OneSpan IAA Auth Tree Nodes

OneSpan Intelligent Adaptive Authentication (IAA) secures your web and mobile applications by analyzing vast and disparate data acquired through user actions and events. Based on this analysis, OneSpan Adaptive Authentication dynamically assesses which authentication and/or transaction security measures are appropriate for each unique end user.

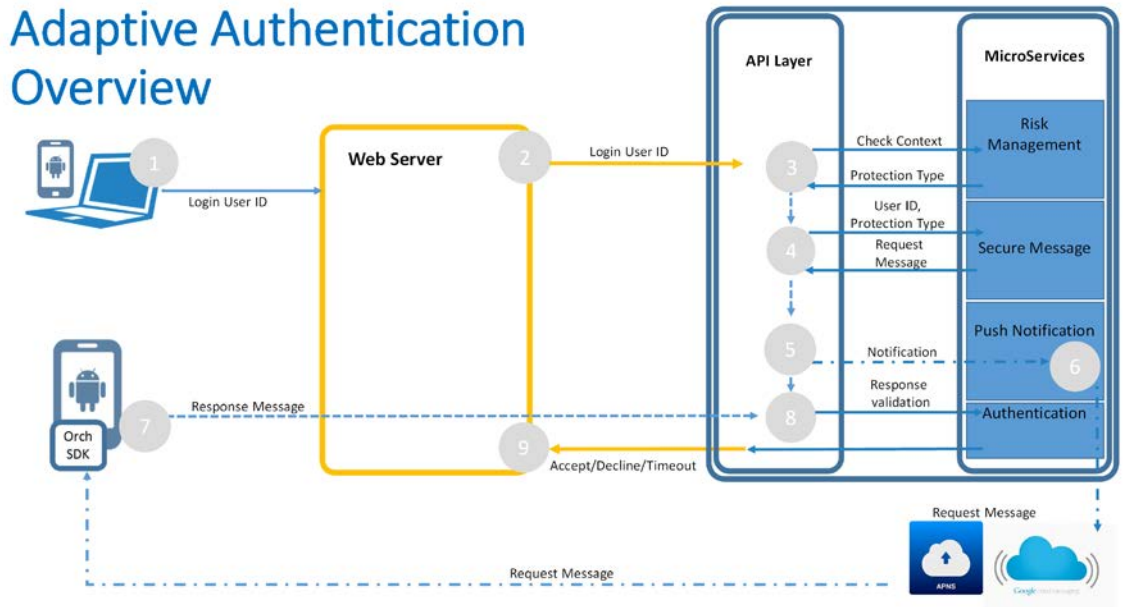
About OneSpan IAA

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.



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

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.

Explore the IAA Demo User Guide for more details.

4. Configure the Intelligent Risk Management (IRM) service. As an important component of IAA, IRM is a fraud management system used for monitoring and designing rules & actions for online banking applications and payment processing that across multi-channels. In order to test through the functionalities of the Tree Nodes, we will set up some simple rules in your IRM system.

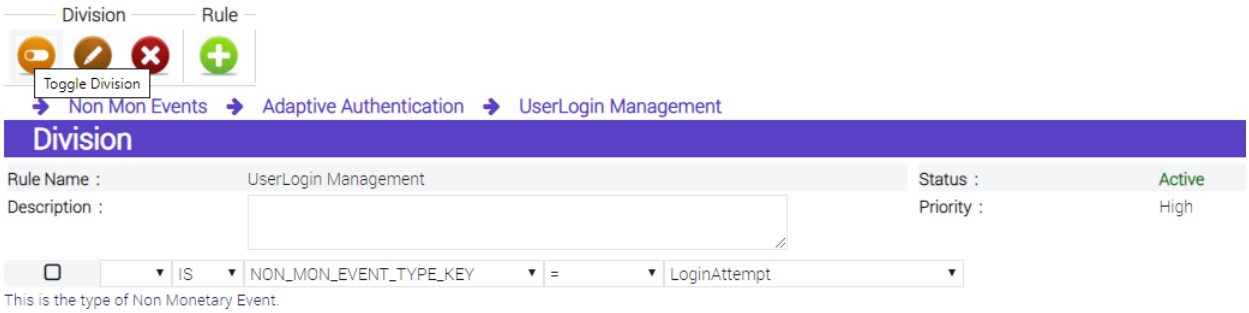
Rule 1: When an end user tried to login, send an extra PIN challenge to user's trusted device.

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

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

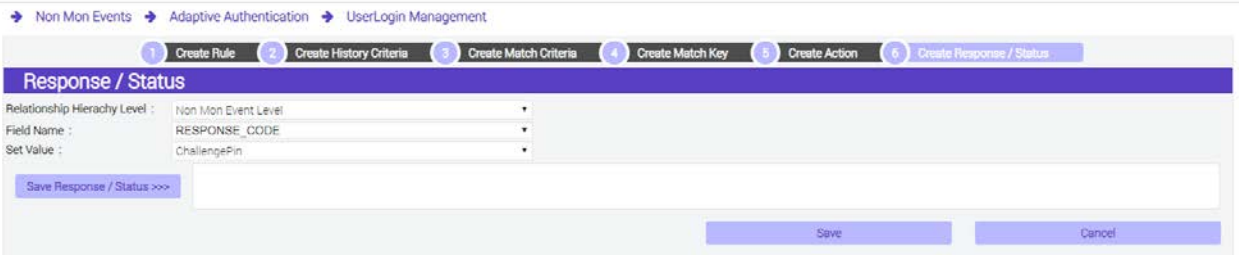
-Create a new **Division** named "UserLogin Management" with a criteria of "is NON_MON_EVENT_TYPE_KEY = LoginAttempt"

- 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.

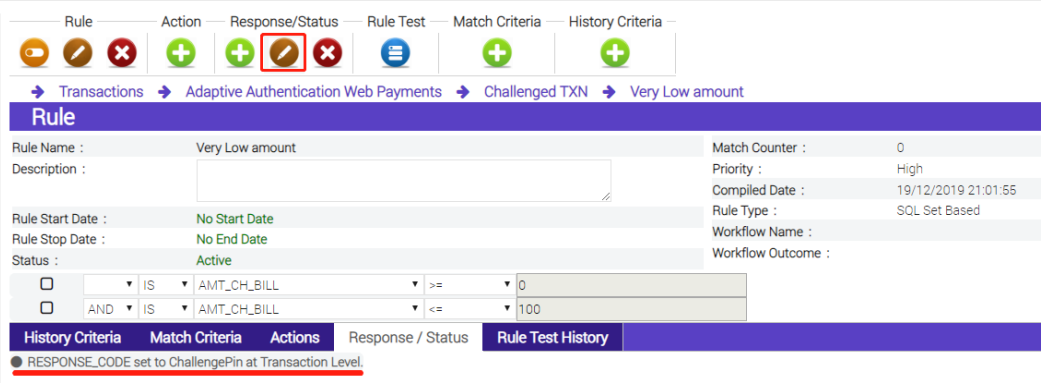


Rule 2: When an end user tried to send a transaction, if the transaction amount was below 100, send an extra PIN challenge to user's trusted device.

- In the Risk Management 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"



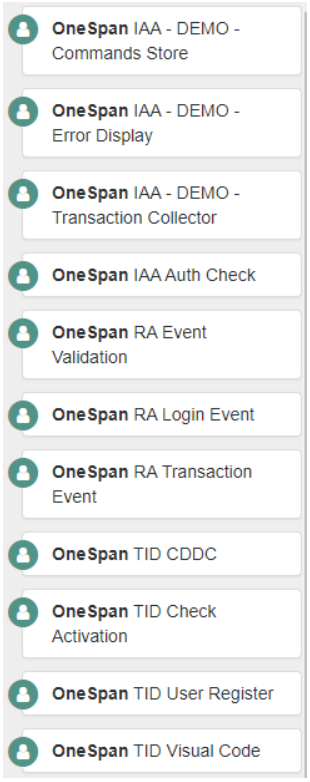
Tips:

- Face/Touch biometrics are also the available options, simply change the IRM 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.

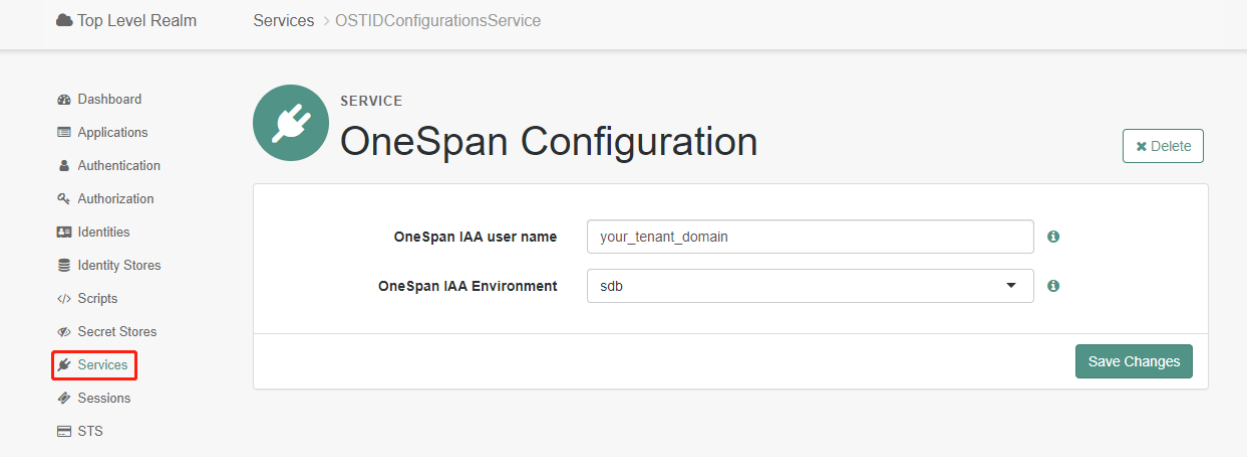
Nodes Overview

The OneSpan IAA Auth Tree Nodes contains 1 Auxiliary Service, 8 nodes, and 3 demo nodes which only used for test purpose. More details will be covered in next sections.



Auxiliary Service

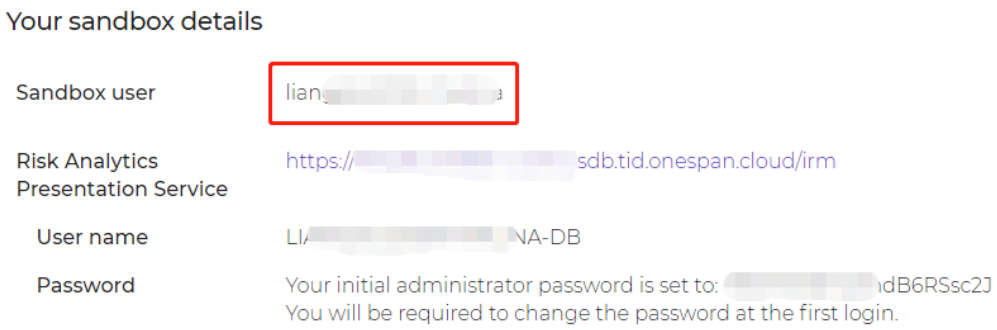
The node provides a realm-specific service named “OneSpan Configuration”, where allows you to specify the OneSpan IAA common configurations.



-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 first level domain of your developer portal URL indicates your environment, for Sandbox accounts (https://sdb.tid.onespan.cloud/devportal/Board#adaptiveAuth), the environment is the “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 the “OneSpan Configuration” service.

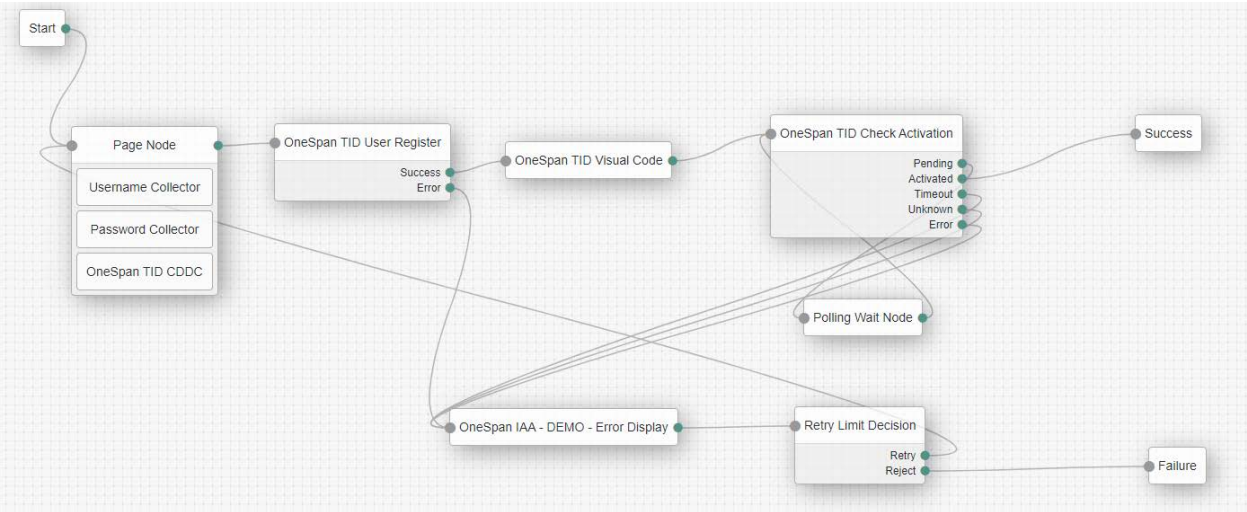
(2) Reproduce below sample trees using either of below two methods:

-Manually create a new tree following the design and remain all the settings default.

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

(3) Launch the Sample AAS Demo App in your mobile, agree the License Agreement and enable the required mobile permissions.

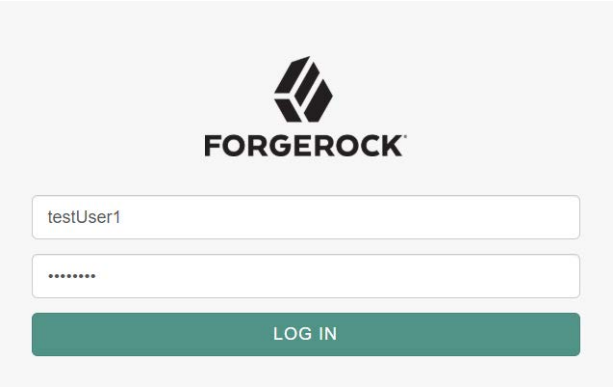
1. OneSpan IAA User Register



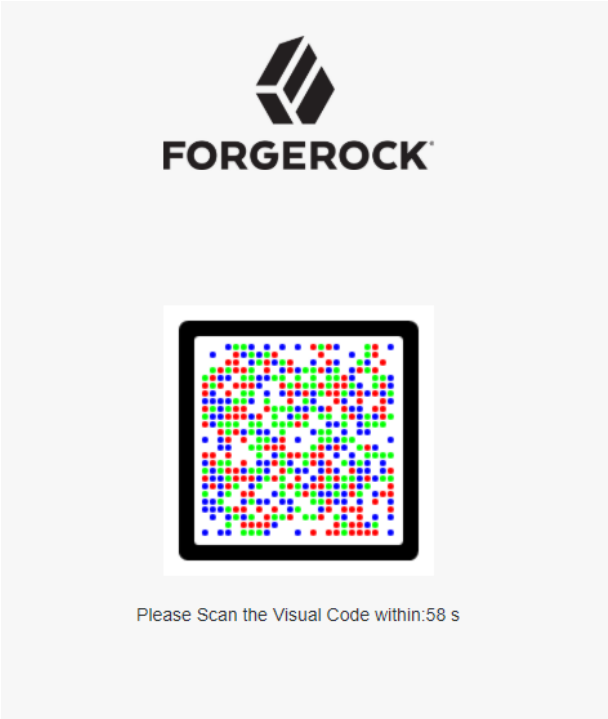
Hit below link in your browser and start the authentication process:

https://{your_instance_url}/openam/XUI/?realm=/&service=OSIAAUserRegister#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)

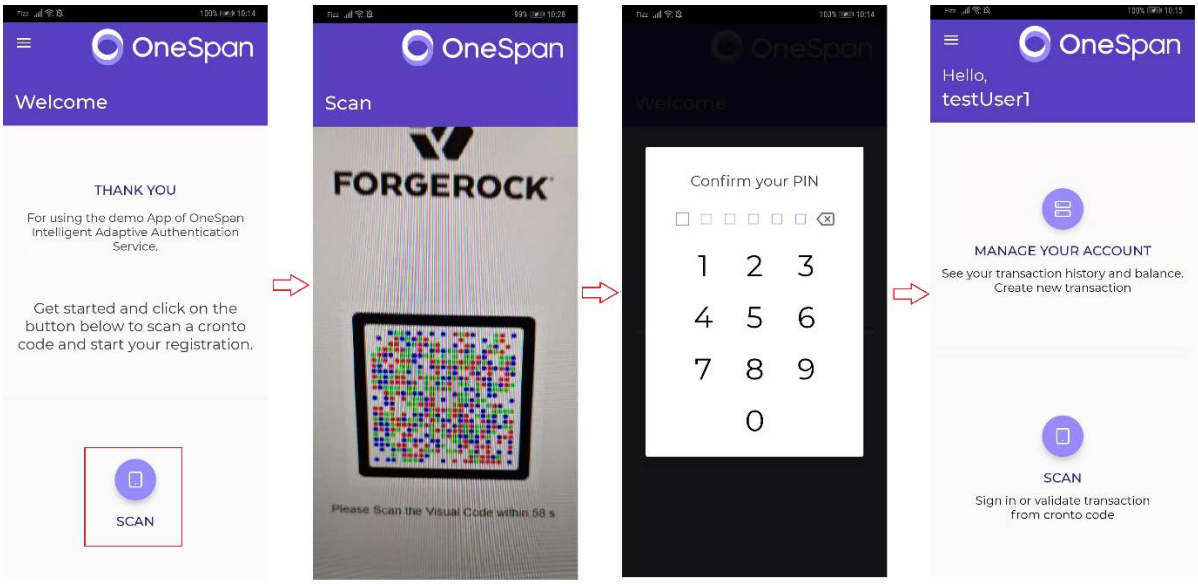


Once the Risk Management service has accepted the user registration, the IAA service creates a Digipass user account and awaits a trusted device to activate the license with an activation token, which is rendered as a visual code.

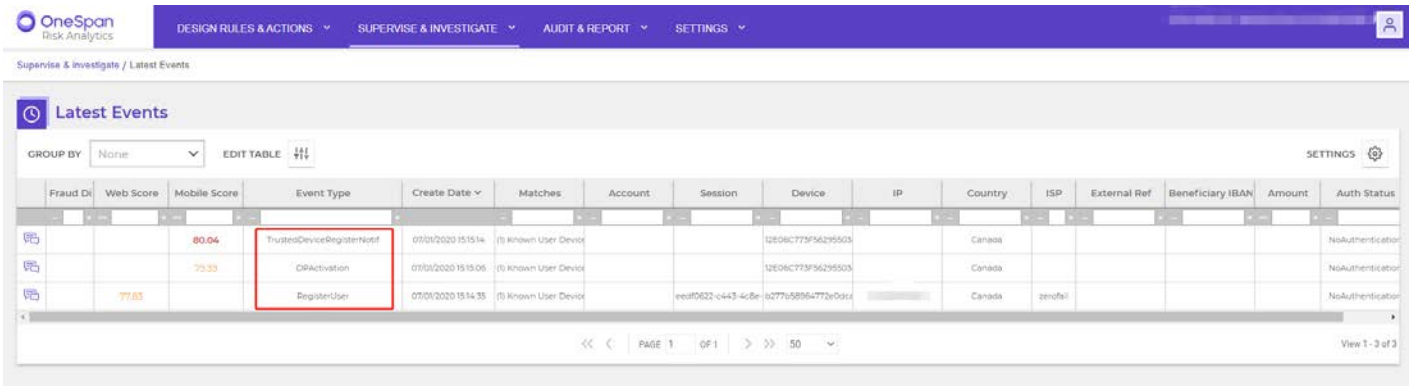


Launch the AAS Demo App, click the "SCAN" button and use the camera to scan the above visual code. Once the code was detected, the app will prompt you to enter a 6 digits security pin twice.

After completion the registration process, the demo app will jump to the user page and the browser will be redirected to the success URL.

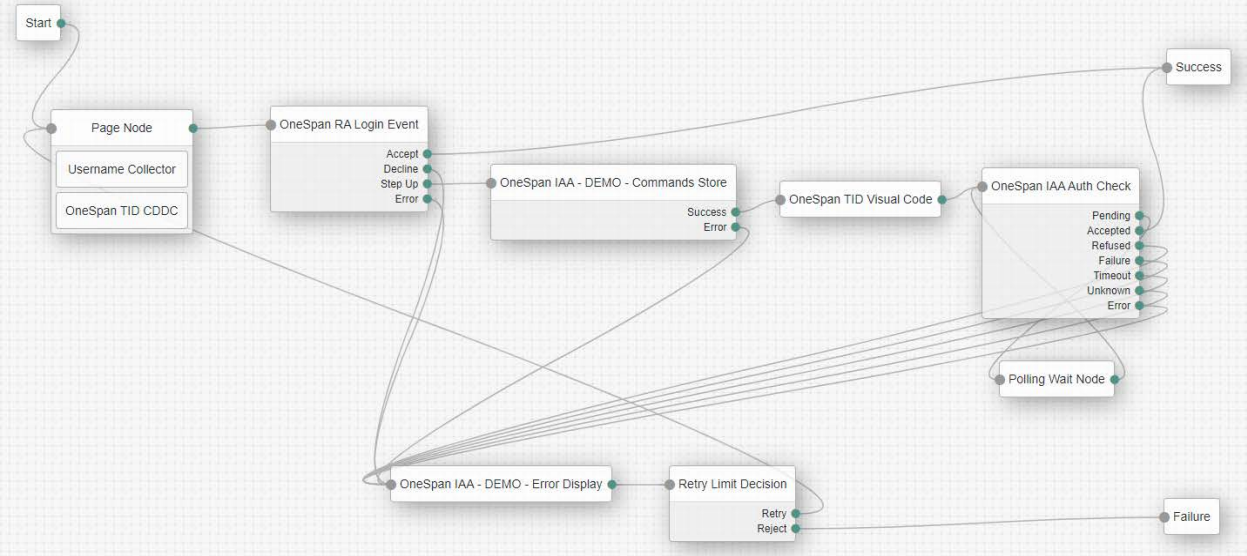


Log onto your IRM system and navigate to SUPERVISE & INVESTIGATE > Latest Events, you will find the user register process has been logged by the system with necessary information.



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

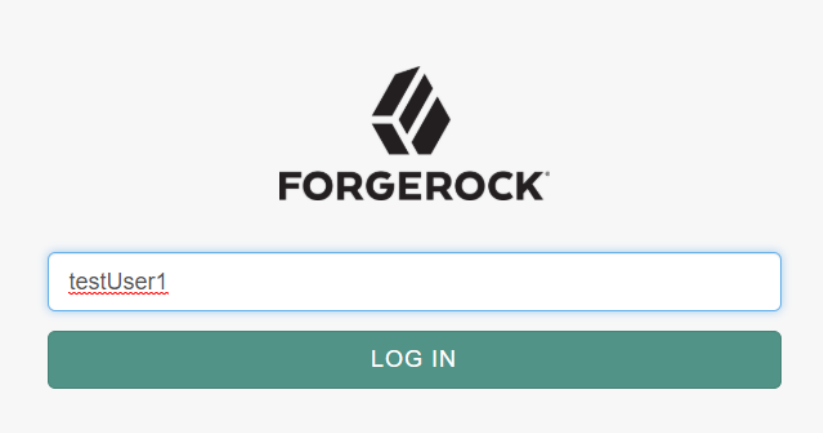
2. OneSpan IAA Login Event



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

https://{your_instance_url}/openam/XUI/?realm=/&service=OSIAAUserLogin#login

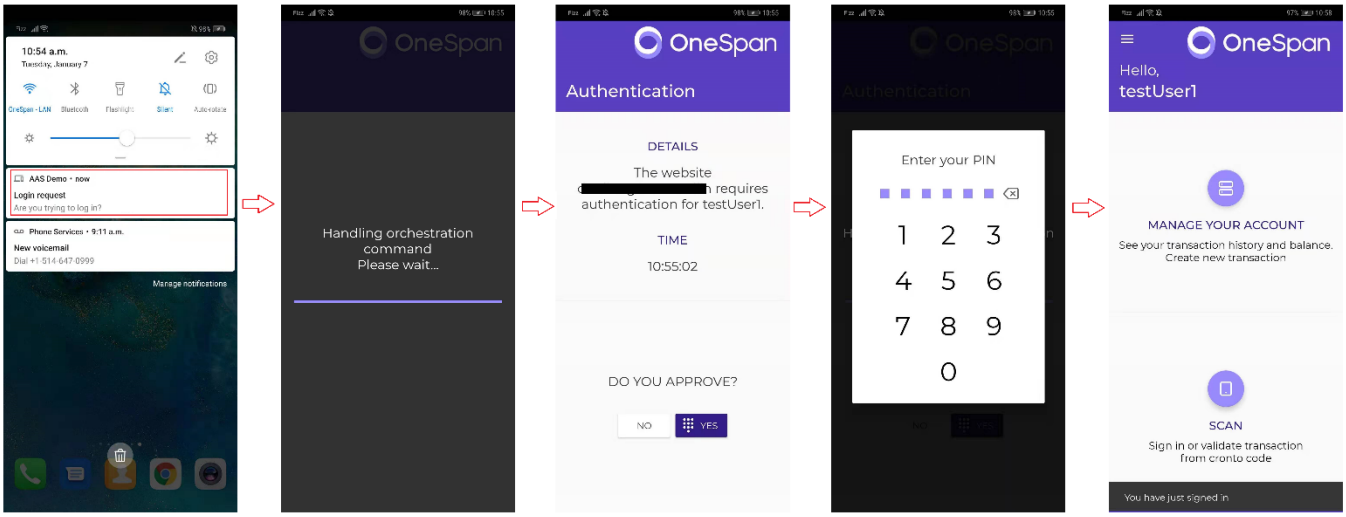
You will be prompt to input the username registered above.



The 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 IRM system, the outcome will directly fall within Accept or Decline depending on the risk score. However, because of the rules we've pre-set in the Risk Management service, the login service will challenge the end user by generating a remote authentication request sent via notification to the trusted device associated with the user.

Once the browser redirected to the visual code page, your mobile should have received a notification prompting for authentication. In case you missed the notification, you can also scan the visual code which contains the same request information and trigger the authentication process actively.

In both way, the demo app will handle the request command automatically, display the event details and prompt you to pass the challenge.

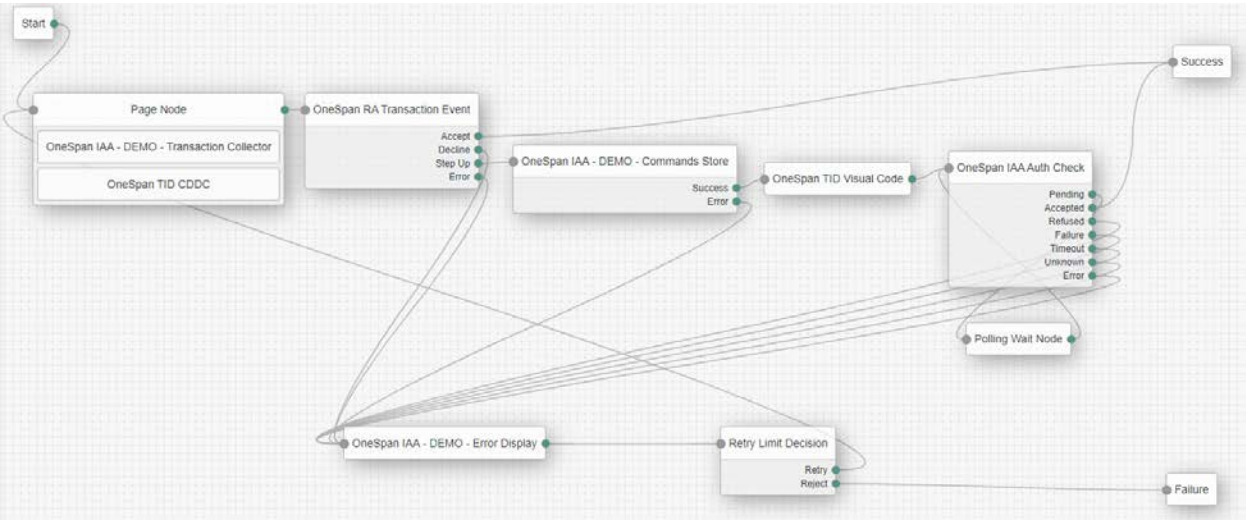


After completion the authentication process, the demo app will indicate that the event was successfully authenticated and the browser will be redirected to the success URL.

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

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

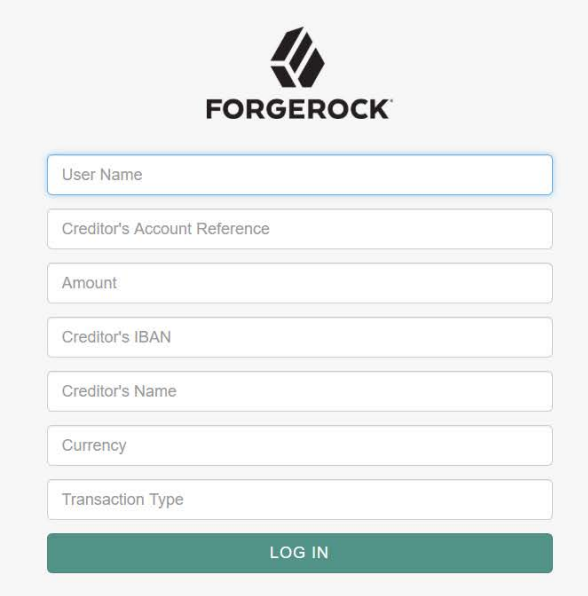
3. OneSpan IAA Transaction Event

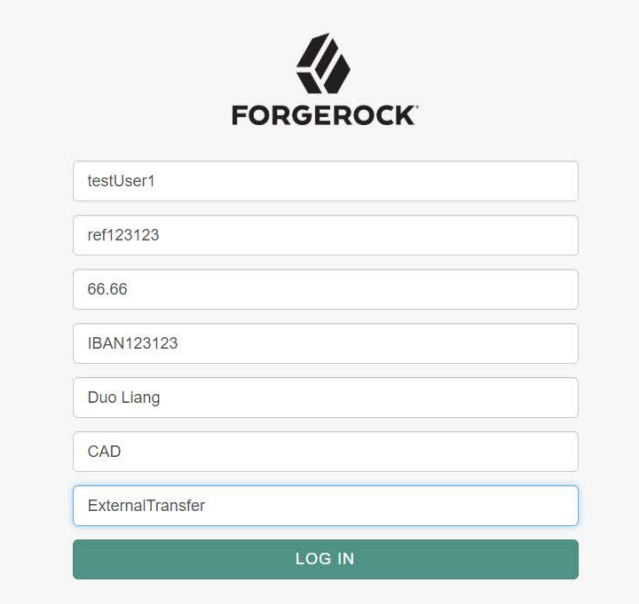


OneSpan IAA also provides the capability to evaluate the risk before an end-user tried to send a transaction, according to the transaction request details and the browser & mobile's context.

In this quick demo, we will use the "OneSpan IAA – DEMO – Transaction Collector" to pass in all the necessary transaction details. You can expect below screen after kicking off the authentication process:

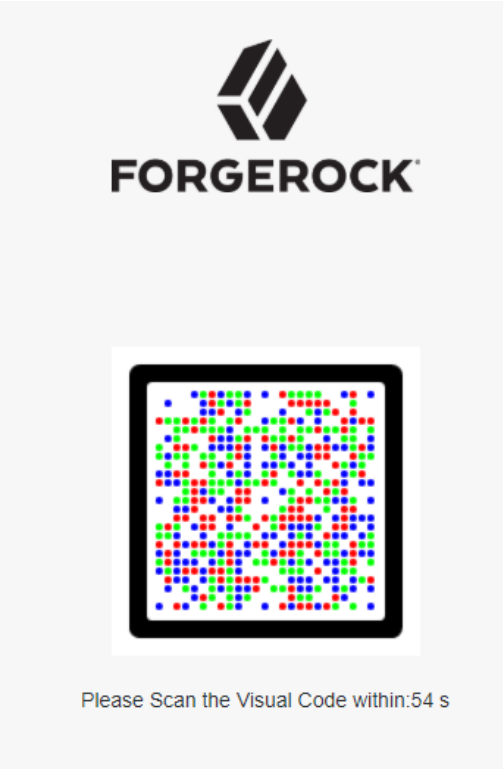
https://{your_instance_url}/openam/XUI/?realm=/&service=OSIAASendTransaction#login



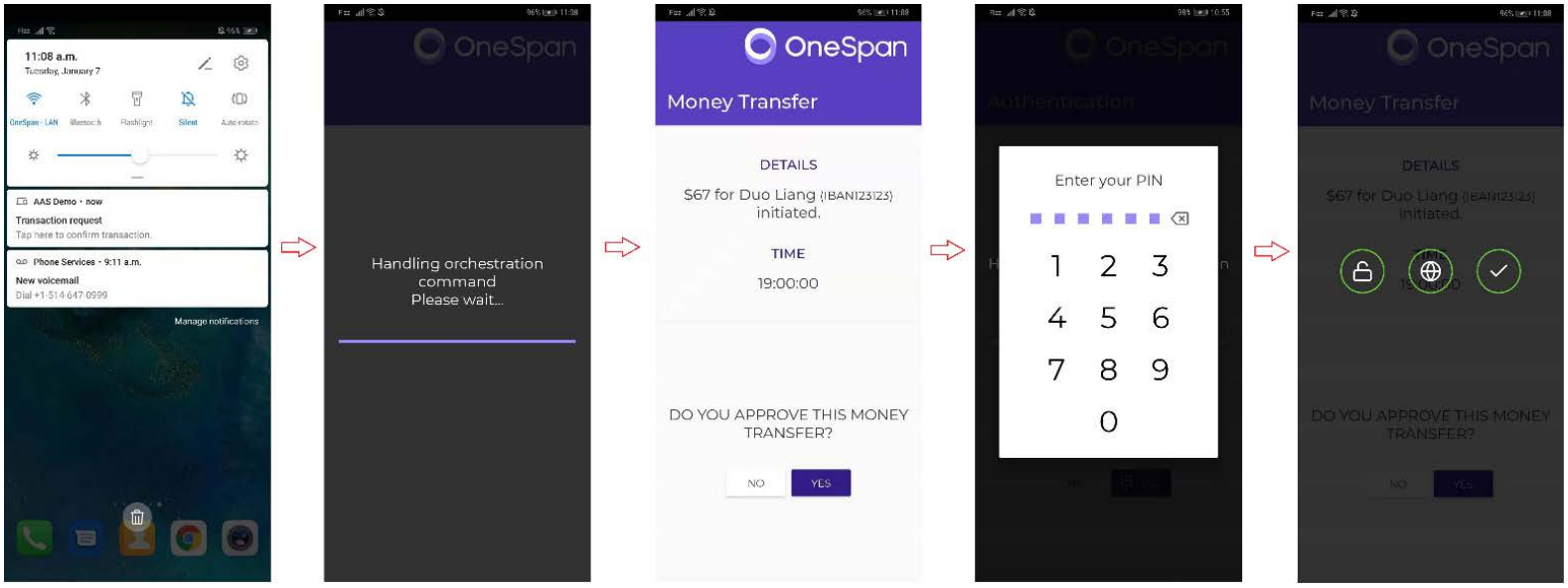


You can find all the available transaction types from the API Specification page > TransactionInput Schema > transactionType Attribute.

Similarly, the 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 IRM system, the outcome will directly fall within Accept or Decline depending on the risk score. However, because of the rules we've pre-set in the Risk Management service, the Transaction service will challenge the end user by generating a remote authentication request sent via notification to the trusted device associated with the user.

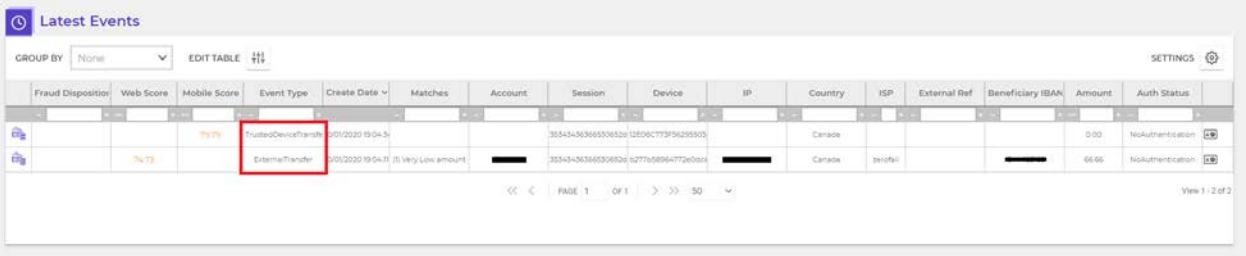


The end user can either scan the visual code or via the mobile notification to start the authentication process.

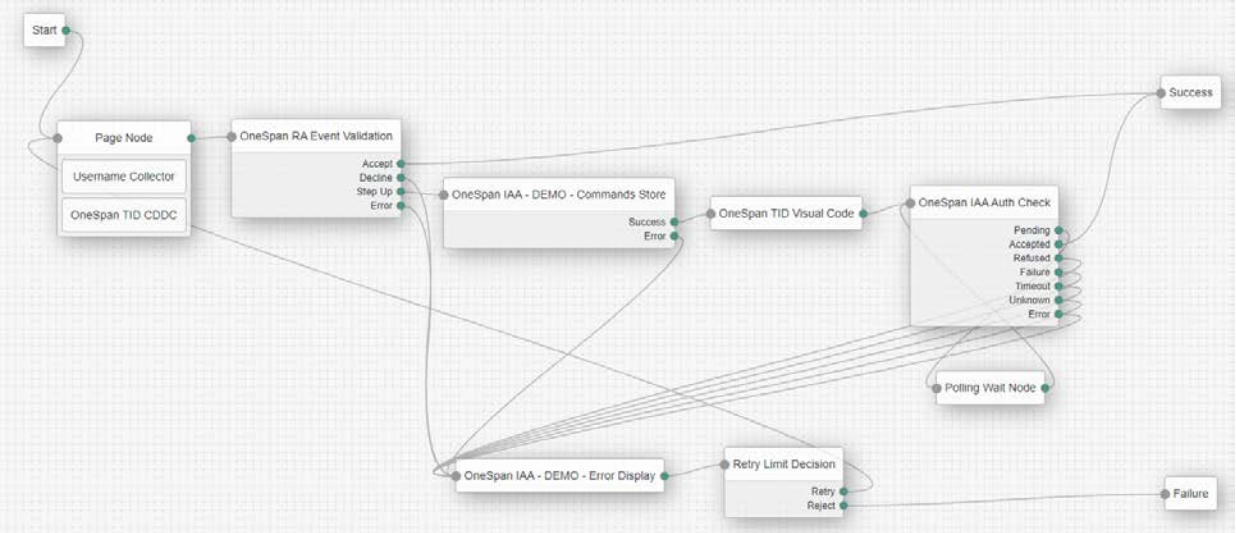


After confirming the details of the transaction event and pass the challenge, the demo app will indicate that the event was successfully authenticated and the browser will be redirected to the success URL.

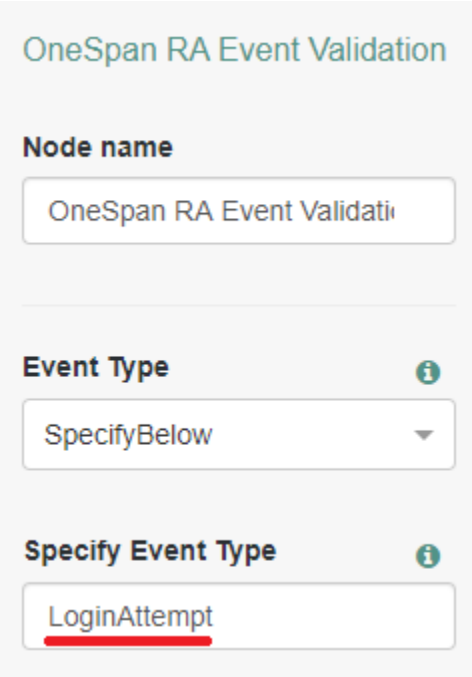
Now if you log onto your IRM system and navigate to SUPERVISE & INVESTIGATE > Latest Events, you will find that this transaction event has been logged by the system with necessary information.



4. OneSpan IAA Event Validation



For other non-monetary events, OneSpan IAA Auth nodes provide a general node to process and validate them. You can either hard code the event type at the node configuration or store the event type at the sharedState during the run time. As a quick demo, we will choose the first option and specify the Event Type as “LoginAttempt”.



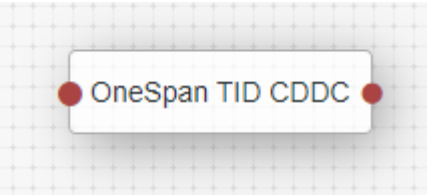
You can find all the available transaction types from the API Specification page > AdaptiveEventValidationInpu Schema > eventType Attribute.

When you start the authentication process with below link, the authentication flow should be exact the same as the use case 2 “OneSpan IAA Login Event”:

https://{your_instance_url}/openam/XUI/?realm=/&service=OSIAANonMonetaryEvents#login

Nodes Features

1. OneSpan TID 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 object. 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	"osstid_cddc_json"	Only when set False above, specify the hidden value ID for the CDDC Json.
CDDC Hash Callback ID	String	"osstid_cddc_hash"	Only when set False above, specify the hidden value ID for the CDDC hash value.

Data Flow:

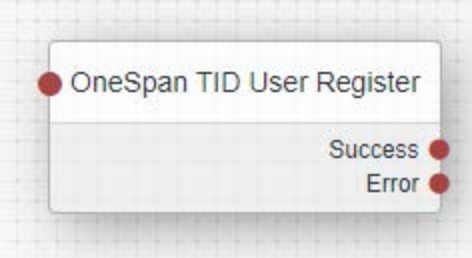
This node requires below inbound data

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	“osstid_cddc_json”	Shared State
CDDC hash value	“osstid_cddc_hash”	Shared State
CDDC client IP	“osstid_cddc_ip”	Shared State

2. OneSpan TID User Register



Introduction:

This node invokes the User Register/Unregister Service API, in order to validate and process the registration/unregistration of a user.

Available Properties:

Property	Type	Default Value	Usage
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.

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 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	"osstid_cddc_json"	Shared State
CDDC hash value	"osstid_cddc_hash"	Shared State
CDDC client IP	"osstid_cddc_ip"	Shared State

This code will store below outbound data

Case 1: When the node function is set to “UserRegister”:

Description	Attribute Name	Storage
Username in sharedState	“ostid_username_in_shared_state”	Shared State
IAA session ID	“osstid_session_id”	Shared State
User activation code	“ostid_activation_code”	Shared State
The visual code message	“ostid_cronto_msg”	Shared State
OneSpan DigiPass serial assigned to the trusted device	“osstid_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 TID Visual Code” node – “Message Options” property for more details.

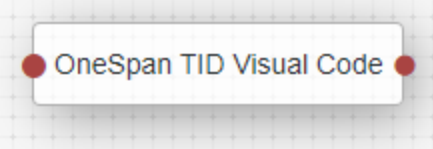
Case 2: When the node function is set to “UserUnregister”:

Description	Attribute Name	Storage
IAA session ID	“osstid_session_id”	Shared State

Case 3: When the outcome is “Error”

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

3. OneSpan TID Visual Code



Introduction:

This node reads the visual code message from the sharedState and renders it as a visual code, which allows the device integrated with the Mobile Security Suite SDKs to scan with. The node can be a part of the customized page node or be used alone.

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	“osstid_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:

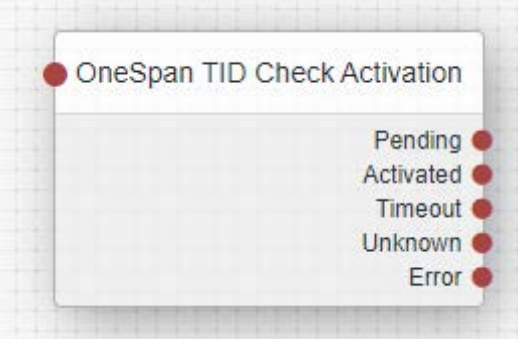
This node requires below inbound data

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 TID Check Activation



Introduction:

This node invokes the Check Activation Status Service API, in order to 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

This code will store below outbound data

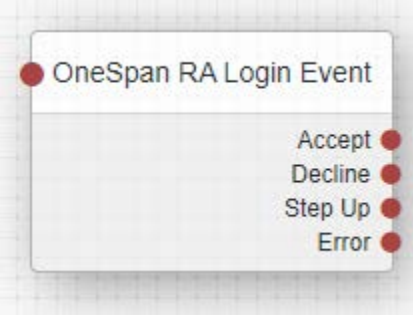
When the outcome is “Error”:

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

API Reference:

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

5. OneSpan RA Login Event



Introduction:

This node invokes the Login Service API, in order to validate a login request against the Risk Management Service and the Authentication Service, and returns the results of the authentication attempt.

If IRM 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
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.
Require Password	boolean	False	Determine whether to include OneSpan IAA user password when attempting to login.
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.
Send Notification	Enum	Default	Determine whether to send a Mobile APP notification to the trusted device.
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.
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:

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)Password	As specified in property	Transient State
(Optional) Other Attributes	As specified in property	Shared State
CDDC Json	“osstid_cddc_json”	Shared State
CDDC hash value	“osstid_cddc_hash”	Shared State

CDDC client IP	“osstid_cddc_ip”	Shared State
(Optional) IAA Session Id	“**osstid_session_id**”	Shared State

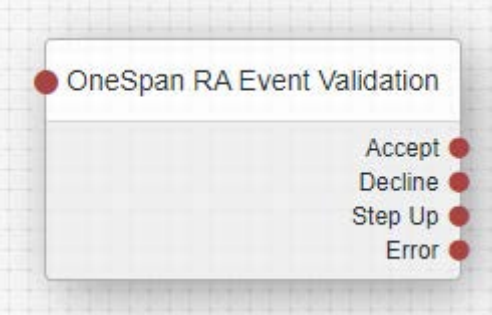
This code will store below outbound data

Description	Attribute Name	Storage
The visual code message	“ostid_cronto_msg”	Shared State
IAA session ID	“osstid_session_id”	Shared State
IAA request ID	“ostid_request_id”	Shared State
OneSpan IRM response	“ostid_irm_response”	Shared State
IAA 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

6. OneSpan RA Event Validation



Introduction:

This node invokes the Event Validation Service API, in order to validate a non-monetary event against the Risk Management Service and the Authentication Service, and returns the result.

If IRM 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
Require Password	boolean	False	Determine whether to include OneSpan IAA user password when attempting to login.
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
Send Notification	Enum	Default	Determine whether to send a Mobile APP notification to the trusted device.
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.
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	“osstid_cddc_json”	Shared State

CDDC hash value	“osstid_cddc_hash”	Shared State
CDDC client IP	“osstid_cddc_ip”	Shared State
(Optional) IAA Session Id	“**osstid_session_id**”	Shared State

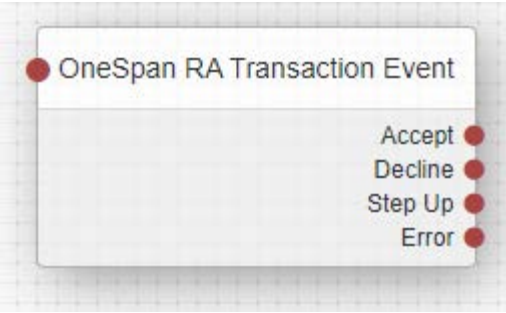
This code will store below outbound data

Description	**Attribute Name**	**Storage**
The visual code message	“ostid_cronto_msg”	Shared State
IAA session ID	“osstid_session_id”	Shared State
IAA request ID	“ostid_request_id”	Shared State
OneSpan IRM response	“ostid_irm_response”	Shared State
IAA 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

7. OneSpan RA Transaction Event



Introduction:

This node invokes the Transaction Service API, in order to validate a monetary transaction request against the Risk Management Service and the Authentication Service, and returns the result.

If IRM 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
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
Require Password	boolean	False	Determine whether to include OneSpan IAA user password when attempting to login.
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
Transaction Type In SharedState	String	“transactionType”	Specify the name of a key in the sharedState object in which to represent the Transaction Type.
Currency In SharedState	String	“currency”	Specify the name of a key in the sharedState object in which to represent the Transaction Currency.
Amount In SharedState	String	“amount”	Specify the name of a key in the sharedState object in which to represent the Transaction Amount.
Creditor IBAN In SharedState	String	“creditorIBAN”	Specify the name of a key in the sharedState object in which to represent the Transaction Creditor’s IBAN.
Creditor Account Reference In SharedState	String	“accountRef”	Specify the name of a key in the sharedState object in which to represent the Transaction Creditor’s Account Reference.
Creditor Name In SharedState	String	“creditorName”	Specify the name of a key in the sharedState object in which to represent the Transaction Creditor’s Name.
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
Send Notification	Enum	Default	Determine whether to send a Mobile APP notification to the trusted device.
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.
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 Service API for more details.

Data Flow:
This node requires below inbound data

Description	Attribute Name	Source
Username	As specified in property	Shared State
(Optional) Password	As specified in property	Transient State
Transaction Type	As specified in property	Shared State

Transaction Currency	As specified in property	Shared State
Transaction Amount	As specified in property	Shared State
Creditor's IBAN	As specified in property	Shared State
Creditor's Account Reference	As specified in property	Shared State
Creditor's Name	As specified in property	Shared State
(Optional) Other attributes	As specified in property	Shared State
CDDC Json	"osstid_cddc_json"	Shared State
CDDC hash value	"osstid_cddc_hash"	Shared State
CDDC client IP	"osstid_cddc_ip"	Shared State
(Optional) IAA Session ID	"**osstid_session_id**"	Shared State

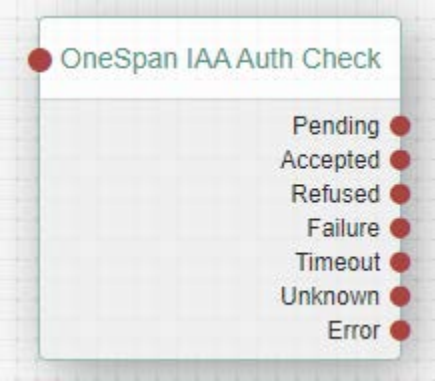
This code will store below outbound data

Description	**Attribute Name**	**Storage**
The visual code message	"ostid_cronto_msg"	Shared State
IAA session ID	"osstid_session_id"	Shared State
IAA request ID	"ostid_request_id"	Shared State
OneSpan IRM response	"ostid_irm_response"	Shared State
IAA 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 IAA Auth Check



Introduction:

This node invokes the Check Session Status Service API, in order to checks 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.