

# Authentication Types

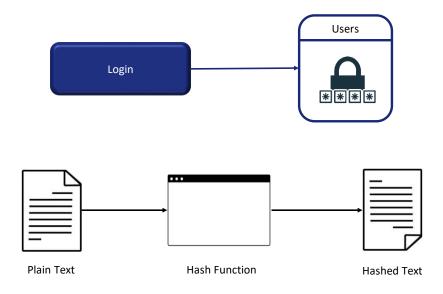
Nicolas Adrién | GeneXus Training

Authentication Types

Internal

External

As we said in previous videos, GAM offers different types of authentication, both internal (against the GAM database) and external (such as web services, social networks, or Google, also called Remote). Let's go into detail on them.

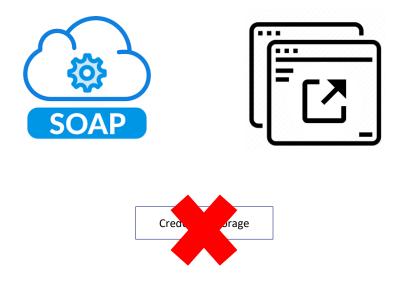


As for Internal Authentication Types, we have **Local Authentication** where user credentials are stored in the "Users" table of GAM.

GAM does not store the users' password but rather stores a hash of it. A hash is an algorithm that, given a plain text string, always results in the same string. Given this string, the original string cannot be obtained.

The hash is obtained from a unique key for each user and an algorithm named SHA-512, which will not be discussed in detail here.

This means that when retrieving GAM Users from the repository, the password property always has an empty value.



When integrating one application with another to exchange information, the first essential aspect is to solve the authentication issue.

In the External authentication type, a first solution is to have the application we need to integrate expose a SOAP web service that performs the authentication.

Another scenario involves an external program that meets the authentication needs and is not necessarily a SOAP service.

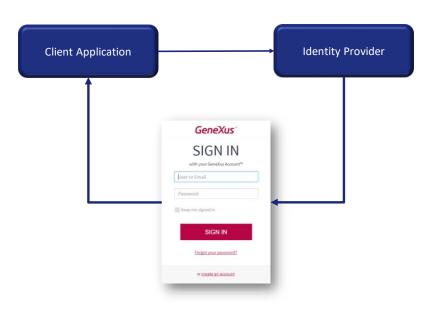
The solution for this scenario is to configure the GAM custom authentication type in the GAM repository.

In both cases, GAM must be configured to accept the external program as the Identity Provider.

When using either of these authentication types, the GAM Client is not the owner of the user's credentials; only the username and other information that depends on the output of the external program will be stored in the Repository.

When authenticating to other external services, such as LDAP, an external program or web service can be used to act as a bridge between the GAM application and LDAP.





Among the other authentication types, first we find Oauth 2.0. GAM allows authenticating with any OAuth 2.0 provider just by following a few

simple steps. When this type of authentication is selected, an application login is redirected to the configured identity provider.

The login is displayed by the provider; there, users enter their credentials and are redirected back to the application.

This Authentication type is defined in the same way as any of the other GAM types we have already mentioned, only that it requires a detailed configuration of the protocol used by the Provider. Therefore, to configure the OAuth authentication type in GAM, you must follow the documentation of the Identity Provider you wish to connect to.

This protocol also provides SSO between different client applications.



General Authorization	Token	User Information	
Client Id	Tag	client_id	Value
Client Secret	Tag	client_secret	Value
Redirect URL	Tag	redirect_uri	Value
Custom Redirect URL?			
Redirect to authenticate?			

/oauth/gam/callback

Oauth 2.0 has a second authentication flow. With the option "Redirect to authenticate?" set to False, it provides OAuth 2.0 authentication using REST without redirecting to the identity provider, since GAM skips the redirection configured in the Authorization tab.

The other option (Custom Redirect URL?) indicates GAM that the specified return URL is customized. If set to False, it will then concatenate "/oauth/gam/callback".

Otherwise, if it is set to True, the developer must implement this URL and read the IDP responses.

Both properties can be configured from the OAuth authentication type in the GAM back end.

# Authentication types | External

Enable OpenID Connect Protocol? OpenID Connect	SINNO?
Validate ID Token?	
Issuer URL	
Path to server certificate filename	
Allow only users with verified email?	

Token

User Information

Authorization

General

Then we have OpenID Connect.

This is an authentication protocol that works with OAuth 2.0 by implementing authentication as an extension of the OAuth authorization process and is becoming one of the most common today.

The advantage it gives compared to OAuth is that this protocol allows us to obtain the user's information, unlike the OAuth standard that doesn't enable us to do so. For this reason, it is no longer necessary to configure the User Information section in the Authentication Type.

For the protocol to work, you must activate the Validate ID Token property and include the provider and the local public certificate on a server. With this information, a JSON Web Token signed and returned by the provider, called ID Token, is obtained.

### Authentication types | External

PRODUCTS (+) Privacy Policy URL Impersonate I	
▶ Dashboard       App ID       373225729315598       Function       Only Authentication         Basic       Display Name       Enabled?       Image name	
Settings       App ID       Function       Only Authentication         Basic       373225729815598       Enabled?       Image: Comparis and the second of the	
Basic     373225729815598     Enabled?       Advanced     Display Name     Description       Pacebook1     TestJavaGXCloud     Small Image name       App Review     App Domains     Big image name       Privacy Policy URL     Impersonate     1       Privacy policy for Login dialog     Client M.     125707037	
3 Roles       Display Name       Description       Pacebook1         ▲ Alerts       TestJavaGXCloud       Small mage name       Description       Pacebook1         > App Review       App Domains       Big image name       Description       Description       Description         > RoDUCTS +       Privacy Policy URL       Impersonate       I       Description       Description         > Facebook Login       Privacy Policy URL       Impersonate       I       Description       Description         App Icon (1024 x 1024)       Client M.       125707037       Description       Description       Description	
G Roles       TestJavaGXCloud         Alerts       App Domains         App Review       Big image name         PRODUCTS +       Privacy Policy URL         Privacy Policy URL       Impersonate         Impersonate       1         Orient Ud.       125707037-         App Icon (1024 x 1024)       Client Macret	
App Domains App Domains Big image name I Privacy Policy URL Privacy Policy URL Privacy policy for Login dialog App Icon (1024 x 1024) Client ld. Eigensonate I Client ld. Eigensonate Eige	
PRODUCTS (+)     Privacy Policy URL     Impersonate     I       Privacy policy for Login dialog     App Icon (1024 x 1024)     Client Mc.     125797037	
Privacy Policy URL impersonate I Privacy policy for Login dialog App Icon (1024 x 1024) Cient Secret 602/cr857a031984e2	
Privacy policy for Login dialog App Icon (1024 x 1024) Client M. 125797037- Client Secret 6926/1837a931984e2.	
App Icon (1024 x 1024) Client Secret 6921:1837a931984e2	
5.15	1550
Local site URL http://apps6.genexus.com	
	m
Additional Scope	
	CANCEL CONFIRM

In second place, we have Facebook.

For this type, two steps must be followed: First, you need to create a "Facebook client application" on your site and obtain an ID and key (called "Secret") for your application. Second, define the "Facebook Authentication Type" in the GAM back end or API.

By following these steps thoroughly you will have the authentication type configured correctly.

This type can be used in web applications and native mobile applications, and in the background it is handled by Oauth 2.0.

#### Authentication types | External

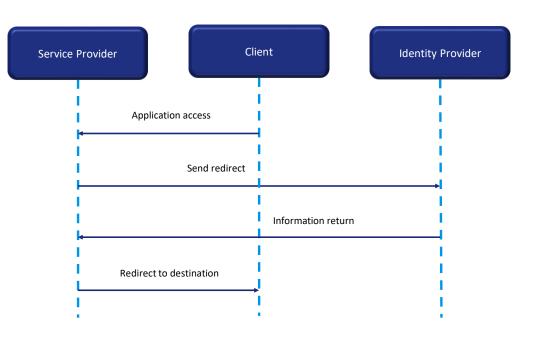
Details     Settings     Keys and Access To     Type     Twitter       Application Settings     Name     twitterb       Application Settings     Function     Only Authentication       Keep the "Consumer Secret" a secret. This ke     Function     Only Authentication       Consumer Key (API Key)     DBZOIErjkqROXK     Description     twitterb       Consumer Secret (API Secret)     86mBGvc     Small image name       Access Level     Read-only (modif)     Impersonate       Dwner ID     408684964     Consumer Key     SelOTIBBeySdryJTZBBgS2110       Consumer Secret     408684964     Consumer Secret     eKyyVRRw6ituMF0RmJ1bUFFMBsJVqTf5J5pge.jsmaller		Authentication Type	
Application Setting:       Function       Only Authentication         Keep the "Consumer Secret" a secret. This ke       Enabled?       Enabled?         Consumer Key (API Key)       DBZOIErjkqROXK       Description       twitterb         Consumer Secret (API Secret)       86mBGvc       mail image name       small image name         Access Level       Read-only (modified)       Impersonate       impersonate         Owner ID       408684964       Consumer Key       SelOT3BeySdryJTZBBgs2/Mice         Consumer Secret (API Secret)       Gosumer Key       SelOT3BeySdryJTZBBgs2/Mice	Details Settings Keys and Access To	Туре	Twitter
Keep the "Consumer Secret" a secret. This ke     Enabled?       Consumer Key (API Key)     DBZOIErjkqROXk       Description     twitterb       Consumer Secret (API Secret)     86mBGvc       Access Level     Read-only (modif;       Dwner     TestGX       Dwner ID     408684964       Consumer Secret (API Secret)     SeloTi3BeySdryJTZBBgs3/Me       Consumer Key     Consumer Key		Name	twitterb
Consumer Key (API Key)     DBZOIErjkqROXk       Consumer Secret (API Secret)     86mBGvc       Access Level     Read-only (modified)       Dwner     TestGX       Dwner ID     408684964       Consumer Secret (API Secret)     Small image name       Consumer Key (API Key)     TestGX       Dwner ID     408684964       Consumer Secret (API Secret)     Consumer Key       Secret (API Secret)     Selotic Secret (API Secret)	Application Settings	Function	Only Authentication
Consumer Secret (API Secret)     86mBGvc     small image name       Access Level     Read-only (modified in the secret)     Big image name       Dwner ID     408684964     Impersonate       Consumer Secret     exxyvvRws11uMF0RmJ1bUFFMBsUVqTf6J5pgo.lg/mm2/bevCovG	Keep the "Consumer Secret" a secret. This ke	Enabled?	
Access Level     Read-only (modified in the state in the	Consumer Key (API Key) DBZOIErjkqROXk	Description	twitterb
Dwner     TestGX     Impersonate       Dwner ID     408684964     Consumer Key     SelOTi3BeySdryJTZBBgS20100       Consumer Secret     eKXyyVRRw511uMF0Rmj1bUFFMBsUYqTf5J5pgo.jg/millet/Cw/G	Consumer Secret (API Secret) 86mBGvc	Small image name	
Dwner ID     408684964     Consumer Key     SelOTi3BeySdryJTZBBgS3/Mod       Consumer Secret     eKXyyVRRw5l1uMF0RmJ1bUFFMBsUYqTf6J5pgo.jg//file/bt/VcwG	Access Level Read-only (modify	Big image name	
Consumer Key SelOT3BeySdryJTZBBos2444 Consumer Secret eKXyyVRRw5I1uMF0RmJ1bUFFMBsUYqTT6J5pgo.1447474VCwG	Dwner TestGX	Impersonate	
	Dwner ID 408684964	Consumer Key	SelOTi3BeySdryJTZBBgs211.
Callback URL http://apps5.genexus.com		Consumer Secret	eKXyyVRRw5I1uMF0RmJ1bUFFMBsUYqfT6J5pgq.jg////////////wCwG
		Callback URL	http://apps5.genexus.com

Then we have Twitter.

This case is executed in the same way as Facebook. Step 1 is to create a Twitter application on your site, and obtain the consumer's key and secret for that application. Step 2 is to define the Twitter authentication type using the GAM backend.

Once again, just like with Facebook, this type of authentication can be used in web apps and also in native mobile apps.

In the GeneXus Wiki, you can find detailed information about this and all the existing authentication types for GAM.



GAM provides authentication using any SAML 2.0 provider.

SAML is a secure XML-based communication mechanism for exchanging identities between organizations.

One of the use cases addressed by SAML is also SSO, so it avoids the need to maintain several credentials in multiple locations and increases security while reducing time-consuming administration tasks.

SAML involves two entities besides the client: a service provider and an identity provider.

A login flow is roughly as follows:

First, the user tries to access an application hosted by a service provider. This provider generates an authentication request and redirects it to the user's browser.

Next, the identity provider receives the request, authenticates the user by requesting valid login credentials or checking for correct session cookies, and generates the response to be returned to the user's browser. Lastly, the user is redirected to the target URL.



Do not use self-signed certificate

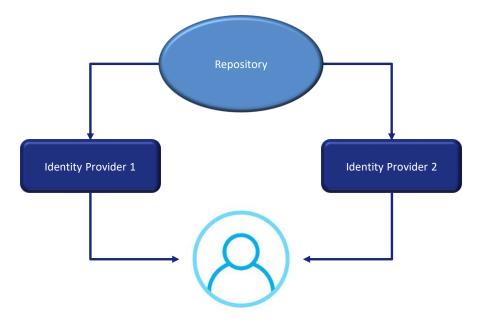
Use HTTPS protocol and include server and virtual directory

General Credentials User Information	
Local Site URL*	https://server/virtualDirectory
Service Provider Entity ID*	
Identity Provider Entity ID	
Execute SAML requests using GET	

The following should be mentioned about SAML:

- Regarding certificates, it is recommended not to use self-signed certificates.
- The Local Site URL property must have HTTPS protocol and include a server and virtual directory as shown on the screen.

#### **GAM** Impersonation



When the GAM Repository allows end users to authenticate with different identity providers, by default they are assigned to different GAM Users. For security reasons, users can authenticate using different mechanisms depending on the login source being used.

However, the login information must be assigned to the same logical GAM user.

Impersonation allows the repository to have two different authentication mechanisms that converge on the same user.

This is useful, for example, when it is not possible to use the same type of authentication from the intranet and from the internet, but the user should be the same.

It is also used for migrating from one type of authentication to another, where the "impersonated" authentication type is the one being migrated.

Depending on the type of authentication, there are different criteria for mapping users, which are described in the GeneXus Wiki.

To end this topic, we will move on to a series of demos in order to show the cases in practice in more detail.



training.genexus.com wiki.genexus.com training.genexus.com/certifications