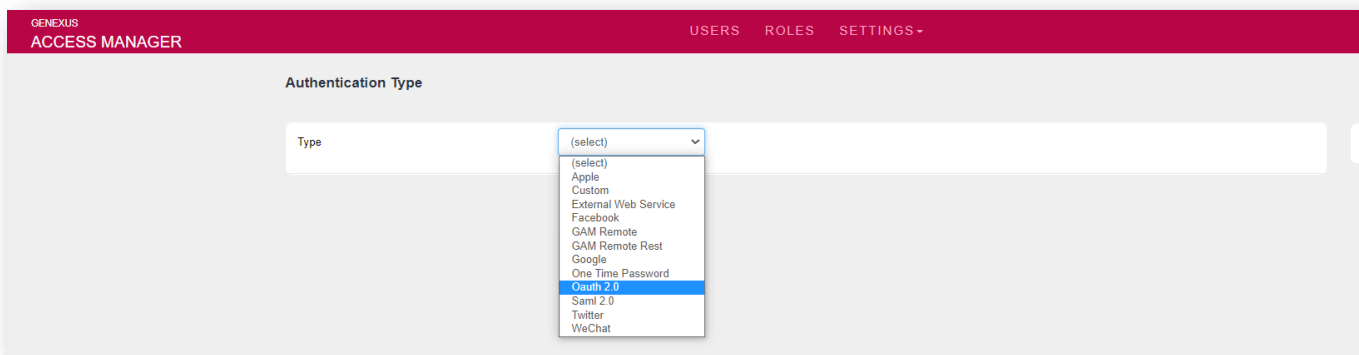
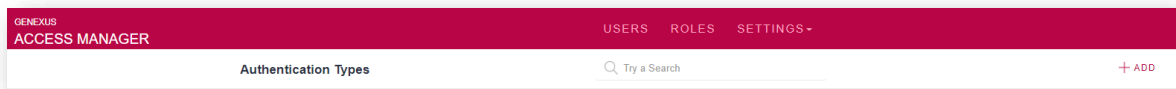


DEMO: OpenID Connect

Primer demo: OpenID Connect



Para esta demo, utilizaremos el protocolo Oauth 2.0 en GAM. Nuestro proveedor de identidad será Azure Active Directory a través de Microsoft.

Asumiremos que la configuración desde el lado de Azure ya se realizó correctamente y no entraremos en detalle de la misma. Para ver como realizarla puede encontrar en la Wiki de GeneXus un artículo detallado sobre esto.

En primer lugar, debemos crear un nuevo Tipo de autenticación GAM Oauth 2.0 y definir los conceptos básicos, como lo es el Nombre, Descripción, etc.

General	Authorization	Token	User Information	
Client Id	Tag	<input type="text" value="client_id"/>	Value	<input type="text" value="2d55e4aa-22c6-476e-acc6-8f3728018bc9"/>
Client Secret	Tag	<input type="text" value="client_secret"/>	Value	<input type="text" value=""/>
Redirect URL	Tag	<input type="text" value="redirect_uri"/>	Value	<input type="text" value="http://localhost:8080/GAMCourse.JavaEnvironment/"/>
Custom Redirect URL?	<input type="checkbox"/>			
Redirect to authenticate?	<input type="checkbox"/>			

En la pestaña General, se debe definir lo siguiente:
Primero seteamos el Client ID y Client Secret que obtenemos desde Azure.
La URL de redirección debe ser la URL Base del Backend de nuestra aplicación.

Como comentamos en el video anterior, no marcamos la opción Redireccionar para autenticar ya que queremos loguearnos desde el GAM mismo.

General	Authorization	Token	User Information		
			https://login.microsoftonline.com/{tenat}/oauth2/v2.0/authorize		
URL	https://login.microsoftonline.com/.../oauth2/v2.0/authorize				
Advanced configuration					
ResponseType	<input checked="" type="checkbox"/>	Tag	response_type	Value	code
Scope	<input checked="" type="checkbox"/>	Tag	scope	Value	https://graph.microsoft.com/user.read
State	<input checked="" type="checkbox"/>	Tag	state		
Include Client Id	<input checked="" type="checkbox"/>				
⋮					
Response					
Access Code Tag	<input type="text" value="code"/>				
Error Description Tag	<input type="text" value="error_description"/>				

Ahora nos dirigimos a la pestaña Authorization.

Acá debemos setear la URL de Azure obtenida desde su portal, la cual luce de la siguiente manera.

Lo segundo a modificar, debe ser el Scope, el cual debe contener la URL que vemos en pantalla.

El resto queda todo por defecto.

General	Authorization	Token	User Information
			https://login.microsoftonline.com/{tenat}/oauth2/v2.0/token
URL		https://login.microsoftonline.com/ /oauth2/v2.0/token	
<i>Advanced configuration</i>			
Token Method	POST		
Header	Tag	Content-type	Value
			application/x-www-form-urlencoded
Include Authentication header?	<input type="checkbox"/>	Method	Basic
			Realm
Include Authorization header with Basic value?	<input type="checkbox"/>		
Grant Type	<input checked="" type="checkbox"/>	Tag	grant_type
			Value
			password
⋮			
Additional Parameters	scope=https://graph.microsoft.com/user.read		
⋮			

En la pestaña Token, nuevamente seteamos la URL de Azure obtenida desde su portal, la cual luce de la siguiente manera.

El resto queda por defecto, excepto los campos Grant Type y Additional Parameters, los cuales seteamos con lo que vemos en pantalla.

Cabe aclarar que esto ultimo solo se debe cambiar cuando queremos que no se redirija al momento de loguearse y se realice desde el login de GAM. En caso contrario, el Grant Type deben dejarlo con el valor por defecto (que es *authorization_code*) y sin parámetros adicionales.

El resto de las opciones quedan configuradas por defecto.

General	Authorization	Token	User Information
URL			
User Info Method			
Header			
Include Access Token			
Include Client Id			
Include Client Secret			
Include User Id			
User Email Tag			mail
User Verified Email Tag			
User External Id Tag			id
User Name Tag			userPrincipalName
User First Name	Tag		givenName
User Last Name	Tag		surname
User Gender	Tag		gender
User Birthday Tag			birthday
User URL Image Tag			picture
User URL Profile Tag			link
User Language Tag			locale
User Time Zone Tag			timezone
Error Description Tag			message

[Advanced configuration](#)

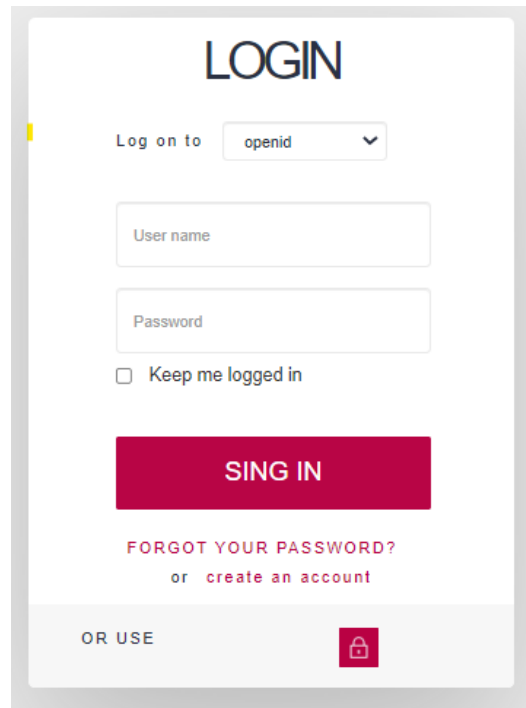
charset=utf-8

Finalmente, en la pestaña User Information, aquí simplemente seteamos la URL que vemos en pantalla (también obtenida desde Azure) y no incluimos nada.

De esta forma es como se crea el usuario en el GAM local, y es desde donde se mapea la obtención de la información del usuario según el IDP configurado. El IDP debe retornar un identificador único de usuario, el cual se debe configurar en "User External Id Tag", y por este es que en los sucesivos login de GAM se tiene certezas de que se está autenticando el mismo usuario.

El resto de los parámetros, los seteamos como vemos en pantalla.

Con esto concluimos la configuración.



The image shows a login form with the following elements:

- Header:** The word "LOGIN" in a large, bold, black font.
- Log on to:** A dropdown menu currently showing "openid".
- User name:** A text input field with the placeholder text "User name".
- Password:** A text input field with the placeholder text "Password".
- Keep me logged in:** A checkbox followed by the text "Keep me logged in".
- SING IN:** A prominent red button with white text.
- FORGOT YOUR PASSWORD? or create an account:** Text in red, with "create an account" being a link.
- OR USE:** Text in grey, followed by a red icon of a padlock.

Ahora simplemente basta con ir al Login, seleccionar loguearnos con el tipo de autenticación recién creado e ingresar las credenciales de un usuario definido en Azure.

Eso es todo.

DEMO: IDP

Segunda demo: IDP.

Application

General Rem

Client ID WEB (Identity Provider, SSO)

Client secret Allow authentication?

Oauth single user Can get user roles?

WEB (Identity Can get user additional data?

Allow authenticati Can get session initial properties?

REST OAUT Image URL

Allow authenticati Local login URL

Callback URLs

Para esta demo, volveremos a utilizar el protocolo Oauth 2.0. El GAM a través de el, será nuestro proveedor de identidad.

En primer lugar, debemos configurar nuestra aplicación GAM definida en el servidor del IDP que interactuará como el proveedor.

Para esto, debemos dirigirnos a la pestaña “Remote Authentication” en las configuraciones de la aplicación desde el Backend de GAM.

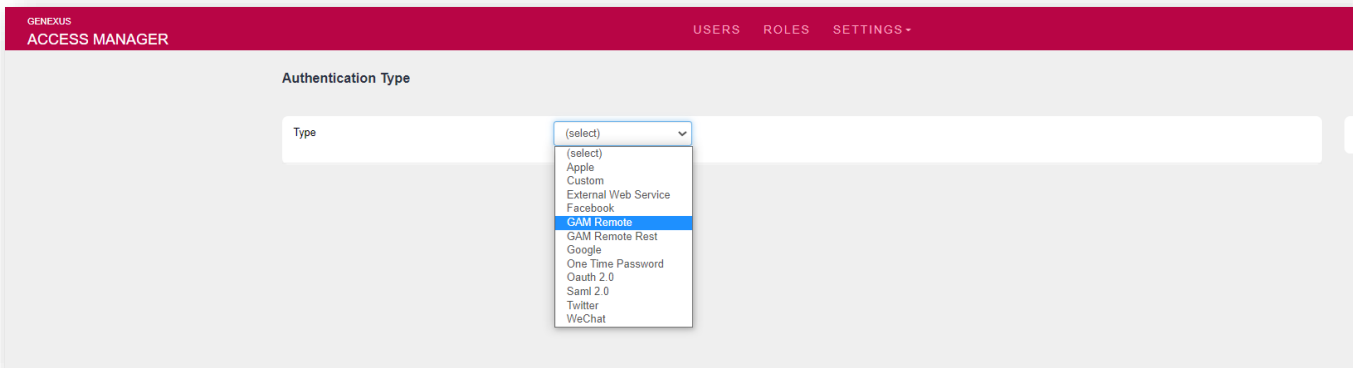
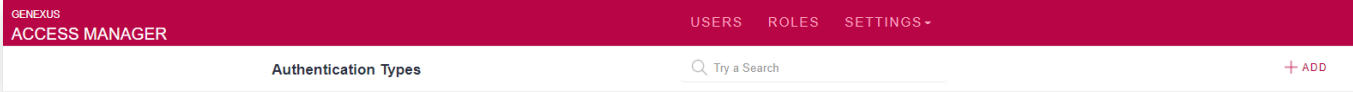
Acá nos guardamos el Client ID y Client Secret para setearlos luego en la aplicación cliente.

Luego, debemos marcar la opción de permitir la autenticación en el apartado WEB (Identity Provider, SSO). Allí, se puede indicar si se quiere compartir con el Cliente los roles de los usuarios, la información adicional, etc.

Lo importante a mostrar en la demo, son las URL de login local y callback. La primera debe corresponder al Web Panel GAMRemoteLogin de la KB, quien será la que realice el proceso de login en el IDP. La segunda, debe ser el path de la aplicación cliente desde donde se invocará al IDP la cual será llamada después de que haya terminado el proceso de Login. Este ultimo parámetro puede estar compuesto por mas de una URL, las cuales deben ser separadas por punto y coma.

Obviamente es en este GAM donde se deben tener definidos los usuarios que serán utilizados para loguearse en el IDP.

Teniendo esta configuración y un usuario creado, ya finalizamos el proceso desde el lado del IDP.



Veamos el lado del Cliente.

El primer paso es dirigirnos a Tipos de Autenticación desde el menú del backend de GAM, y crear uno de tipo GAM Remote.

Authentication Type	
Type	GAM Remote
Name*	gamremote
Function	Only Authentication ▼
Enabled?	<input checked="" type="checkbox"/>
Description	<input type="text" value="gamremote"/>
Small image name	<input type="text"/>
Big image name	<input type="text"/>
Impersonate	(none) ▼
Client Id.*	<input type="text" value="b62c8a436ca34614821066e7d1c94ff8"/>
Client Secret*	<input type="text" value=""/>

Lo importante a configurar aquí es lo siguiente:

La propiedad Function, la setearemos en Only Authentication dado que en el lado del servidor de IDP no le indicamos que se compartan los roles de usuario. En caso de que pongamos la otra opción, Autenticacion y roles, al momento de loguearnos nos devolverá un error.

Lo siguiente a configurar, son el Client Id y Client Secret que nos guardamos desde el IDP.

Local site URL*	<input type="text" value="http://localhost:8080/App1.JavaEnvironment"/>		
Custom callback URL?	<input type="checkbox"/>		
Add gam_user_additional_data scope?	<input type="checkbox"/>	Add gam_session_initial_prop scope?	<input type="checkbox"/>
Additional Scope	<input type="text"/>		
Remote server URL*	<input type="text" value="http://localhost:8080/AppIDP.JavaEnvironment"/>		
Private encryption key	<input type="text"/>		
Repository GUID	<input type="text"/>		
Validate external token	<input type="checkbox"/>		

\$Server/<Base_URL>

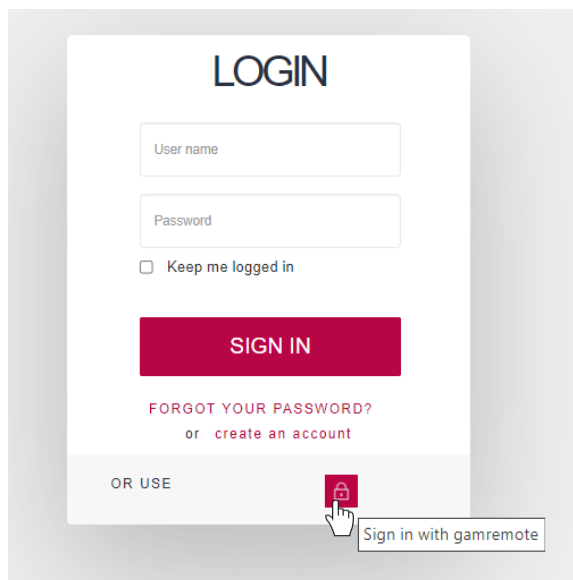
Posteriormente configuraremos la propiedad “Local site URL” con la dirección de nuestra aplicación cliente, la misma que ya especificamos en la Callback URL en el servidor; también la propiedad “Remote server URL” con la dirección del IDP, siguiendo el formato que vemos en rojo.

Como comentarios adicionales:


La propiedad “Add gam_user_additional_data scope?” **debe activarse cuando queremos pasar datos adicionales del usuario**. En el lado del servidor, se debe seleccionar la propiedad Permitir autenticación, en la sección Web (Proveedor de identidad, SSO).

La propiedad “Add gam_session_initial_prop scope?” consiste en pedirle al IDP que devuelva al cliente las propiedades iniciales establecidas dinámicamente en el inicio de sesión. Por supuesto, en el IDP también se debe configurar que se envíe esta información.

Finalmente, la propiedad “Validate External Token” valida el vencimiento de la sesión según la expiración del token y lo renueva automáticamente sin tener que realizarlo nosotros manualmente.



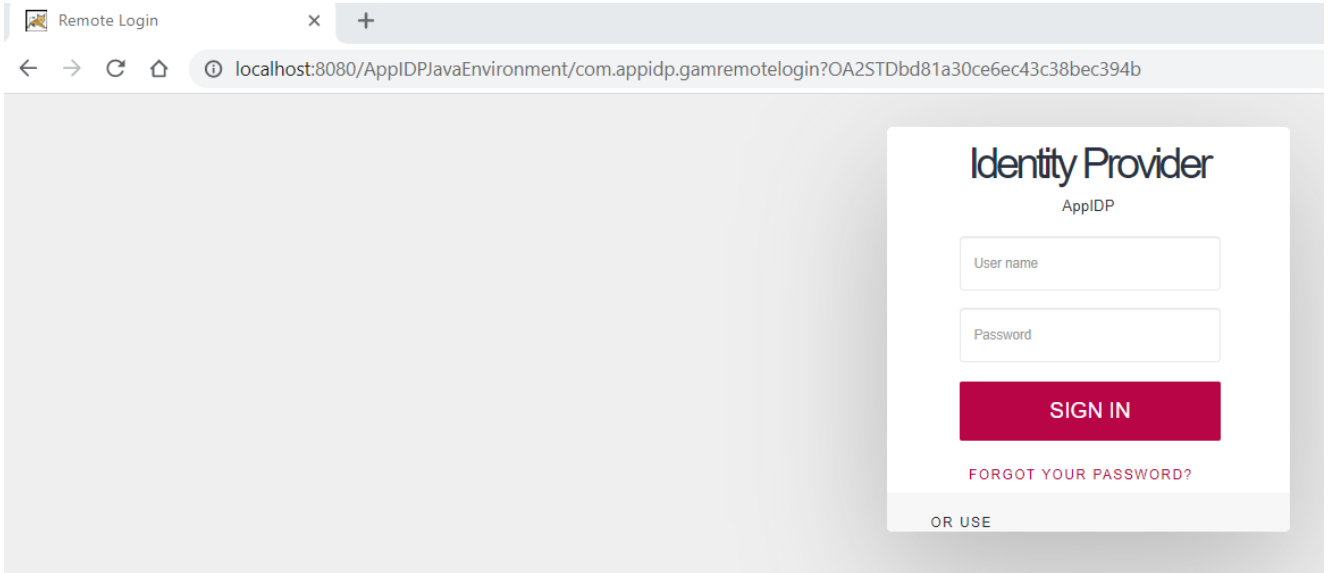
The image shows a login interface with the following elements:

- LOGIN** (Title)
- (User name)
- (Password)
- Keep me logged in
- SIGN IN** (Red button)
- FORGOT YOUR PASSWORD?** (Text)
- or **create an account** (Text)
- OR USE** (Text)
-  (Red icon)
- Sign in with gamremote** (Text in tooltip)

A efectos de la demo, creamos un Web Panel en la aplicación Cliente, donde nos muestra los datos del usuario logueado. Obviamente este objeto tiene activado la seguridad integrada con valor Autenticación.

Cuando queremos acceder a el, dado que no estamos logueados nos redirige al login.

Veamos que ahora que tenemos definido el tipo de autenticación Oauth, desde el login tenemos la opción de acceder a través de esa misma.



Al hacer clic en esa opción, vemos que nos redirige al IDP y su login remoto. Procedemos a loguearnos con el usuario que habíamos definido en el IDP.

App Client

User

GENEXUS ACCESS MANAGER USERS ROLES SETTINGS - A Administrator

✓ SHOW FILTERS Users 🔍 Try a Search + ADD

User Name	First Name	Last Name	Authentication	
nadrien@mail.com	Nicolas	Adrién	idp	EDIT
admin	Administrator	User	local	EDIT

FIRST / PREV / NEXT

Username	nadrien@mail.com
E-Mail	nadrien@mail.com
First Name	Nicolas
Last Name	Adrién

Efectivamente vemos como ahora nos redirigió a nuestro Web Panel con la información del usuario logueado.

Si nos dirigimos al Backend de la aplicación cliente, podemos ver como fue creado el usuario que habíamos creado en el IDP con su información.

DEMO: Custom Authentication

Tercer demo: Autenticación Custom.


```

Parm(in:&StrInput, out:&StrOutput); //&StrInput and &StrOutput are varchar(256)

&Key = '03E1E1AAA5BCA19FBA8C42058B4ABF28'
&GAMWSLoginIn.FromJson(&StrInput) // &GAMWSLoginIn is &GAMWSLoginInSDT data type

//Decrypt parameters
&UserLogin = Decrypt64( &GAMWSLoginIn.GAMUsrLogin, &Key )
&UserPassword = Decrypt64( &GAMWSLoginIn.GAMUsrPwd, &Key )
&GAMWSLoginOut = New GAMWSLoginOutSDT() //&GAMWSLoginOut is &GAMWSLoginOutSDT data type
&GAMWSLoginOut.WSVersion = GAMAutExtWebServiceVersions.GAM10
&GAMWSLoginOut.User = New GAMWSLoginOutUserSDT()

Do 'ValidUser'

&StrOutput = &GAMWSLoginOut.ToJson()

Sub 'ValidUser'
  If &UserLogin = !"user"
    If &UserPassword = !"password"
      &GAMWSLoginOut.WSStatus = 1
      &GAMWSLoginOut.User.Code = !"code"
      &GAMWSLoginOut.User.FirstName = !"FirstName"
      &GAMWSLoginOut.User.LastName = !"LastName"
      &GAMWSLoginOut.User.Email = !"name2@domain.com"
      Do 'GetRoles' //optional
    Else
      &GAMWSLoginOut.WSStatus = 3
    EndIf
  Else
    &GAMWSLoginOut.WSStatus = 2
  EndIf
EndSub

Sub 'GetRoles'
  &GAMWSLoginOutUserRol = New()
  &GAMWSLoginOutUserRol.RoleCode = "role_1"
  &GAMWSLoginOut.User.Roles.Add(&GAMWSLoginOutUserRol)
  &GAMWSLoginOutUserRol = New()
  &GAMWSLoginOutUserRol.RoleCode = "role_2"
  &GAMWSLoginOut.User.Roles.Add(&GAMWSLoginOutUserRol)
EndSub

```

Para realizar una autenticación Custom, debemos crear un procedimiento. En la Wiki de GeneXus pueden encontrar el ejemplo que vemos en pantalla, con una lógica muy simple ya definida. Queda a cargo del desarrollador modificarla bajo sus condiciones.

En primer lugar, vemos que como reglas se definen dos Varchar: uno de entrada y otro de salida, los cuales traerán los datos ingresados por el usuario y devolverán el resultado del login con determinada información del usuario (esto ultimo en caso de éxito por supuesto).

Luego, se define una clave que mas adelante entraremos en detalle, y se descripan los parámetros de ese parámetro de entrada procedente de las reglas, además de crearse un data type que será cargado en el parámetro de salida al finalizar.

Posteriormente, en el método **ValidUser** se realiza la validación del nombre de usuario y contraseña, el cual está hecho a modo de ejemplo verificando que el nombre de usuario es "user" y la contraseña es "password". En caso contrario, se devuelven distintos errores según la falla.

Este método debe cambiarse por una lógica de login mas segura y que no haga diferenciación entre los errores según usuario o contraseña.

Opcionalmente, se puede utilizar el método **GetRoles** para definirle determinados roles al usuario logueado.

Este método es de utilidad cuando queremos programar nosotros como validamos

la contraseña de un usuario, ya sea para validarlo contra una base de datos local, contra un LDAP o contra otro lugar donde se encuentren almacenadas las credenciales de los usuarios.

GENEXUS
ACCESS MANAGER

USERS ROLES

SECURITY POLICIES
APPLICATIONS
REPOSITORY CONFIGURATION
REPOSITORY CONNECTIONS
AUTHENTICATION TYPES
CHANGE PASSWORD
CHANGE WORKING REPOSITORY
EVENT SUBSCRIPTIONS
GAM CONFIGURATIONS

A Administrator

X HIDE FILTERS

Users

+ ADD

GENDER
(All)

AUTHENTICATION TYPE
(All)

ROLE
(All)

User Name	First Name		Authentication	
custom	FirstName		custom	EDIT
admin	Administrator	User	local	EDIT
test	Test	GAM	local	EDIT


FIRST / PREV / NEXT

Ahora que ya contamos con un procedimiento personalizado para la autenticación, debemos configurarlo en GAM.

El primer paso es dirigirse a Authentication Types, y crear uno nuevo de tipo Custom.

GENEXUS ACCESS MANAGER USERS ROLES SETTINGS - A Administrator

Authentication Type

Type	Custom	 Delete
Name*	custom	
Function	Authentication and Roles ▾	
Enabled?	<input checked="" type="checkbox"/>	
Description	Custom Authentication Type	
Small image name	<input type="text"/>	
Big image name	<input type="text"/>	
Impersonate	(none) ▾	
Enable Two Factor Authentication?	<input type="checkbox"/>	
JSON version	Version 1.0 ▾	
Private encryption key	03E1E1AA5BCA19FBAB8C42058B4ABF	<input type="button" value="Generate Key Custom"/>
File name*	agamlogincustom.class	
Package	com.gamcourse	
Class name*	agamlogincustom	

Aquí, las configuraciones a resaltar son las siguientes:

Función: Permite especificar si queremos que el tipo de autenticación sea Autenticacion y roles, o solo Autenticacion. En nuestro caso dejamos la primer opción.

Private encryption key, aquí se debe configurar la clave de cifrado utilizada en el procedimiento para descifrar el usuario y contraseña recibido. Si recuerdan en la slide del procedimiento GeneXus que mostré anteriormente se definía una clave la cual es la que ingresamos en esta propiedad. Esta es útil porque la función de encriptación de GeneXus la utiliza para cifrar el nombre de usuario y contraseña cuando se pasan al programa.

Nombre de archivo: aquí se especifica el nombre del archivo que corresponde al procedimiento externo. En el caso de Java es opcional.

Paquete: aquí se especifica en el caso de modelos Java el mismo valor de propiedad de nombre de paquete de Java, y en el caso de modelos NET el valor de la propiedad de espacio de nombres de la aplicación. Esta propiedad es opcional y depende si el procedimiento o programa utilizado tiene paquete o no.

Finalmente **Nombre de la clase,** una propiedad obligatoria que especifica el nombre de la clase del procedimiento.

LOGIN

Log on to

Custom Authentication Type ▾

User name

Password

Keep me logged in

SING IN

[FORGOT YOUR PASSWORD?](#)
or [create an account](#)

Luego de tener todo configurado, simplemente en nuestro login se setea el tipo de autenticación custom, y se realiza el login.

Un detalle a mencionar, es que en este caso el tipo de autenticación se selecciona a través del combo resaltado debido a que le indicamos que no se redireccione al IDP. En caso contrario, el tipo de autenticación se muestra como un icono en la parte inferior del login como veíamos en la Demo del IDP.

DEMO: OTP

OTP.

Repository Configuration

General Users Session **E-Mail**

Server Host	<input type="text" value="smtp.mail.outlook.com"/>	Server Port	<input type="text" value="587"/>
Timeout (seconds)	<input type="text" value="20"/>	Secure	<input checked="" type="checkbox"/>
Sender email address	<input type="text" value="admin@outlook.com"/>	Sender name	<input type="text" value="Mail"/>
Server require authentication?	<input checked="" type="checkbox"/>		
User name	<input type="text" value="admin@outlook.com"/>	Password	<input type="password" value=""/>
Send email when user activate account?	<input type="checkbox"/>		
Send email when user change password?	<input type="checkbox"/>		
Send email when user change email/username?	<input type="checkbox"/>		
Send email for recovery password?	<input type="checkbox"/>		

Un requisito previo para hacer funcionar OTP, es que el repositorio debe tener configurado el servicio de mail para enviar los códigos. Esto se configura en la opción “Configuración del Repositorio” del backend de GAM.

GENEXUS ACCESS MANAGER

USERS ROLES

SECURITY POLICIES
APPLICATIONS
REPOSITORY CONFIGURATION
REPOSITORY CONNECTIONS
AUTHENTICATION TYPES
CHANGE PASSWORD
CHANGE WORKING REPOSITORY
EVENT SUBSCRIPTIONS
GAM CONFIGURATIONS

ADMINISTRATOR

HIDE FILTERS

Users

Try a Search

+ ADD

User Name	First Name	Authentication	
custom	FirstName	custom	EDIT
admin	Administrator	local	EDIT
test	Test	local	EDIT

FIRST / PREV / NEXT

GENDER (All)

AUTHENTICATION TYPE (All)

ROLE (All)

Ahora si, para definir este tipo de autenticación, todo se realiza y configura nuevamente a través del backend de GAM.
Como en la Demo anterior, nos dirigimos a Authentication Types y damos de alta un nuevo tipo.

GENEXUS
ACCESS MANAGER

USERS ROLES SETTINGS -

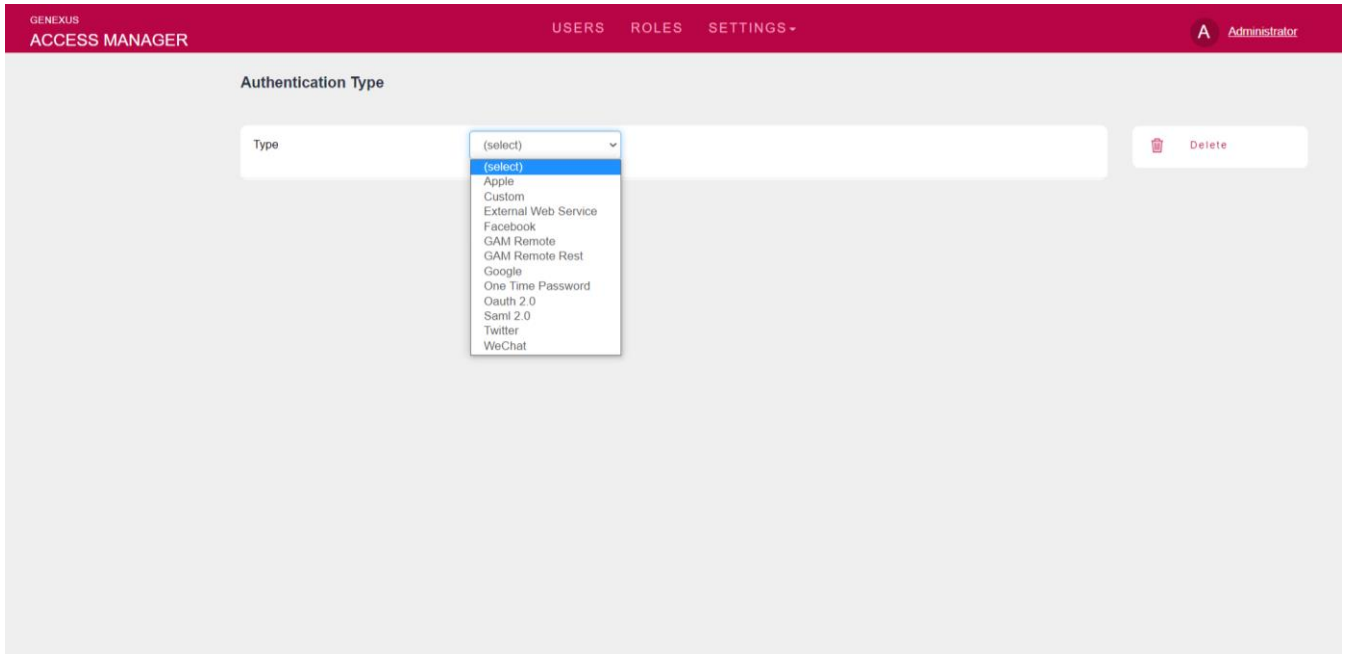
A Administrator

Authentication Type

Type

- (select)
- Apple
- Custom
- External Web Service
- Facebook
- GAM Remote
- GAM Remote Rest
- Google
- One Time Password
- Oauth 2.0
- Saml 2.0
- Twitter
- WeChat

Delete



En este caso, seleccionamos el tipo One Time Password.

Authentication Type

Type	One Time Password
Name*	otp
Function	Only Authentication
Enabled?	<input checked="" type="checkbox"/>
Description	<input type="text" value="One Time Password"/>
Small image name	<input type="text"/>
Big image name	<input type="text"/>
Impersonate	<input type="text" value="local"/>
Use For First Factor Authentication?	<input checked="" type="checkbox"/>
User validation event	<input type="text" value="(none)"/>
Code generation type	<input type="text" value="OTP"/>

 Delete

Pasemos a describir las propiedades mas importantes:

Impersonate: Aquí se especifica el tipo de autenticación donde los usuarios van a ser validados al usar OTP. Como mencioné anteriormente en el teórico de esto, los usuarios ya deben existir. Este es el único tipo de autenticación que se requiere configurar esta propiedad ya que los usuarios deben existir en la base de datos de GAM.

Usar como autenticación de primer factor: Si no configura esta propiedad, OTP solo podría usarse como un segundo factor. En nuestro caso, lo habilitamos.

Autogenerated OTP code length	<input type="text" value="6"/>
Generate code only with numbers?	<input checked="" type="checkbox"/>
Code expiration timeout (seconds)	<input type="text" value="1800"/>
Maximum daily number of codes	<input type="text" value="12"/>
Number of unsuccessful retries to lock the OTP	<input type="text" value="3"/>
Automatic OTP unlock time (minutes)	<input type="text" value="60"/>
Number of unsuccessful retries to block user based on number of OTP locks	<input type="text" value="3"/>
Send code using	<input type="text" value="Email by GAM"/>
Mail message subject	<input type="text" value="We have sent the code to access %1"/>
Mail message HTML text	<input type="text" value="To access the application %1 enter the following code: %2"/>
Validate code using	<input type="text" value="GAM"/>

El resto de las propiedades, se utilizan para definir propiedades del código a enviar y el formato del mail.

En nuestro caso usaremos el por defecto de GAM que es el email, pero recordar que existe la posibilidad del envío del código a través de un SMS. Si el desarrollador decide optar por esta segunda forma, debe implementar y configurar el evento de GAM que luego debe seleccionar en la propiedad "Send code using".

LOGIN

Log on to

One Time Password ▾

User name

 Keep me logged in

SEND ME A CODE

We have sent the code to access GAMCourse

Mail [redacted]
para mí ▾

To access the application GAMCourse enter the following code: 784693

LOGIN

user

Code

VALIDATE CODE

BACK TO LOGIN

Finalmente yendo al Login, seleccionamos el tipo OTP, donde vemos que nos pedirá solamente el nombre de usuario para el envío de código. Luego de llegado el código a través del mail, simplemente se ingresa en el login para autenticarse en el sistema.

DEMO: TOTP

En esta demo, los pasos para configurar un nuevo tipo de autenticación TOTP son prácticamente los mismos que OTP, excepto por una salvedad.

GENEXUS
ACCESS MANAGER

USERS ROLES

SECURITY POLICIES
APPLICATIONS
REPOSITORY CONFIGURATION
REPOSITORY CONNECTIONS
AUTHENTICATION TYPES
CHANGE PASSWORD
CHANGE WORKING REPOSITORY
EVENT SUBSCRIPTIONS
GAM CONFIGURATIONS

A Administrator

X HIDE FILTERS

Users

+ ADD

GENDER
(All)

AUTHENTICATION TYPE
(All)

ROLE
(All)

User Name	First Name		Authentication	
custom	FirstName		custom	EDIT
admin	Administrator	User	local	EDIT
test	Test	GAM	local	EDIT

FIRST / PREV / NEXT

Para definir este tipo de autenticación, nuevamente nos dirigimos a Authentication Types y damos de alta un nuevo tipo.

GENEXUS
ACCESS MANAGER

USERS ROLES SETTINGS -

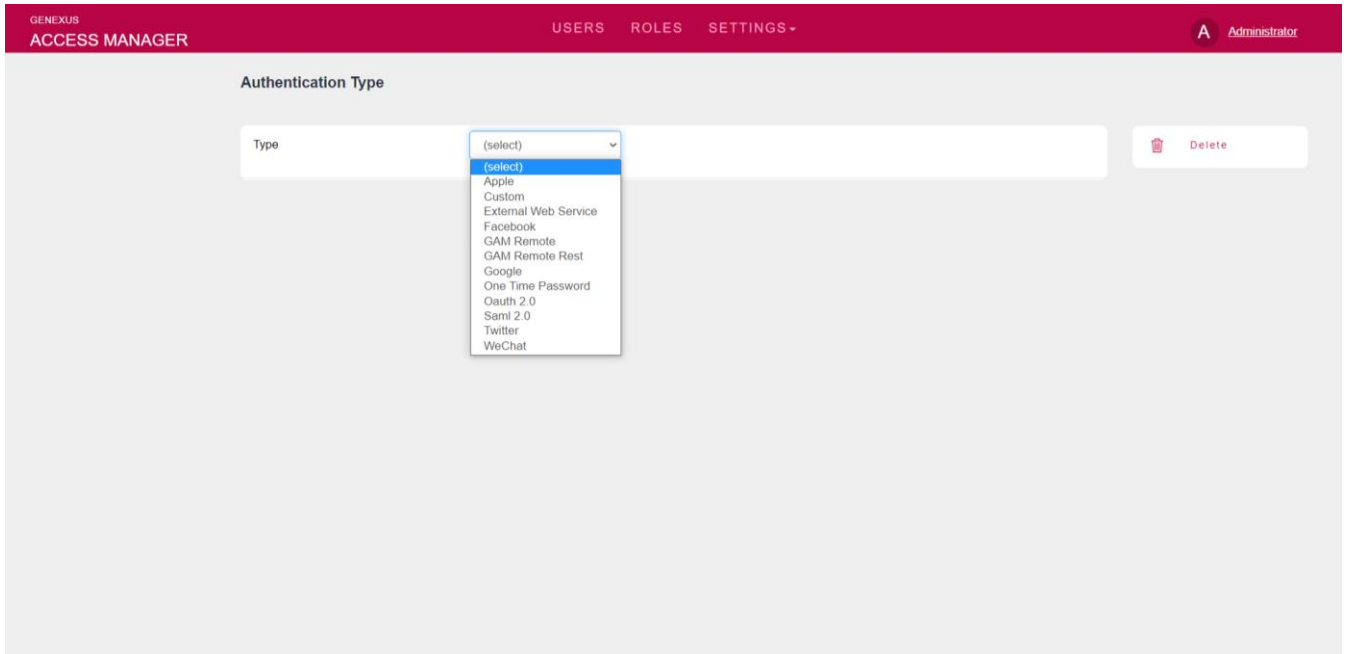
A Administrator

Authentication Type

Type

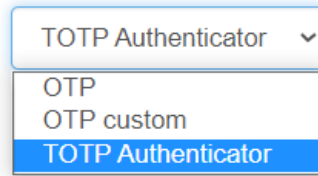
- (select)
- Apple
- Custom
- External Web Service
- Facebook
- GAM Remote
- GAM Remote Rest
- Google
- One Time Password
- OAuth 2.0
- Saml 2.0
- Twitter
- WeChat

Delete

The screenshot shows the 'Authentication Type' configuration page in the GeneXus Access Manager. At the top, there is a navigation bar with 'GENEXUS ACCESS MANAGER' on the left and 'USERS ROLES SETTINGS -' on the right. A user profile 'A Administrator' is visible in the top right corner. The main content area is titled 'Authentication Type'. It features a form with a 'Type' field, a 'Delete' button, and a dropdown menu. The dropdown menu is open, showing a list of authentication types: '(select)', 'Apple', 'Custom', 'External Web Service', 'Facebook', 'GAM Remote', 'GAM Remote Rest', 'Google', 'One Time Password', 'OAuth 2.0', 'Saml 2.0', 'Twitter', and 'WeChat'. The 'One Time Password' option is highlighted in blue, indicating it is the selected option.

En este caso, también seleccionamos el tipo One Time Password.

Code generation type



A dropdown menu with a light blue border and a white background. The menu is open, showing three options: 'TOTP Authenticator' (selected and highlighted in blue), 'OTP', and 'OTP custom'. A small downward arrow is visible on the right side of the dropdown box.

TOTP Authenticator
OTP
OTP custom
TOTP Authenticator

La diferencia con OTP es la propiedad mostrada en pantalla, donde en este caso elegimos TOTP Authenticator.

El resto de las propiedades son para configuraciones del código que no vienen al caso.

User

GUID	e7483297-a3e3-440e-a543-8b1801d09226
Name Space	GAMCourse
Authentication Type	GAM Local
User Name*	test
E-Mail*	user@mail.com


[Edit](#)[Permissions](#)[Roles](#)[Change Password](#)[Unblock OTP Codes](#)[Enable authenticator](#)[Delete User](#)

Veamos la salvedad mas importante que tiene con OTP.

Cada usuario debe activar la autenticación a través de sus configuraciones. La diferencia más importante es que este algoritmo del código está basado en el tiempo y los códigos los generan las diferentes aplicaciones autenticadoras.

A efectos de la demo, se creó utilizando el usuario administrador del backend de GAM para un usuario "test" de una aplicación de ejemplo. Los pasos a realizar para esta manera, se basan en dirigirse al usuario en cuestión y presionar en Enable authenticator.

Enable TOTP authenticator

User Name	test
Email	user@mail.com
	
Secret Key	EQ75LTFDWE62CVQK
Type a code	<input type="text"/>

[BACK](#)[ENABLE](#)

Una vez allí, se proporcionará el código QR para configurarse en un software o aplicación móvil basada en autenticación con contraseña de un solo uso, el cual luego de haberlo leído, nos retornará el código a ingresar en el campo "Type a code".

LOGIN

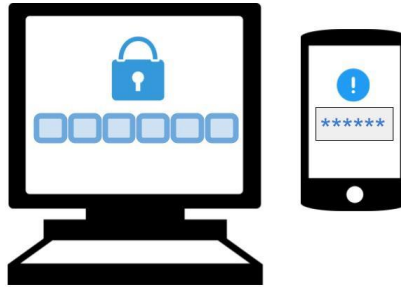
Log on to

Time-Based One Time Passw 

User name

Keep me logged in

NEXT



LOGIN

user

Code

VALIDATE CODE

BACK TO LOGIN

Finalmente el login es igual que todos los tipos anteriores, donde en este caso existe una aplicación intermediaria la cual nos proporciona el código de acceso.

GeneXus™

training.genexus.com

wiki.genexus.com

training.genexus.com/certifications