

Oracle Banking Digital Experience

Security Guide
Release 18.1.0.0.0

Part No. E92727-01

January 2018

ORACLE®

Security Guide

January 2018

Oracle Financial Services Software Limited

Oracle Park

Off Western Express Highway

Goregaon (East)

Mumbai, Maharashtra 400 063

India

Worldwide Inquiries:

Phone: +91 22 6718 3000

Fax: +91 22 6718 3001

www.oracle.com/financialservices/

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are “commercial computer software” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate failsafe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Table of Contents

1. Preface	5
1.1 Audience	5
1.2 Documentation Accessibility	5
1.3 Access to OFSS Support	5
1.4 Related Information Sources	5
2. General Security Principles	6
2.1 Restrict Network Access to Critical Services	6
2.2 Follow the Principle of Least Privilege	6
2.3 Monitor System Activity	6
2.4 Keep Up To Date on Latest Security Information	6
3. Secure Installation and Configuration	7
3.1 Architecture Diagram	7
3.2 Installing WebLogic	7
3.3 Configuring SSL	7
3.4 Disable SSLv3	10
3.5 HTTP Response Header Configurations	11
3.5.1 X-Frame-Options	11
3.5.2 Content-Security-Policy	11
3.5.3 X-XSS-Protection	11
3.5.4 Strict-Transport-Security	11
3.5.5 Cache-Control	12
3.6 Password Policy Guidelines	12
3.6.1 Password Policy Configuration	12
3.7 Configuring 2FA for login	15
3.8 Configuring 2FA attributes	17
3.9 Choosing a non blocking PRNG	20
3.10 Mobile App SSL Pinning Configuration	21
4. Guidance for Implementation Teams	23
4.1 CSRF Mitigation – Generating Nonces	23
4.2 Indirect Object Reference Implementation	23
4.2.1 What it means	23
4.2.2 How OBDX supports it	24

4.3	Output Encoding	25
4.4	Configuring the Hashing Algorithm for OTPs.....	25
4.5	Implementing a custom 2FA mechanism	26
4.6	Disable Password Protection for PDF Documents	27

1. Preface

This document provides a comprehensive overview of security for Oracle Banking Digital Experience. It includes conceptual information about security principles, descriptions of the product's security features, and procedural information that explains how to use those features to secure Oracle Banking Digital Experience.

This preface contains the following topics:

- Audience
- Documentation Accessibility
- Access to Oracle Support
- Related Documents

1.1 Audience

This Security Guide is intended for Bank IT Staff responsible for application installation and security configuration as well as Product Implementation teams.

1.2 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

1.3 Access to OFSS Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if your hearing is impaired.

1.4 Related Information Sources

For more information, see the following documentation:

Hardening Tips for Default Installation of Oracle Enterprise Linux 6 at

https://docs.oracle.com/cd/E37670_01/E36387/E36387.pdf

Oracle® Fusion Middleware Installation Guide for Oracle WebLogic Server at

http://docs.oracle.com/cd/E17904_01/doc.1111/e14142/toc.htm

Oracle® Fusion Middleware Application Security Guide - Configuring and Managing Auditing at

http://docs.oracle.com/cd/E23943_01/core.1111/e10043/audpolicy.htm

For installation and configuration information, see the Oracle Banking Digital Experience Installation Guide

For the complete list of Oracle Banking licensed products and the Third Party licenses included with the license, see the Oracle Banking Licensing Guide.

2. General Security Principles

The following principles are fundamental for using any application securely.

2.1 Restrict Network Access to Critical Services

Keep both the Oracle Banking Digital Experience middle-tier and the database behind a firewall. In addition, place a firewall between the middle-tier and the database. The firewalls provide assurance that access to these systems is restricted to a known network route, which can be monitored and restricted, if necessary. As an alternative, a firewall router substitutes for multiple, independent firewalls.

If firewalls cannot be used, be certain to configure the TNS Listener Valid Node Checking feature which restricts access based upon IP address. Restricting database access by IP address often causes application client or server programs to fail for DHCP clients. To resolve this, consider using static IP addresses, a software or a hardware VPN or Windows Terminal Services or its equivalent.

2.2 Follow the Principle of Least Privilege

The principle of least privilege states that users should be given the least amount of privilege to perform their jobs. User privileges should be reviewed periodically to determine relevance to current job responsibilities.

2.3 Monitor System Activity

System security largely depends on the following practices:

- Good security protocols
- Proper system configuration
- System monitoring

The system needs to be constantly monitored from a monitoring tool.

2.4 Keep Up To Date on Latest Security Information

Oracle continually improves its software and documentation. It is recommended to keep your software updated.

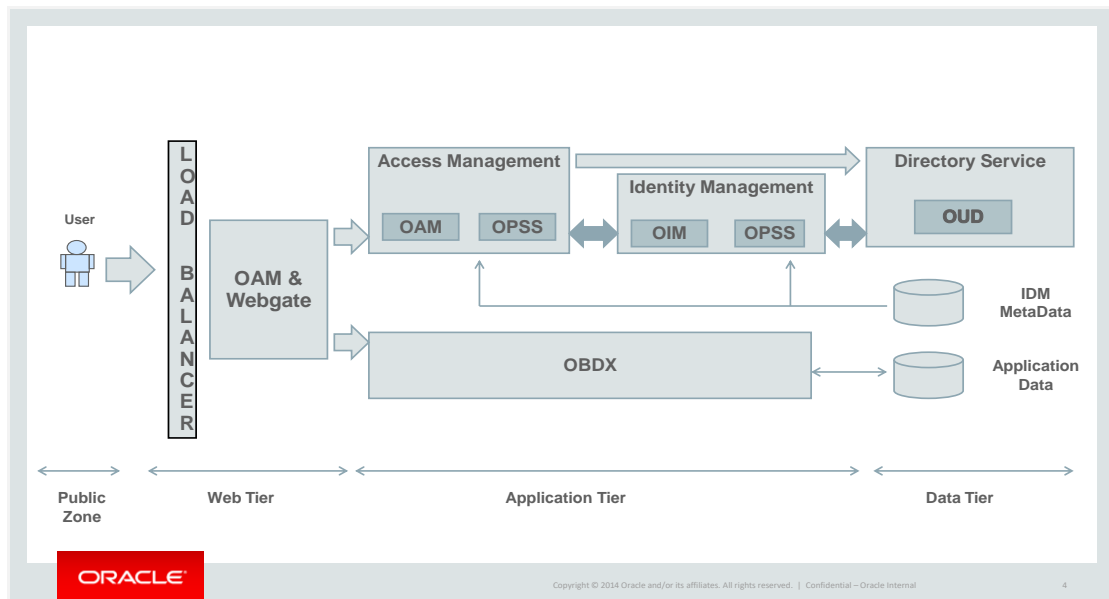
[Home](#)

3. Secure Installation and Configuration

This chapter provides an overview of the architecture of the deployment and describes the installation and configuration procedure for Oracle Banking Digital Experience.

Please note that this is only a guide to securing the Oracle Banking Digital Experience application and does not replace periodic reviews of the security architecture of the entire ecosystem of multiple applications maintained by the customer. The guidance provided in this document must always be augmented by specific understanding of the security considerations of the specific deployment architecture.

3.1 Architecture Diagram



3.2 Installing WebLogic

Installation of Weblogic Server can be done by referring to the documentation published at https://docs.oracle.com/cd/E24329_01/doc.1211/e24492/toc.htm.

3.3 Configuring SSL

One way SSL between the presentation tier and the application on WebLogic server is supported. The detailed configuration is explained below:

Note: Procure an external CA signed certificate before proceeding further. Follow the instructions below to install the certificate once the certificate is available

1. Import the Certificate into a Java Trust Keystore

Execute the following command:

```
keytool -import -trustcacerts -alias sampletrustself -keystore
SampleTrust.jks -file SampleSelfCA.cer.der -keyalg RSA
keytool -import -alias `hostname -f` -file `hostname -f`.cer -keystore
<JAVA_HOME>/jre/lib/security/cacerts -storepass changeit -noprompt
```

2. Configure Application Domain's WebLogic with Custom Identity and Trust keystores

- a. Open the WebLogic admin console and navigate to *Home --> Summary of Servers --> AdminServer*.
- b. Click the **Keystores** tab.

- Click the **Change** button.
- Select Custom Identity and Java Standard Trust option from the list.
- Click the **Save** button.
- Enter the following details in the **Identity** and **Trust** sections:

Field	Value
Custom Identity Keystore	Absolute path of the custom keystore
Custom Identity KeyStore Type	JCEKS
Custom Identity KeyStore Passphrase	<Passphrase>

Field	Value
Confirm Custom Identity KeyStore Passphrase	<Re-enter the same Passphrase>

Enter the passphrases that were used while creating the custom Identity Keystore and certificate.

- c. Click the **Save** button.
- d. Click the **SSL** Tab.

The screenshot shows the Oracle WebLogic Server Administration Console interface. The 'SSL' tab is selected under the 'Configuration' section. The 'Identity and Trust Locations' section is expanded, showing the following settings:

- Private Key Location:** from Custom Identity Keystore
- Private Key Alias:** weblogic
- Private Key Passphrase:** (masked with asterisks)
- Confirm Private Key Passphrase:** (empty)
- Certificate Location:** from Custom Identity Keystore
- Trusted Certificate Authorities:** from Java Standard Trust Keystore

Enter the following details in the **Identity** section:

Field	Value
Private Key Alias	<Alias>
Private Key Passphrase	<Passphrase>
Confirm Private Key Passphrase	<Re-enter passphrase>

- i. Enter the passphrases that were used while creating the certificate.
- ii. Click the Save button.
- iii. Click the Advanced link.
- iv. Ensure that **Two Way Client Cert Behavior** is set to **Client Certs Not Requested**.

- e. Click the **General** tab.
- f. Select the **SSL Listen Port Enabled** check box.

The screenshot shows the Oracle WebLogic Server Administration Console. The main content area is titled 'Settings for OBCServer' and is under the 'General' tab. The 'SSL Listen Port Enabled' checkbox is checked. The 'Listen Port' is set to 7001 and the 'SSL Listen Port' is set to 7002. The 'Name' is OBCServer, 'Hostname' is (None), and 'Cluster' is (Stand Alone). The 'Listen Address' is 8020. The 'Java Compiler' is javac. The 'Diagnostic Volume' is set to a default value.

- g. Click the **Save** button.

3.4 Disable SSLv3

By default, SSLv3 should be disabled.

Specifying the `weblogic.security.SSL.protocolVersion` system property in a command-line argument that starts the WebLogic Server lets you specify the protocol that is used for SSL connections.

The following command-line arguments can be specified so that WebLogic Server supports only TLS connections:

```
-Dweblogic.security.SSL.protocolVersion=TLS1
```

Note: If you don't specify the above property, weblogic assumes SSLv3 by default.

3.5 HTTP Response Header Configurations

The following are some HTTP Response Headers that mitigate certain vulnerabilities.

Vulnerability	HTTP Response Header
Clickjacking	X-Frame-Options
XSS	Content-Security-Policy
	X-XSS-Protection
Cookie hijacking Protocol Downgrade attacks	Strict-Transport-Security
Retrieving Sensitive data from browser cache	Cache-Control

The sections below specify how to configure these response headers in the `httpd.conf` file of the web server.

3.5.1 X-Frame-Options

```
Header always append X-Frame-Options SAMEORIGIN
```

3.5.2 Content-Security-Policy

```
Header set Content-Security-Policy "default-src 'none'; img-src 'self'
data:; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self'
https://fonts.googleapis.com 'unsafe-inline'; object-src 'none'; frame-src
'none'; font-src 'self' https://fonts.googleapis.com/
https://fonts.gstatic.com/ data:; connect-src 'self'
https://fonts.googleapis.com/ https://fonts.gstatic.com/ http://<OAM
Server>:<OAM Port>; manifest-src 'self'; child-src 'self'"
```

Please note that the policy mentioned here is for the base product. If the product gets customized and content from different URLs needs to be allowed to be executed by the browser, then this policy will have to be modified accordingly.

3.5.3 X-XSS-Protection

```
Header set X-XSS-Protection "1; mode=block"
```

3.5.4 Strict-Transport-Security

Set this for your top level domain. The header directive needs to be included inside the `VirtualHost` directive

```
<VirtualHost *:443>
  Header always set Strict-Transport-Security "max-
age=31540000; includeSubDomains"
</VirtualHost>
```

Consider submitting your website to be included in the HSTS preload list of websites maintained by Google Chrome at <https://hstspreload.appspot.com/>.

Other browsers like MS IE 11, MS Edge, Firefox and Opera also refer to this list maintained by Google and therefore the security offered by this mechanism will extend to other browsers too.

3.5.5 Cache-Control

```
Header set Cache-Control "max-age=0, no-cache, no-store, must-revalidate"
Header set Pragma "no-cache"
Header set Expires 0
```

3.6 Password Policy Guidelines

Our recommendations for setting a password policy are in line with the latest recommendations from NIST as of July 2017.

1. The minimum length of a password must be at least 8 characters. You can choose to increase this number to 10 or 12.
2. The maximum length of a password must be at least 64 characters. You can choose to increase this number to 80 or 100.
3. Do not cause passwords to expire without reason. A password must be expired only when the user has forgotten it and has requested a reset.
4. Allow all printable ASCII characters, including spaces, and accept all UNICODE characters too.
5. Do not force the user to use a combination of upper case characters, lower case characters, numbers and special characters.
Instead recommend to him that he uses “passphrases” instead of passwords, and that’s the reason why the recommended minimum length must be at least 8 and the maximum length must be at least 64. Passphrases are sentences like “Wow, I like the freedom to choose this password!!” (yes, with spaces, a comma and exclamation marks in it)

3.6.1 Password Policy Configuration

The password policy can be maintained in the Oracle Banking Digital Experience database or you can also set it in the LDAP user repository, if one is being used.

You need to specify your preference in the property shown below:

The screenshot shows a SQL query result in a database client. The query is: `select * from DIGX_FW_CONFIG_ALL_B where PROP_ID='PASSWORD_POLICY_VALIDATION_LOCAL' and CATEGORY_ID='dayoneconfig';`

Row 1	Fields
PROP_ID	PASSWORD_POLICY_VALIDATION_LOCAL
CATEGORY_ID	dayoneconfig
PROP_VALUE	false
FACTORY_SHIPPED_FLAG	N
PROP_COMMENTS	configure to decide either use local or remote password policy validation
SUMMARY_TEXT	configure to decide either use local or remote password policy validation
CREATED_BY	ofsuser
CREATION_DATE	22-MAR-17 06:29:04.000000 AM
LAST_UPDATED_BY	ofsuser
LAST_UPDATED_DATE	22-MAR-17 06:29:04.000000 AM
OBJECT_STATUS	Y
OBJECT_VERSION_NUMBER	1

A value of “true” in PROP_VALUE indicates that the password policy set in the local database needs to be applied.

A value of “false” in PROP_VALUE indicates that the password policy set in the LDAP user repository needs to be applied.

When PROP_VALUE is set to “true”, the password policy must be set in the database table DIGX_UM_PWD_POLICY. The following table lists the columns of the table and significance of each column:

Column Name	Significance	
PASSWORDPOLICYID	Unique Identifier for the password policy.	
PWD_POLICY_NAME	Human friendly name given to the policy.	
PWD_MIN_LENGTH	Minimum length of the password.	
PWD_MAX_LENGTH	Maximum length of the password.	
CHAR_ALLOWED	Characters allowed in the password.	
	1	Upper Case Letters
	2	Lower Case Letters
	9	Numbers
	-1	Special Characters
FIRST_CHAR_ALLOWED	Characters allowed as the first character of a password.	
LAST_CHAR_ALLOWED	Characters allowed as the last character of a password.	
NBR_UPPER_CASE	Minimum number of upper case characters required in the password.	
NBR_LOWER_CASE	Minimum number of lower case characters required in the password.	
NBR_NUMERIC_CHAR	Minimum number of numeric characters required in the password.	
NBR_SPECIAL_CHAR	Minimum number of special characters required in the password.	
SPECIAL_CHAR_ALLOWED	List of special characters allowed in the password.	
NBR_REPEATED_CHAR	Maximum number of times a character is repeated in the password.	
NBR_SUCESSIVE_CHAR	Maximum number of times a character can be repeated successively in the password.	

The following image shows the recommended values for the key columns of the database table DIGX_UM_PWD_POLICY

Row 1	Fields
PASSWORDPOLICYID	123789
PWD_POLICY_NAME	CLIP Password Policy
PWD_MIN_LENGTH	8
PWD_MAX_LENGTH	64
CHAR_ALLOWED	1,2,9,-1
FIRST_CHAR_ALLOWED	1,2,9,-1
LAST_CHAR_ALLOWED	1,2,9,-1
NBR_UPPER_CASE	0
NBR_LOWER_CASE	0
NBR_NUMERIC_CHAR	0
NBR_SPECIAL_CHAR	0
SPECIAL_CHAR_ALLOWED	@.#,\$.%&!.,~*.-
NBR_REPEATED_CHAR	100
NBR_SUCESSIVE_CHAR	100

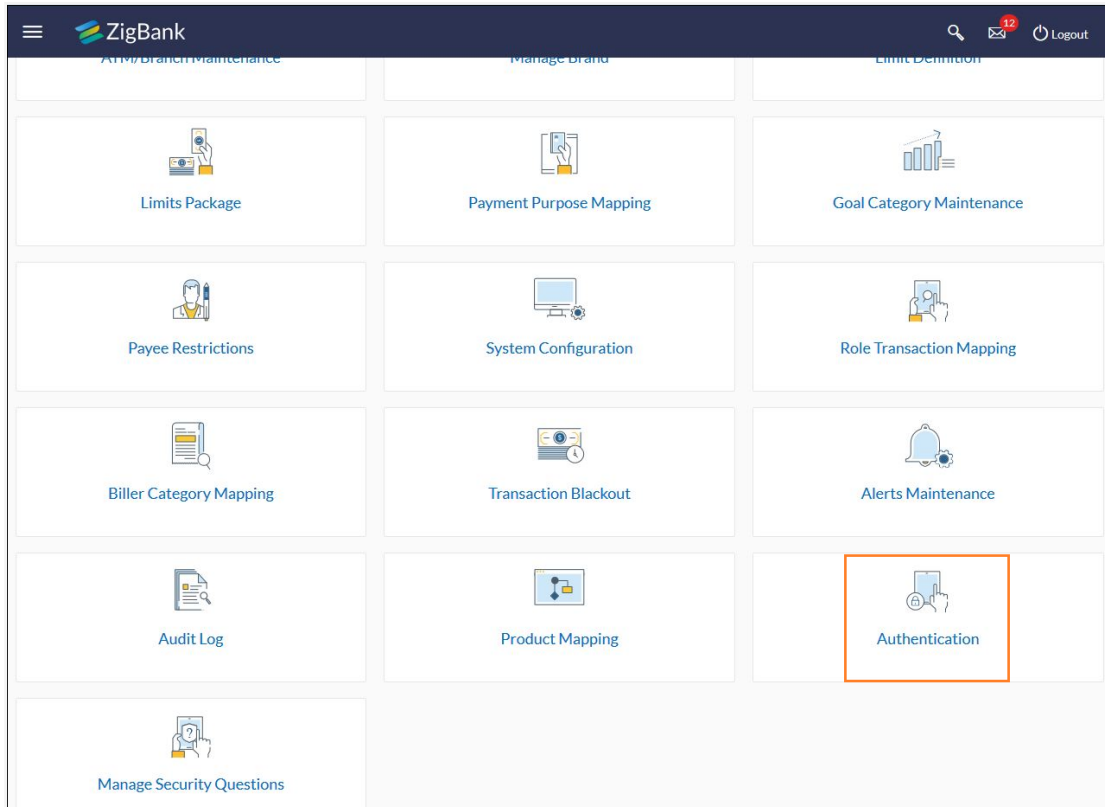
Note:

- 1) The password policy ID and name are at your discretion and no specific recommendations are being made for them.
- 2) A different password policy can be set for each user group that you define. The password policy applicable for a particular user group can be defined in the table DIGX_UM_PWD_GROUP_MAP. That's where the password policy ID will come in handy.
- 3) If you want the same password policy to be applicable for all groups then you can simply define just one policy in DIGX_UM_PWD_POLICY. You can leave the mapping table DIGX_UM_PWD_GROUP_MAP empty.

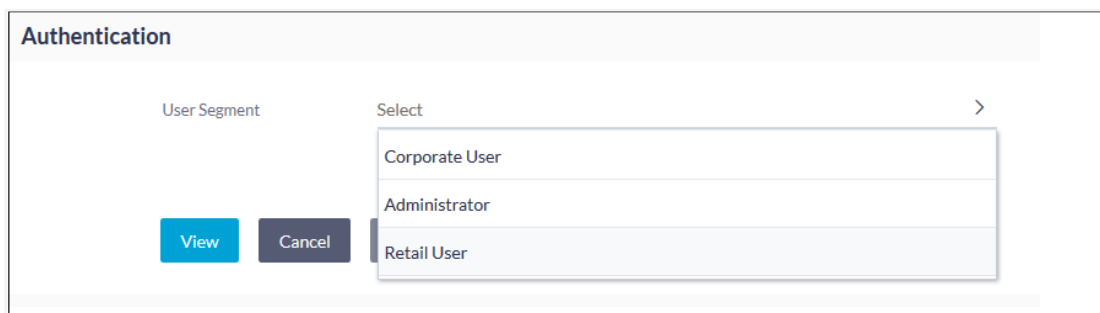
3.7 Configuring 2FA for login

Oracle Banking Digital Experience supports a 2nd factor of authentication during login.

1. Login as the Admin user
2. Click on “Authentication”



3. Choose the user segment for which you want to configure 2FA for login and click on “View”



4. You will see the following screen where you can configure 2FA for virtually every transaction, including Login.

Authentication			
View			
User Segment	Retail		
Transaction Type	Level 1	Level 2	
Self Transfer	One Time Password	Soft Token	
Bill Payment			
Domestic Payer			
Domestic Demand Draft Pay Later			
International Demand Draft			
Instruction Cancellation			
Mobile Device Registration			
Create Peer To Peer Payee			
Domestic Payment			
User Security Question Update			
Login			
International Payout			
Domestic Payment Pay Later/SI			
International Demand Draft Pay Later			
Reset Debit Card Pin			
Internal Transfer			
Domestic Payin Pay Later			

- Click on the “Edit” button at the bottom of the screen.

Domestic Payin Pay Later
Reset Credit Card Pin
Domestic Demand Draft
External Transfer
<input type="button" value="Edit"/> <input type="button" value="Cancel"/> <input type="button" value="Back"/>

- You can now configure up to 2 factors (levels) of authentication / re-authorization. However please note that the system will not let you set “Security Questions” as a factor of authentication / re-authorization for the Login transaction. You will have to choose either OTP or Soft Token.

Edit

User Segment: Retail

Transaction Type	Level 1	Level 2
Self Transfer	One Time Password >	Soft Token >
Bill Payment	None >	None >
Domestic Payer	None >	None >
Domestic Demand Draft Pay Later	None >	None >
International Demand Draft	None >	None >
Instruction Cancellation	None >	None >
Mobile Device Registration	None >	None >
Create Peer To Peer Payee	None >	None >
Domestic Payment	None >	None >
User Security Question Update	None >	None >
Login	None >	None >
International Payout	None >	None >

Apply to all

- Click on the “Save” button at the bottom of the screen, followed by the “Confirm” button seen in the subsequent verification screen.

3.8 Configuring 2FA attributes

This section covers some key attributes of the 2nd factor of authentication (re-authorization). Attributes like the maximum number of times a user is allowed to hit the “Resend” button after an OTP is generated, the pool of security questions etc are a couple of examples of 2FA attributes.

These attributes are seen in the database in the PROP_ID column of the table DIGX_FW_CONFIG_ALL_B (CATEGORY_ID = 'authenticationConfig'). The following table lists down all possible attributes and their significance. Their values must be set in the column PROP_VALUE.

PROP_ID	SIGNIFICANCE
OTP.EXPIRATION_TIME	Time in milliseconds after which an OTP will expire.
T_SOFT_TOKEN.EXPIRATION_TIME	Time in milliseconds after which a time based soft token will expire.
R_SOFT_TOKEN.EXPIRATION_TIME	Time in milliseconds after which a random soft token will expire.
SEC_QUE.EXPIRATION_TIME	Time in milliseconds after which answers to the security questions presented to the user will no longer be considered for re-authorization.
EXPIRATION_TIME	Time in milliseconds after which the re-authorization factor will expire. This is the default property that will be looked up in case factor specific expiration times are not maintained.
OTP.MAX_NO_ATTEMPTS	Max number of unsuccessful attempts of entering a valid OTP after which all 2FA enabled transactions for the user will be locked for a "cooling period" amount of time.
T_SOFT_TOKEN.MAX_NO_ATTEMPTS	Max number of unsuccessful attempts of entering a valid time based soft token after which all 2FA enabled transactions for the user will be locked for a "cooling period" amount of time.
R_SOFT_TOKEN.MAX_NO_ATTEMPTS	Max number of unsuccessful attempts of entering a valid random soft token after which all 2FA enabled transactions for the user will be locked for a "cooling period" amount of time.
SEC_QUE.MAX_NO_ATTEMPTS	Max number of unsuccessful attempts of entering valid answers to security questions after which all 2FA enabled transactions for the user will be locked for a "cooling period" amount of time.
MAX_NO_ATTEMPTS	Max number of unsuccessful attempts of entering valid 2FA after which all 2FA enabled transactions for the user will be locked for a "cooling period" amount of time. This is the default property that will be looked up in case factor specific Max Attempts are not maintained.

PROP_ID	SIGNIFICANCE
TFA_LOCK_COOLING_PERIOD	This is the cooling period in milliseconds after which 2FA transactions which were locked out because of exceeding MAX_NO_ATTEMPTS, are enabled once again.
OTP.MAX_ACTIVE_REF_NO	<p>Max number of attempts to generate 2FA reference numbers for a transaction after which no more attempts can be made for EXPIRATION_TIME units of time for that factor of authentication. This one is specific to OTPs.</p> <p>This property is in place as a basic mechanism to protect the application against DOS attacks where the end user can keep generating OTPs by initiating transactions and making the system generate the 2nd factor of authentication, but not going through and completing the transaction.</p>
T_SOFT_TOKEN.MAX_ACTIVE_REF_NO	Max number of attempts to generate 2FA reference numbers for a transaction after which no more attempts can be made for EXPIRATION_TIME units of time for that factor of authentication. This one is specific to Time Based Soft Tokens.
R_SOFT_TOKEN.MAX_ACTIVE_REF_NO	Max number of attempts to generate 2FA reference numbers for a transaction after which no more attempts can be made for EXPIRATION_TIME units of time for that factor of authentication. This one is specific to Random Soft Tokens.
SEC_QUE.MAX_ACTIVE_REF_NO	Max number of attempts to generate 2FA reference numbers for a transaction after which no more attempts can be made for EXPIRATION_TIME units of time for that factor of authentication. This one is specific to Security Questions.
MAX_ACTIVE_REF_NO	Max number of attempts to generate 2FA reference numbers for a transaction after which no more attempts can be made for EXPIRATION_TIME units of time for that factor of authentication. This is the default property that will be looked up in case factor specific Max Active Reference Number attempts are not maintained.

PROP_ID	SIGNIFICANCE
OTP.RESEND_COUNT	Max number of times a user can hit the "Resend" button in case of OTPs. After exceeding this count, the user will need to re-initiate the transaction all over again.
retailuser.NO_QUE_ANS	Number of security questions that a retail user needs to setup (answer). During an actual transaction he will be asked a sub set of these questions.
corporateuser.NO_QUE_ANS	Number of security questions that a corporate user needs to setup (answer). During an actual transaction he will be asked a sub set of these questions.
administrator.NO_QUE_ANS	Number of security questions that an admin user needs to setup (answer). During an actual transaction he will be asked a sub set of these questions.
NO_QUE_ANS	Number of security questions that a user segment needs to setup (answer). During an actual transaction he will be asked a sub set of these questions. This is the default property that will be looked up in case factor specific NO_QUE_ANS is not maintained

3.9 Choosing a non blocking PRNG

OBDX uses Java's random number generation capabilities internally. However the out of the box algorithm for PRNG configured in the JDK can block the thread after a certain time if there isn't enough randomness available. This is because the default configuration uses `/dev/random` on Linux for PRNG.

Therefore we recommend that you navigate to `<JDK_HOME>/jre/lib/security` and edit the `java.security` file. Comment out the old property and change its value as shown below

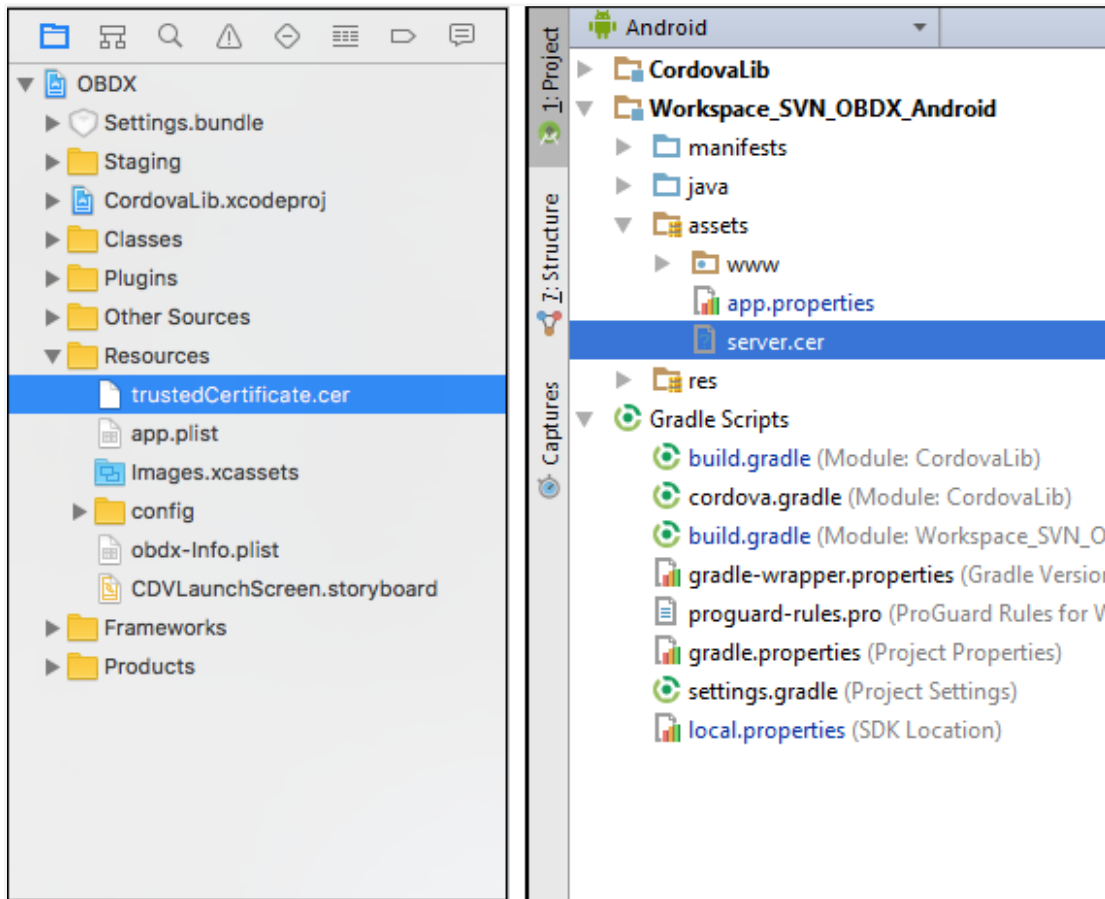
```
#securerandom.strongAlgorithms=NativePRNGBlocking:SUN
securerandom.strongAlgorithms=NativePRNGNonBlocking:SUN
```

This will ensure that the application uses `/dev/urandom` for PRNG.

Needless to say, make sure you make this change in the JDK that your weblogic server is going to be using.

3.10 Mobile App SSL Pinning Configuration

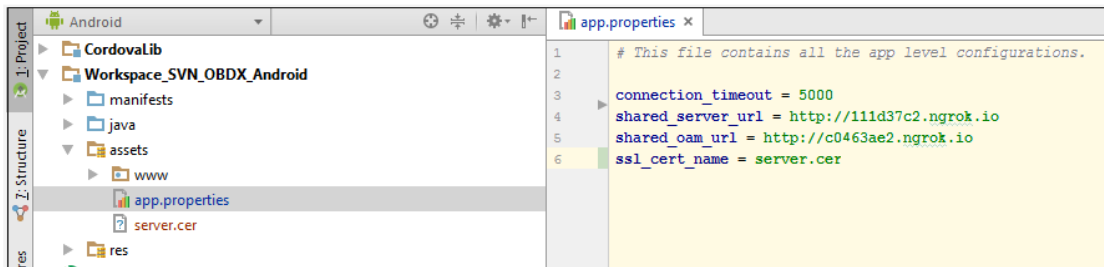
SSL Pinning has been implemented in the mobile apps, both iOS and Android. The public key certificate of the server needs to be imported into these apps for the connection to the server to be successful. The certificate needs to have an extension .cer and needs to be placed in mobile app workspaces as shown in the images below:



The name of the certificate file needs to be configured in a property file. For iOS it is the app.plist file

Key	Type	Value
▼ Root	Dictionary	(3 items)
PinnedCertificateName	String	trustedCertificate
CertificateType	String	cer
ConnectionTimeout	String	5

For Android it is the `app.properties` file



The screenshot shows an IDE window with a project structure on the left and a code editor on the right. The project structure includes folders for CordovaLib, Workspace_SVN_OBDX_Android, manifests, java, assets, www, app.properties, server.cer, and res. The code editor displays the contents of the app.properties file, which contains the following configuration:

```
1 # This file contains all the app level configurations.
2
3
4 connection_timeout = 5000
5 shared_server_url = http://111d37c2.ngrok.io
6 shared_oam_url = http://c0463ae2.ngrok.io
7 ssl_cert_name = server.cer
```

[Home](#)

4. Guidance for Implementation Teams

4.1 CSRF Mitigation – Generating Nonces

A nonce is a pseudo random number that may be used only once. If a nonce is sent across in every request from the client to the server and the server validates the sent nonce every single time, then it mitigates the risk of Cross Site Request Forgery (CSRF).

The product provides a REST Service to generate nonces – each nonce can be used only once to identify each request uniquely, for each session. The product also has an inbuilt framework that will validate the nonce sent in the request.

Therefore post a successful login you need to make a call to <https://<Host>:<Port>/digx/v1/session/nonce> before you make a call to any other service. This service will return back an array of nonces in the response header. You can pick up any one nonce from the array and use it to send across the nonce required in a subsequent request. A nonce can be used only once. You need to discard it after usage.

Please note that unless you send across a nonce, the services that are accessed post login will not work.

4.2 Indirect Object Reference Implementation

4.2.1 What it means

It is a good security practice to hide sensitive data objects from the end user. Although the system needs to play around with sensitive data objects, it is recommended to refer to these sensitive data objects via pointers – tokens that temporarily point to the sensitive data objects but themselves do not contain any sensitive data.

For example consider a credit card application on the web which offers the following 2 transactions:

- Credit Cards Summary – Displays a list of all credit cards the user owns.
- Credit Card Details – Displays the details of one specific Credit Card that the user selects

The Credit Cards Summary page will typically list all credit card numbers in a masked format. Let's assume that the end user holds 2 Credit Cards C1 and C2. When the end user hits the Summary link, the server returns back the following in its response:

- a. Masked Credit Card Number C1 (visible to the user)
- b. Masked Credit Card Number C2 (visible to the user)
- c. Token T1 (not visible to the user)
- d. Token T2 (not visible to the user)

T1 and T2 are random tokens – difficult to guess – which the server has generated as proxies for C1 and C2 respectively. The server has internally stored this mapping of C1-T1 and C2-T2 somewhere. Please note that T1 and T2 are tied to the current session. The moment the session expires, T1 and T2 get discarded. Next time the user logs in, the server generates different tokens T1x and T2x for C1 and C2 respectively.

Whenever the user clicks on say Credit Card Details for C1, the client sends T1 to the server instead of C1, as a request parameter. The server internally figures out that the request is actually for C1 and processes the request accordingly.

Thus we refer to sensitive data indirectly via tokens that are generated with different values for every session.

4.2.2 How OBDX supports it

To implement the above mechanism the framework offers interception of both the request and the response. For the interception to work automatically, the sensitive fields holding the Personally Identifiable Information (PII) must be defined as a Java type, which extends the abstract class

```
com.ofss.digx.datatype.complex.MaskedIndirectedObject.
```

The abstract class essentially exposes 2 fields, namely value and displayValue. The value field holds the indirected value which is used for data transmission. Whereas, the displayValue field holds the masked value of the data.

The following data types are supported out-of-box:

1. `com.ofss.digx.datatype.complex.Account` - Account number
2. `com.ofss.digx.datatype.complex.Applicant` – The unique identifier to identify an applicant. Typically a party ID.
3. `com.ofss.digx.datatype.complex.ApplicationId` – The unique identifier of an application for account opening.
4. `com.ofss.digx.datatype.complex.ContentId` – The unique identifier for content such as documents for a party.
5. `com.ofss.digx.datatype.complex.CreditCard` – Credit Card Number
6. `com.ofss.digx.datatype.complex.DebitCard` – Debit Card Number
7. `com.ofss.digx.datatype.complex.Email` – Email ID
8. `com.ofss.digx.datatype.complex.Party` – The unique identifier for a party.
9. `com.ofss.digx.datatype.complex.PhoneNumber` – Phone Number
10. `com.ofss.digx.datatype.complex.SSN` – Social Security Number
11. `com.ofss.digx.datatype.complex.SubmissionId` – The unique identifier for a submission containing 1 or more applications for account opening.

To modify the existing/base product-masking pattern for any of the above data types, the following entries need to be copied/cloned from the table DIGX_FW_CONFIG_ALL_B to the table DIGX_FW_CONFIG_ALL_O and then modified as required in DIGX_FW_CONFIG_ALL_O.

Note: Please DO NOT MODIFY these entries IN DIGX_FW_CONFIG_ALL_B.

Data Type	Category ID	Property ID
Account	MaskingPattern	AccountNumberMasking
Applicant	MaskingPattern	ApplicantIdMasking
ApplicationId	MaskingPattern	ApplicationIdIdMasking
ContentId	MaskingPattern	ContentIdMaskingPattern
CreditCard	MaskingPattern	CreditCardNumberMasking
DebitCard	MaskingPattern	DebitCardNumberMasking
Email	MaskingPattern	EmailIdMasking
Party	MaskingPattern	PartyIdMasking

Data Type	Category ID	Property ID
PhoneNumber	MaskingPattern	PhoneNumberMasking
SSN	MaskingPattern	SSNMasking
SubmissionId	MaskingPattern	SubmissionIdMaskingPattern

The characters allowed in the making pattern are as below:

N – Keeps the character transparent. Does not mask.

Any other character – Replaces the character at the location to the character specified.

For example: XXXXXNNNN will keep the last 4 characters in clear text and mask the first 5 characters using the character 'X'.

4.3 Output Encoding

To mitigate inline Cross Site Scripting attacks, the product provides a framework to encode the data sent in the response. In the previous versions there was guidance in this section of the security guide on the steps needed to be followed to implement output encoding in your service response. However, in OBDX 17.2.0.0.0 this is something that is handled implicitly in the framework for all services, base as well as any custom services that you might write.

There is nothing that you need to do explicitly to achieve this.

4.4 Configuring the Hashing Algorithm for OTPs

The hashing algorithm used to hash the One Time Passwords used for payments and other transactions is configurable.

This has been made configurable since Cryptography is an ever changing field. Algorithms that are considered as secure today might be rendered insecure tomorrow by attackers doing research in cryptography. Making the hashing algorithm configurable will help the customer upgrade to the latest algorithm prevailing at the time in the future, with reasonable ease.

Configuration Steps:

1. Implement your own java class that contains the code to generate the hash.
2. The class should implement the interface `IHashHelper`.

```
package com.ofss.digx.framework.security.authentication.helper;

/**
 * Hashing is the transformation of a string of characters into a fixed-length value or key that represents the original.
 */
public interface IHashHelper {
    /**
     * Creates the hash value for the token. Hashing is one way encryption. Every string has a unique hash value. To add
     * extra security we add secret to original token value while creating hash.
     *
     * @param token
     *           {@link String} Value that needs to be hashed
     * @param secret
     *           {@link String} Salt added for hashing.
     * @return {@link String} type hashed value
     */
    public String createHash(String token, String secret);
}
```

3. Add/Edit the value of the column PROP_VALUE in the following entry in the database table DIGX_FW_CONFIG_ALL_B (PROP_ID = 'TOKEN_HASH_LOCAL' and CATEGORY_ID = 'authenticationConfig'). You need to provide the fully qualified class name of your java class.

```
select * from digx_fw_config_all_b where PROP_ID = 'TOKEN_HASH_LOCAL' and category_id = 'authenticationConfig'
```

Row 1	Fields
PROP_ID	TOKEN_HASH_LOCAL
CATEGORY_ID	authenticationConfig
PROP_VALUE	com.ofss.digx.framework.security.authentication.helper.HashGeneratorClassName
FACTORY_SHIPPED_FLAG	Y
PROP_COMMENTS	
SUMMARY_TEXT	To generate hashed token
CREATED_BY	ofssuser
CREATION_DATE	15-SEP-16 03:27:04.000000 PM
LAST_UPDATED_BY	ofssuser
LAST_UPDATED_DATE	15-SEP-16 03:27:04.000000 PM
OBJECT_STATUS	A
OBJECT_VERSION_NUMBER	1

4. Needless to say, an application server restart is needed.

4.5 Implementing a custom 2FA mechanism

1. You will need to write your own Java class to implement your own custom factor of authentication.
2. The class must be registered in the table DIGX_AU_AUTH_TYPE_MST. Choose a custom ID.

```
select * from digx_au_auth_type_mst
```

ID	PROVIDER_CLASS_NAME	NAME
1	com.ofss.digx.framework.security.authentication.provider.impl.HardTokenAuthenticationProvider	Hard Token
2	com.ofss.digx.framework.security.authentication.provider.impl.OTPBasedProvider	One Time Password
3	com.ofss.digx.framework.security.authentication.provider.impl.RandomNumBasedSoftTokenAuthenticationProvider	Random Number Based Soft Token Authentication
4	com.ofss.digx.framework.security.authentication.provider.impl.SoftTokenAuthenticationProvider	Soft Token
5	com.ofss.digx.framework.security.authentication.provider.impl.TimeBasedSoftTokenAuthenticationProvider	Time Based Soft Token Authentication
6	com.ofss.digx.framework.security.authentication.provider.impl.SecurityQueBasedAuthenticationProvider	Security Question

3. The custom class must implement the interface

```
com.ofss.digx.framework.security.authentication.provider.I2FactorAuthenticationProvider
```

4. To configure your custom authenticator as an additional option available to the admin during the 2FA configuration of transactions, set the custom ID used in Step 2 in the table DIGX_FW_CONFIG_VAR_B

```
select * from DIGX_FW_CONFIG_VAR_B where prop_id like '%SUPPORTED_AUTH_TYPE%'
```

	PROP_ID	ENV_ID	PROP_VALUE
1	SUPPORTED_AUTH_TYPE	OBDX	OTP~SOFT_TOKEN~SEC_QUE
2	administrator.SUPPORTED_AUTH_TYPE	OBDX	OTP~SOFT_TOKEN~SEC_QUE
3	corporateuser.SUPPORTED_AUTH_TYPE	OBDX	OTP~SOFT_TOKEN~SEC_QUE
4	retailuser.SUPPORTED_AUTH_TYPE	OBDX	OTP~SOFT_TOKEN~SEC_QUE
5	PC_CM_ME.SUPPORTED_AUTH_TYPE	OBDX	OTP~SOFT_TOKEN

- The configuration already seen in the above image suggests that an admin will have the option of setting one of OTP, Soft Token and Security Questions as an additional factor of authentication when configuring 2FA for user segments Retail, Corporate and Administrator.
- The PROP_ID that the system must look up in this table (DIGX_FW_CONFIG_VAR_B) is maintained in the table DIGX_FW_CONFIG_ALL_B against the PROP_ID SUPPORTED_AUTH_TYPE.
- If `$_PROPERTY_` is the value maintained against retailuser.SUPPORTED_AUTH_TYPE in the table DIGX_FW_CONFIG_ALL_B, then for retail users the application will look up the table DIGX_FW_CONFIG_VAR_B where PROP_ID = `$_PROPERTY_` to check what options are available to the admin.

```
select * from DIGX_FW_CONFIG_ALL_B where CATEGORY_ID='authenticationConfig'
AND PROP_ID LIKE '%SUPPORTED_AUTH_TYPE%';
```

	PROP_ID	CATEGORY_ID	PROP_VALUE
1	PC_CM_ME.SUPPORTED_AUTH_TYPE	authenticationConfig	\${PC_CM_ME.SUPPORTED_AUTH_TYPE}
2	SUPPORTED_AUTH_TYPE	authenticationConfig	\${SUPPORTED_AUTH_TYPE}
3	administrator.SUPPORTED_AUTH_TYPE	authenticationConfig	\${administrator.SUPPORTED_AUTH_TYPE}
4	corporateuser.SUPPORTED_AUTH_TYPE	authenticationConfig	\${corporateuser.SUPPORTED_AUTH_TYPE}
5	retailuser.SUPPORTED_AUTH_TYPE	authenticationConfig	\${retailuser.SUPPORTED_AUTH_TYPE}

4.6 Disable Password Protection for PDF Documents

By default all the pdf documents are password protected. There is configuration to override the feature by updating value for "PDF_PASSWORD_HELPER" in digx_fw_config_all_b table.

Note: Overriding this value will make all the documents which user can download (in pdf format) password non-protected.

Scripts:

```
update digx_fw_config_all_b set prop_value ='' where prop_id ='PDF_PASSWORD_HELPER'
and category_id ='documentConfig';
```

Note: Server/DB restart required.

[Home](#)