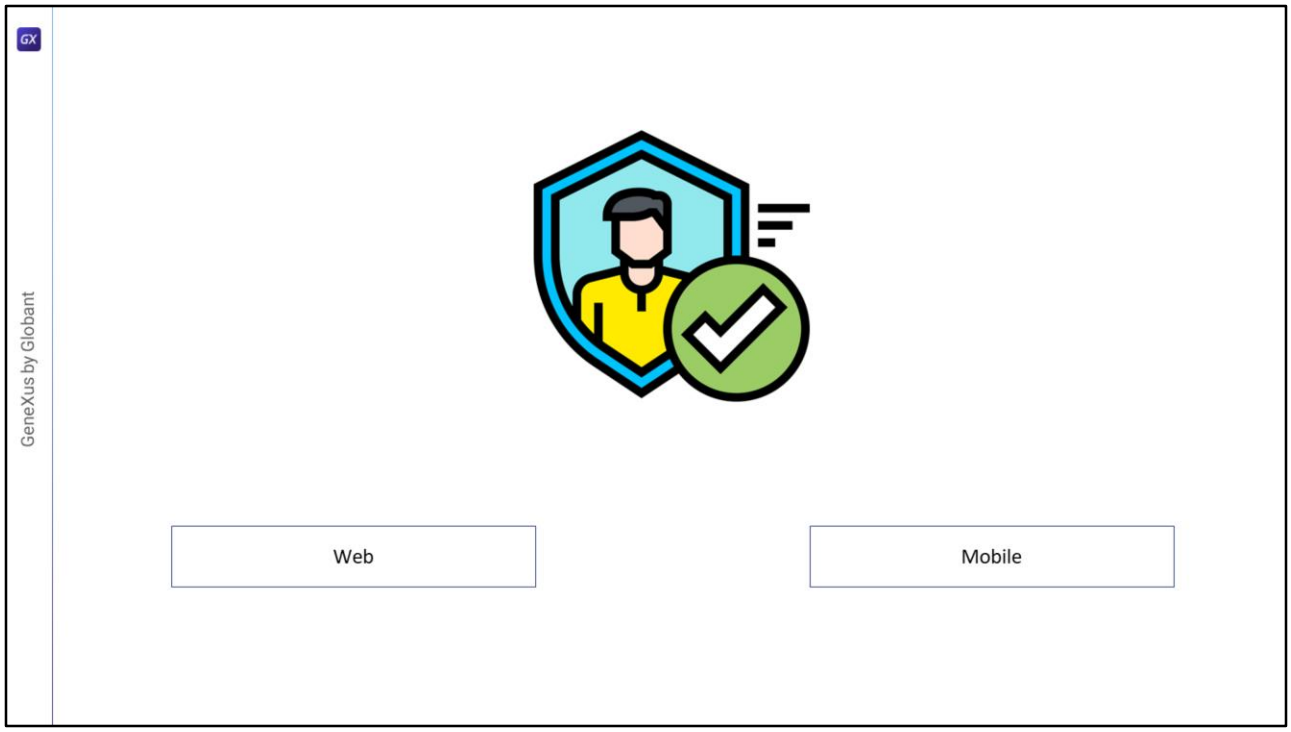

GeneXus by Globant
GeneXus™
by Globant
training.genexus.com

GeneXus by Globant

Authentication

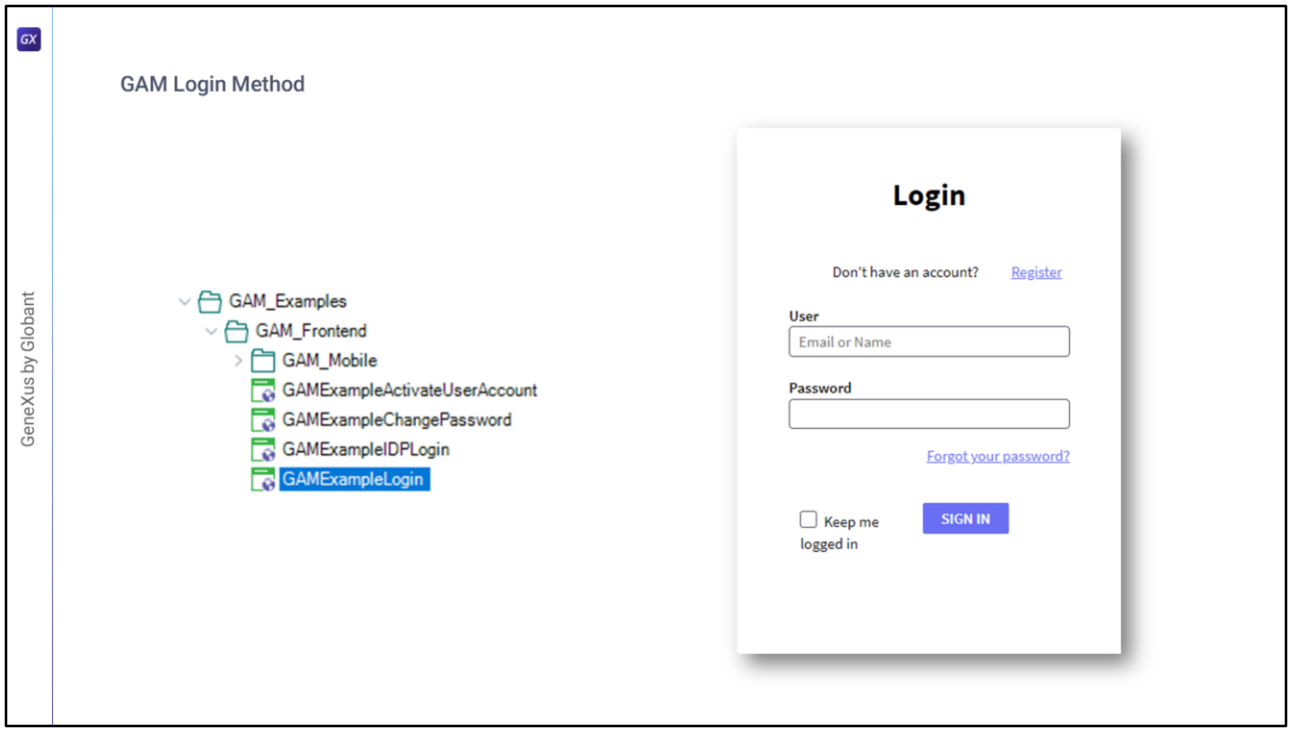
Nicolas Adrién



As we've said in previous videos, authentication is the act or process of confirming that something (or someone) is who they claim to be.

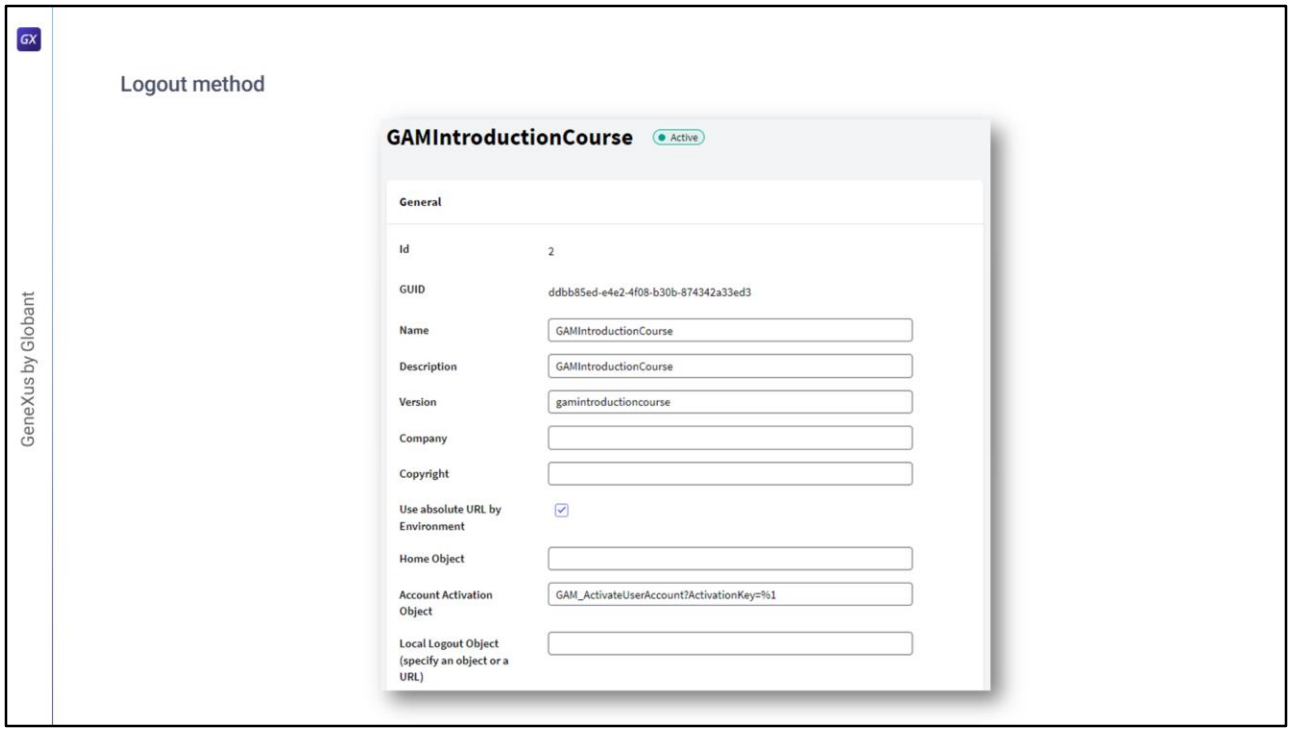
All GAM Authentication scenarios include the possibility of entering a username and password, and validating this data against an existing DB.

The modes to which GAM can be applied depend on the environment for which the applications are developed or deployed. The available options are Web and Mobile.



Let's take a look at the GAM Login method.
As we said before, GAM provides example objects that we can use as a guide.

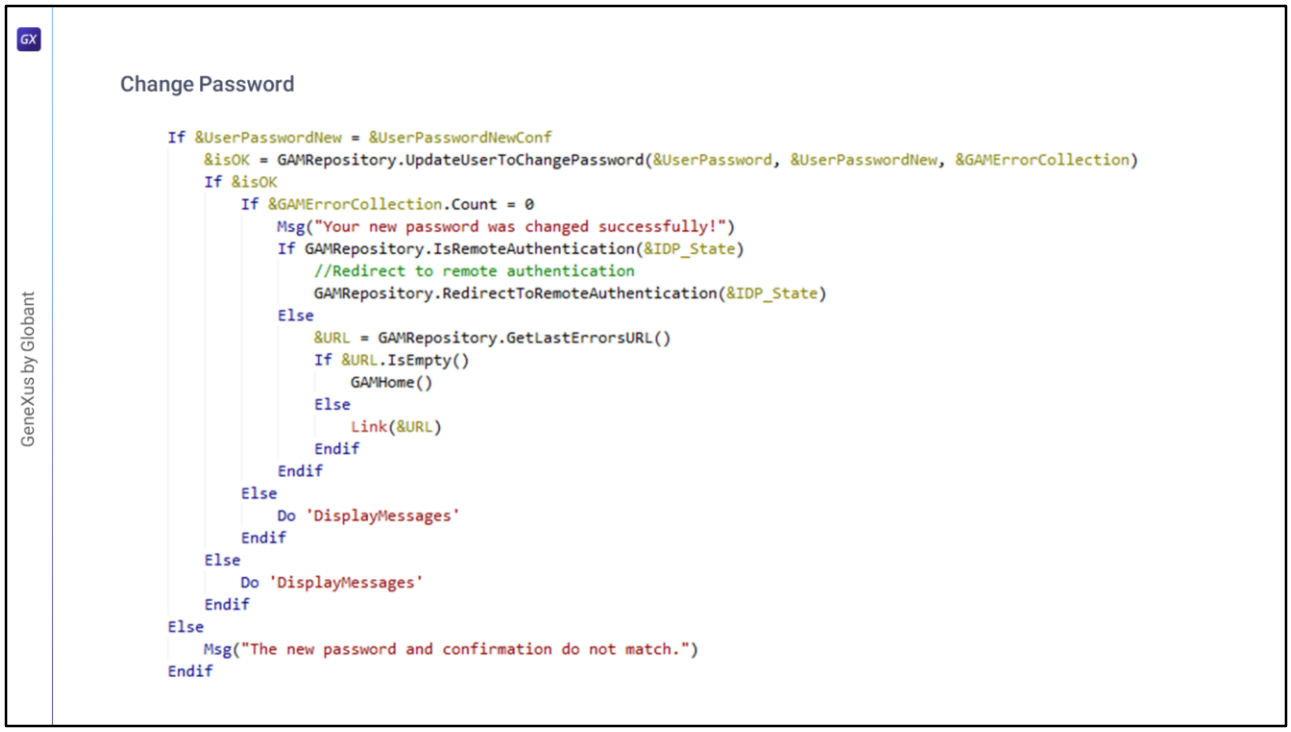
In particular, for the GAM method there is the **GAMExampleLogin** object, which authenticates through GAM.
It uses the login method of **GAMRepository**, which is an external object that is part of the GAM library and will not be discussed in detail for the moment.



As for logout, a possible implementation can be as follows.

The first instruction loads the user's data into the GAM database repository, and in case of errors these are received in the *&Errors* SDT.
The second instruction transfers the flow to the **GAMExampleLogin** Web Panel. That's all.

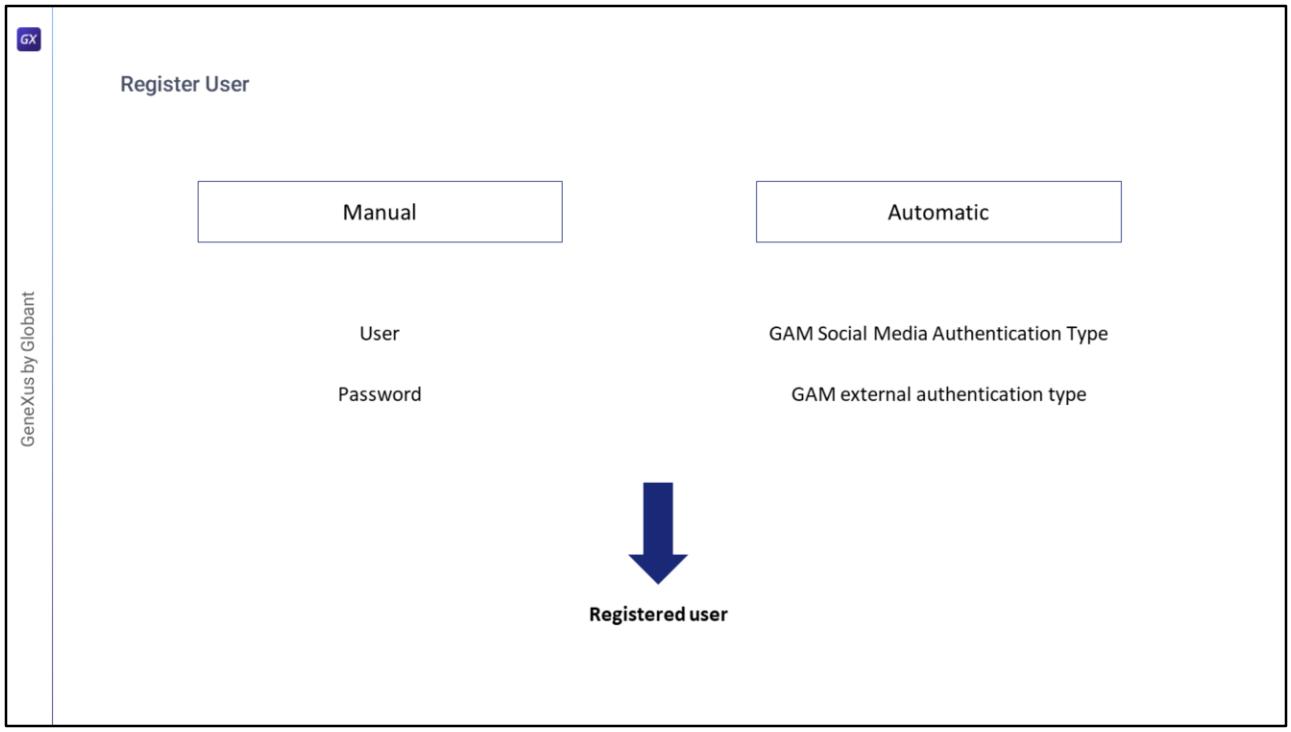
The application logout is configured using the GAM web back office in the application settings (or programmatically, using the GAM API).
Its purpose is to determine the URL of the object to be redirected after Logout in SSO applications.



As we already mentioned the existence of GAM examples, this one also provides one for password changes.

Its mechanism is quite simple and consists of using the methods provided by GAM, such as **UpdateUserToChangePassword**.

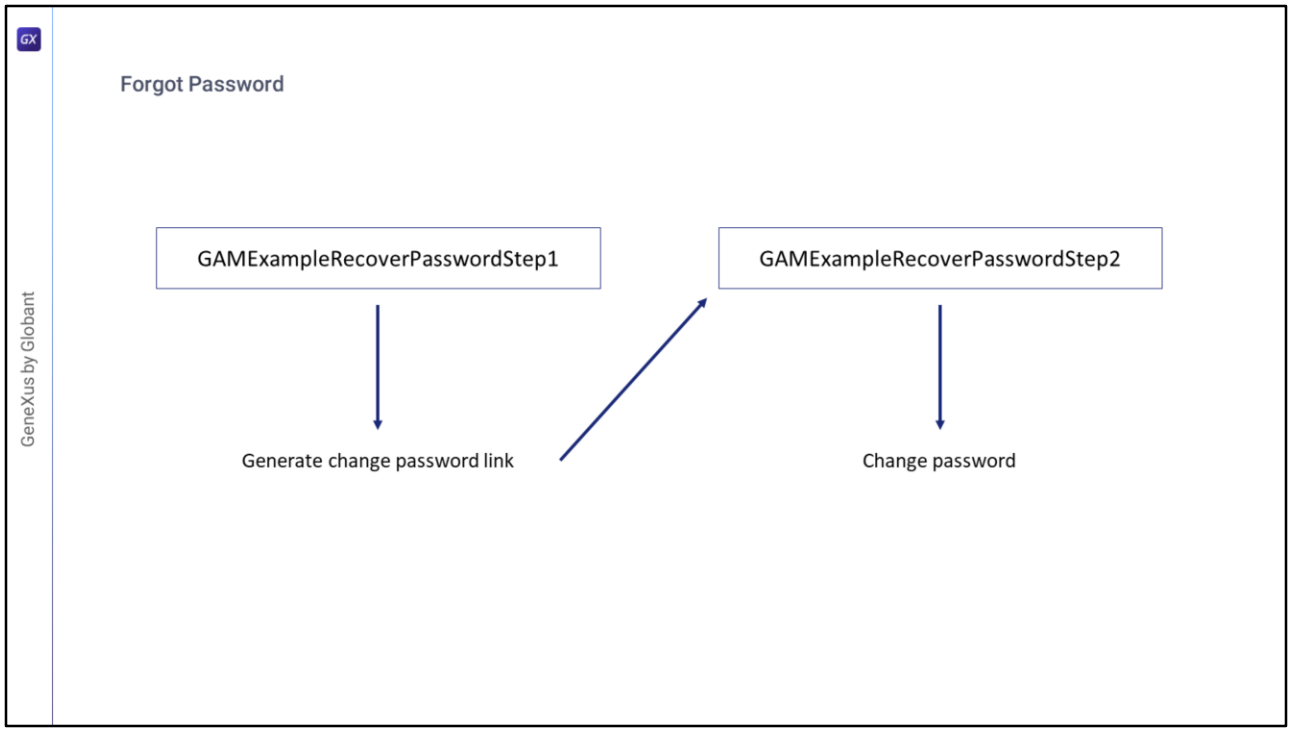
We will not describe this method's implementation in detail, but note that after calling it, it is only checked whether there were errors in the process. If there were not, it is also checked if the authentication was remote in order to finish the Login process in
the external applications, if applicable; otherwise, it is redirected to the GAM home.



As for user registration, there are two ways to do so:

The first one is manual, where a GeneXus user registers in GAM and creates a username and password to have access to the application. This occurs in the case of *GAM Local Authentication Type*.

The second way is automatic and is *GAM Authentication Type through social networks or external authentication*, where user registration is made when the user enters the application for the first time.

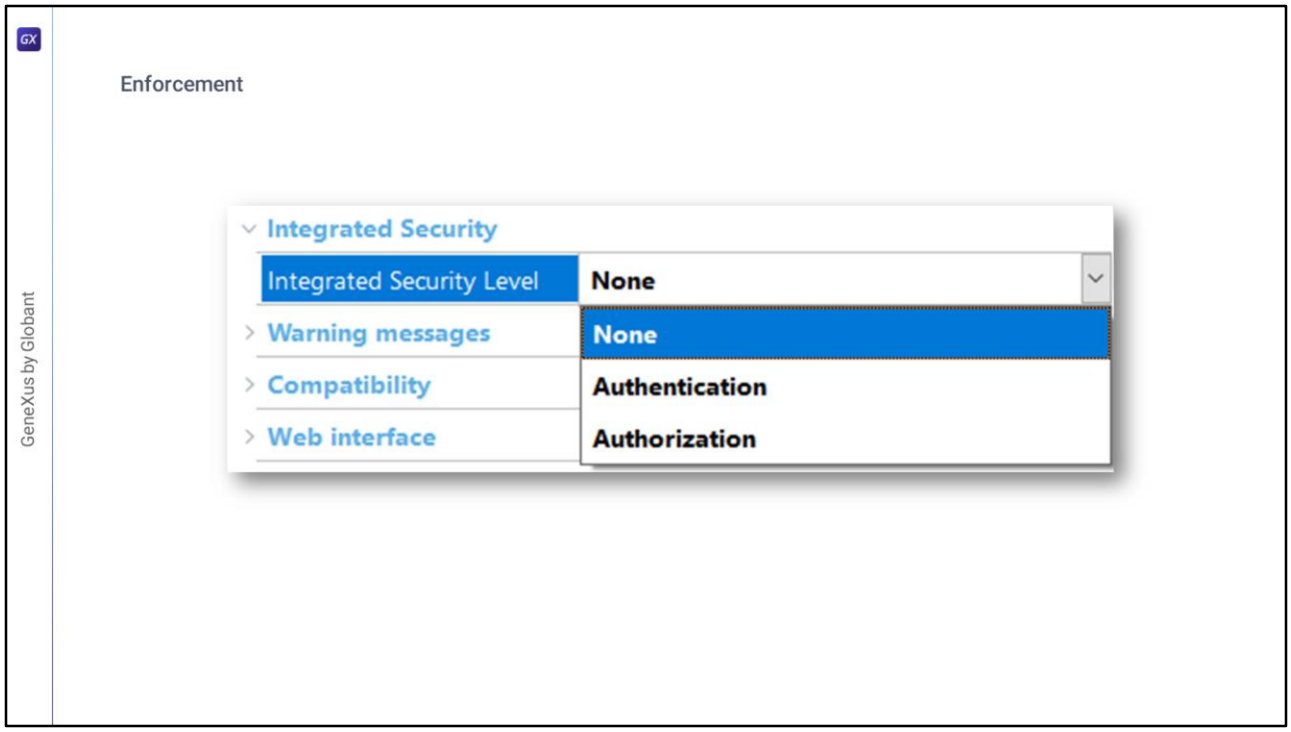


Two objects are distributed with the GAM examples library: **GAMExampleRecoverPasswordStep1** and **GAMExampleRecoverPasswordStep2**, which provide a solution in case of a forgotten password. The idea is that the GeneXus user completes these examples as needed.

With the first one, application users can enter their username or email in order to change their password through it.

In order to protect user confidentiality, a link to the second web object is sent to the user's email address, through which he/she can finally change the password.

More details on this implementation can be found in the GeneXus Wiki.



In all web panels that require authentication, GAM verifies it automatically without the need to ask the user to log in every time.

The easiest way to configure and define this is by setting the **Integrated Security Level** property to Authentication in the properties of the objects that should only be accessed by authenticated users.

By activating this property with this value, security will be enforced.

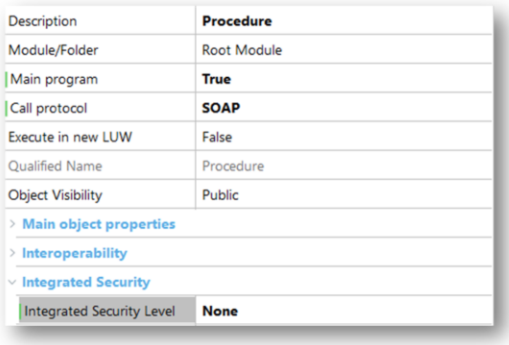
In the case of web objects, verification is also performed on each AJAX call that is executed. This is the default value at the version level.

If the user is not authenticated, a Login Object for the Web property or a Login Object for the SD property (depending on the application) will be displayed to allow the user to authenticate and access the application.

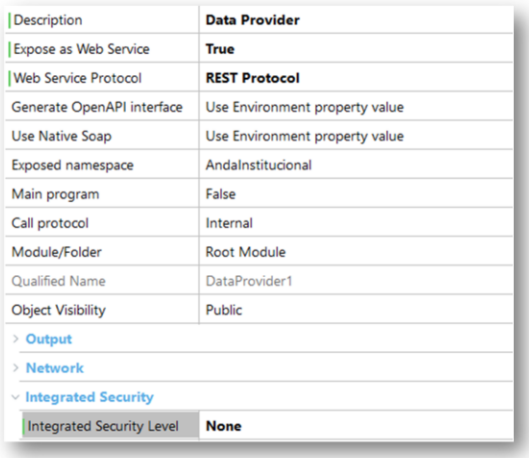
If the **Authorization** option is selected, the system will not only check whether the user is authenticated, but also if the user has permissions to access the object.

GeneXus by Globant

Enforcement



Description	**Procedure**
Module/Folder	Root Module
Main program	**True**
Call protocol	**SOAP**
Execute in new LUW	False
Qualified Name	Procedure
Object Visibility	Public
Main object properties	
Interoperability	
Integrated Security	
Integrated Security Level	**None**



Description	**Data Provider**
Expose as Web Service	**True**
Web Service Protocol	**REST Protocol**
Generate OpenAPI interface	Use Environment property value
Use Native Soap	Use Environment property value
Exposed namespace	AndaInstitucional
Main program	False
Call protocol	Internal
Module/Folder	Root Module
Qualified Name	DataProvider1
Object Visibility	Public
Output	
Network	
Integrated Security	
Integrated Security Level	**None**

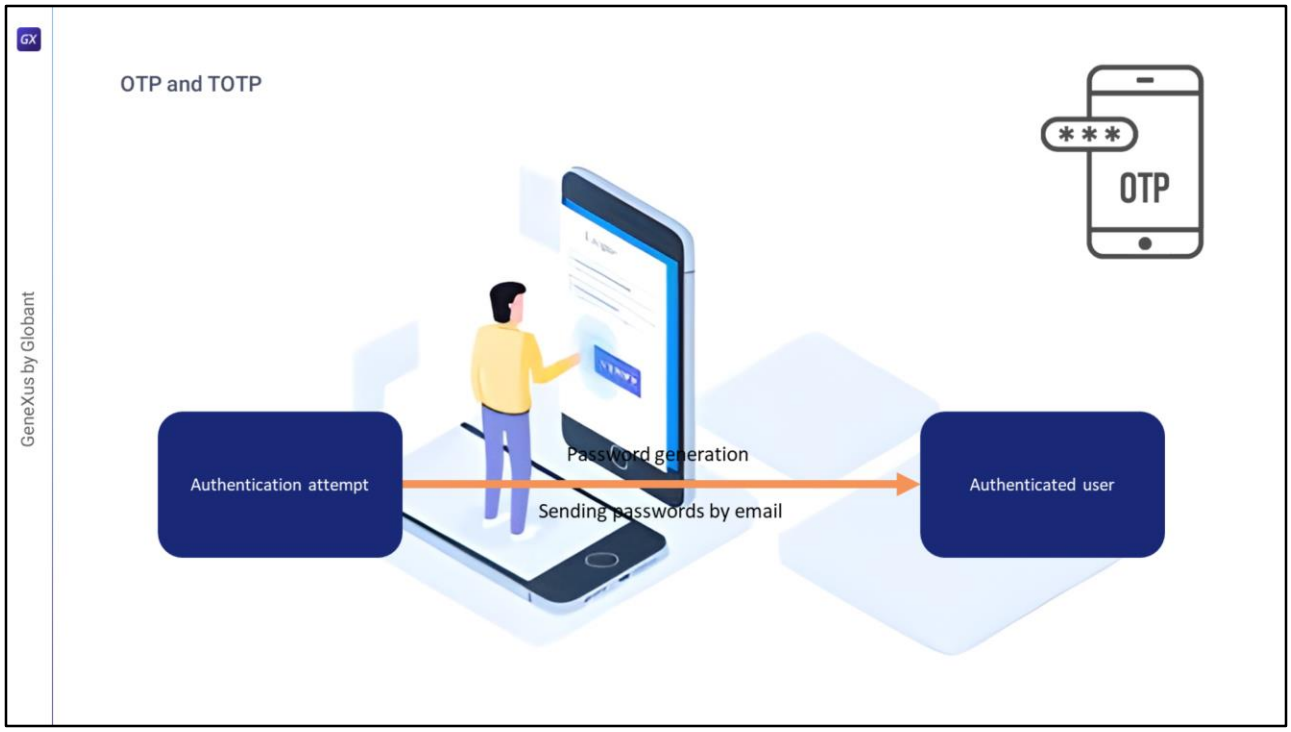
If the Integrated Security Level property is defined at version level, the procedures or Data Providers of Web Service type will also take that value, and this may not be what you are looking for.

To set a specific value or simply disable the property, it is done like with any other object from the Properties menu on the object.

Another way to do this—manually—can be found in the GAM example objects, where it is done with the code displayed on the screen.

In the different External Objects included in GAM, we find **GAMSession**, which has a method for session checking.

Besides checking the Boolean result of this method, we can use a *variable of GAMSession type* in order to use the different methods provided by the External Object.
In this particular case, **IsAnonymous** is used to check if the session user is authenticated or not.



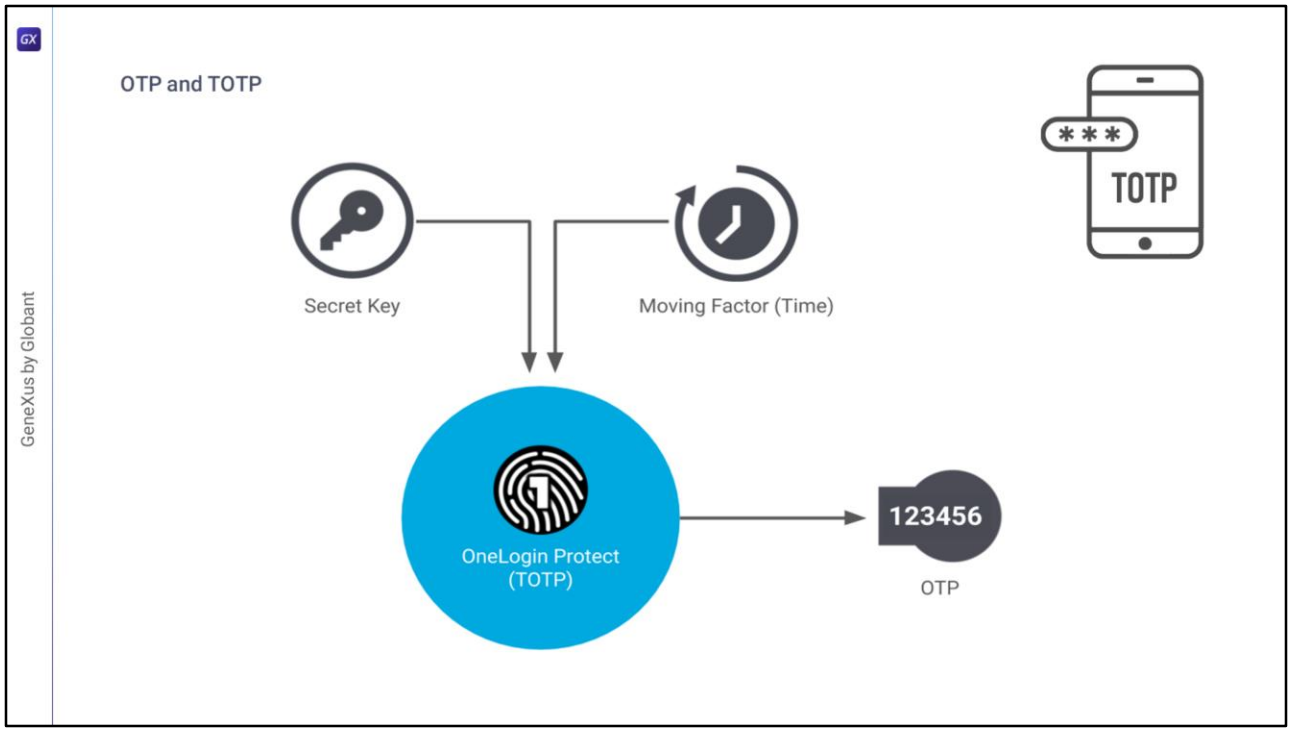
Providing secure access to cloud-based applications and software is a constant challenge for companies in all industries.
One of the ways in which password theft and other types of cyber-attacks have been addressed is through the use of one-time passwords (OTP).

OTP is a type of multi-factor authentication where for each authentication it is possible to generate a temporary password and send it by email or SMS to the user specified in the login form.

How does it work?
When a user authenticates to a web application, a password is generated and an email/SMS is sent with the generated password. By default, GAM sends passwords by email; however, you can customize how codes are sent to users (for example, by sending passwords via SMS).
The same steps apply for mobile applications.

The auto-generated password must not be expired and can be used for a **single** login.
However, OTP must have an expiration date for security reasons, and it is up to the application administrator to set this time.

To allow a user to use OTP with GAM, the user must exist in GAM and be previously registered. In addition, the authenticity of the method used to receive the OTP must have been validated, either by email or SMS to verify that it actually belongs to the registered user.

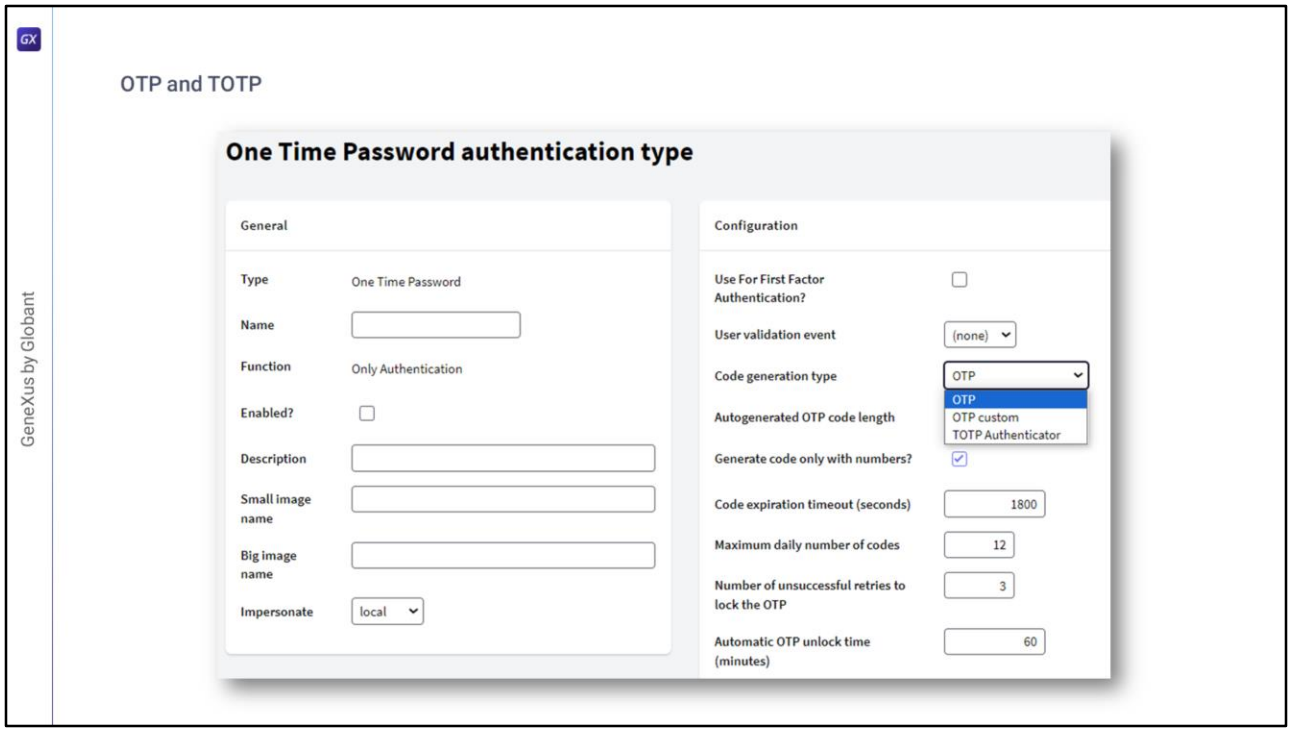


The other way in which password theft and other types of cyber-attacks have been addressed is through the use of time-based one-time passwords (TOTP).

A time-based one-time password is an algorithm that generates a one-time password (OTP) that uses the current time as a source of uniqueness. Therefore, in GAM TOTP is found as a type of OTP Code generation.

This authentication method has the advantage that users do not need to remember a password, since a new code is generated every time they wish to log in. It also adds another level of security because the code is valid for a short period of time.

If someone tries to authenticate with a username that does not belong to them, this method adds another level of difficulty because users need an application on their cell phone to obtain these codes.



These methods can be activated and configured in GAM through the Authentication Type menu option in the GAM Back office.


GeneXus by Globant
GeneXus™
by Globant
training.genexus.com