OneSpan Auth Tree Nodes

OneSpan Intelligent Adaptive Authentication (IAA) helps drive down fraud, improve customer experience, and meet compliance requirements by combining our powerful OneSpan Cloud Authentication (OCA) and Risk Analytics (RA) service.

By analyzing vast and disparate data acquired from user actions, device integrity, and transaction data in real time, the end-user can be dynamically presented with the appropriate authentication level for the current session or transaction.

Watch this video to learn how OneSpan intelligent adaptive authentication works.

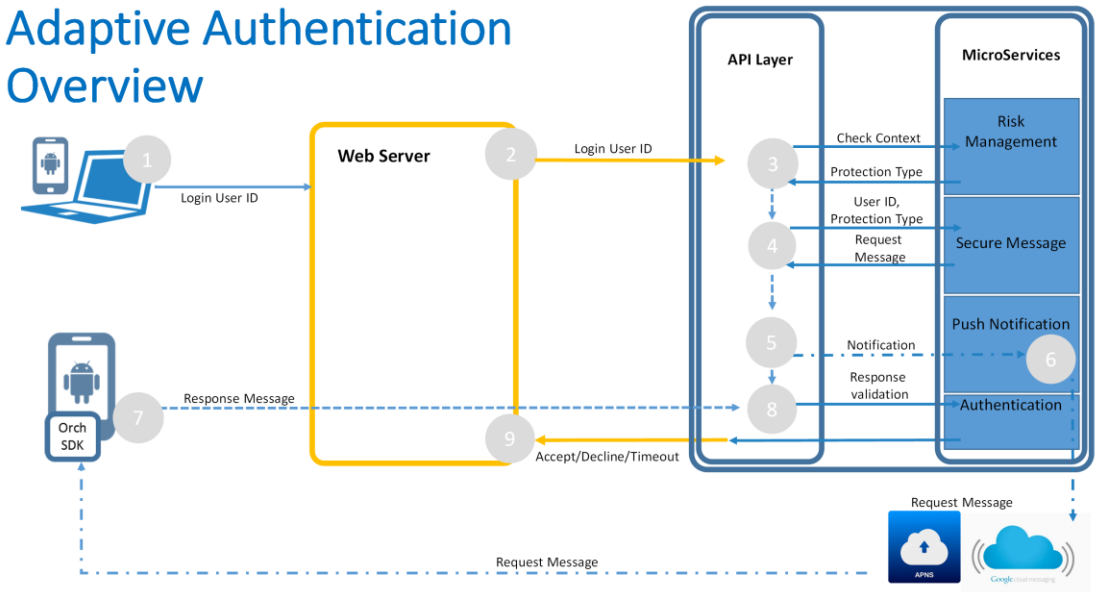
About OneSpan IAA

OneSpan Adaptive Authentication provides hosted solutions to test and build web and mobile applications for login and transaction signing flows.

Integration with OneSpan Adaptive Authentication is incredibly simple and extensible, as it will support future authentication technologies without the need to change anything in your integration code.

OneSpan intelligent Adaptive Authentication uses a 'trusted device' (e.g. a mobile phone using the OneSpan Mobile Security Suite SDKs) to provide strong multi-factor authentication whenever the risk associated with an action is high.

OneSpan Adaptive Authentication evaluates the risk related to an end-user request through vast data collected from the devices which is then scored with a sophisticated machine-learning engine. Depending on the risk, OneSpan Adaptive Authentication can dynamically adjust the end-user security requirements by requesting step-up authentication for higher risk transactions using various configurations of device-based, PIN-based, fingerprint-based, or face recognition-based authentication as needed to fully secure transactions.



About OneSpan OCA

As important components of Adaptive Authentication, OneSpan Cloud Authentication (OCA) and Risk Analytics (RA) can also be leveraged as stand-alone services.

With OCA, OneSpan offers a comprehensive solution for strong authentication in the Cloud, integrating Push Notification and Secure Channel operations.

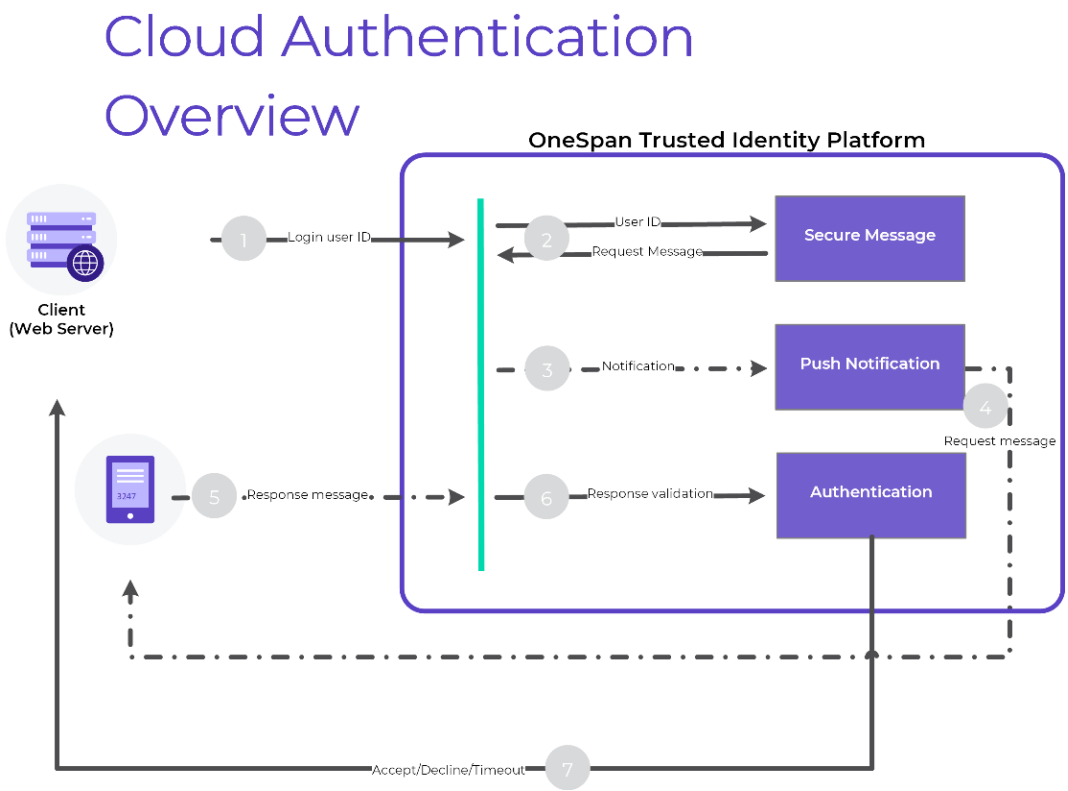
OneSpan Auth Tree Nodes facilitates developers to integrate with below OCA use cases:

User authentication

- With static password
- With offline one-time password (OTP)
- Secure Channel-based
- Push Notification-based
- FIDO-based

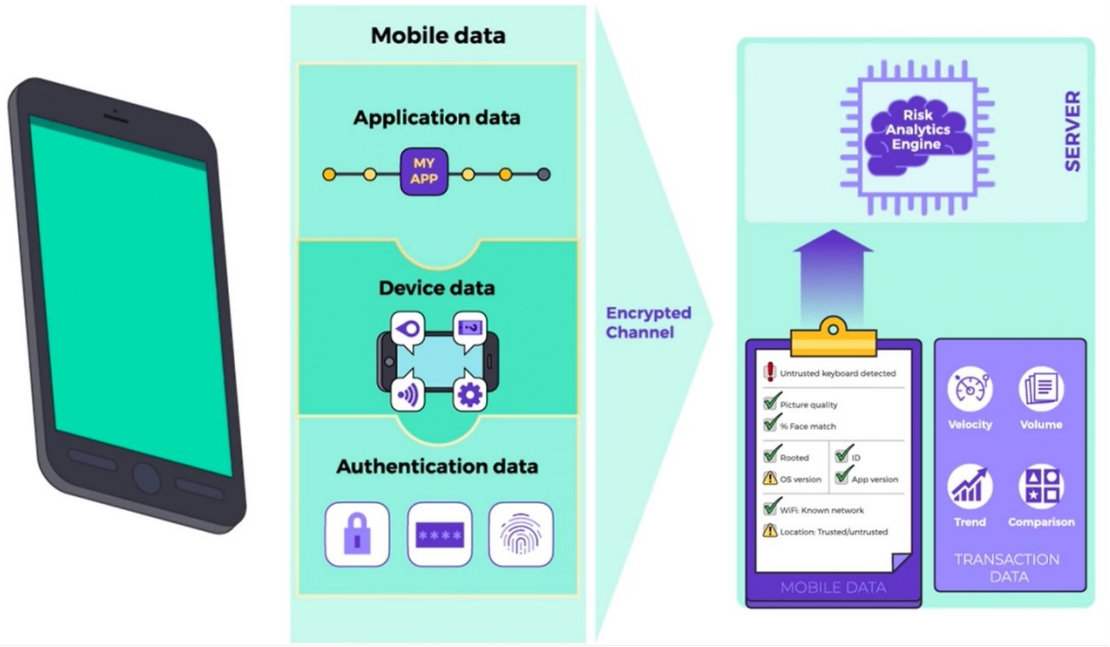
Transaction data signing

- Offline signing
- Secure Channel-based signing
- Push Notification-based signing
- FIDO-based signing (UAF only)



About OneSpan RA

OneSpan Risk Analytics (RA) leverages the latest machine learning and sophisticated data modeling. It analyzes data in real time to produce a transaction risk score. The risk score can then drive intelligent workflows that trigger immediate action based on pre-defined and/or customer-defined security policies and rules.



The combination of intelligent automation and risk scoring streamlines processes and helps prevent account takeover, new account fraud, and mobile fraud.

In below sections, we will showcase you how to configure the Risk Analytics Presentation Service where you can monitor and design rules, factors & actions to prevent various types of fraud attack and to manage risks for corporate banking applications across multi-channels.

Installation

Download the current release here.

Copy the jar file to the "../web-container/webapps/openam/WEB-INF/lib" folder where AM is deployed, then restart the AM. The nodes will be available in the tree designer.

Before You Begin

For OneSpan IAA & RA Users:

1. Create an OneSpan Developer Community account.

2. Once logged in the community portal, you'll be able to create an OneSpan IAA Sandbox account.

3. Set up a mobile application integrated with the Mobile Security Suite. As an easy start up, you can install the OneSpan IAA Demo App on your phone.

4. In order to test through the functionalities of the Tree Nodes, we will set up some simple rules in you Risk Analytics Presentation Service:

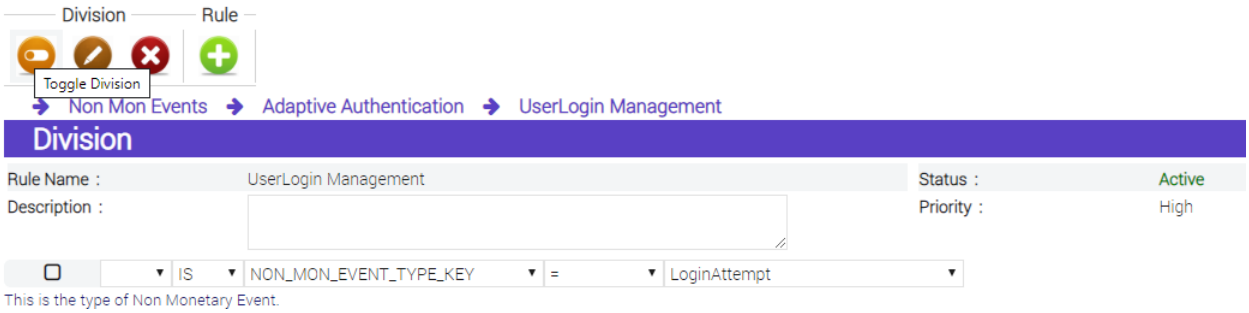
Rule 1: When an end user tried to login, send an extra PIN challenge to user's trusted device.

- In the Risk Analytics Presentation Service, navigate to DESIGN RULES & ACTIONS > Rule Management > Rules.

- Select "Non Mon Events" (**Hierarchy** level) > "Adaptive Authentication" (**Campaign** level).

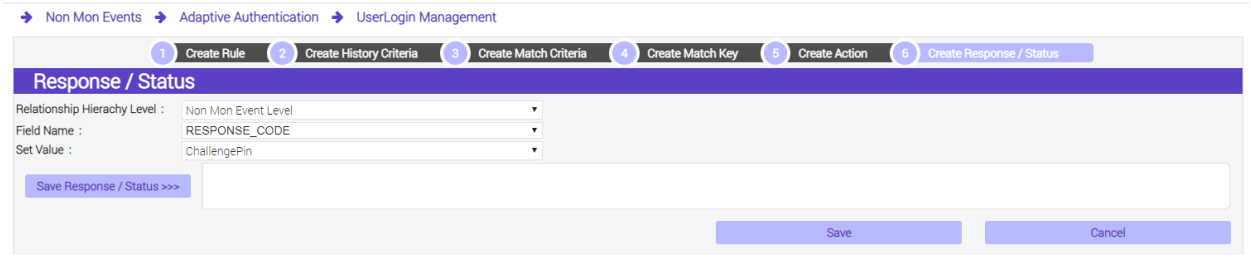
- Create a new **Division** named "UserLogin Management" with a criteria of "is NON_MON_EVENT_TYPE_KEY = LoginAttempt"

- Toggle the newly created Division.



- Create a **Rule** named "ChallengePIN". No need to add specific criteria here, so directly click "Save & Next", and skip creating "History Criteria", "Match Criteria", "Match Key", "Action", until the "Response / Status" where we'll set the response value as "ChallengePin"

- Toggle the newly created Rule.

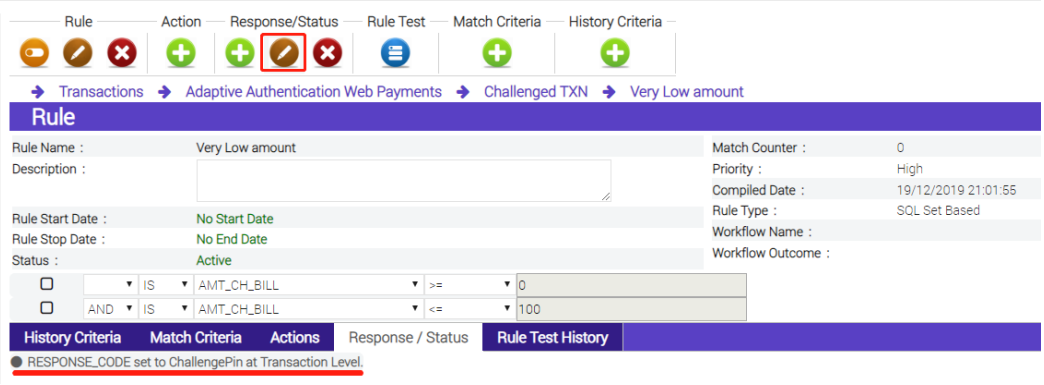


Rule 2: When an end user tried to validate a transaction event, if the transaction amount was below 100, send an extra PIN challenge to user's trusted device.

- In the Risk Analytics Presentation Service, navigate to DESIGN RULES & ACTIONS > Rule Management > Rules.

- Select "Transactions" (**Hierarchy** level) > "Adaptive Authentication Web Payments" (**Campaign** level) > "Challenged TXN" (**Division** level) > "Very Low amount" (**Rule** level).

- Tweak the existing rule by changing the response to "ChallengePin"



Tips:

- Face/Touch biometrics are also the available options, simply change the response to the particular challenge and don't forget to enroll your face/fingerprint at the Demo App.

- Explore the Risk Analytics Admin Guides for more details.

For OneSpan OCA Users:

1. Create an OneSpan Developer Community account.

2. Once logged in the community portal, you'll be able to create an OneSpan IAA Sandbox account.

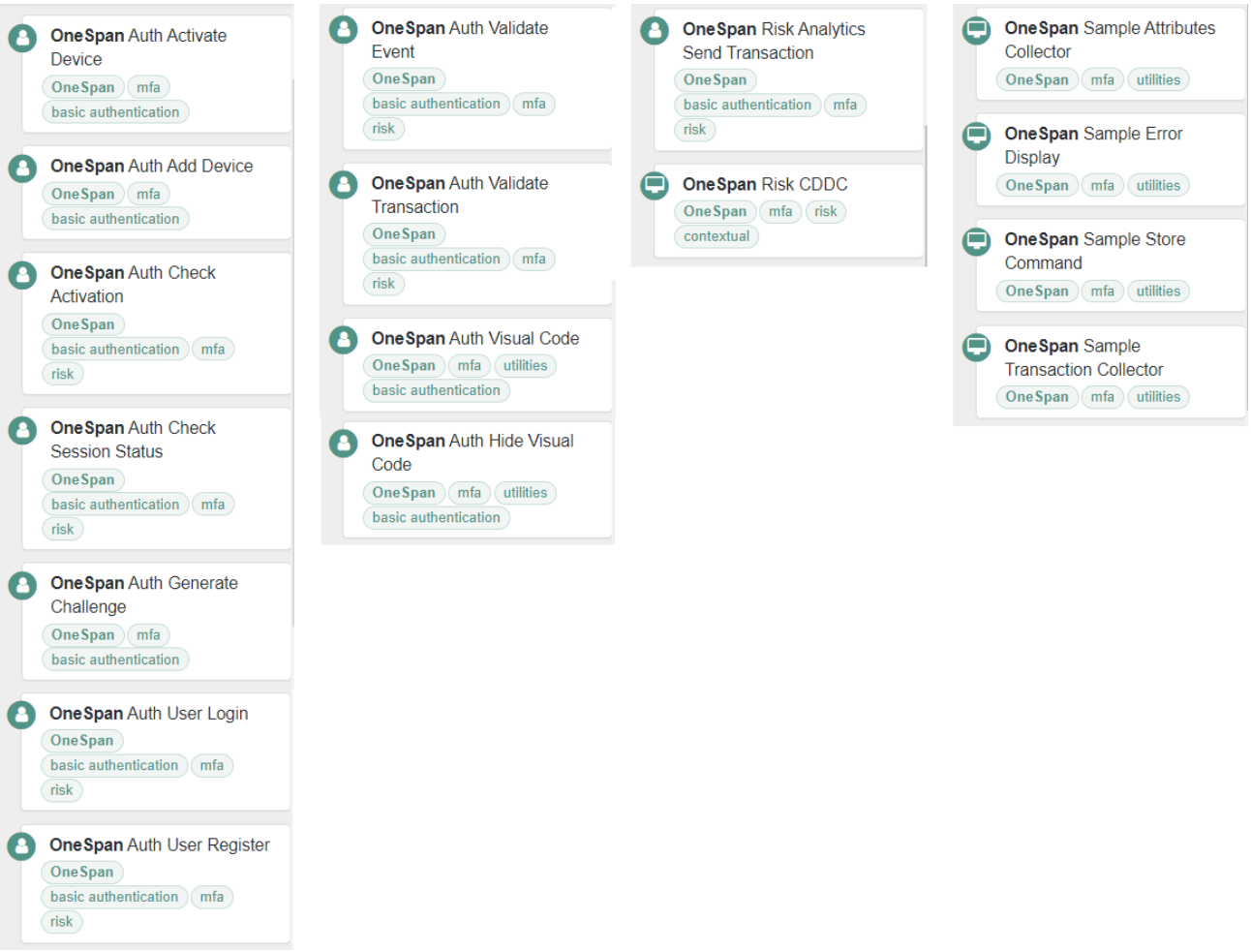
3. Install a mobile application which acts as the Digipass authenticator. You can follow either of below two options:

- To customize and build your own Mobile Authenticator Studio app. As a quick start, you can download a sample Mobile Authenticator app through our Demo Site after entering your IAA domain.

- Download "OneSpan Mobile Authenticator" from Google Play or Apple Store.

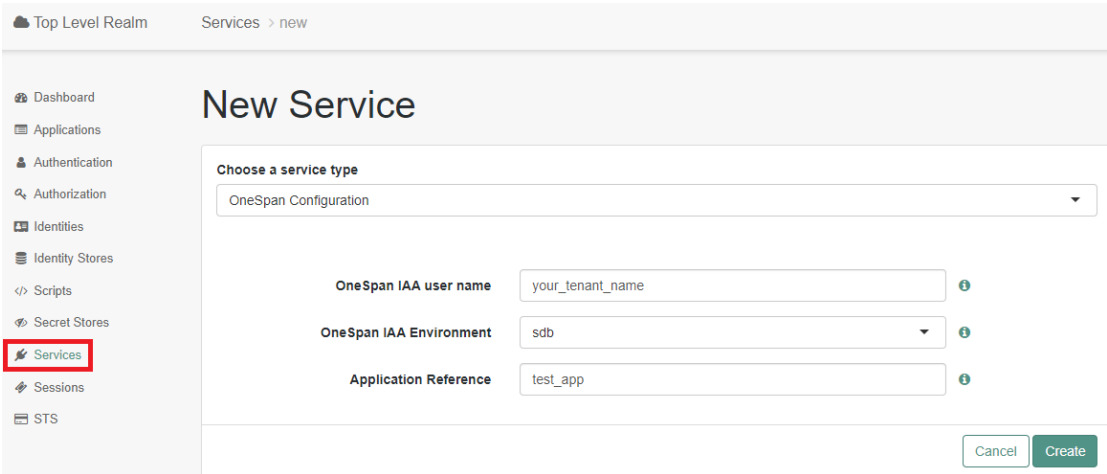
Nodes Overview

The OneSpan Auth Tree Nodes contains 1 Auxiliary Service, 13 nodes, and 4 demo nodes which will only be used for testing purpose. More details will be covered in next sections.



Auxiliary Service

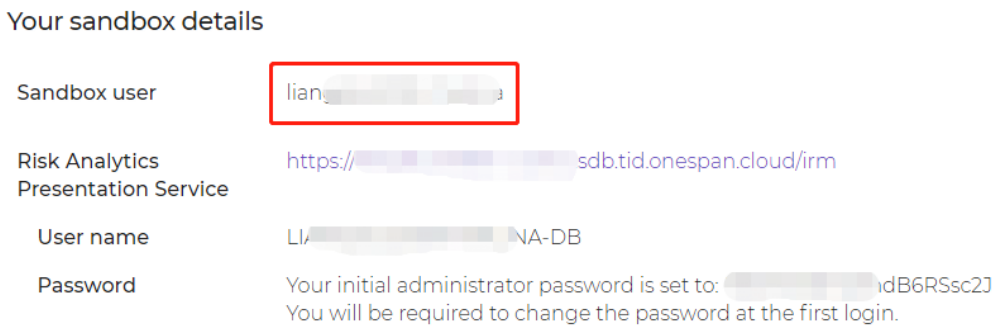
The node provides a realm-specific service named “OneSpan Configuration”, where allows you to specify the OneSpan IAA common configurations.



- In your AM dashboard, navigate to REALMS > your_realm > Services

- Add a new service and choose “OneSpan Configurations” from the dropdown list

- You can find your OneSpan IAA user name from the IAA Sandbox index page



-The fourth level domain indicates your IAA environment. Take a Sandbox account for example (https://tenant_name.sdb.tid.onespan.cloud), the IAA environment is “sdb”.

Quick Start

Below sample trees help you to address the most common use cases. Before start, make sure you’ve followed below steps:

(1) Add an “OneSpan Configuration” auxiliary service.

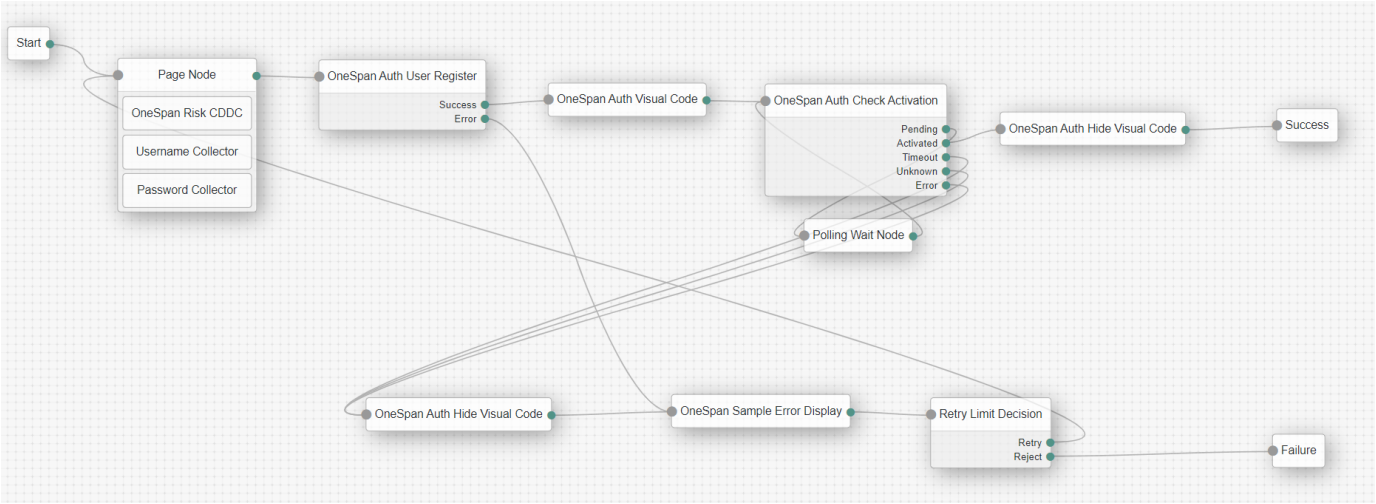
(2) Reproduce below sample trees using either of the below two methods:

- Import the JSON files under the “/sample” folder through AM Treetool.

- Follow the design and manually create the trees.

For OneSpan IAA Users:

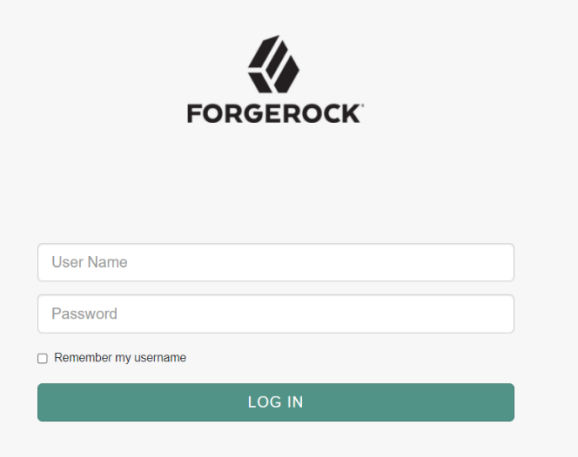
1. User Registration



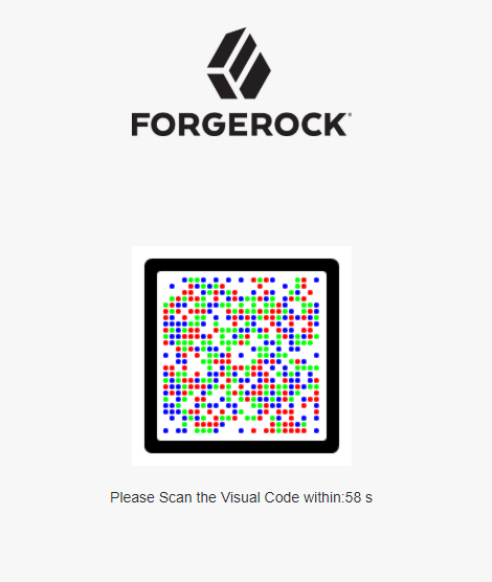
Hit below link in your browser and start the registration process:

https://{your_instance_url}/openam/XUI/?realm=/&service=OneSpan-XUI-Adapative-Authentication-User-Register-Sample-Tree#login

You will be prompt to input the username and password. (Password should include at least one lowercase, one uppercase, one number, 8 digits in length, and doesn't include part of the username for any 3 characters)

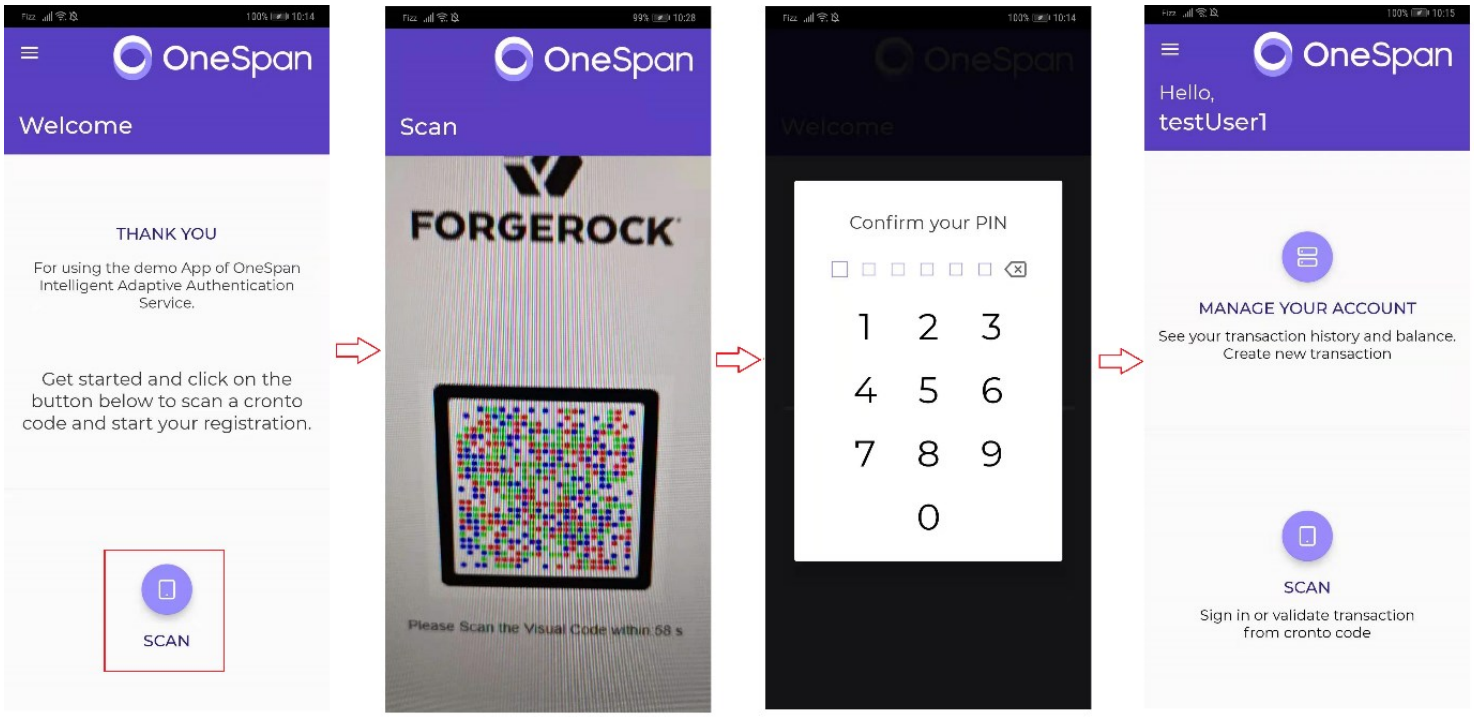


Once the Risk Analytics has accepted the user registration, the IAA service creates a Digipass user account and awaits a trusted device to activate the license with an activation token, which is rendered as a visual code.

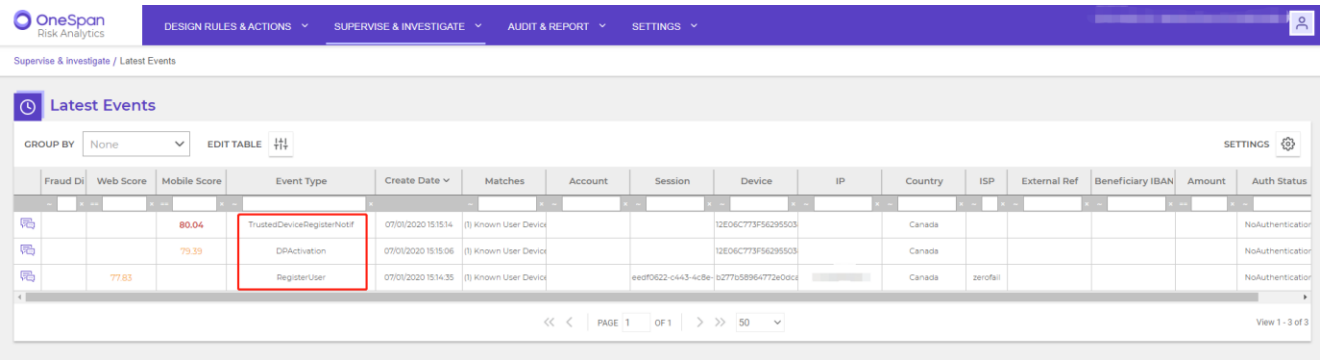


Launch the Sample AAS Demo App in your mobile, agree the License Agreement and enable the required mobile permissions. Click the "SCAN" button in the main screen and scan the above visual code, the app will prompt you to enter a 6 digits security pin twice after successfully detected the image.

When user completes the registration process, the AAS demo app will jump to the user page and the browser will be redirected to the success URL.

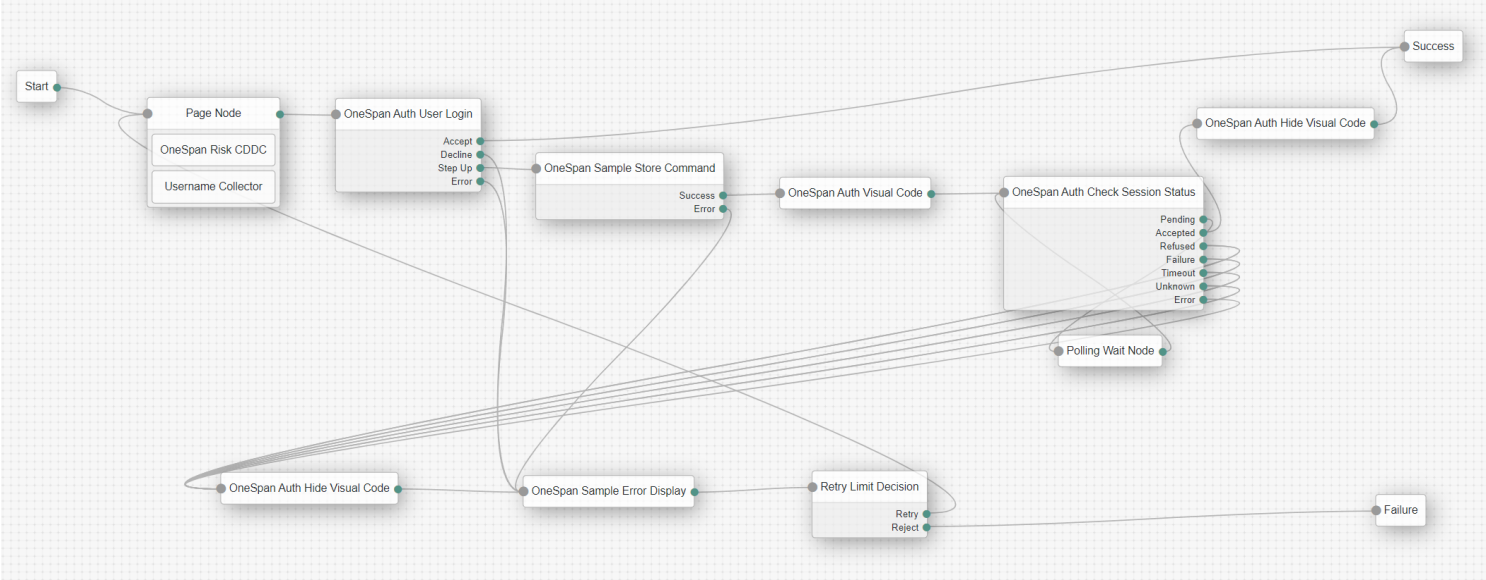


Log onto your Risk Analytics system and navigate to SUPERVISE & INVESTIGATE > Latest Events, you will find the user register process has been logged by the system with all the relevant information.



Explore the Integrating end-user registration and Digipass activation guide for more detailed information.

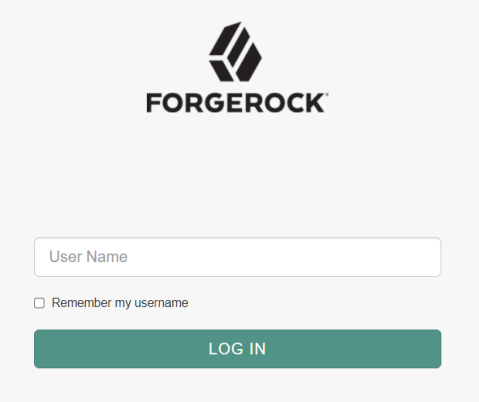
2. User Login



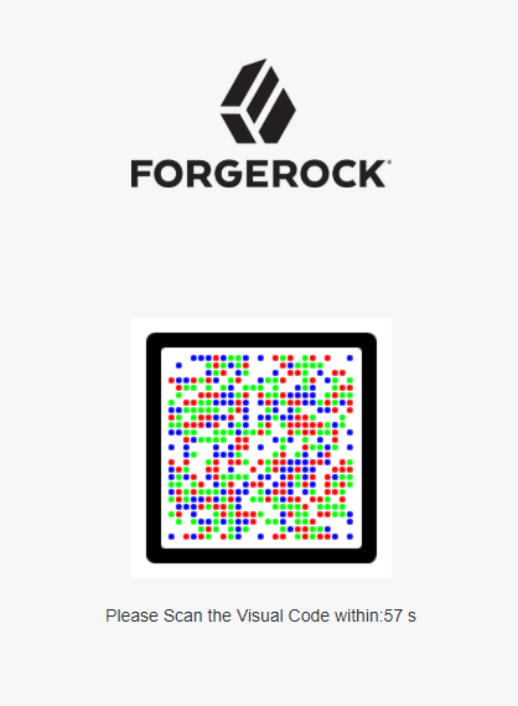
Hit below link in your browser and start the authentication process:

https://{your_instance_url}/openam/XUI/?realm=/&service=OneSpan-XUI-Adapative-Authentication-User-Login-Sample-Tree#login

You will be prompt to input the username registered above.

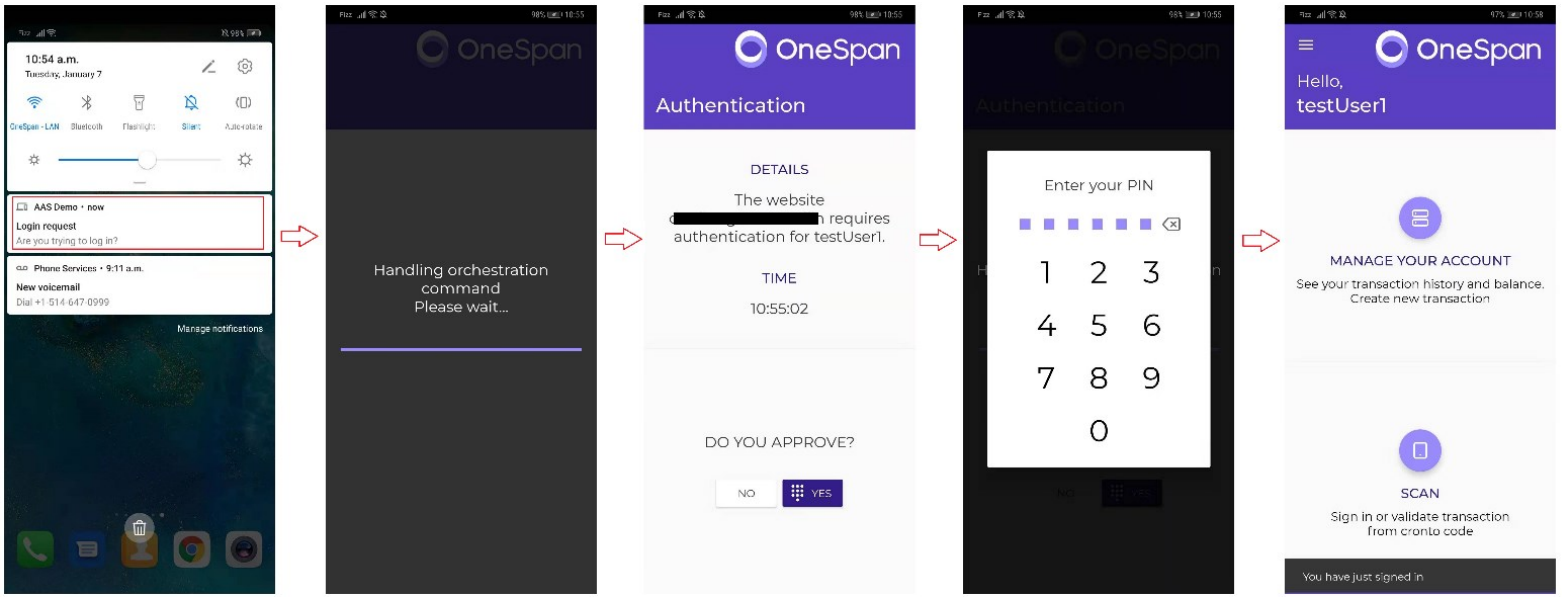


The User Login service checks the browsing context and analyzes the risk of the end-user login. If there's no user step up configured in the Risk Analytics Representation Service, the outcome will directly fall within Accept or Decline depending on the risk score. However, because of the rules we've pre-set, the User Login service will challenge the end user by sending a remote authentication request to the trusted device associated with the user.



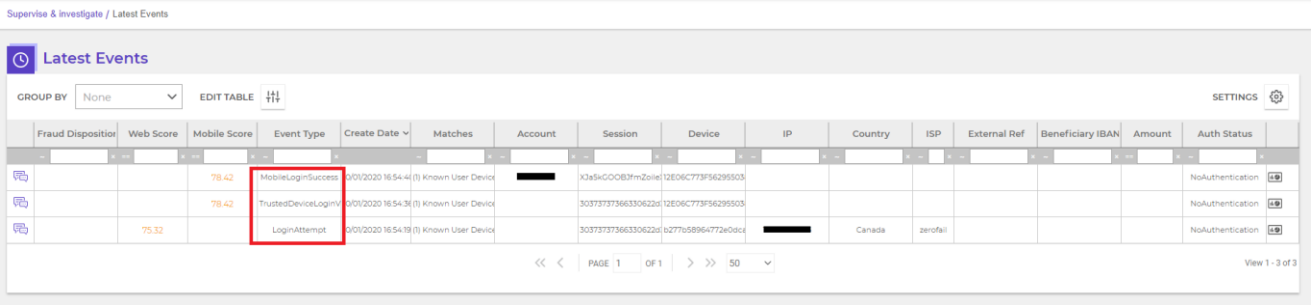
At this point, your trusted device should have received a notification prompting for step up authentication. In case you missed the notification, you can also scan the visual code which contains the same request information and manually trigger the authentication process.

Either way, the AAS demo app will handle the orchestration command, display the event details and prompt you to pass the challenge.



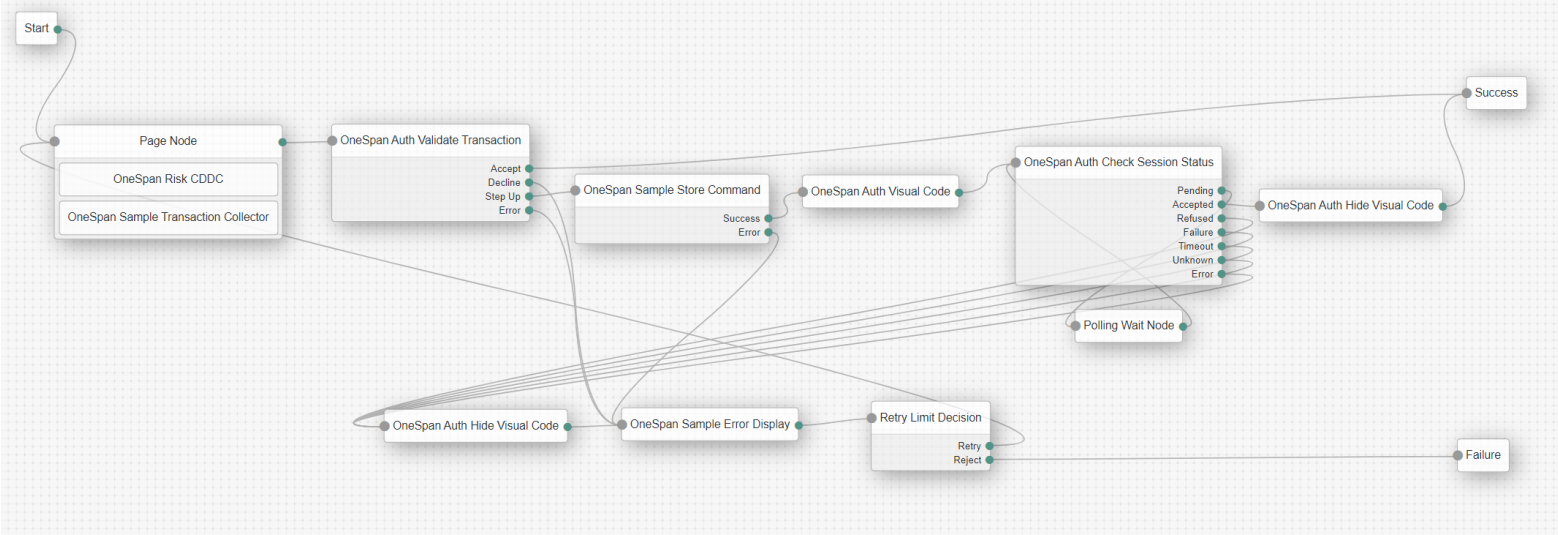
When user completes the authentication process, the AAS demo app will indicate that the event was successfully authenticated and the browser will be redirected to the success URL.

Now if you log onto your Risk Analytics system and navigate to SUPERVISE & INVESTIGATE > Latest Events, you will find the user login process has been logged by the system with all the relevant information.



Explore the [Integrating end-user login via notification] guide for more detailed information.

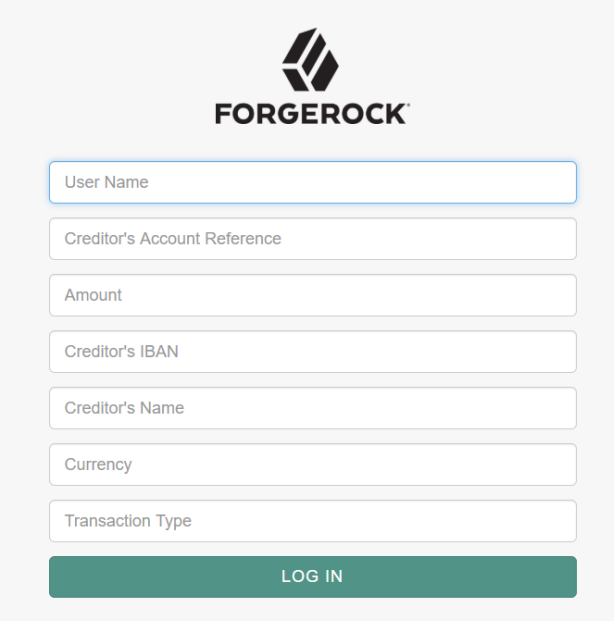
3. Validate Transaction Event

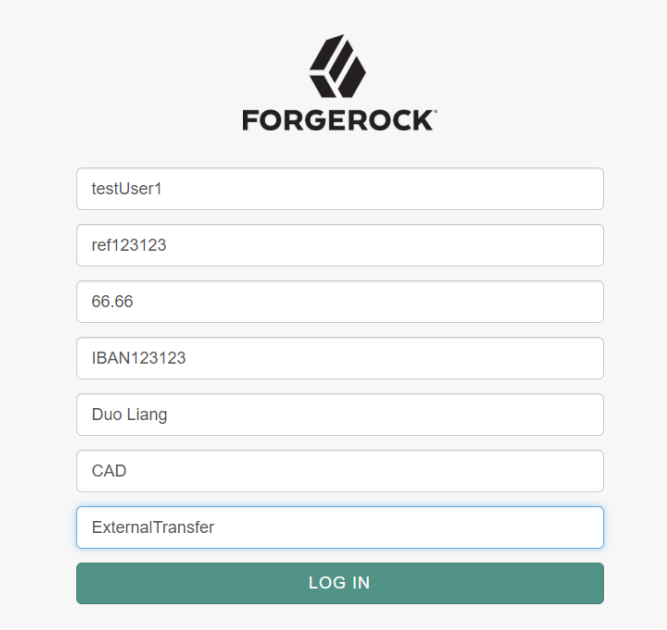


OneSpan IAA also provides the capability to evaluate the risk before an end-user tried to send a transaction, depending on the transaction details and the browser / mobile's context. Build below link and kick off the validation process:

https://{your_instance_url}/openam/XUI/?realm=/&service=OneSpan-XUI-Adapative-Authentication-Validate-Transaction-Sample-Tree#login

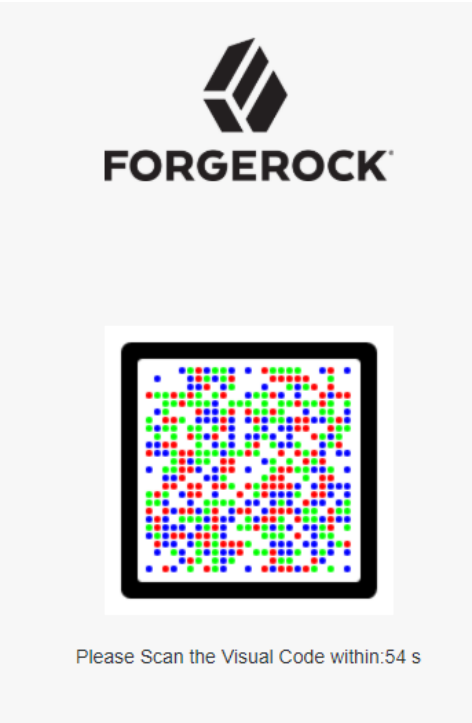
In this quick demo, we will use the "OneSpan Sample Transaction Collector" to pass in all the necessary transaction details.



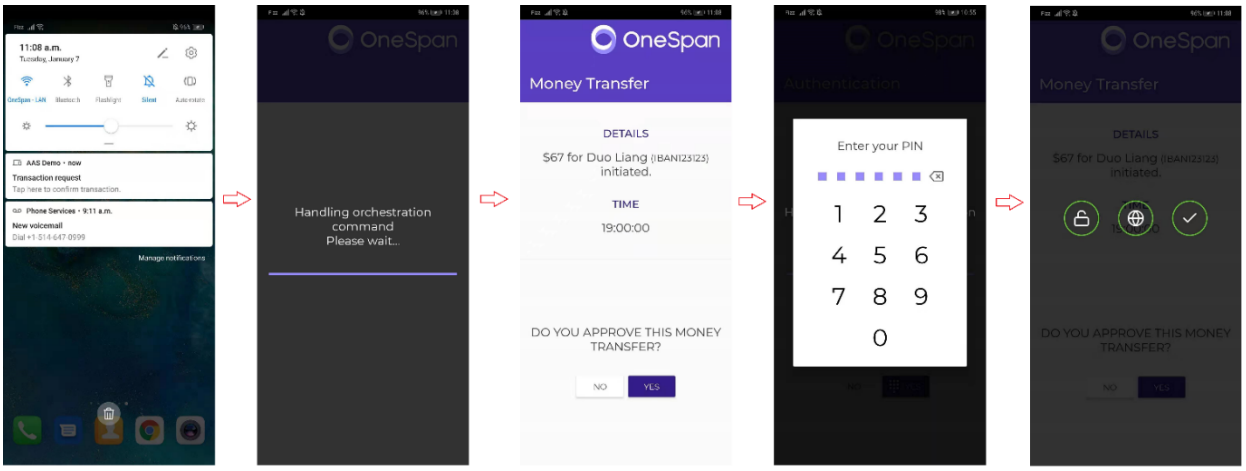


You can find all the available transaction types from the API Specification page > TransactionInput Schema > transactionType Attribute.

Similarly, the Validate Transaction service checks the browsing context and analyzes the risk of the end-user's request. If there's no user step up configured in the Risk Analytic Representation Service, the outcome will directly fall within Accept or Decline depending on the risk score. However, because of the rules we've pre-set, the Validate Transaction service will challenge the end user by generating a remote authentication request via push notification.

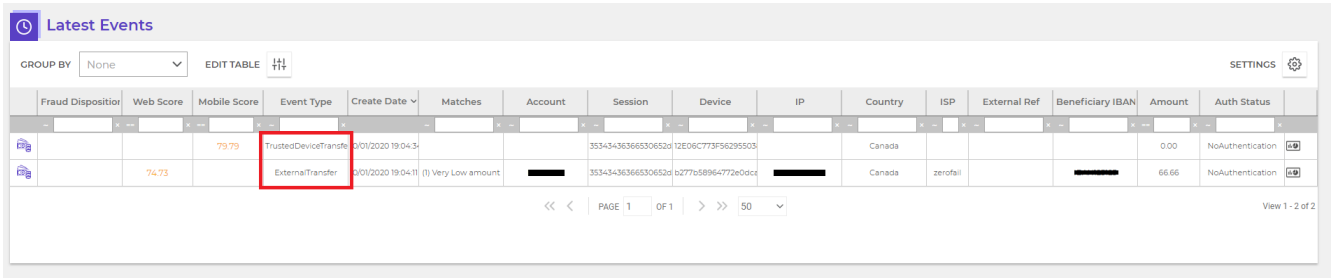


The end user can either scan the visual code or via the mobile notification to kick off the authentication process.

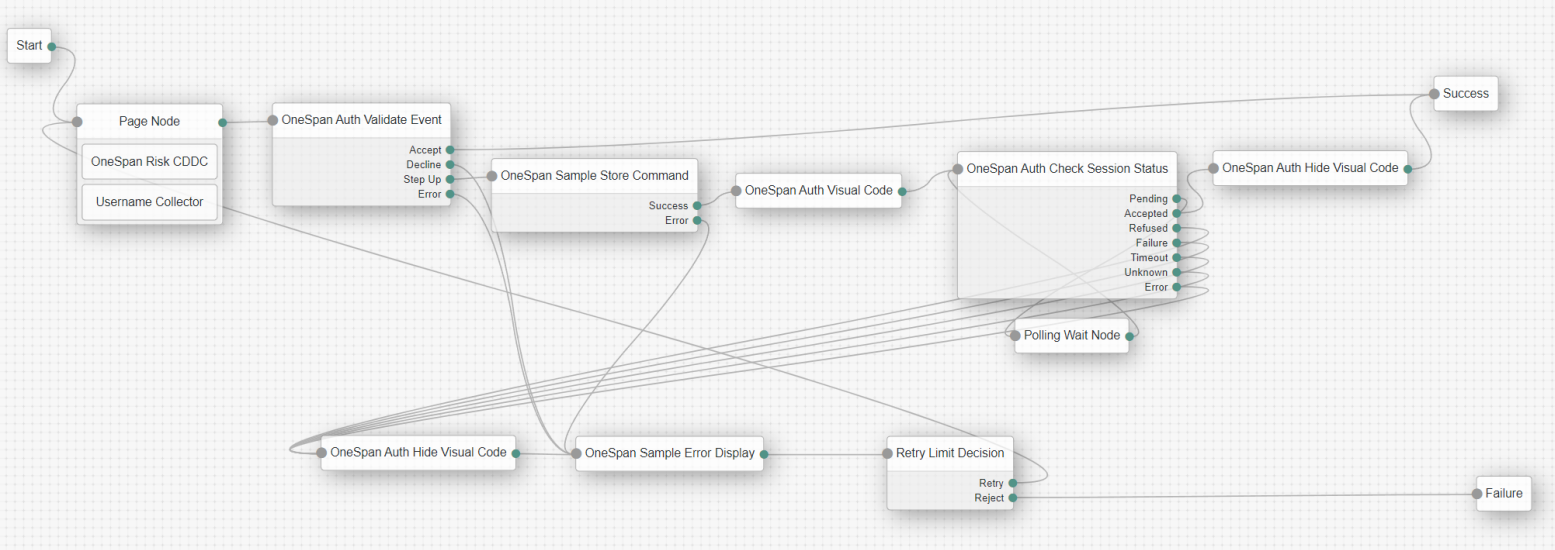


After confirming the transaction details and pass the challenge, the demo app will indicate that the event was successfully authenticated and the browser will be redirected to the success URL.

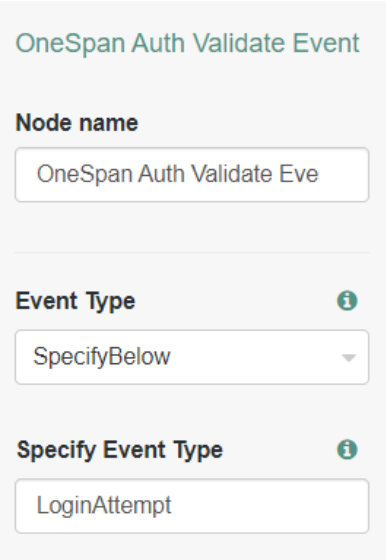
Now if you log onto your Risk Analytics Presentation Service and navigate to SUPERVISE & INVESTIGATE > Latest Events, you will find that this transaction event has been logged by the system with all the relevant information.



4. Validate Non-Monetary Events



For other non-monetary events, OneSpan Auth nodes provide a general node to validate these requests. You can either hard code the event type at the node configuration or store the event type in Shared State during the run time. As a quick start, we will hard code the event type as “LoginAttempt”.



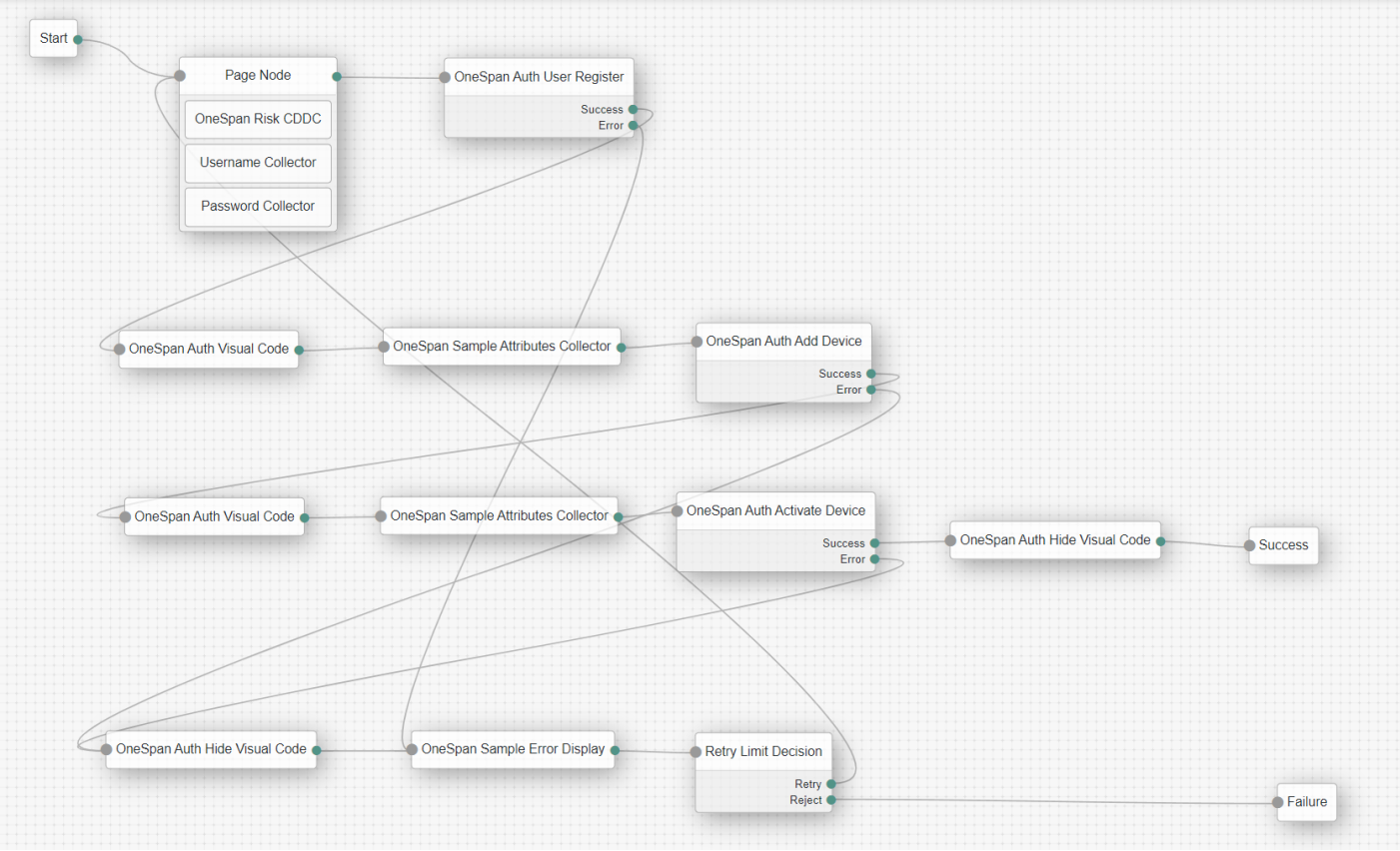
You can find all the available Event Types from the API Specification page > AdaptiveEventValidationInpu Schema > eventType Attribute.

Once you started the authentication process with below link, the authentication flow should be exactly the same as the second use case “User Login”:

https://{your_instance_url}/openam/XUI/?realm=/&service=OneSpan-XUI-Adapative-Authentication-Validate-Event-Sample-Tree#login

For OneSpan OCA Users:

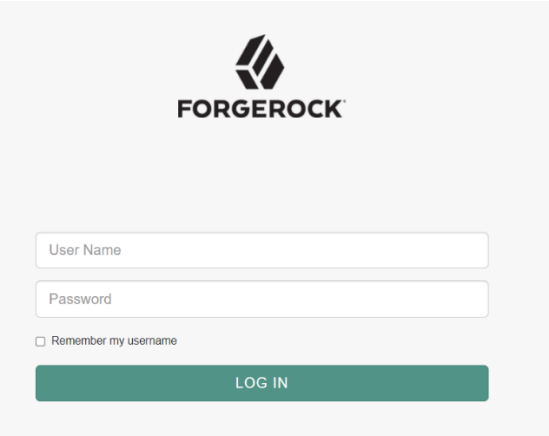
1. Offline User Registration and Digipass Activation



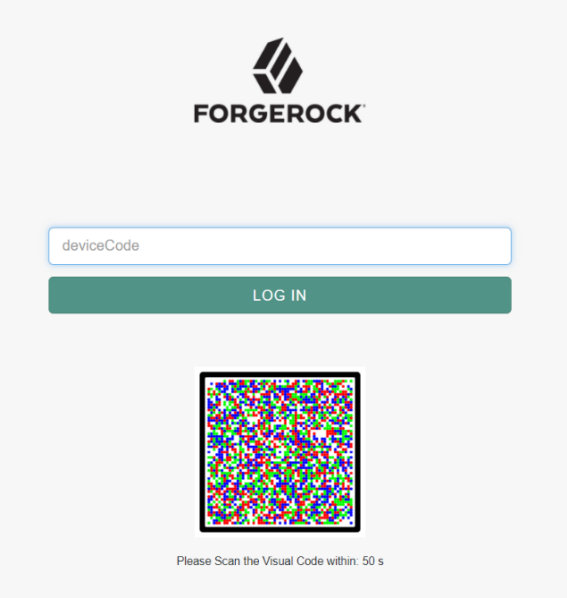
Hit below link in your browser and kick off the registration process:

https://{your_instance_url}/openam/XUI/?realm=/&service=OneSpan-XUI-Cloud-Authentication-User-Register-Sample-Tree#login

You will be prompted with the username and password. (Password should include at least one lowercase, one uppercase, one number, 8 digits in length, and doesn't include part of the username for any 3 characters)

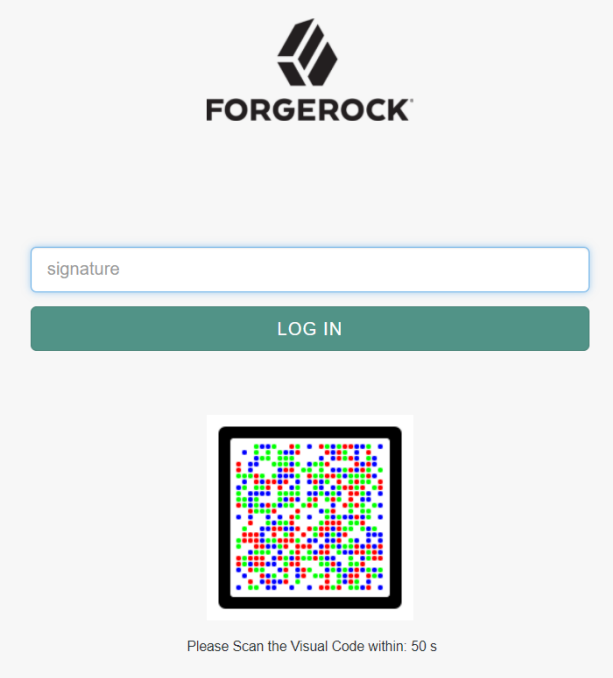


Behind the scene, the OCA service creates a Digipass user account and awaits a Digipass Authenticator to activate the license. The activation password will be rendered as a visual code:

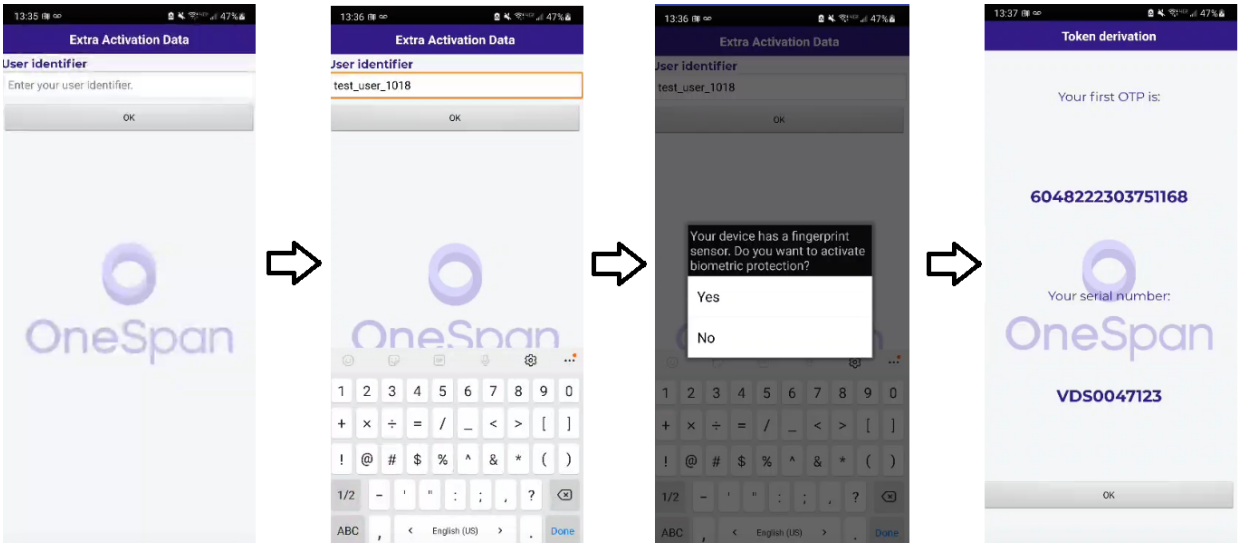


In this quick start guide, we will use a Mobile Authenticator Studio (MAS) app which is built and downloaded from the OCA Sample Site. Launch the app and scan the above Cronto image, a device code will be received after the visual code was successfully detected.

Enter the device code and a second Cronto image will be displayed:



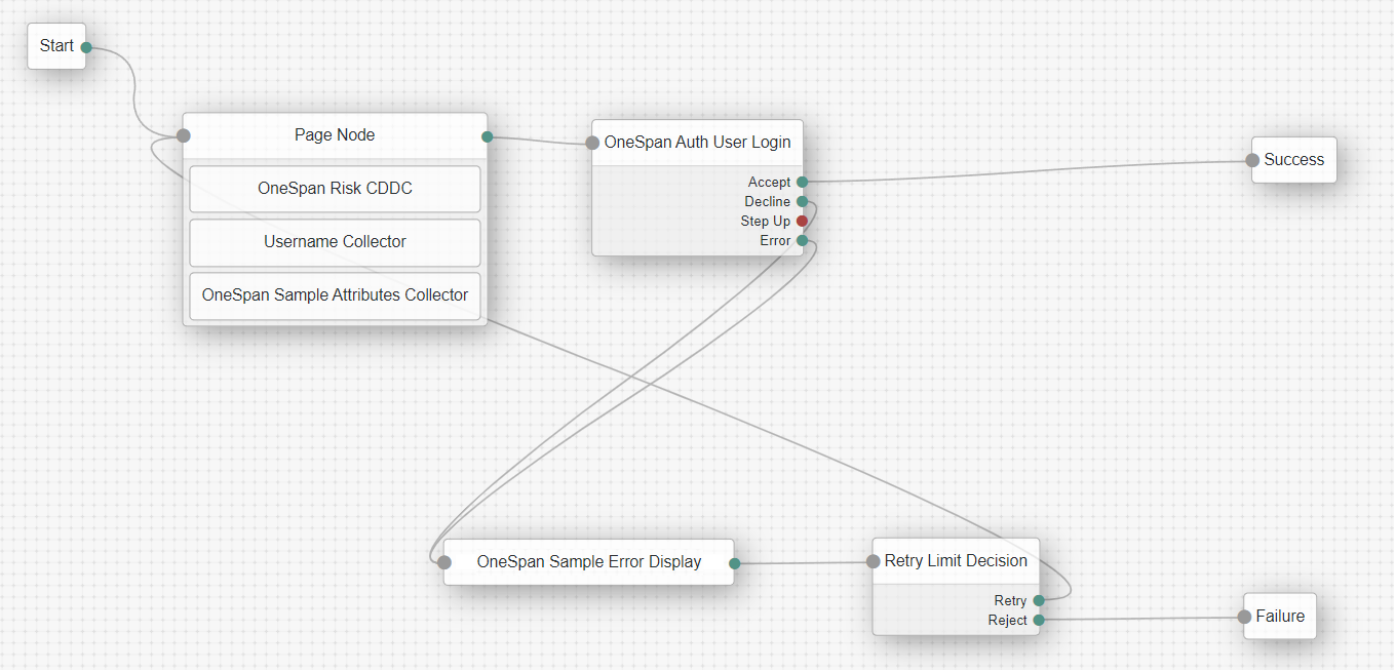
Scan the visual code with the MAS app again, the app will prompt for the same username as well as to activate fingerprint protection. Afterwards, the first OTP will be obtained for the final activation:



Enter the first OTP as the signature and if the activation process succeeded, the MAS app will jump to the Applications page and the browser will be redirected to the success URL.

Explore the Integrating Offline End-User Registration and Digipass Activation guide for more detailed information.

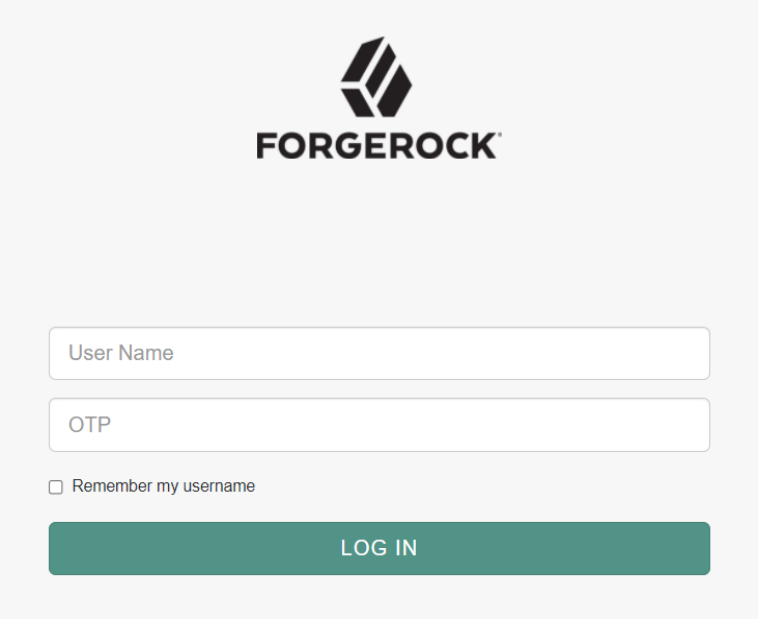
2. User Login with one-time password (OTP)



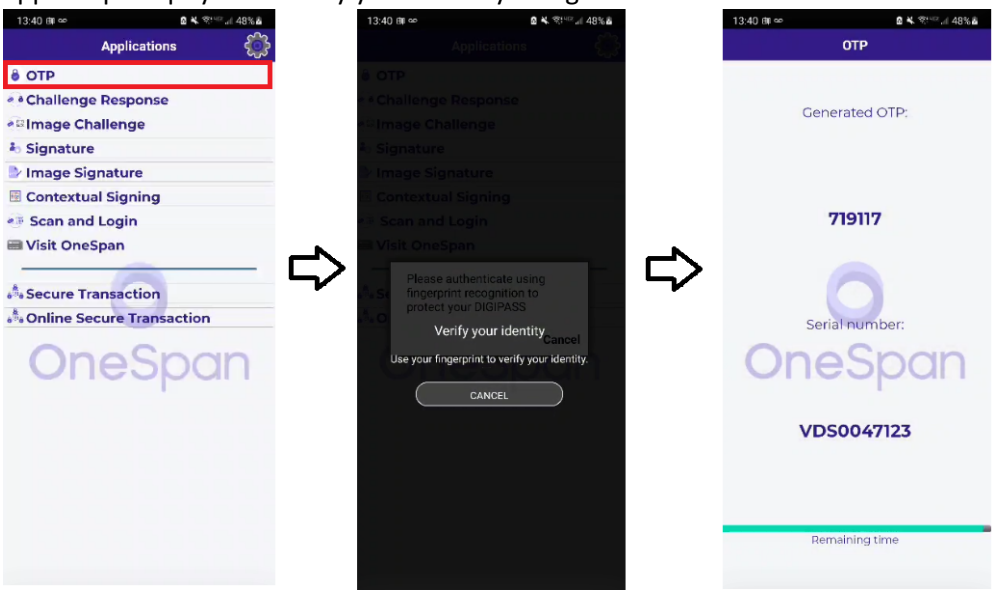
Hit below link in your browser and start the authentication process:

https://{your_instance_url}/openam/XUI/?realm=/&service=OneSpan-XUI-Cloud-Authentication-User-Login-OTP-Sample-Tree#login

You will be prompted with the username and an OTP:



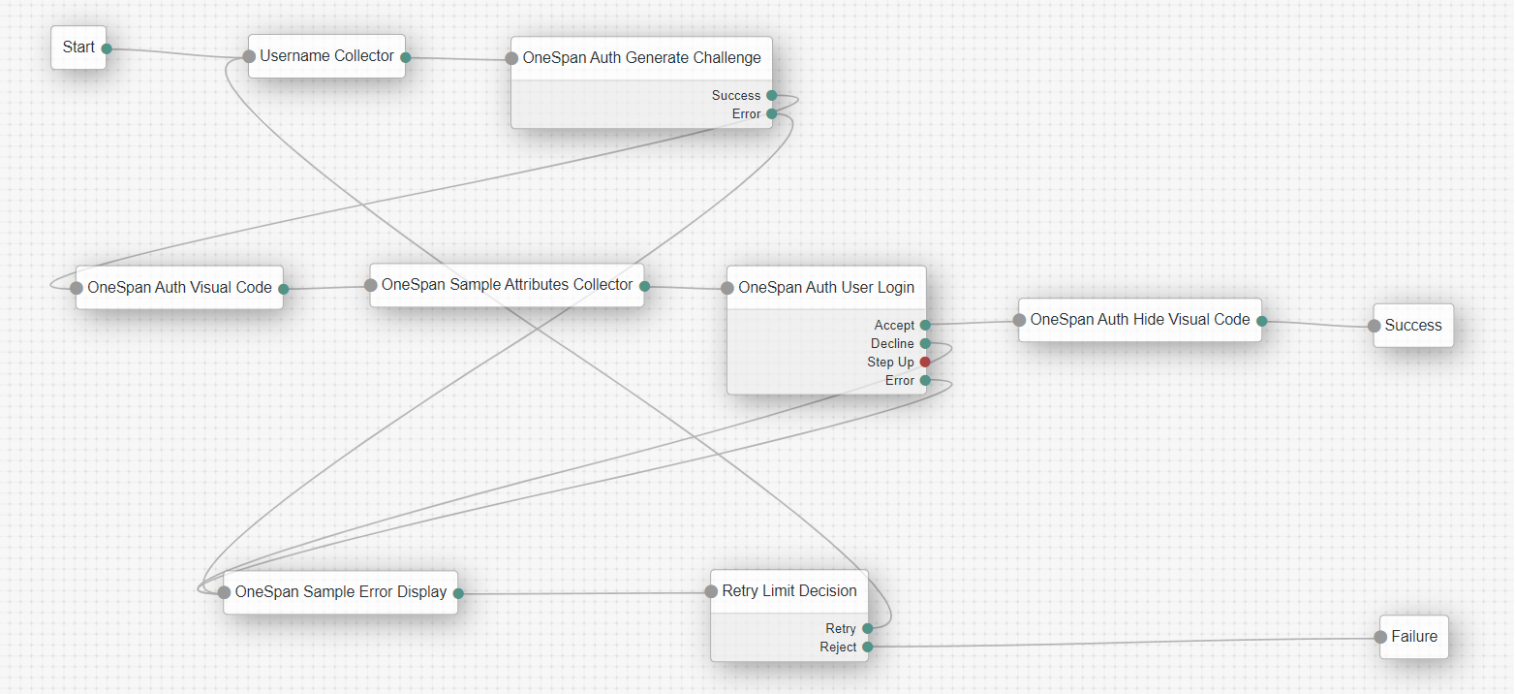
In order to receive an OTP, launch the MAS app and click the “OTP” option from the applications list, the app will prompt you to verify your identity and generate an OTP.



The OCA Login service validates the OTP and returns the validation result. If the authentication has succeeded, the browser will be redirected to the success URL.

Explore the Integrating end-user login with one-time password (OTP) guide for more detailed information.

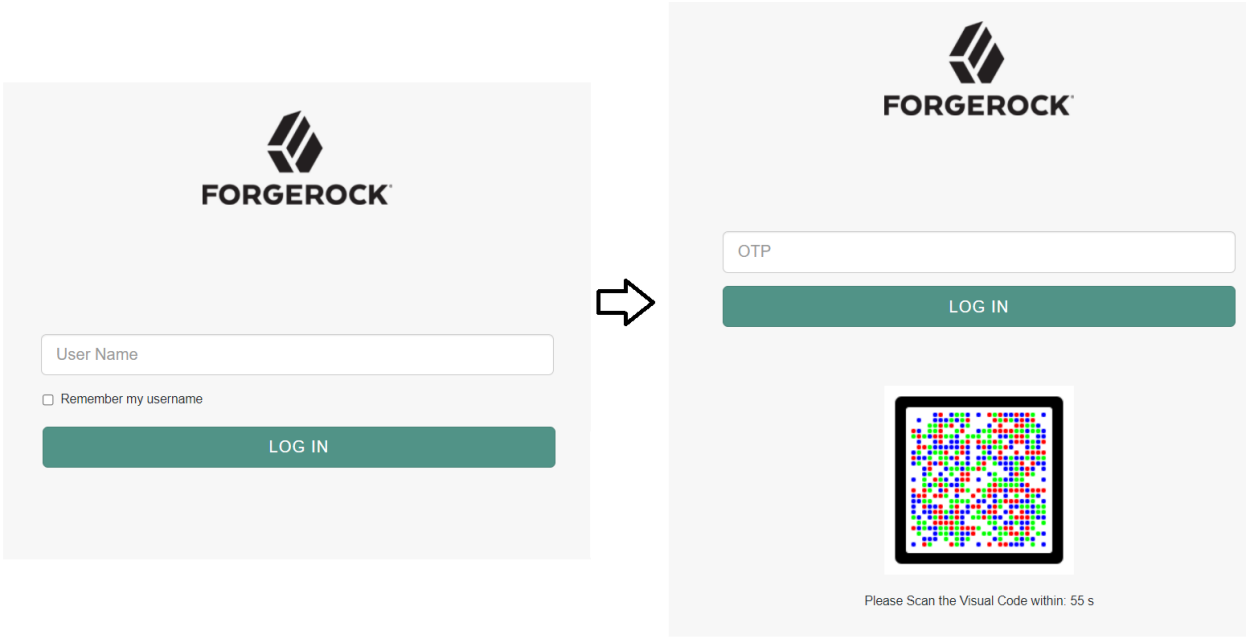
3. User Login with Challenge/Response (CR)



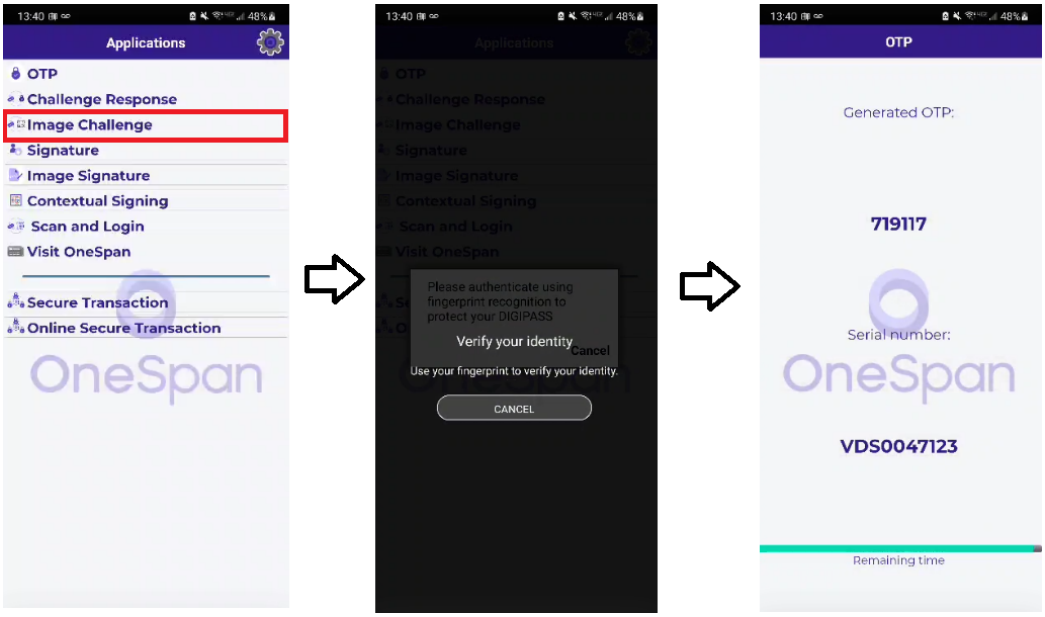
Hit below link in your browser and start the authentication process:

https://{your_instance_url}/openam/XUI/?realm=/&service=OneSpan-XUI-Cloud-Authentication-User-Login-Challenge-Response-Sample-Tree#login

After entering the username, a Cronto image carrying the challenge code will be displayed:



In order to retrieve the corresponding response, launch the MAS app and click the “CR” option from the applications list, the app will prompt you to verify your identity and generate a response code based on the challenge:

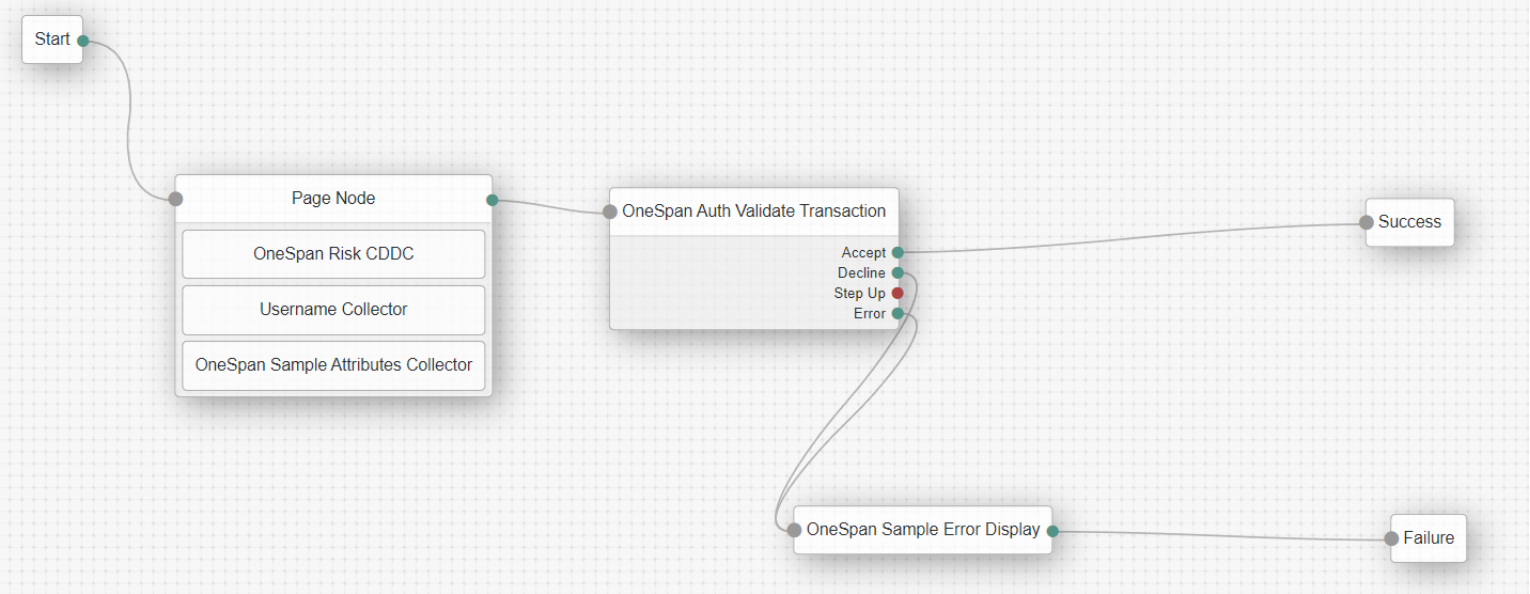


Input the generated OTP as well as the username to the authentication page, and the OCA Login service will validate the OTP and return the validation result.

If the authentication has succeeded, the browser will be redirected to the success URL.

Explore the [Integrate end-user login with Challenge/Response] guide for more detailed information.

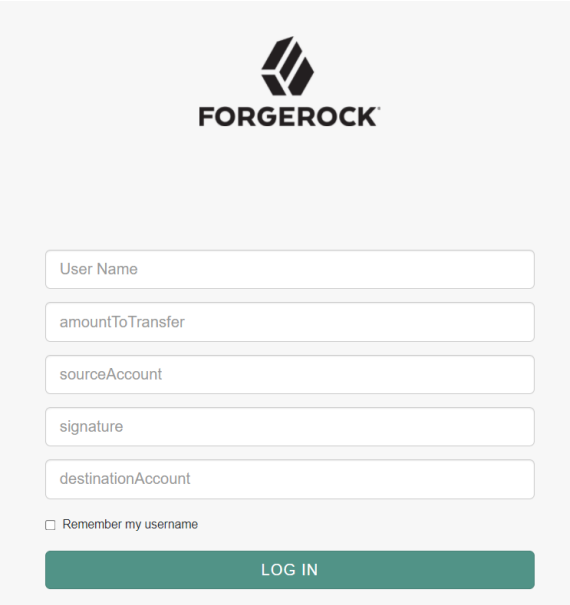
4. Offline Transaction Data Signing

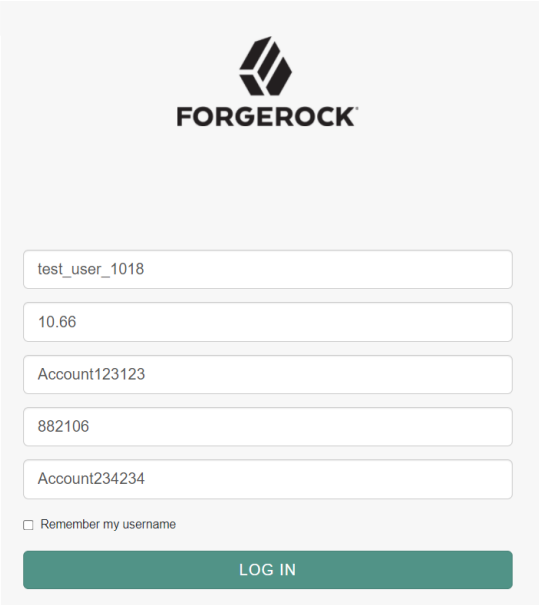


With OneSpan Cloud Authentication (OCA), your users can sign their transaction data offline. Build below link and kick off the validation process:

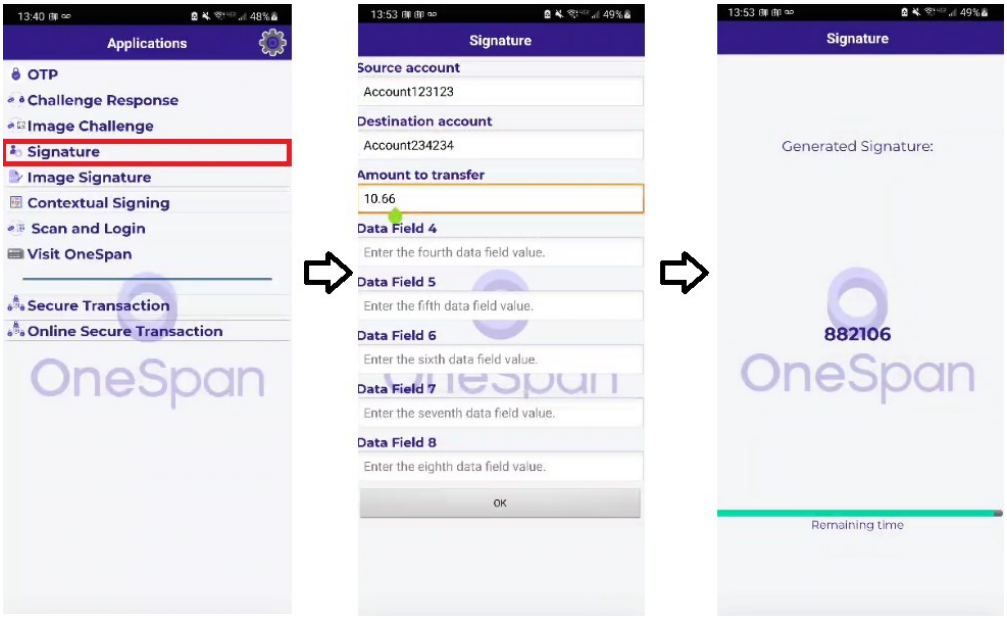
https://{your_instance_url}/openam/XUI/?realm=/&service=OneSpan-XUI-Cloud-Authentication-Validate-Transaction-Sample-Tree#login

In this quick demo, we will use the “OneSpan Sample Attributes Collector” to pass in all the necessary transaction details.





In order to sign the transaction data and obtain a signature, launch the MAS app and click the "Signature" option from the applications list, the app will prompt the same transaction information and generate a signature.

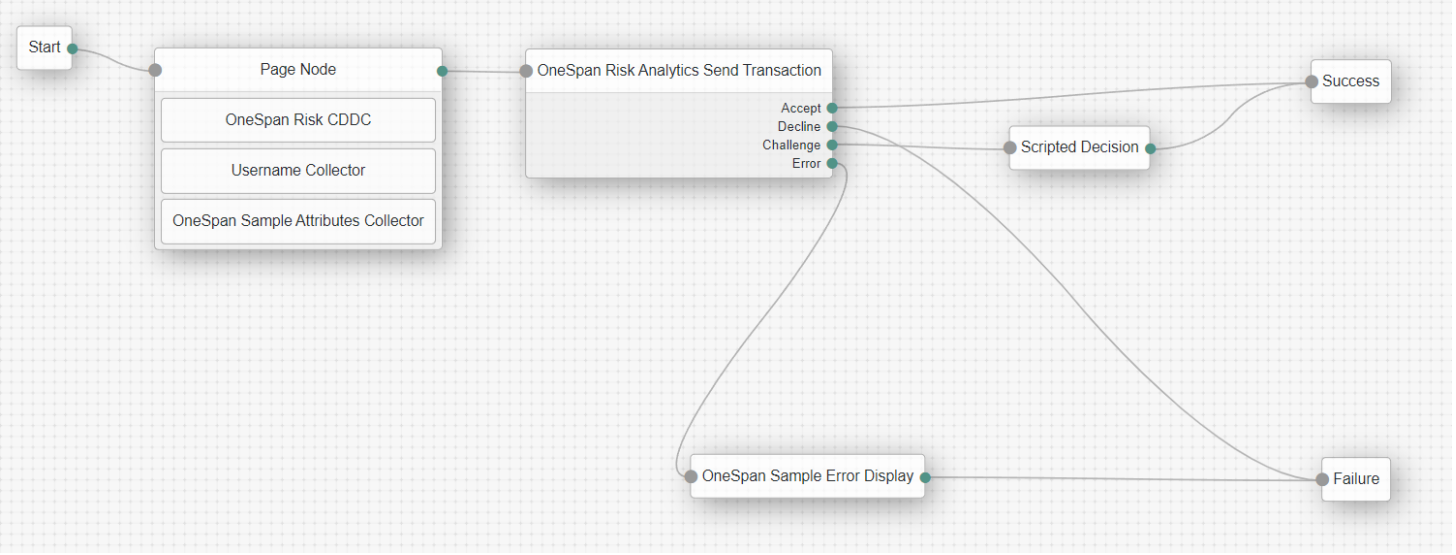


Input the transaction detail along with the generated signature to the authentication page, and the OCA Validate Transaction service will validate the signature and return the validation result. If the authentication has succeeded, the browser will be redirected to the success URL.

Explore the [Integrating Offline Transaction Data Signing] guide for more detailed information.

For OneSpan RA Users:

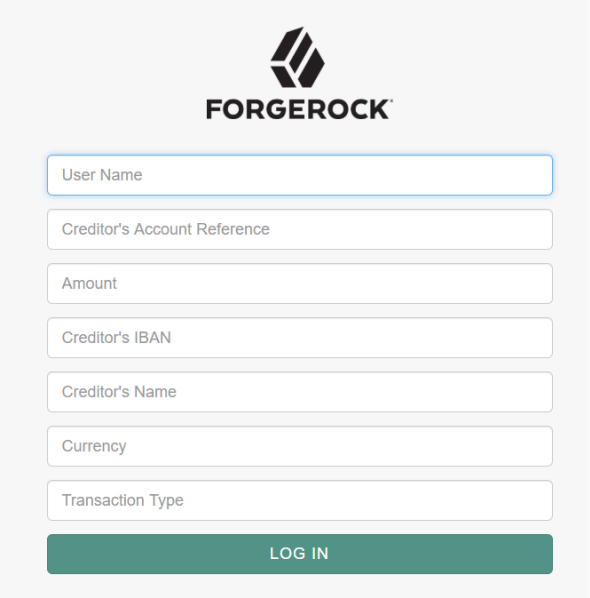
1. Insert Transaction

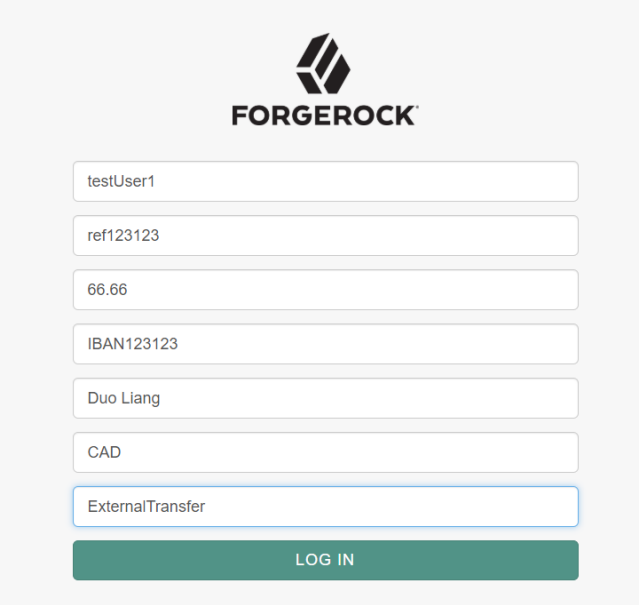


Similar to the Auth Validate Transaction workflow, if you only want to leverage OneSpan Risk Analytics and get a response code without sending the adaptive authentication request. Build below link and kick off the validation process:

https://{your_instance_url}/openam/XUI/?realm=/&service=OneSpan-XUI-Cloud-Authentication-User-Login-Challenge-Response-Sample-Tree#login

In this quick demo, we will use the "OneSpan Sample Transaction Collector" to pass in all the necessary transaction details.





If there's no user step up configured in the Risk Analytic Representation Service, the outcome will directly fall within Accept or Decline depending on the risk score. However, because of the rules we've pre-set, when the transaction amount was less than 100, the Risk Analytics system will return a step up authentication "ChallengePin".

According to the API Specification page > TransactionOutput Schema > riskResponseCode Attribute, if the challenge type is Challenge PIN, the API returns a risk response code of 22.

The "OneSpan Risk Analytics Send Transaction" node then stores the risk response code in the shared state in name of "riskResponseCode", which can later be retrieved by Scripted Decision node and determine the path the authentication journey takes.

Nodes Features

1. OneSpan Risk CDDC

Introduction:

This node utilizes the Client Device Data Collector (CDDC) library to collect the end user's device fingerprint and browser data and to store the data to the sharedState. This information will be later used by OneSpan Risk Analytics to assess the risk of the web session context.

Available Properties:

Property	Type	Default Value	Usage
Push CDDC Script	boolean	True	If set to True, the node will push the CDDC Javascript to the ForgeRock page and automatically collect the CDDC Json and Hash value with two hidden value callbacks. If set to False, integrators are supposed to pass the values through hidden value callbacks.
CDDC Json Callback ID	String	"ostid_cddc_json"	Only when set False above, specify the hidden value ID for the CDDC Json.
CDDC Hash Callback ID	String	"ostid_cddc_hash"	Only when set False above, specify the hidden value ID for the CDDC hash value.

Data Flow:

This node requires below inbound data

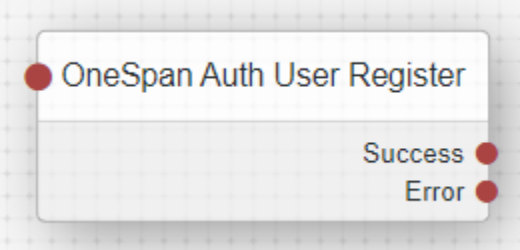
Description	Attribute Name	Source

CDDC Json	As specified in property	hidden value callback
CDDC hash value	As specified in property	hidden value callback

This code will store below outbound data

Description	Attribute Name	Storage
CDDC Json	“ostid_cddc_json”	Shared State
CDDC hash value	“ostid_cddc_hash”	Shared State
CDDC client IP	“ostid_cddc_ip”	Shared State

2. OneSpan Auth User Register



Introduction:

This node fits both the IAA and OCA use cases. Behind the scene, it invokes the User Register / Unregister API, which validates and processes the registration / unregistration of a user.

Available Properties:

Property	Type	Default Value	Usage
Object Type	Enum	IAA	IAA / OCA use cases
Node Function	Enum	UserRegister	Choose the node function from user register / unregister.
User Name In SharedState	String	“username”	Specify the name of a key in the sharedState object in which to represent the OneSpan IAA User Name.
Password In TransientState	String	“password”	Specify the name of a key in the transientState object in which to represent the OneSpan IAA User Password.
Activation Type	Enum	onlineMDL	If to activate the authenticator with online or offline flow, or using FIDO device (UAF or FIDO2).

Optional Attributes	Map<String,String>	Empty Collection	Specify other optional attributes like user email, user phone number, etc. The key of the map represents the name of the key in the sharedState object, while the value of map represents the key that will be additional added to the API payload. For example, with a pair like "emailAddressInSharedState" : "emailAddress", the node will look for the key "emailAddressInSharedState" in the sharedState and add a pair "emailAddress" : "{valueInSharedState}" to the OneSpan IAA API payload.
Event Expiry	int	60	Specify the event expiry. The priority is: ForgeRock Session Expiry > OneSpan IAA Session Expiry > Event Expiry. Make sure the ForgeRock session expiry and the OneSpan IAA session expiry are no shorter than the value specified here.

API Reference:

Refer to the User Register API and User Unregister API for more details.

Data Flow:

This node requires below inbound data:

Description	**Attribute Name**	**Source**
Username	As specified in property	Shared State
Password	As specified in property	Transient State
(Optional) Other Attributes	As specified in property	Shared State
CDDC Json	"ostid_cddc_json"	Shared State
CDDC hash value	"ostid_cddc_hash"	Shared State
CDDC client IP	"ostid_cddc_ip"	Shared State

This code will store below outbound data:

Case 1: For IAA use cases, when the node function is set to "UserRegister":

Description	Attribute Name	Storage
session ID	"ostid_session_id"	Shared State
User activation code	"ostid_activationPassword" & "activationPassword"	Shared State
The visual code message	"ostid_cronto_msg"	Shared State
DigiPass serial	"ostid_digi_serial"	Shared State
The expiry date	"ostid_event_expiry_date"	Shared State

Note:

The visual code message follows the particular syntax which is used for the demo app, which looks like

"02;{username};111;{instance_tenant_name};{activation_code};{instance_tenant_name}"

To facilitate your integration, you can (1) follow the same syntax in your custom mobile app, or (2) store the custom value in sharedState, refer to "OneSpan Auth Visual Code" node – "Message Options" property for more details.

Case 2: For OCA use cases, when the node function is set to "UserRegister":

Description	Attribute Name	Storage
session ID	"ostid_session_id"	Shared State
User activation code	"ostid_activationPassword" & "activationPassword"	Shared State
The visual code message	"ostid_cronto_msg"	Shared State
DigiPass serial	"ostid_digi_serial"	Shared State
The registration ID	"ostid_registration_id"	Shared State
The expiry date	"ostid_event_expiry_date"	Shared State

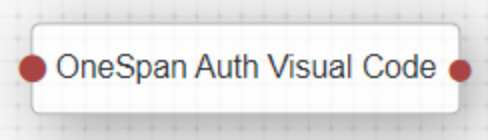
Case 3: When the node function is set to "UserUnregister":

Description	Attribute Name	Storage
session ID	"ostid_session_id"	Shared State

Case 4: When the outcome is "Error"

Description	Attribute Name	Storage
The error message	"ostid_error_message"	Shared State

3. OneSpan Auth Visual Code



Introduction:

This node reads the visual code message from the sharedState and renders it as a visual code. The end users can scan the image with a mobile app integrated with the Mobile Security Suite SDKs or Digipass authenticators with Cronto image support.

The node can be part of a page node or be leveraged independently of the UI flow.

Available Properties:

Property	Type	Default Value	Usage
Message Options	Enum	DemoMobileApp	If set to “DemoMobileApp”, the node will look up the message stored by the prior node, which was formatted in a particular way that fits the demo mobile app. To edit your own message format, set the option to “CustomMessage”, and use your own Auth Node to store the visual code message in the sharedState.
Custom Message In SharedState	String	“”	Only when choose “CustomMessage” above, allows to specify the name of a key in the sharedState object in which to represent the customized visual code message.
Visual Code Callback ID	String	“ostid_cronto”	The node will return a hidden value callback with this ID, containing the image URL of the visual code.
Render Visual Code	boolean	True	If set to True, the node will return a JavaScript callback displaying the visual code and the countdown timer. If set to False, no JavaScript callback will be returned, integrators can use the image URL in hidden value callback to render their own visual code page.
Visual Code DOM ID	String	“dialog”	The DOM ID used to locate the visual code.
Visual Code Type	Enum	Cronto	Depend on which type of visual code your mobile app can detect and handle with.
Visual Code Size	int	210	Specify the size of the visual code
Visual Code Alt Text	String	“OneSpan TID Cronto Image”	The alternative text in case the image can’t be displayed properly.
Please Scan Text	String	“Please Scan the Visual Code within:”	The text label prompting the user to scan the visual code. The countdown seconds will be automatically attached.
Please Scan CSS	String	“”	The CSS for the label. For example: “font-size: 14px; color: blue;”
Expired Text	String	“Your Activation Code has been expired!”	The text label after the countdown expired.

Expired CSS	String	“”	The CSS for the label. For example: “font-size: 14px; color: red;”

Data Flow:

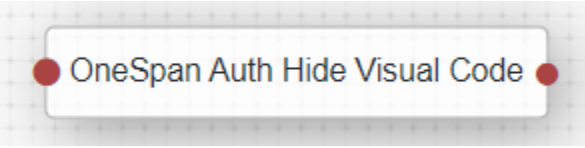
This node requires below inbound data:

Description	Attribute Name	Source
Visual code message	“ostid_cronto_msg” / As specified in property	Shared State
The expiry date	“ostid_event_expiry_date”	Shared State
A tag indicates whether the custom application has consumed the cronto URL. In an API flow, you’d set the hidden value callback input value as “true”, so that the node knows to proceed	“ostid_cronto_has_rendered”	Hidden Value Callback

This code will store below outbound data:

Description	Attribute Name	Storage
Visual code image URL	As specified in property	Hidden Value Callback & Shared State
The expiry date in Unix time, specific for API flow	“ostid_event_expiry_date”	Hidden Value Callback
A tag indicates whether the custom application has consumed the cronto URL. In an API flow, you’d set the hidden value callback input value as “true”, so that the node knows to proceed	“ostid_cronto_has_rendered”	Hidden Value Callback

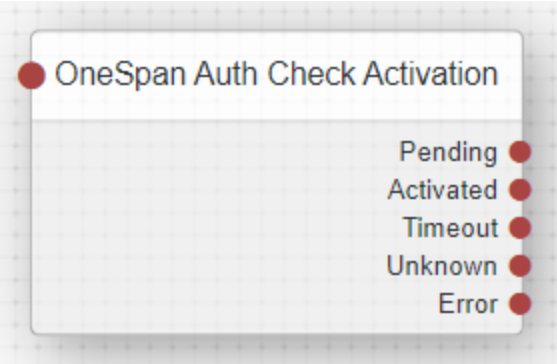
4. OneSpan Auth Hide Visual Code



Introduction:

When the OneSpan Auth Visual Code node was used in a UI flow, this node provides the capability to hide the visual code from UI.

5. OneSpan Auth Check Activation



Introduction:

This node invokes the Check Activation Status API, which checks the status of a pending activation of a device.

Data Flow:
This node requires below inbound data:

Description	Attribute Name	Source
Username in sharedState	“ostid_username_in_shared_state”	Shared State
The expiry date	“ostid_event_expiry_date”	Shared State

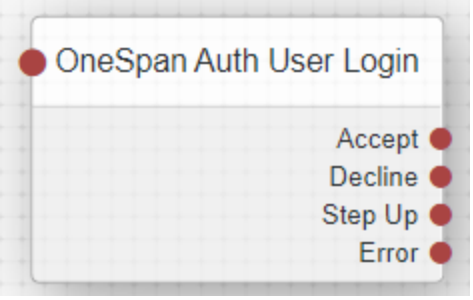
When the outcome is “Error”, this code will store below outbound data:

Description	Attribute Name	Storage
The error message	“ostid_error_message”	Shared State

API Reference:

Explore the Integrating end-user registration and Digipass activation guide for detailed descriptions of the API and the outcomes.

6. OneSpan Auth User Login



Introduction:

This node fits both the IAA and OCA use cases. In general, the node invokes the User Login API, which validates the end user's login request the authentication service, then returns the result of the authentication attempt.

For IAA use cases, the request will further be validated against the Risk Analytics system and If RA requires an extra challenge, a multi-factor authentication flow needs to be designed after the "Step Up" outcome.

Available Properties:

Property	Type	Default Value	Usage
Object Type	Enum	AdaptiveLoginInput	Choose "AdaptiveLoginInput" for IAA use cases, choose "LoginInput" for OCA use cases.
Credentials Type	Enum	none	For OCA use cases, select the credentials type.
User Name In SharedState	String	"username"	Specify the key name in sharedState which represents the OneSpan IAA/OCA User Name
Password In TransientState	String	"password"	Only when the above option was True, specify the key name in transientState which represents the password
Optional Attributes	Map<String,String>	Empty Collection	Specify other optional attributes like user email, user phone number, etc. The key of the entry represents the key name in sharedState, while the value of entry represents the attribute name defined in API schema. For example, with a pair like [emailAddressInSharedState : emailAddress], the node will look for the key

			"emailAddressInSharedState" in sharedState and add an attribute "emailAddress" : "{valueInSharedState}" to the API payload.
Orchestration Delivery	Enum	Default	Indicates whether a push notification should be sent, and/or if the orchestration command should be included in the response requestMessage.
Login Timeout	int	60	Specify the event expiry. The priority is: ForgeRock Session Expiry > OneSpan IAA Session Expiry > Event Expiry. Make sure the ForgeRock session expiry and the OneSpan IAA session expiry are no shorter than the value specified here.
Visual Code Message	Enum	SessionId	Determine what visual code message will be used to render the visual code. To send your own customized message format, refer to "OneSpan Auth Visual Code" node – "Message Options" property for more details.

API Reference:

Explore the Integrating end-user login via notification guide for more details.

Data Flow:

This node requires below inbound data

Description	**Attribute Name**	**Source**
Username	As specified in property	Shared State
(Optional) Other Attributes	As specified in property	Shared State
CDDC Json	"ostid_cddc_json"	Shared State
CDDC hash value	"ostid_cddc_hash"	Shared State
CDDC client IP	"ostid_cddc_ip"	Shared State
(Optional) IAA Session Id	"ostid_session_id"	Shared State
(Optional) AuthenticationResponse	"authenticationResponse"	Shared State

from respective FIDO protocol		
(Optional) FIDO protocol used in this operation	“fidoProtocol”	Shared State
(Optional) One-time password generated by the authenticator	“OTP”	Shared State
(Optional)Password	As specified in property	Transient State

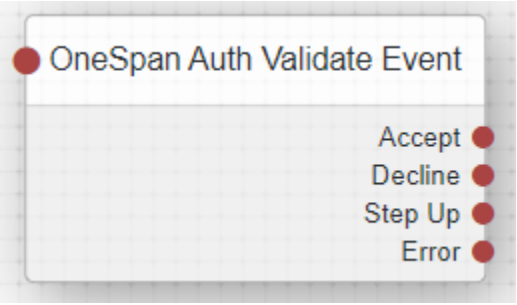
This code will store below outbound data:

Description	Attribute Name	Storage
The visual code message	“ostid_cronto_msg”	Shared State
session ID	“ostid_session_id”	Shared State
request ID	“ostid_request_id”	Shared State
OneSpan IRM response	“ostid_irm_response”	Shared State
command	“ostid_command”	Shared State
The expiry date	“ostid_event_expiry_date”	Shared State

Only when the outcome is “Error”:

Description	Attribute Name	Storage
The error message	“ostid_error_message”	Shared State

7. OneSpan Auth Validate Event



Introduction:

This node invokes the Event Validation API, which validates a non-monetary event against the Risk Analytics system and the Authentication Service, and returns the result.

If Risk Analytics system required an extra challenge, a multi-factor authentication flow has to be designed after the “Step Up” outcome.

Available Properties:

Property	Type	Default Value	Usage
Event Type	Enum	SpecifyBelow	If set to “SpecifyBelow”, integrators can hard code the event type in below configuration. If set to “ReadFromSharedState”, integrators can determine the event type at run time by pre-store the event type in the sharedState. The name of the key can be specified in below configuration.
Specify Event Type	String	“”	Only when choose “SpecifyBelow” above. All the available event types can be found at the API Specifications.
Event Type in SharedState	String	“”	Only when choose “ReadFromSharedState” above. Specify the name of a key in the sharedState object in which to represent the Event Validation type.
User Name In SharedState	String	“username”	Specify the name of a key in the sharedState object in which to represent the OneSpan IAA User Name
Password In TransientState	String	“password”	Only when choose True above, specify the name of a key in the transientState object in which to represent the OneSpan IAA User Password
Optional Attributes	Map<String,String>	Empty Collection	Specify other optional attributes like user email, user phone number, etc. The key of the map represents the name of the key in the sharedState object, while the value of map represents the key that will be additional added to the API payload.

			For example, with a pair like [emailAddressInSharedState : emailAddress], the node will look for the key "emailAddressInSharedState" in the sharedState and add an attribute "emailAddress" : "{valueInSharedState}" to the API payload
Orchestration Delivery	Enum	Default	Indicates whether a push notification should be sent, and/or if the orchestration command should be included in the response requestMessage.
Event Validation Timeout	int	60	Specify the event expiry. The priority is: ForgeRock Session Expiry > OneSpan IAA Session Expiry > Event Expiry. Make sure the ForgeRock session expiry and the OneSpan IAA session expiry are no shorter than the value specified here.
Visual Code Message	Enum	SessionId	Determine what visual code message will be used to render the visual code. To send your own customized message format, refer to "OneSpan TID Visual Code" node – "Message Options" property for more details.

API Reference:

Refer to the Event Validation API for more details.

Data Flow:

This node requires below inbound data:

Description	**Attribute Name**	**Source**
Event type	As specified in property	Configuration / Shared State
Username	As specified in property	Shared State
(Optional)Password	As specified in property	Transient State

(Optional) Other attributes	As specified in property	Shared State
CDDC Json	“ostid_cddc_json”	Shared State
CDDC hash value	“ostid_cddc_hash”	Shared State
CDDC client IP	“ostid_cddc_ip”	Shared State
(Optional) Session Id	“ostid_session_id ”	Shared State

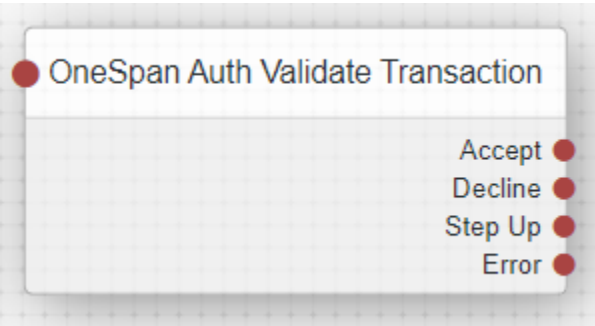
This code will store below outbound data:

Description	Attribute Name	Storage
The visual code message	“ostid_cronto_msg”	Shared State
session ID	“ostid_session_id”	Shared State
request ID	“ostid_request_id”	Shared State
OneSpan IRM response	“ostid_irm_response”	Shared State
command	“ostid_command”	Shared State
The expiry date	“ostid_event_expiry_date”	Shared State

When the outcome is “Error”:

Description	Attribute Name	Storage
The error message	“ostid_error_message”	Shared State

8. OneSpan Auth Validate Transaction



Introduction:

This node fits both the IAA and OCA use cases. In general, the node invokes the Transaction Service API, which validates monetary transaction request against the Authentication Service and returns the result.

For IAA use cases, it further validates the request against the Risk Analytics system, if RA requires an extra challenge, a multi-factor authentication flow needs to be designed after the “Step Up” outcome.

Available Properties:

Property	Type	Default Value	Usage
Object Type	Enum	AdaptiveTransactionValidationInput	Choose "AdaptiveTransactionValidationInput" for IAA use cases, choose "TransactionValidationInput" for OCA use cases.
User Name In SharedState	String	“username”	Specify the name of a key in the sharedState object in which to represent the OneSpan IAA User Name
Data To Sign	Enum	transactionMessage	Choose "fido", "standard" or "secureChannel" for OCA use cases. Choose "transactionMessage" for IAA use cases.
Standard Data Fields	List<String>	"sourceAccount","destinationAccount","amountToTransfer"	If selected "standard" option, the signature will be generated out of a sorted list of data. Please store the data-to-sign in the sharedState and supply the key names here.
Signature In SharedState	String	“signature”	If selected "standard" or "secureChannel" option, you'll be prompted for a signature generated by your authenticator. Please store the generated signature in the shared state and supply the key name here.

Fido Attributes	Map<String, String>	{
		"fidoProtocol": "fidoProtocol" ,
		"authenticationResponse": "authenticationResponse"
		}
			If selected "fido" option, you'll be prompted for "fidoProtocol" ("UAF11" or "FIDO2") and "authenticationResponse". Please store the values in the shared state as per this map, where the "key" of

			the pair refers to the JSON attribute and "value" is the ShareState attribute name.

Adaptive Attributes	Map<String, String>	{
		"accountRef": "accountRef" ,
		"amount": "amount",
		"currency": "currency",
		"transactionType": "transactionType",
		"creditorBank": "creditorBank",
		"creditorIBAN": "creditorIBAN",
		"creditorName": "creditorName",
		"debtorIBAN": "debtorIBAN",
		}
			If selected "transactionMessage" option, there are some mandatory attributes like accountRef, amount, currency, transactionType, etc. Specify the API field names as the "key" and the Shared State attribute names where you've stored their values as the "value".

Adaptive Data Fields	Map<String, String>	Empty Collection	If selected "transactionMessage" option, you can pass in additional data to be displayed in your mobile application. Specify the API field names as the "key" and the Shared State attribute names where you've stored their values as the "value".
Optional Attributes	Map<String,String>	Empty Collection	Specify other optional attributes like user email, user phone number, etc. The key of the map represents the name of the key in the sharedState object, while the value of map represents the key that will be additional added to the API payload. For example, with a pair like [emailAddressInSharedState : emailAddress], the node will look for the key “emailAddressInSharedState” in the sharedState and add an attribute “emailAddress” : “{valueInSharedState}” to the API payload

Orchestration Delivery	Enum	Default	Indicates whether a push notification should be sent, and/or if the orchestration command should be included in the response requestMessage.
Validation Timeout	int	60	Specify the event expiry. The priority is: ForgeRock Session Expiry > OneSpan IAA Session Expiry > Event Expiry. Make sure the ForgeRock session expiry and the OneSpan IAA session expiry are no shorter than the value specified here.
Visual Code Message	Enum	SessionId	Determine what visual code message will be used to render the visual code. To send your own customized message format, refer to "OneSpan TID Visual Code" node – "Message Options" property for more details.

API Reference:
Refer to the Transaction Validation API for more details.

Data Flow:
This node requires below inbound data:

Description	**Attribute Name**	**Source**
Username	As specified in property	Shared State
CDDC Json	"ostid_cddc_json"	Shared State
CDDC hash value	"ostid_cddc_hash"	Shared State
CDDC client IP	"ostid_cddc_ip"	Shared State
(Optional) Standard Data Fields	As specified in property	Shared State
(Optional) Generated Signature	As specified in property	Shared State
(Optional) Fido Attributes	As specified in property	Shared State

(Optional) Adaptive Attributes	As specified in property	Shared State
(Optional) Adaptive Data Fields	As specified in property	Shared State
(Optional) Other attributes	As specified in property	Shared State
(Optional) IAA Session ID	“ostid_session_id”	Shared State

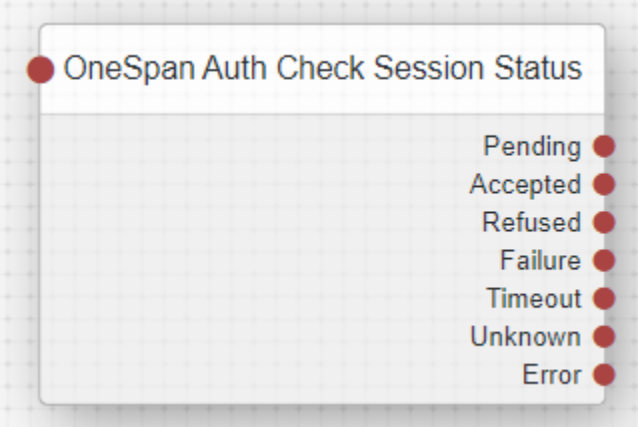
This code will store below outbound data:

Description	Attribute Name	Storage
The visual code message	“ostid_cronto_msg”	Shared State
Session ID	“ostid_session_id”	Shared State
Request ID	“ostid_request_id”	Shared State
OneSpan IRM response	“ostid_irm_response”	Shared State
Mobile Command	“ostid_command”	Shared State
The Expiry Date	“ostid_event_expiry_date”	Shared State

When the outcome is “Error”:

Description	Attribute Name	Storage
The error message	“ostid_error_message”	Shared State

9. OneSpan Auth Check Session Status



Introduction:

This node invokes the Check Session Status API and returns the status of a request.

Data Flow:

This node requires below inbound data:

Description	Attribute Name	Source

IAA Request ID	“ostid_request_id”	Shared State
The expiry date	“ostid_event_expiry_date”	Shared State

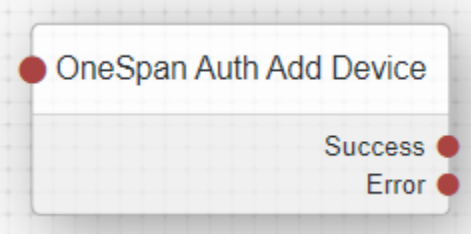
This code will store below outbound data when the outcome is “Error”:

Description	Attribute Name	Storage
The error message	“ostid_error_message”	Shared State

API Reference:

Refer to the Check Session Status Service API for more details.

10. OneSpan Auth Add Device



Introduction:

This node is specifically for OCA use cases. In a provisioning process, after scanning the activation password, the DIGIPASS authenticator will return its device code. This node prompts for the device code and continues the process.

Data Flow:

This node requires below inbound data:

Description	Attribute Name	Source
Registration ID	“ostid_registration_id“	Shared State
The device code of the DIGIPASS Authenticator	“deviceCode”	Shared State

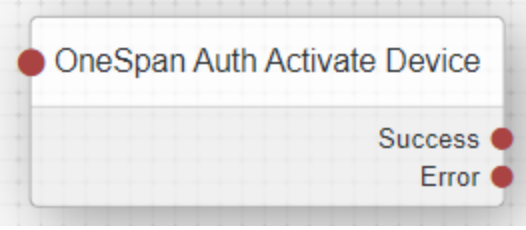
This code will store below outbound data when the outcome is “Error”:

Description	Attribute Name	Storage
The error message	“ostid_error_message”	Shared State

API Reference:

Refer to the Add Device API for more details.

11. OneSpan Auth Activate Device



Introduction:

This node is specifically for OCA use cases. In a provisioning process, after scanning the second activation password, the DIGIPASS authenticator will return the first OTP. Activate Device node prompts for this signature and finalizes the activation process.

Data Flow:
This node requires below inbound data:

Description	Attribute Name	Source
Registration ID	“ostid_registration_id“	Shared State
The first OTP created by the DIGIPASS Authenticator	“signature”	Shared State

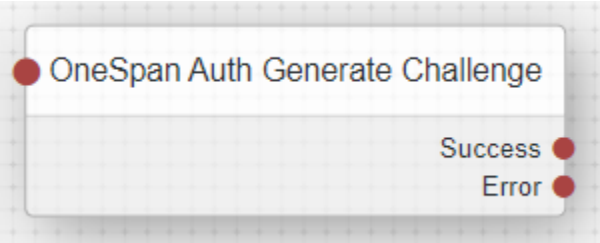
This code will store below outbound data when the outcome is “Error”:

Description	Attribute Name	Storage
The error message	“ostid_error_message”	Shared State

API Reference:

Refer to the Activate Device API for more details.

12. OneSpan Auth Generate Challenge



Introduction:

This node is specifically for OCA Challenge/Response authentications. It requests a random challenge which will later be presented to the user. The user enters it in their authenticator and enters the response in the authentication page.

Data Flow:
This code will store below outbound data:

Description	Attribute Name	Source
Request ID	“ostid_request_id“	Shared State
The visual code message	“ostid_cronto_msg”	Shared State

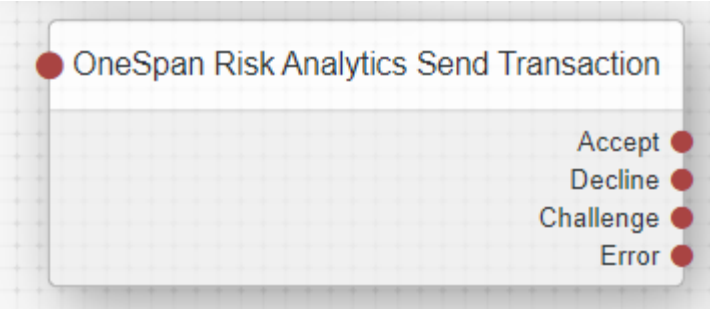
When the outcome is “Error”:

Description	Attribute Name	Storage
The error message	“ostid_error_message”	Shared State

API Reference:

Refer to the Generate Challenge API for more details.

13. OneSpan Risk Send Transaction



Introduction:

This node invokes Risk Analytics transaction insertion API, which validates monetary transaction request against the Risk Analytics system without sending the adaptive authentication request.

Available Properties:

Property	Type	Default Value	Usage
User Name In SharedState	String	“username”	Specify the key name in sharedState which represents the OneSpan IAA User Name
Adaptive Attributes	Map<String, String>	{ "accountRef": "accountRef" , "amount": "amount", "currency": "currency", "transactionType": "transactionType", "creditorBank": "creditorBank", "creditorIBAN": "creditorIBAN", "creditorName": "creditorName", "debtorIBAN": "debtorIBAN", }	Specify the input payload for Risk Analytics transaction requests. The "key" refers to the JSON attribute as defined in API schema ("transactionType", "amount", "currency" and "accountRef" are mandatory) and "value" refers to the name of the ShareState attribute. For example, given

			a pair like "emailAddress" : "emailAddressInSharedState", the node will first look for the key "emailAddressInSharedState" in the sharedState then add a pair "emailAddress" : "{valueInSharedState}" to the OneSpan API payload.

API Reference:
Refer to the RA Insert Transaction API for more details.

Data Flow:
This node requires below inbound data:

Description	Attribute Name	Source
Username	As specified in property	Shared State
CDDC Json	“ostid_cddc_json”	Shared State
CDDC hash value	“ostid_cddc_hash”	Shared State
CDDC client IP	“ostid_cddc_ip”	Shared State
Adaptive Attributes	As specified in property	Shared State

This code will store below outbound data:

Description	Attribute Name	Storage
Risk Response Code	“ostid_risk_response_code” & “riskResponseCode”	Shared State

When the outcome is “Error”:

Description	Attribute Name	Storage
The error message	“ostid_error_message”	Shared State