JWT string).
- **Inspect Response:** Analyze the HTTP response code and body from the ARX Token Endpoint. Common error responses are:
  - **400 Bad Request** with an OAuth error code like `invalid_client` or `unauthorized`.
  - **500 Internal Server Error** (which often points to a certificate/cryptographic failure on the server side).

2. JWT Content Validation (Client-Side)

Decode the **JWT Assertion** (`<jwt>`) being sent in the request (using an online tool or internal library) and inspect the **Payload Claims** section.

- **Claim 1: `iss` (Issuer/Client ID):** Verify the value is the **exact and active Client ID** registered in ARX.
- **Claim 2: `sub` (Subject/User ID):** Verify the value is for a **known and existing user** in the ARX user database.
- **Claim 3: `aud` (Audience):** Verify the value is set to the **ARX Token Endpoint URL** or a configured ARX audience value.
- **Claim 4: `exp` (Expiration):** Verify the timestamp has not already expired.

3. Application Configuration & Key Check

- **Keystore Path:** Confirm the application's configuration properties (`java.security.keystore.path`, `java.security.keystore.password`, `java.security.key.alias`) are pointing to a valid, accessible `.jks` file and the correct private key alias.
- **Public Key Match:** The application team must confirm that the **Public Key** corresponding to the Private Key used for signing is **identical** to the one registered in the ARX configuration (e.g., in the `arx-idam config map` under the key `assertion_public_key`).

🔎 Root Cause Analysis (RCA)

1. 🔑 Assertion Rejected Due to Certificate Mismatch

Finding: The **Public Key** registered(or present in the arx-idam config map with the name of key:`assertion_public_key`) in ARX under the client's configuration does not correspond to the **Private Key** used by the application to sign the JWT Assertion. The signature cannot be verified.

Impact: ARX returns an **exception while validating assertion token with public key** or a standard OAuth error like `invalid_client` or `unauthorized`. This is a **Cryptographic Failure**—the token's origin cannot be trusted.

2. 📝 Assertion Rejected Due to Client ID Mismatch

Finding: The value of the mandatory **`iss` (client_id) claim** in the JWT payload is not set to the correct and active **Client ID** registered in ARX. The JWT signature replaces the Client Secret, making the `iss` claim the primary client identifier.

ImpactARX rejects the assertion because the client_id (the application) is unrecognized or unauthorized. ARX may specifically return the error **`Invalid_oauth_credentials`**. This is a **Client Configuration Failure**.

3. 👤 Assertion Rejected Due to Non-Existent User in Database

Finding: The value of the **`sub` (Subject) claim** in the JWT payload refers to a user ID that does not exist in the ARX user database.

Impact: The authentication process halts during the user lookup phase. ARX may return an error indicating the **user ID is not present**. The consuming application will generally receive a generic **401 Unauthorized** HTTP response because the request could not be completed successfully. This is a **Data Integrity Failure**.

4. ⚙️ Assertion Rejected Due to Incorrect Internal Validation

Finding: The application's **custom internal JWT library** (used for pre-check before calling ARX) validates the token against a locally configured or expected `iss` (or `aud`) value. If the assertion's claim value does not match the internal validator's expectation, it is rejected locally.

Impact: The application throws a runtime error **before** making the external HTTP call to ARX's Token Endpoint. If the internal validator is specifically checking the *ARX* issuer, ARX may eventually return an error like **`Invalid issuer`** (if it ever received the token), but the immediate failure is a **401 Unauthorized** at the application side, leading to a functional failure that is difficult to trace to ARX. This is an **Internal Configuration Failure**.

✅ Mandatory Permanent Solution (Application Teams Action)

The following flow is **mandatory** for ARX jwt assertion creation that must be implemented by all application teams to resolve the timeout issue permanently.

Step (Finding)	Action (Mandatory Permanent Solution)	Rationale (Intended Outcome)
1. Certificate Mismatch	Verify and Sync Public Key: The application team must ensure the exact same Public Key corresponding to the Private Key used for signing the JWT Assertion is registered in the ARX configuration map (`arx-idam config map` / `assertion_public_key`).	Ensures the cryptographic trust chain is intact, allowing ARX to successfully verify the JWT signature and prevent Cryptographic Failure.
2. 📝 Client ID Mismatch	Correct `client_id` Claim Value: The application must set the mandatory `iss` (Client ID) claim in the application side configuration to the correct and currently active Client ID registered for the application in ARX.	Correctly identifies the application as the legitimate client id of the JWT, preventing ARX from returning `invalid_client` or `Invalid_oauth_credentials` due to Client Configuration Failure.
3. 👤 Non-Existent User	Pre-validate User Existence: The application must ensure the value used for the `sub` (Subject/User ID) claim exists in the ARX user database before generating and sending the JWT Assertion.	Halts the authentication flow early on invalid data, preventing the Data Integrity Failure where ARX fails the user lookup and returns a generic `401 Unauthorized`.

4. ⚙️ Incorrect JWT_Issuer	Disable or Correct Internal Validation: The application team must maintain JWT_issuer for validation of the ARX assertion, OR correct it to use the exact expected ARX JWT_Issuer/Audience value for the `iss`/`aud` claims.	Eliminates internal application runtime errors. Ensures the assertion is actually sent to ARX for proper validation, moving the source of truth to the authoritative ARX Token Endpoint.
5. 📁 Key Store Path is `null`	Validate Configuration: Application code must perform a null check on the `keyStorePath` configuration property before attempting to instantiate `new FileInputStream()`. A non-null, valid path must be provided.	Prevents a `java.lang.NullPointerException` during key store loading, ensuring the Private Key is successfully initialized for JWT signing.

🛠️ Additional Technical Checks & Temporary Workaround

Category	Action	Environment Check / Rationale
1. Client Configuration	Sync Client ID/Secret	Verify that the **`client_id`** and **`client_secret`** used by the application are an **exact match** for the values recorded in the identity database table **`ims_oauth_registration`**. Mismatched credentials lead to validation failure.
2. Certificate Path	Mount Path Validation	Ensure the configured **certificate file path** is a **permanent, accessible mount path** on the application server. This guarantees file availability and read permissions for key loading.
3. Certificate Format	Use JKS KeyStore	The certificate containing the Private Key for signing must be in the **Java KeyStore (`.jks`)** format. This proprietary Java format is required for `KeyStore.getInstance("JKS")` to successfully

		load and access the cryptographic keys used to sign the JWT Assertion.