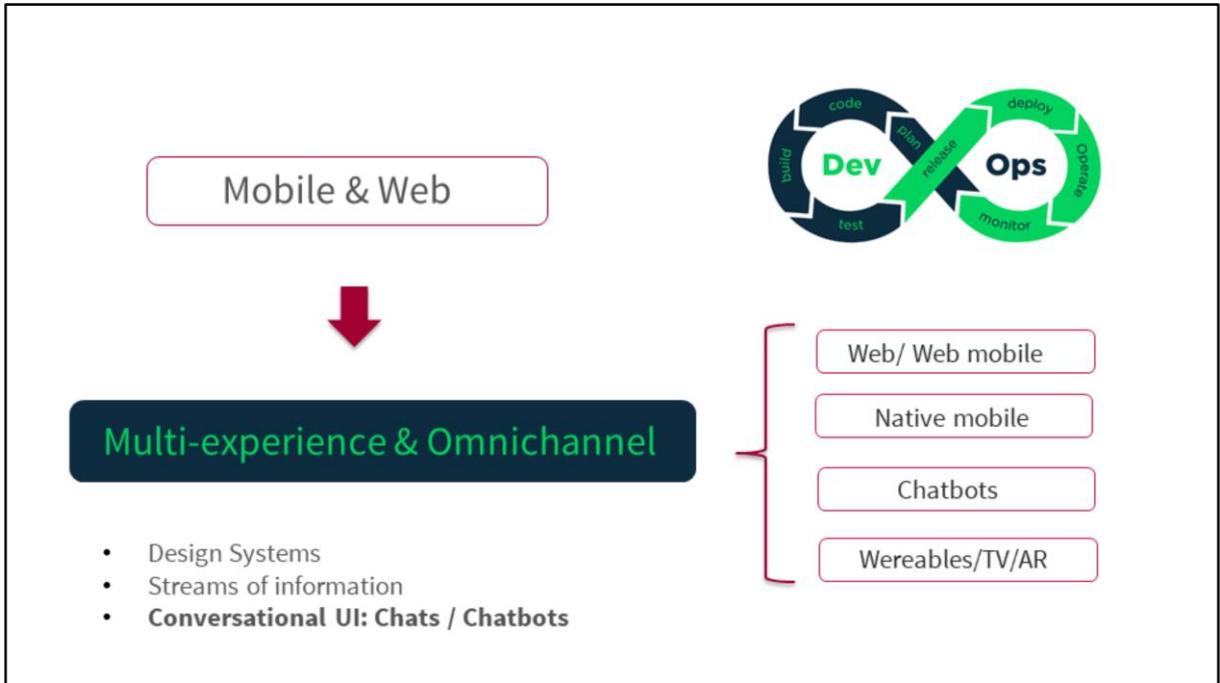


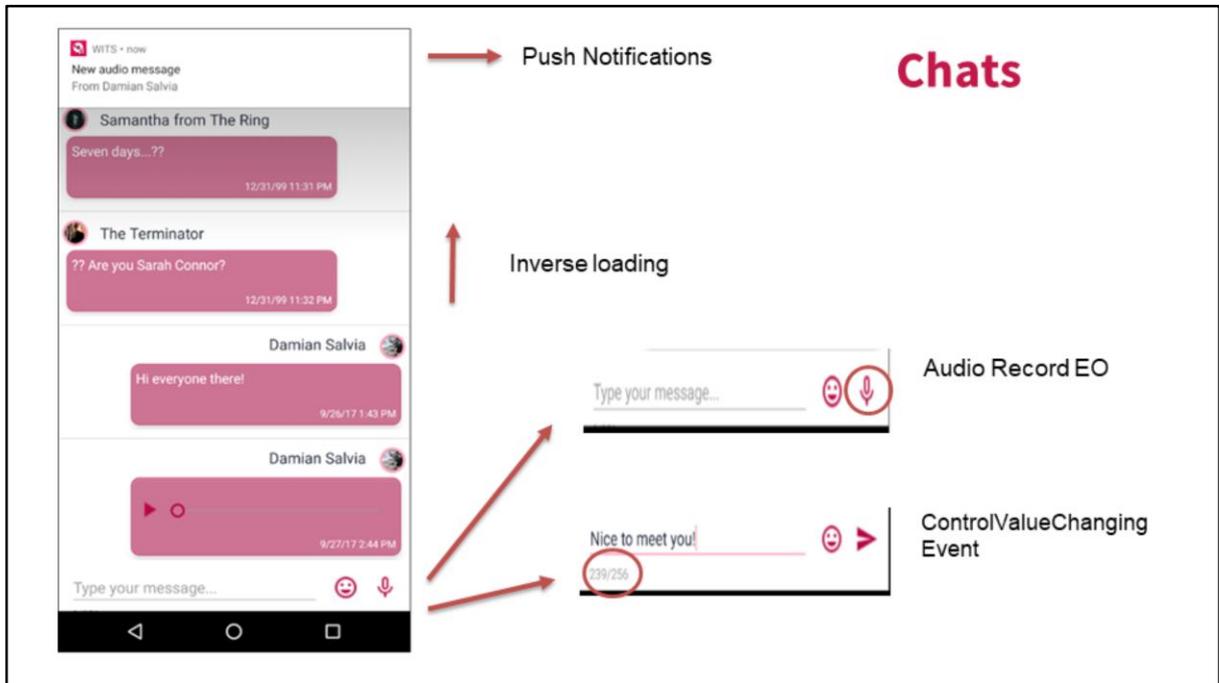
GeneXus[™]
The power of doing

Conversational UI:
Chats / Chatbots

GeneXus™ 16



Now we'll see the next point: the features to implement Simple Chats and the facilities to implement Chatbots.

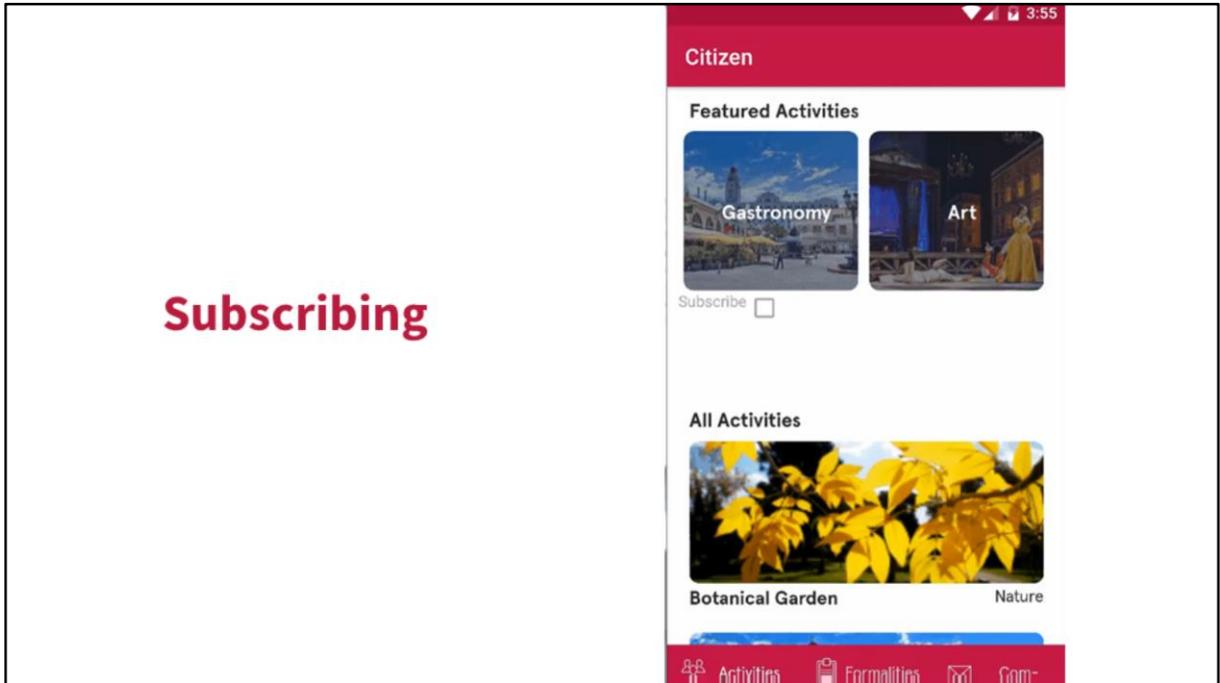


To implement a simple chat we have:

- Inverse Loading, which is a grid property that allows you to view the latest information from the bottom up.
- Audio Record, an external object that allows you to record an audio
 - Control Value Changing is an event that lets you know if the user has changed the value of an editable field (numeric or character), in this example it can be used to count the characters of the message to be sent.
- Push Notifications, that allow you to inform the user about some event by means of an alert, in this context, we inform that there is a new activity of interest.

Push Notification Demo

Now, we'll focus on Push Notifications and the implementation of the Socket API; later on we'll talk about the new features of version 16.
Let's see an example in execution.



In the Citizen application, users can subscribe to receive notifications of new activities.

Inserting from Backend Web

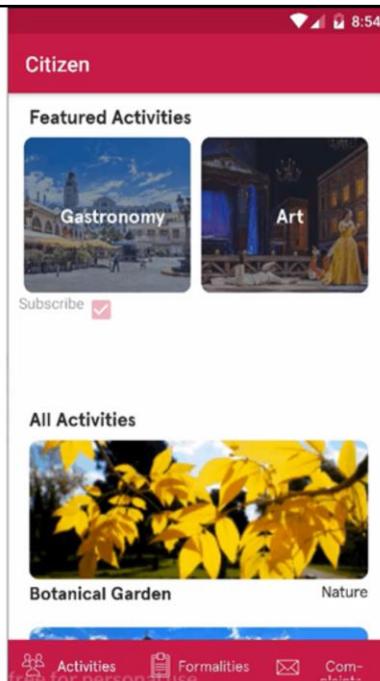
The screenshot shows a web application interface for 'Citizen Service' by GeneXus. The page title is 'Recent Cultural Activity — Cultural Activities'. There is a search bar for 'Activity Name' and an '+ Import' button. Below is a table of cultural activities with columns for Name, Category, Photo, Featured, and Static Map. Each row includes 'UPDATE' and 'DELETE' buttons.

Name	Category	Photo	Featured	Static Map		
Botanical Garden	Nature		<input type="checkbox"/>		UPDATE	DELETE
Camaval Museum	Culture		<input type="checkbox"/>		UPDATE	DELETE
Centenario Stadium	Sports		<input checked="" type="checkbox"/>		UPDATE	DELETE
Dámaso Antonio Larrañaga Zoological Museum	Culture		<input type="checkbox"/>		UPDATE	DELETE
Japanese Garden	Nature		<input type="checkbox"/>		UPDATE	DELETE
Juan Manuel Bienes Museum	Art		<input type="checkbox"/>		UPDATE	DELETE
Mercado del Puerto	Gastronomy		<input checked="" type="checkbox"/>		UPDATE	DELETE

Afterwards, someone inserts a new activity from the web backend and after saving, the notification is sent to the subscribed users.

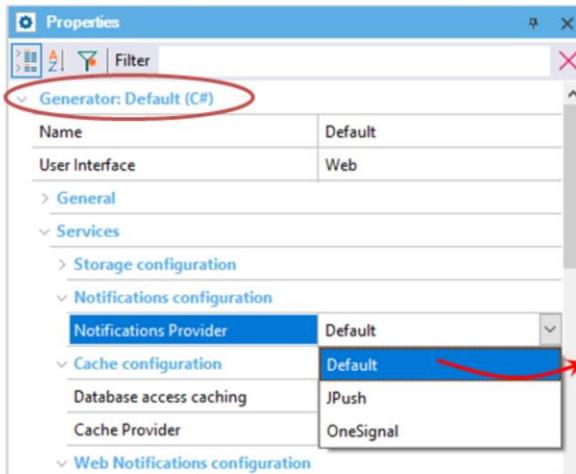
Receiving the Push Notification

How..?



The user receives the notification in his/her device and when tapping on it, he can see the detail of the new activity.
And how is this done?

Notifications Provider property



OneSignal (since GX15 u4)

JPush (since GX15 u8)

GeneXus Notifications by using RemoteNotification external object.
JPush Notifications
OneSignal Notifications

For this we have the property Notification Provider (Gx15 U4) at the web generator level, allowing us to choose the external provider with which we will handle the notifications. When you choose one provider or another, several properties will be displayed for its configuration.

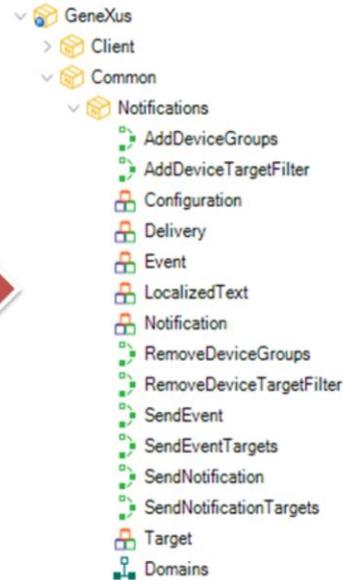
Note: The main difference between OneSignal and Jpush is that the latter is used for sites where Google service is not supported, mainly used in app for China.

Notification Provider API

Notification parameter
Remote notification
Notification configuration
Remote notification result



(Since GeneXus 15 Upgrade 3)



Previously we used several External Objects to handle Notifications; now we unify them into a single API called Notification Provider, which can be found in the module GeneXus-Common-Notifications.

Push Notification GeneXus

Somewhere in the web backend

The screenshot displays the GeneXus IDE interface. On the left, a project tree shows the following structure:

- Common
 - Configuration
 - Notifications
 - AddDeviceGroups
 - AddDeviceTargetFilter
 - Configuration
 - ConfigurationProperty
 - Delivery
 - Event
 - LocalizedText
 - Notification
 - RemoveDeviceGroups
 - RemoveDeviceTargetFilter
 - SendEvent
 - SendEventTargets
 - SendNotification
 - SendNotificationTargets
 - Target

The central data model shows the following entities and their attributes:

- Notification**
 - Id: Character(20)
 - Text: LocalizedText, GeneXus.Common.Noti...
 - Title: LocalizedText, GeneXus.Common.Noti...
 - Appearance
 - Badge: Character(4)
 - BadgeType: NotificationBadgeType, GeneXus.Com...
 - Icon
 - Large: Url, GeneXus
 - Small: Character(100)
 - Sound: Character(100)
 - Picture: Character(100)
 - Actions
 - DefaultAction: Event, GeneXus.Common.Notifications
 - CustomUIActions
 - Advanced
- Configuration**
 - ApplicationId: Character(100)
 - Properties: ConfigurationProperty, GeneXus.Com...
 - Debug
 - Advanced
- Delivery**
 - Priority: PushNotificationPriority, GeneXus
 - Expiration: Numeric(18.0)
- Event**
 - Name: Character(100)
 - Parameters
 - Parameter
 - Name: Character(100)
 - Value: Character(100)

At the bottom, a code snippet shows the parameter list for the `SendNotification` method:

```
parm(in:&Configuration, in:&DeviceToken, in:&NotificationMessage, in:&Delivery, out:&Messages, out:&Success);
```

With this API we can configure the notification using variables of SDT type and sending it from the web backend.

Let's stop to analyze the SDT that will be used:

- The Notification SDT allows configuring the content of the notification as well as the action that will trigger some event, such as opening an sdpanel sending parameters.
- Configuration allows you to indicate the name of the main SD object where the notification is executed in its Application ID item.
- Delivery allows you to set the notification priority and expiration priority.

Finally, there are some procedures to be able to send the notification, for example Send Notification, which is in charge of sending the notification to the device with the configurations made.

GeneXus

Configuring the Notification

```

&TheNotification.Title.DefaultText = "Citizen there is a new activity for you"
&TheNotification.Text.DefaultText = &activityName
&TheNotification.Actions.DefaultAction.Event.Name = "Subscribe"
// Declared in the SD Main object
&TheNotification.Actions.DefaultAction.Event.Parameters.FromJson
('{{"Name":"ActivityID","Value":'+&activityID.ToString()+'}}')

&TheNotificationDelivery.Expiration = 3000
&TheNotificationDelivery.Priority = PushNotificationPriority.High

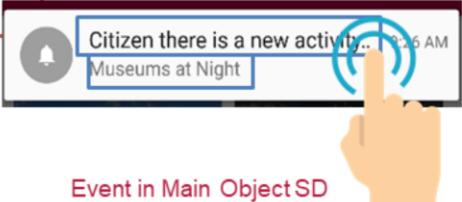
&TheNotificationConfiguration.ApplicationId = !"CitizenMenu"
// Name Main Object SD
  
```

Notification

- SD
- Text

Configuration

- ApplicationId Character(100)
- Properties
- Item ConfigurationProperty, GeneXus.Common...
- Debug
- Advanced
- CustomActions
- Advanced



Event in Main Object SD

Event 'Subscribe'
ActivityDetail(&ActivityID)
endevent

&TheNotification is based in Notification SDT
 &TheNotificationDelivery is based in Delivery SDT
 &TheNotificationConfiguration is based Configuration SDT

Within the notification you can configure:

- Title
- Text
- The event that was defined in the main sd object and the one in charge of opening the Activity Detail once the user tapped on the notification.

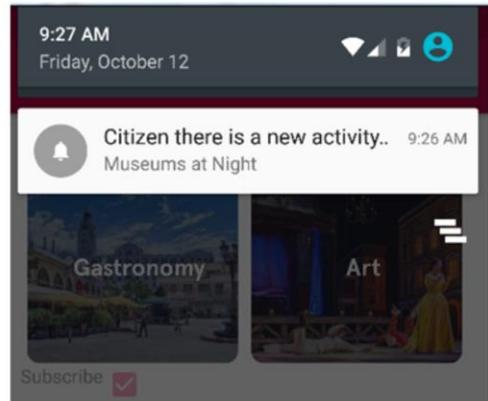
Next, set the priority and expiration of the notification.

Lastly, configure the application ID which is the name of the main object.

Sending the Notification

```
For each Device
  Where DeviceSubscribe= true

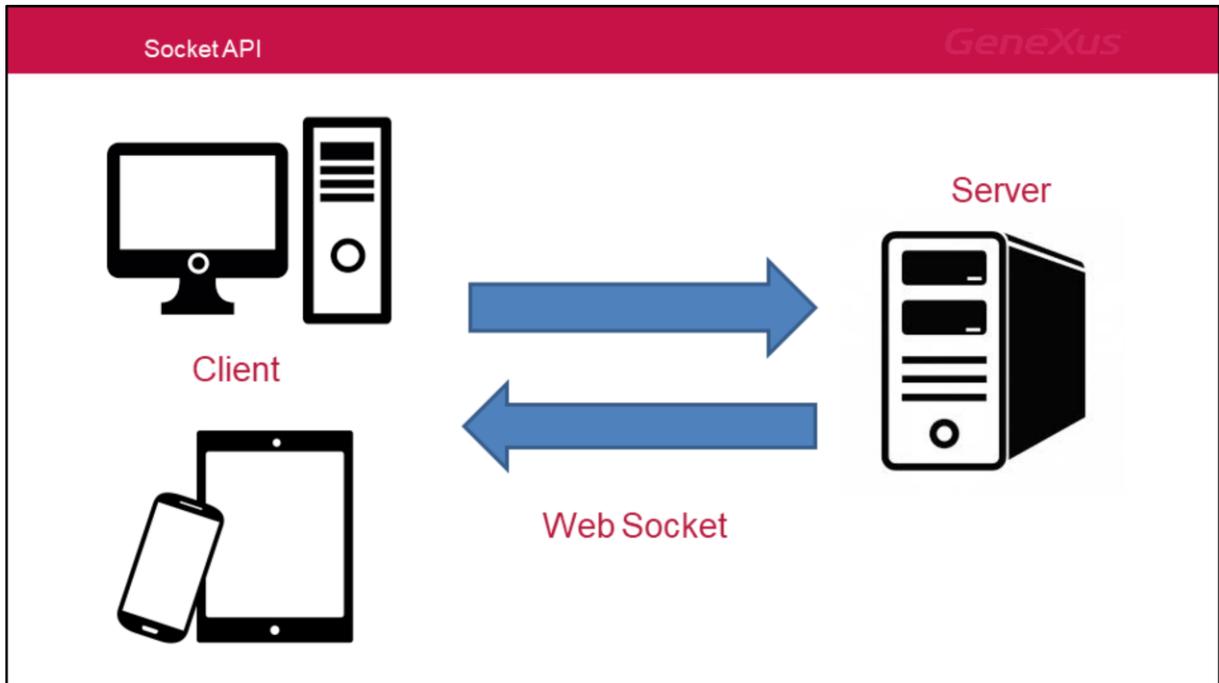
  GeneXus.Common.Notifications.SendNotification
  (
    &TheNotificationConfiguration,
    DeviceToken, // Target device token
    &TheNotification,
    &TheNotificationDelivery,
    &OutMessages,
    &IsSuccessful
  )
endfor
```



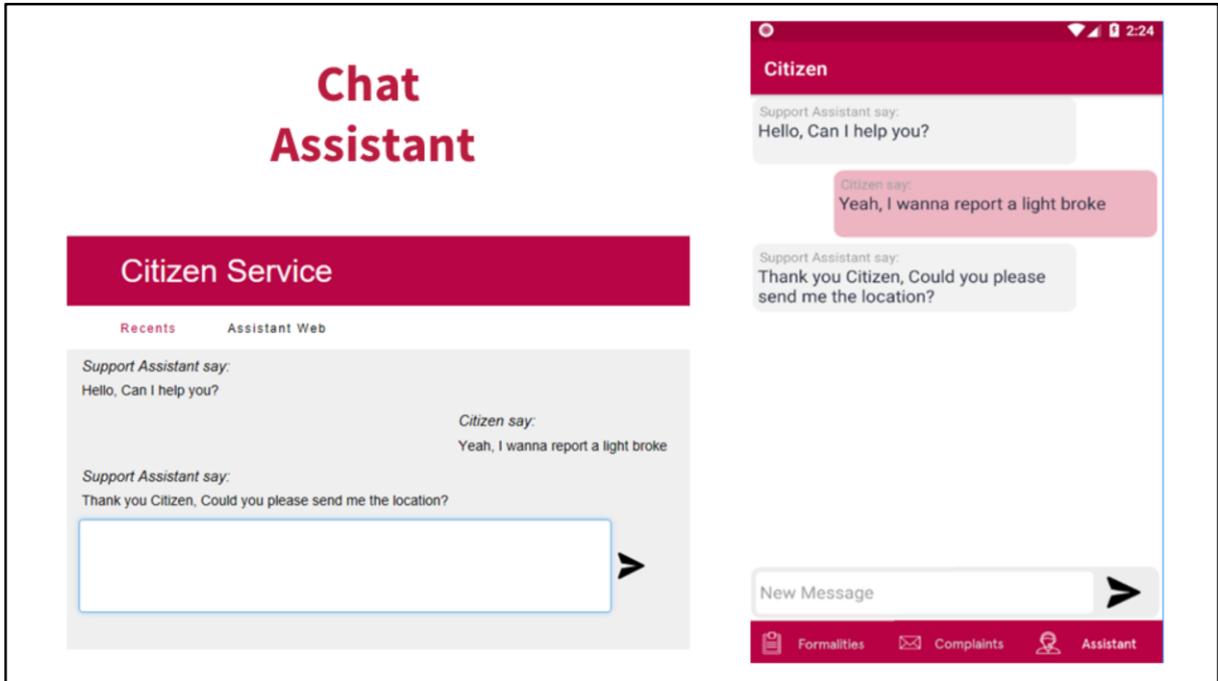
Lastly, you must call the SendNotification procedure passing the variables that you configured and indicating the device that will receive this notification; in this example, the users that subscribed.

Socket API

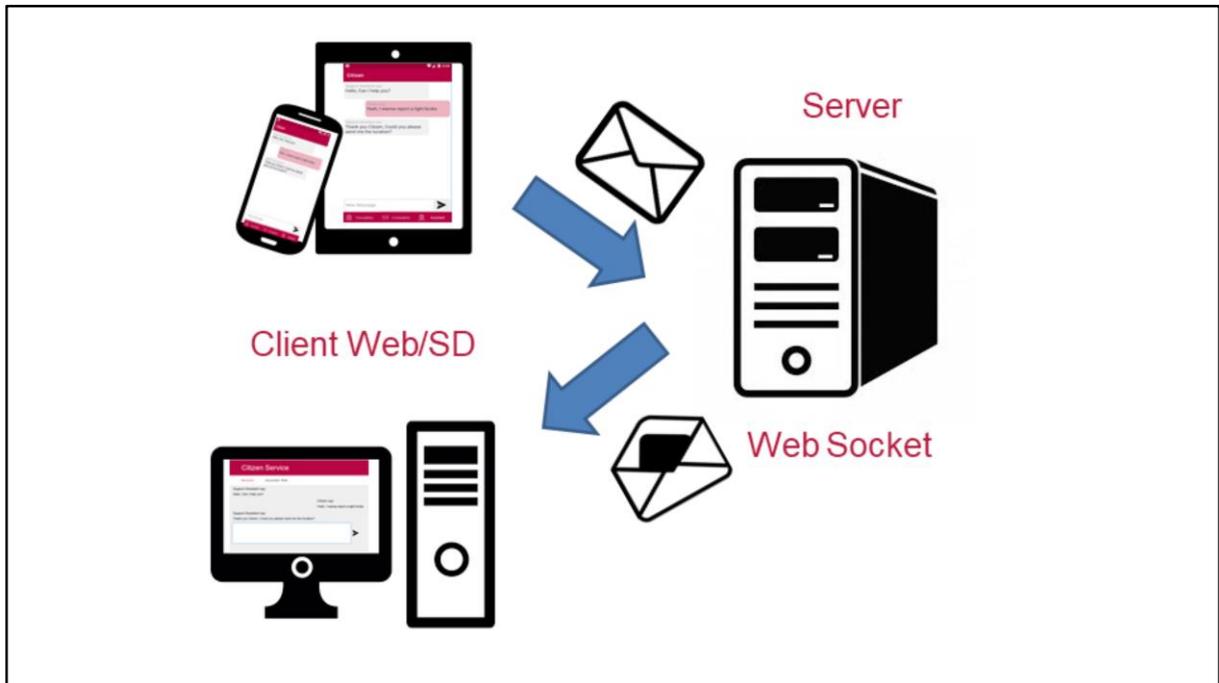
Let's move on to the following topic...



Regarding the Socket API, note that it allows establishing a two-way communication between the client and server through web sockets, which are a communications protocol that provides a communication channel over a tcp connection.



An example of use of the Socket API is an Assistance chat, which in this example allows the user to make a report in real time from his mobile and a wizard will support him from the web part.



It works as follows:

- The SD client sends the message to the Server
- The server sends it to the WEB client
- The web client will refresh its screen to show the new message
- And vice versa.

GeneXus

Server Socket

References

- GeneXus
 - Client
 - Common
 - SD
 - Server
 - NotificationInfo
 - Socket

Structure	Type
Socket	
Properties	
ClientId	Character(100)
ErrCode	Numeric(4.0)
ErrDescription	Character(100)
Methods	
Notify	Numeric(4.0)
@ sdtNotification	NotificationInfo, GeneXus.Server
NotifyClient	Numeric(4.0)
@ clientId	Character(100)
@ sdtNotification	NotificationInfo, GeneXus.Server
Broadcast	None
@ sdtNotification	NotificationInfo, GeneXus.Server
NotifyClientText	Numeric(4.0)
@ clientId	Character(100)
@ message	VarChar(200)
Events	

NotificationInfo

- Id Character(100)
- Object Character(100)
- Message LongVarChar(2M)

For this we have a Server Socket, which allows you to establish the communication from the server to the clients, sending a notification using some methods, such as:

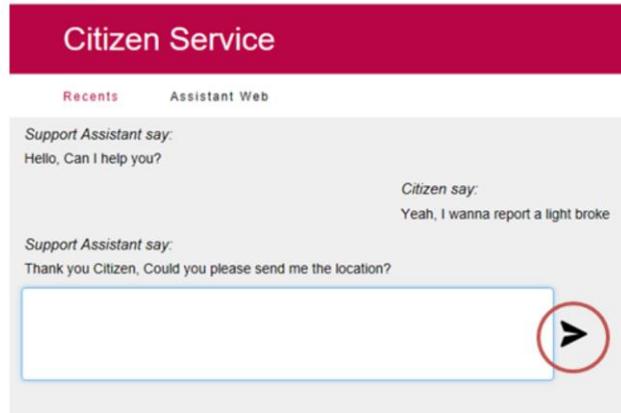
- Notify allows sending a notification to the user who originated an action.
- Notify client; this method allows you to send a notification to a specific user.
- Broadcast allows you to send notifications to all users.
- NotifyClientText allows you to send plain text to a specific client.

NotificationInfo allows you to specify the notification information that will be sent as notification ID, the Object that will receive that notification, and the message.

Optional

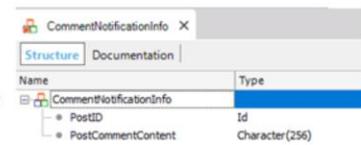
There are also some properties: clientId, which allows obtaining the client ID of the current session, error code and error description which return information in the event of an error.

Sending Message from Web



Event 'Send'

```
&commentNotificationInfo.PostId= random()  
&commentNotificationInfo.PostCommentContent= &NewMessage  
&NotificationInfo.Message= &commentNotificationInfo.ToJson()  
&ServerSocket.NotifyClient(&clientID,&NotificationInfo)  
Endevent
```



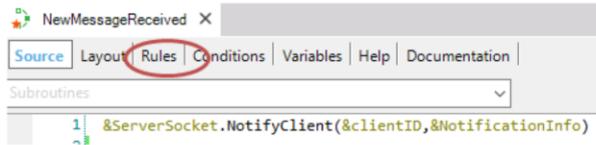
In this example, we send a new message from the Web Chat (a web panel) to the user engaged in the conversation (a Panel for Smart Devices).

Note that the SDT CommentNotificationInfo was created to send the message in json format.

Receiving Message in Web

Web Notifications configuration

Web Notifications Provider	InProcess
Received Handler	NewMessageReceived
Open Handler	NewConnection
Close Handler	LostConnection
Error Handler	WSocketError



Event OnMessage(&NotificationInfo)
//code
refresh
Endevent

```
parm(in:&clientid, in:&NotificationInfo);
```

Subsequently, to receive the answer you need to configure at the KB level the procedures that will handle the connection and the message received. Note that in this case you have the procedure NewMessageReceived. Finally to show it use the OnMessage event in the WebPanel and make a Refresh to load the new message that will be displayed.

GeneXus

Client Socket

- References
- GeneXus
 - Client
 - ClientInformation
 - ClientStorage
 - Socket
 - Domains

Structure	Type
Socket	
Properties	
Status	SocketStatus, GeneXus.Client
Methods	
Open	None
@ url	Character(100)
Close	None
Send	None
@ msg	Character(100)
Events	
Connected	None
ConnectionFailed	None
MessageReceived	None
@ msg	Character(100)

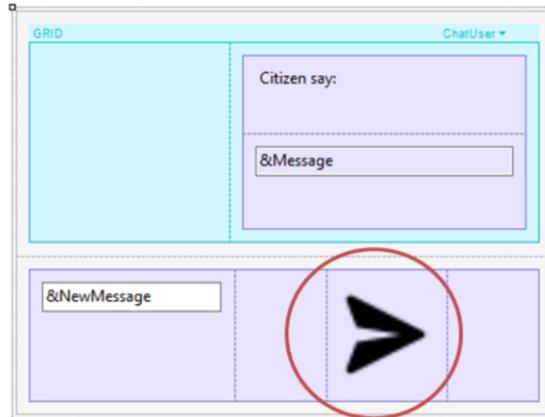
(Only for Smart Devices now)

On the other hand, for the communication between the client and the server there is a Client Socket which has methods for establishing the connection to the Web Socket and a method for sending a message.

In addition, there are events which are executed when establishing the connection or not and when receiving a message.

Note that Client Socket is currently available only for Smart Device Generators.

Sending Message from SD

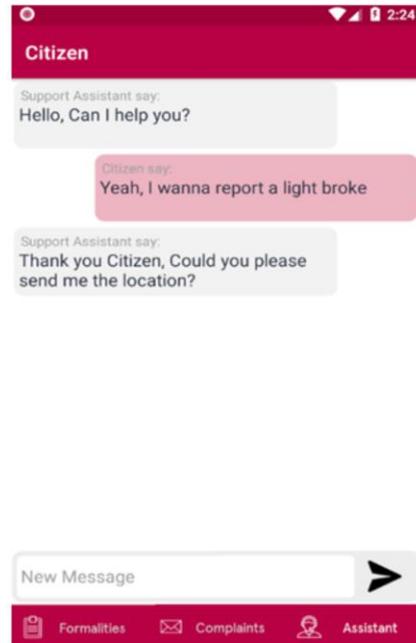


```
| Event ImageSend.Tap  
  GeneXus.Client.Socket.Send(&NewMessage)  
- Endevent
```

Continuing with the example, from the mobile part configure the delivery through the Send method passing the variable of the new message.

Receiving Message in SD

```
Event Client.Socket.MessageReceived(&MessageReceived)
  Composite
    &NotificationInfo.FromJson(&MessageReceived)
    //code
    refresh
  EndComposite
EndEvent
```



To receive messages, the MessageReceived event is used to then display them to the user.

Chatbot

CONVERSATIONAL USER INTERFACES



GeneXus

CHATBOT GENERATOR

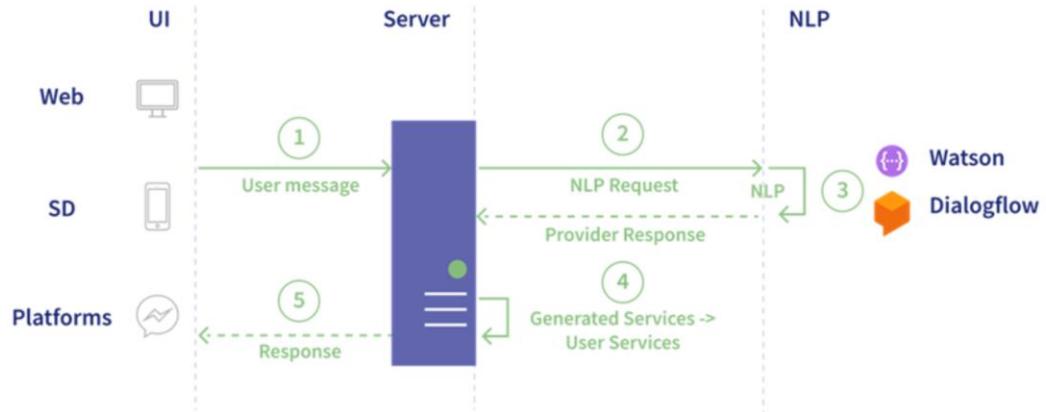
Cross-platform

Integrated to the solution

Hybrid UI

GeneXus

CHATBOT GENERATOR ARCHITECTURE



Chatbot Generator

The screenshot displays the Chatbot Generator interface. On the left, a tree view under 'Conversational Configuration' includes items like 'Default Welcome Messages', 'Default Welcome Triggers', and 'Transaction'. A 'Pattern Settings' window is also visible. On the right, a 'New Object' dialog box is open, showing a list of categories and types. The 'Conversational Flows' type is selected. Below the list, there are input fields for 'Name' (ConversationalFlows1), 'Description' (Conversational Flows1), and 'Module/Folder' (Root Module). 'Create' and 'Cancel' buttons are at the bottom right of the dialog.

New Object

Select a Category: Select a Type:

- Common
- Workflow
- Reporting
- Documentation
- Web
- Extensibility
- Deploy
- Chatbots
- Smart Devices

Conversational Flows

Create a pattern instance of Conversational

Name:

Description:

Module/Folder:

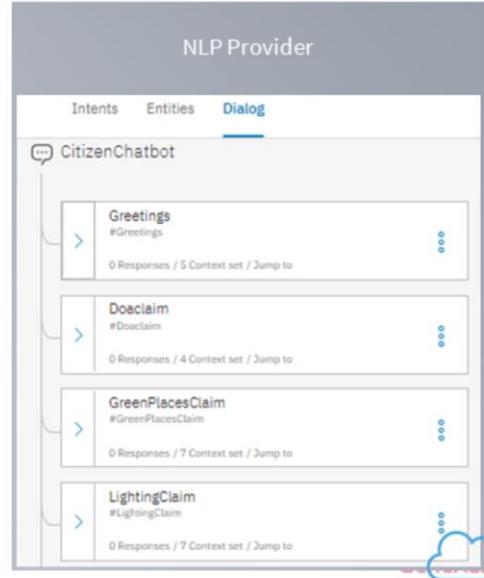
CHATBOT GENERATOR OVERVIEW

 Conversational Flows

Chatbot
Generator



↓
GeneXus objects generated



NLP Provider

Intents Entities **Dialog**

CitizenChatbot

- Greetings
#Greetings
0 Responses / 5 Context set / Jump to
- Doacclaim
#Doacclaim
0 Responses / 4 Context set / Jump to
- GreenPlacesClaim
#GreenPlacesClaim
0 Responses / 7 Context set / Jump to
- LightingClaim
#LightingClaim
0 Responses / 7 Context set / Jump to

Resources update

CommonChatbots module update

Conversational Flows

 An update is available for the Conversational Flows Resources. This action will modify the 'CommonChatbots' module and the Conversational Flows theme classes. Do you agree?



Conversational x

Properties

Pattern Settings

- Conversational Configuration
 - Default Welcome Messages
 - Default Welcome Triggers
 - Default Insert Trigger Messages
 - Default Search Trigger Messages
 - Default Update Trigger Messages
 - Default Delete Trigger Messages
 - Transaction

Filter

Config: Conversational Configuration

Keep Resources Updated	Prompt
------------------------	--------

- CommonChatbots
 - Context
 - DeserializeContext
 - GenericContext
 - Data
 - GetChatContextById
 - GetChatMeta
 - GetUserDevice
 - GetUserId
 - GetWebComponentById
 - Gx00N0
 - Gx00O0
 - GXChatMessage
 - GXChatUser
 - NewMessage
 - PanelChatSDOfflineDatabase
- Synchronization
 - Notifier
- UI
 - SD
 - PanelChatSD
 - PanelShowImage
 - Web
 - PanelChatWeb

Chatbot Generator Resources

GXChatMessage, GXChatUser

Carmines, CarminesD

PanelChatWeb, PanelChatSD

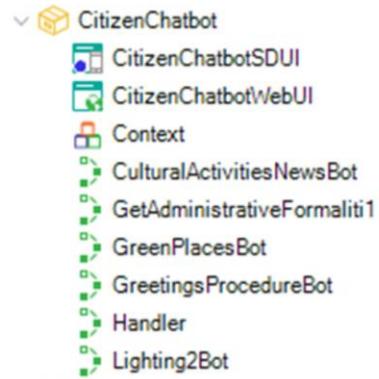
The image shows a screenshot of a chatbot interface. At the top, there is a header labeled "GRID". Below this, there is a list of messages, each in a light blue box with a thin border. The messages are: "GXChatMessageMessage", "< Component>", "GXChatMessageDate", "GXChatMessageType", "GXChatUserId", and "GXChatMessageInstance". At the bottom of the interface, there is a white input field containing the text "&Send" and a red circular button with a white right-pointing arrow.

Generated Objects

Module containing generated objects

CitizenChatbotWebUI:

```
Event Start  
└─ CommonChatbots.PanelChatWeb(Chatbot.Conversational.Watson, !"Citizen")  
Endevent
```

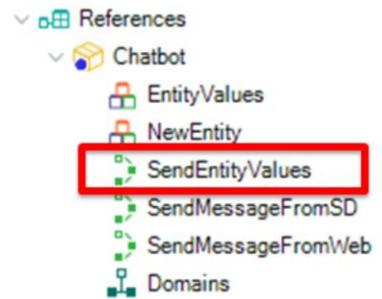


<https://wiki.genexus.com/commwiki/servlet/wiki?37102,Chatbot+generator,#+2+Generated+objects+of+the+Pattern>

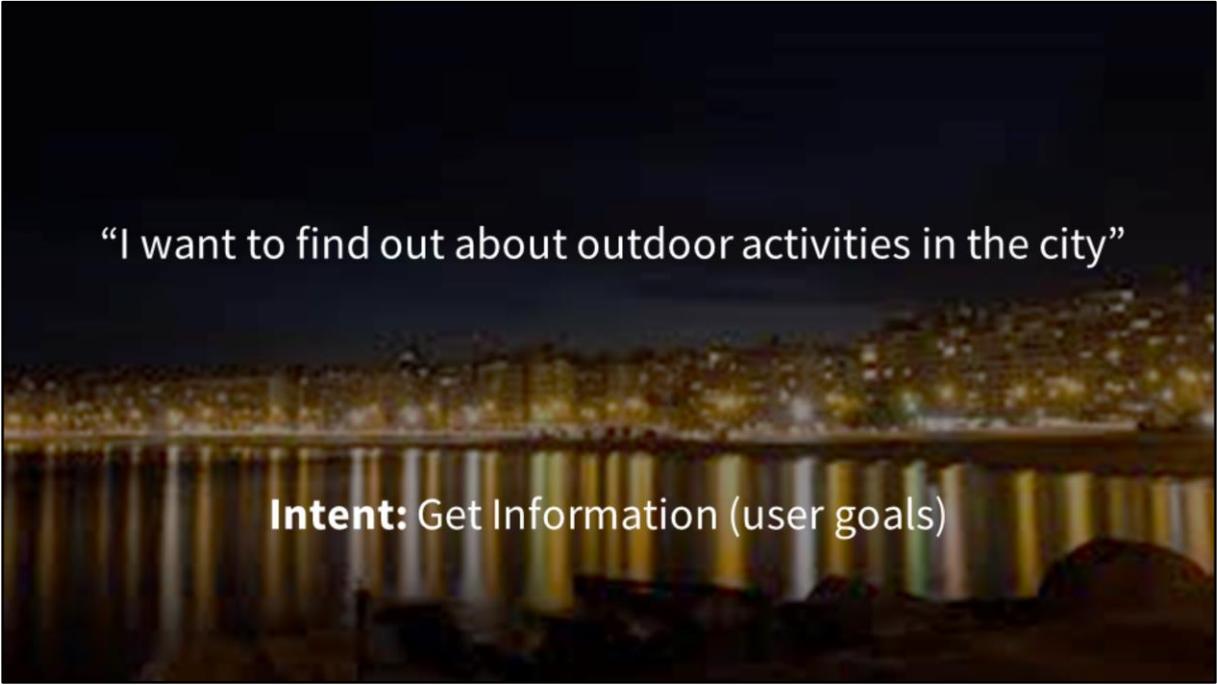
Chatbot External module

Communication with the NLP Providers

```
Chatbot.SendEntityValues(&Provider,&SDTEntityValues,  
!"UserIdentification",&InstanceName,&messages)
```



How to create a Chatbot using GeneXus



“I want to find out about outdoor activities in the city”

Intent: Get Information (user goals)

<https://www.youtube.com/watch?v=wyWgsF9eYc8>

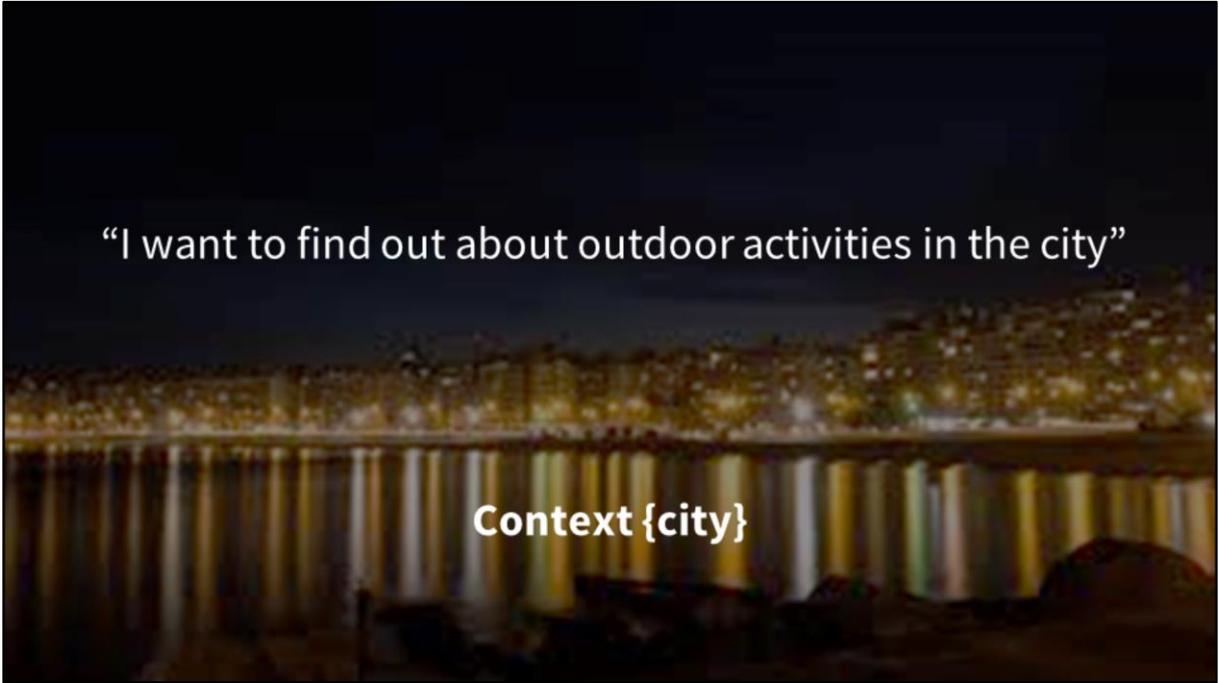


“I want to find out about outdoor activities in the city”

Entity : Activities {artistic, outdoors, cultural}
Values & synonyms

“I want to find out about outdoor activities in the city”

Context {city}



Citizen service chatbot

Do a traffic claim

Find out about activities

Get debt refinancing info

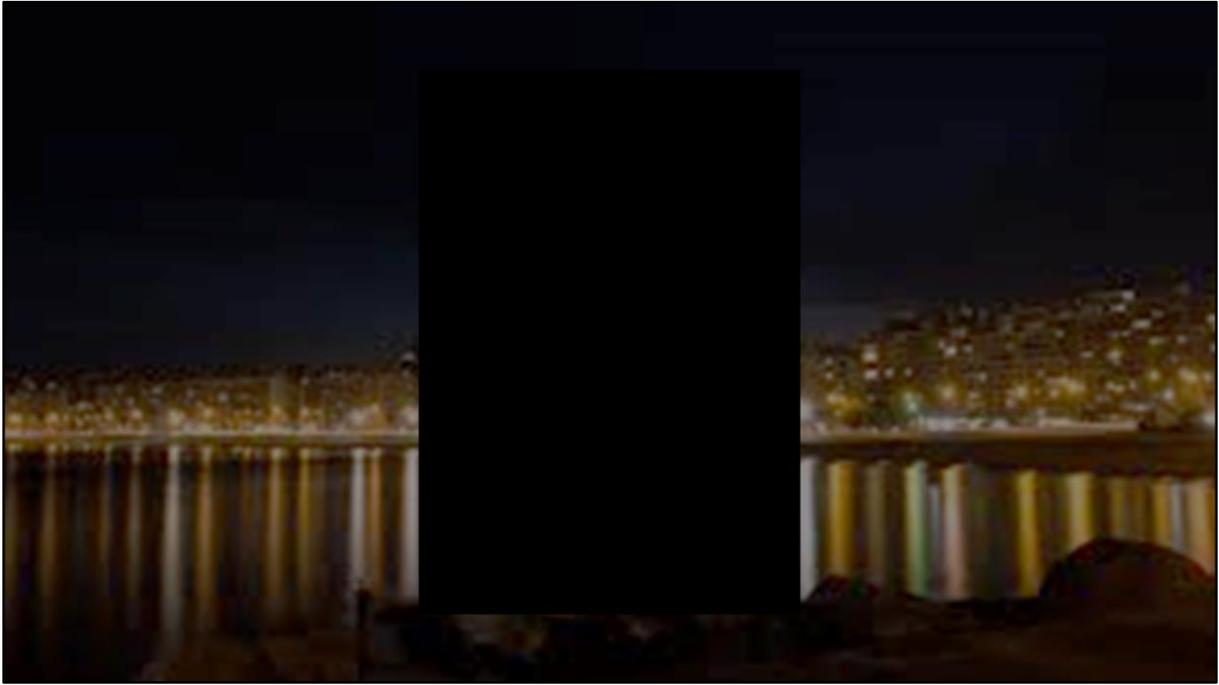
Setup an appointment

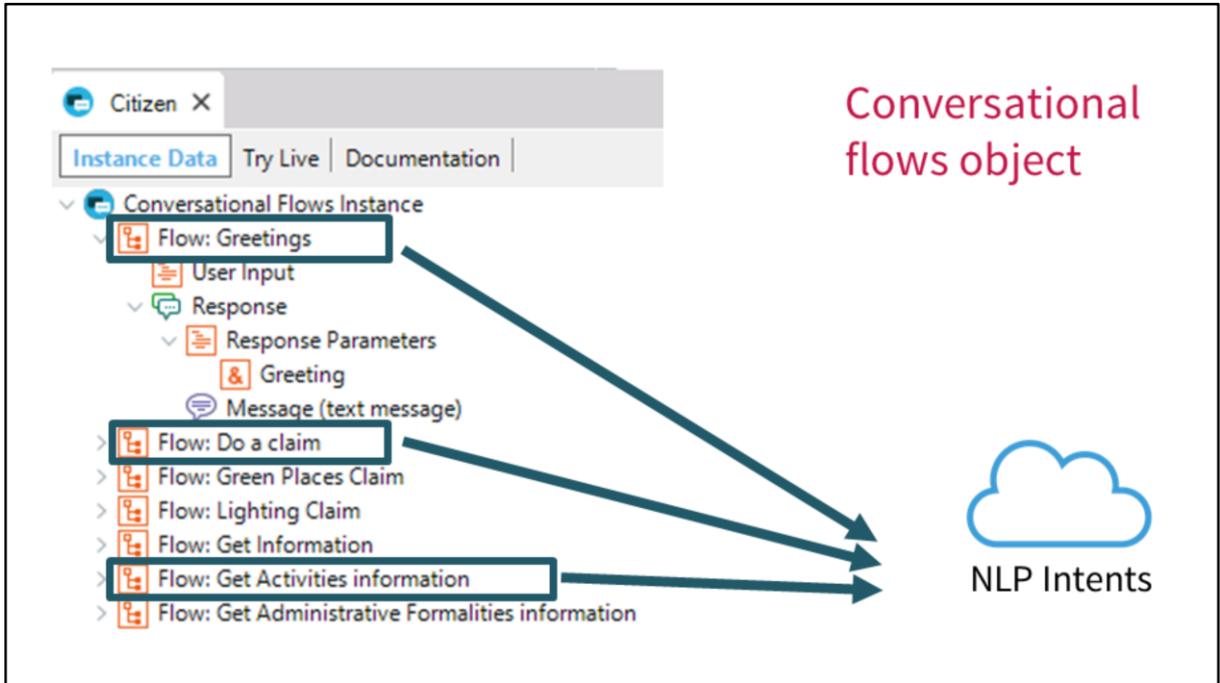


NLP response

```
{  
  "intents": [  
    {  
      "intent": "Doaclaim",  
      "confidence": 1.0  
    }  
  ],  
  "entities": [],  
  "input": {  
    "text": "do a claim"  
  }  
}
```







Trigger messages

- Flow: Get Activities information
 - User Input
 - Response
 - Response Parameters
 - Message (component view)



NLP Messages

Properties	
Filter	
flow: Flow: Get Activities information	
Conversational Object	CulturalActivitiesNews
Name	Get Activities information
Trigger Messages	Social events;info social events;activities

Know about activities

Find out about leisure

Social events information

--

User input

"I want to find out about activities in the city"



I'm very glad to help you! What type of activities are you interested in? It can be Culture, Art, or Nature.

NLP Entity



Activities

{artistic (art, music, theatre),
outdoors (nature, fresh air),
culture (cultural, museum)}

```
SendEntityValues(in:&Provider,  
in:&EntityValues, in:&Entity,  
in:&ChatbotInstance,  
out:&Messages);
```

User input



"Outdoors activities! Thanks!"

- Flow: Get Activities information
 - User Input
 - CulturalActivitiesCategory
 - Response

"outdoors"

Properties	
variable: CulturalActivitiesCategory	
Name	CulturalActivitiesCategory
Description	CulturalActivitiesCategory
Data Type	CulturalActivitiesCategory
Match With Entity	True
Entity	Activities
Ask again	True
Try Limit	2
Collection	False
Ask Messages	I'm very glad to help you! What type of activities are you interested in? It can be Culture, Art, or Natu...
On Error Messages	&UserName I don't know about &GXUserInput yet, sorry. Please, enter Culture, Art, or Nature,;
Clean Context Value	True

Response

- Flow: Get Activities information
 - User Input
 - Response
- Flow: Get Activities information
 - User Input
 - CulturalActivitiesCategory
 - Response
 - Response Parameters
 - CulturalActivitiesId
 - CulturalActivitiesName
 - CulturalActivitiesDescription
 - CulturalActivitiesCategory
 - CulturalActivitiesPhoto
 - Message (component view)

Properties

Flow: Get Activities Information

Conversational Object CulturalActivitiesNews

```
parm(in:&CulturalActivitiesCategory);
```

```
1 CulturalActivitiesNews
2 where CulturalActivitiesCategory = &CulturalActivitiesCategory
3 {
4   CulturalActivitiesId = CulturalActivitiesId
5   CulturalActivitiesName = CulturalActivitiesName
6   CulturalActivitiesDescription = CulturalActivitiesDescription
7   CulturalActivitiesCategory = CulturalActivitiesCategory
8   CulturalActivitiesPhoto = CulturalActivitiesPhoto
9 }
```

Response

- Flow: Get Activities information
 - User Input
 - CulturalActivitiesCategory
 - Response
 - Response Parameters
 - CulturalActivitiesId
 - CulturalActivitiesName
 - CulturalActivitiesDescription
 - CulturalActivitiesCategory
 - CulturalActivitiesPhoto
 - Message (component view)

Properties	
Filter	
messages: Message (component view)	
Condition	
Action	component view
Messages	Activities to do in our City
Component view properties for Smart Devices	
Show Response As	Component
Component references	
SD Component	(none)
Generated SD Component	CitizenServiceChatbot.CulturalActivitiesNewsComponentSD
Web Component	(none)
Generated Web Component	

```
parm(in:&CulturalActivitiesCategory);
```

```
Event Start
```

```
&CulturalActivitiesNew = CulturalActivitiesNews(&CulturalActivitiesCategory)
```

```
Endevent
```

Response

Layout * Rules Events Conditions Variables Documentation

Application Bar

<

&CulturalActivitiesCategory

GRID

&CulturalActivitiesNew.item(0).CulturalActivitiesId

&CulturalActivitiesNew.item(0).CulturalActivitiesCategory

&CulturalActivitiesNew.item(0).CulturalActivitiesName

&CulturalActivitiesNew.item(0).CulturalActivitiesDescription



Android Emulator - GeneXus-API26-X86:5554

ConversationalFlowsChatbot BACK

Japanese Garden

Behind the emblematic Municipal Museum of Fine Arts Juan Manuel Blanes is the distinguished Japanese Garden of Montevideo, a symbol of friendship between the Japanese people and the Uruguayan people. The garden is a work of art of nearly two thousand square meters that represents the beauty of nature and the spiritual values of Japanese culture.

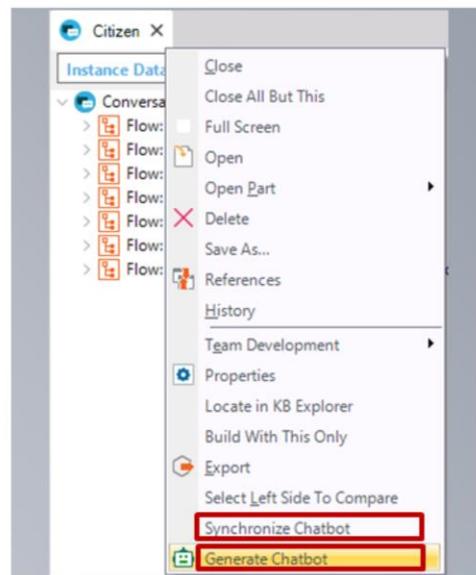


Type to talk...

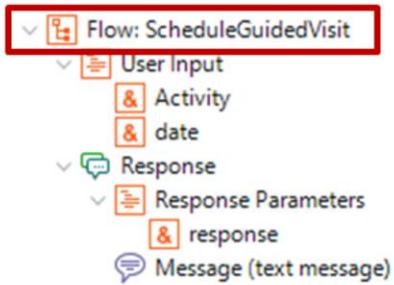


Impacting model changes

Synchronize Chatbot
Generate Chatbot



Another example: scheduling

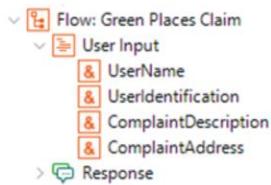
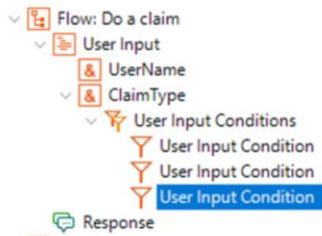


Context: Activity

System Entity: date



Redirect to another flow



Properties	
Filter	
InputCondition: User Input Condition	
Condition	&ClaimType='Sanitation'
Action	Redirect
Redirect to Flow	Green Places Claim

Entity inference from the query

"I'm John Smith and I want to do a claim about a sanitation problem"



Match With Identity

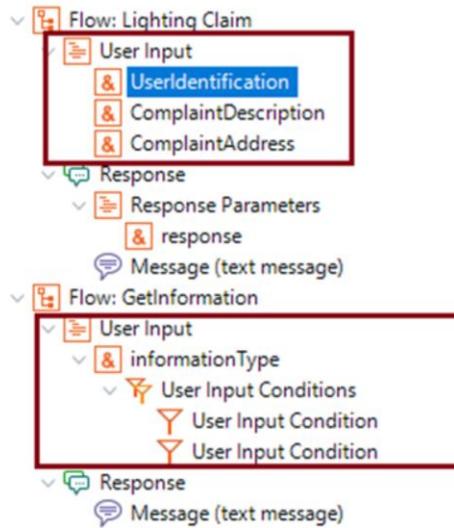
Clean Context Value = FALSE

- ∨  Flow: Green Places Claim
 - ∨  User Input
 -  UserIdentification
 -  ComplaintDescription
 -  ComplaintAddress
 - >  Response

Context

- &GXUserInput
- User Inputs

Clean Context Value = TRUE/FALSE



Try live

The screenshot shows a chat window titled "Citizen X" with tabs for "Instance Data", "Try Live", and "Documentation". The chat history includes:

- Red bubble: "what's new about activities in the city?" (18:59)
- Grey bubble: "I'm very glad to help you! What type of activities are you interested in? It can be Culture, Art, or Nature." (18:59)
- Red bubble: "I'm not sure" (19:00)
- Grey bubble: "I don't know about I'm not sure yet, sorry. Please, enter Culture, Art, or Nature." (19:00)

At the bottom, there is an input field, a "SEND" button, and a "Clear context" link. The status is "Status: 200".

On the right, the "Provider Response" section shows a JSON object:

```
{
  "input": {
    "text": "what's new about activities in the city?"
  },
  "output": {
    "text": null,
    "nodes_visited": null,
    "log_messages": null
  },
  "context": null
}
```



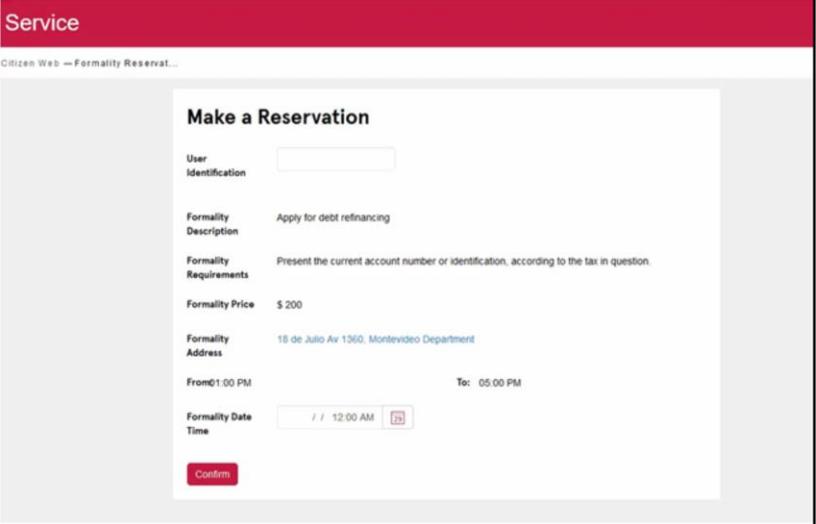
More about SD

GeneXus™ 16

Let's see some new developments in Smart Devices.

Control Value Changed

Web and SD



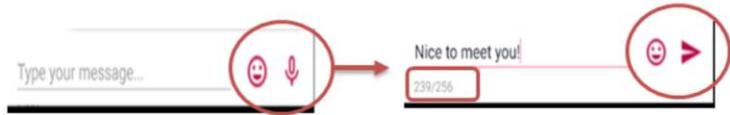
The screenshot shows a web interface with a red header bar labeled 'Service'. Below the header, the page title is 'Citizen Web - Formality Reservat...'. The main content area is titled 'Make a Reservation' and contains the following fields:

- User Identification:** A text input field.
- Formality Description:** 'Apply for debt refinancing'
- Formality Requirements:** 'Present the current account number or identification, according to the tax in question.'
- Formality Price:** '\$ 200'
- Formality Address:** '18 de Julio Av 1360, Montevideo Department'
- Formality Date Time:** A date and time selector showing '11/12:00 AM' and a calendar icon.
- Formality Time:** 'From 01:00 PM' and 'To: 05:00 PM'

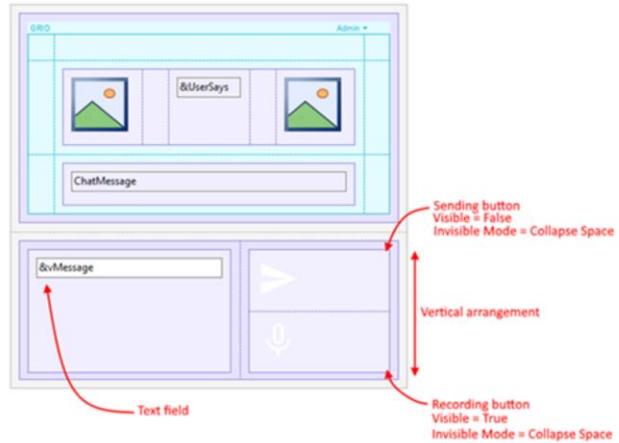
A red 'Confirm' button is located at the bottom of the form.

Control Value Changed. It's an event that allows executing a certain code when the user finishes entering a value. It is available for Web and SD.

ControlValue Changing



```
Event &vMessage.ControlValueChanging(&vMessageNew)
Composite
  &MessageLen = &vMessageNew.Length()
  SendButton.Visible = &MessageLen > 0
  RecButton.Visible = &MessageLen = 0
EndComposite
Endevent
```



The ControlValue Changing event allows validating if the user is changing the value of an editable field and execute a code. For example, when the user hasn't started entering text, the button to record an audio is enabled, and when he begins to write the text the button to send it is displayed.

Audio Recorder

- GeneXus
 - Client
 - Common
 - SD
 - iOS
 - Media
 - Audio
 - AudioPlayerCustomAction
 - AudioPlayerSettings
 - AudioRecorder ←
 - Camera

Structure	Type
AudioRecorder	
Properties	
IsRecording	Boolean
Methods	
Start	Boolean
Stop	Url, GeneXus
Events	

Damian Salvia

▶
○

9/22/17 2:44 PM

Type your message...

😊
🎤

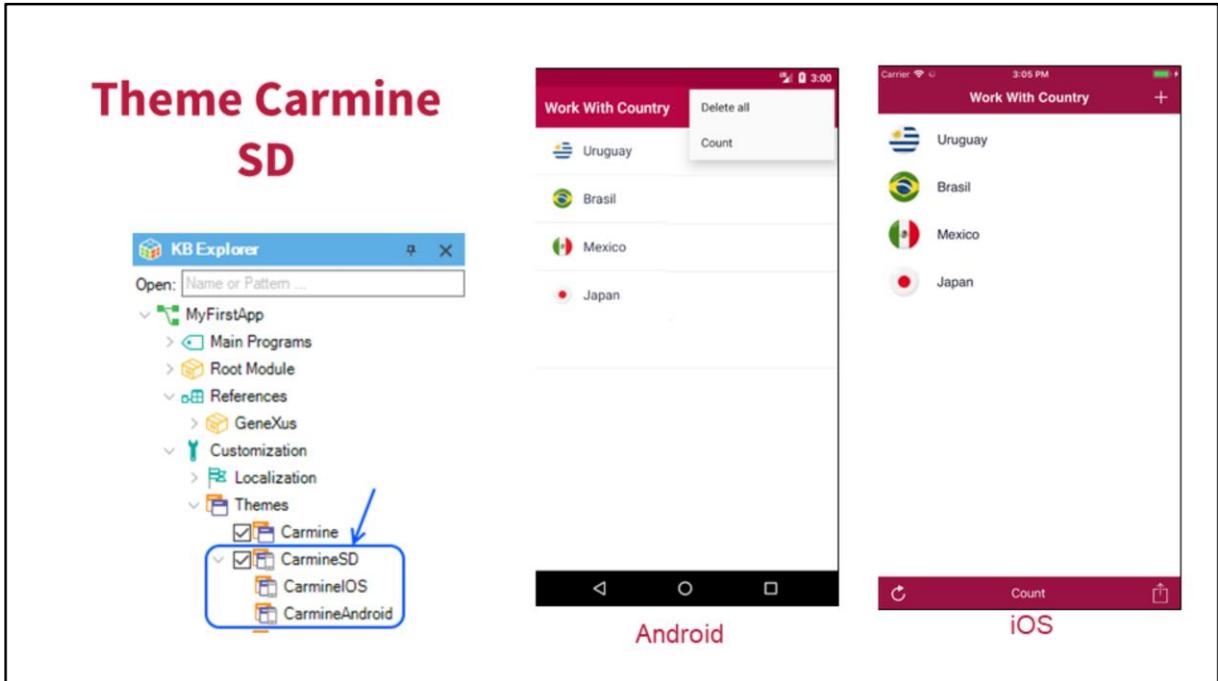
```

Event 'StartRecording'
  &HasSuccess = AudioRecorder.Start()
EndEvent

Event 'StopRecording'
  Composite
    &IsRecording = AudioRecorder.IsRecording
    If &IsRecording
      &FilePath = AudioRecorder.Stop()
      &Audio.AudioURI = &FilePath
      SendAudioMessage(&Audio,&Username)
      Refresh
    EndIf
  EndComposite
EndEvent

```

Audio recorder is an external Object that allows recording an audio track using the Start and Stop methods.



Since upgrade 6 of GeneXus 15 the Carmine theme is available. It allows obtaining a modern and sophisticated look & feel.

WebBrowser

- GeneXus
- Client
- Common
- SD
 - iOS
 - Media
 - Notifications
 - Store
 - Synchronization
 - Actions
 - BeaconInfo
 - ...
 - Search
 - SearchScope
 - WebBrowser ←
 - Domains

Structure	Type
WebBrowser	
Properties	
Methods	
Events	
BeforeNavigate	None
@ url	Url, GeneXus
@ handled	Boolean

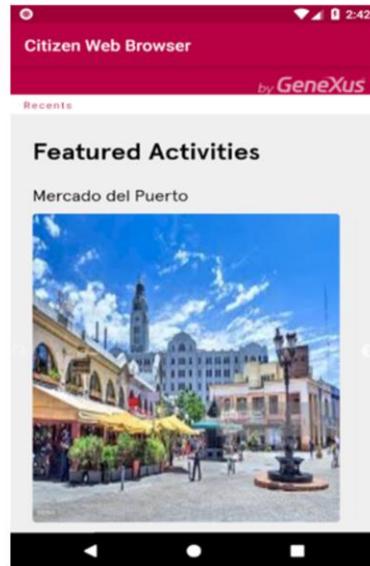
Web Browser is an External Object that allows managing the events triggered from a URL embedded in a Smart Device application.

Web app Embedded in SD app

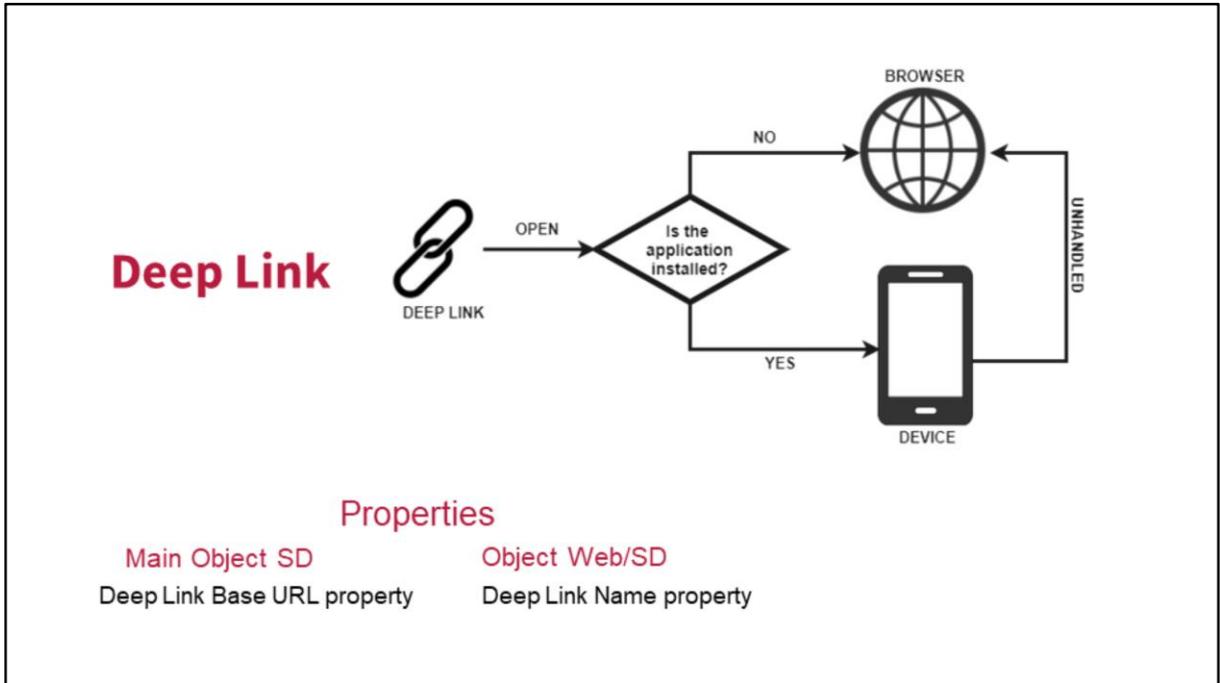
Sample: WebBrowser

Before Navigate to URL or Web app Embedded

```
Event GeneXus.SD.WebBrowser.BeforeNavigate(&Url, &Handled)  
  composite  
    if &Url = //something  
      //<your_code>  
      &Handled = true  
      return  
    endif  
  endcomposite  
EndEvent
```



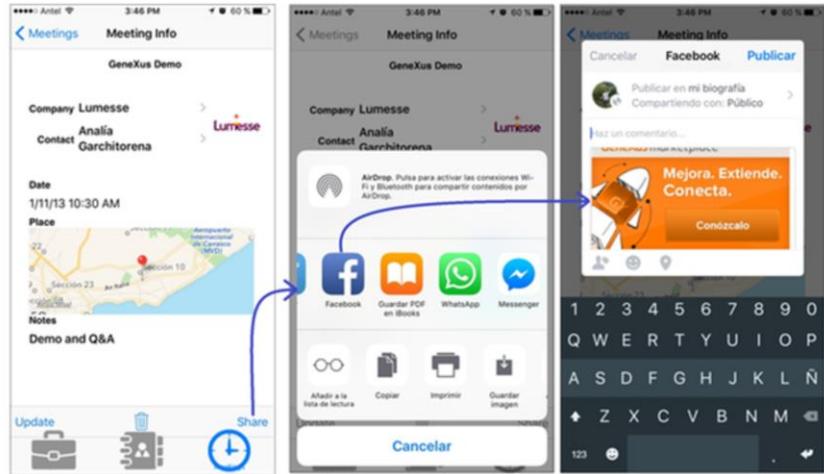
To manage navigation, use the event Before Navigate, which is triggered before browsing a URL or embedded web application.
In this way you can control if the user is accessing an external URL to your application.



Deep Link allows choosing how to open a URL. If the SD app is installed it will open it there; otherwise, it will open the corresponding object in the Web browser.

Share

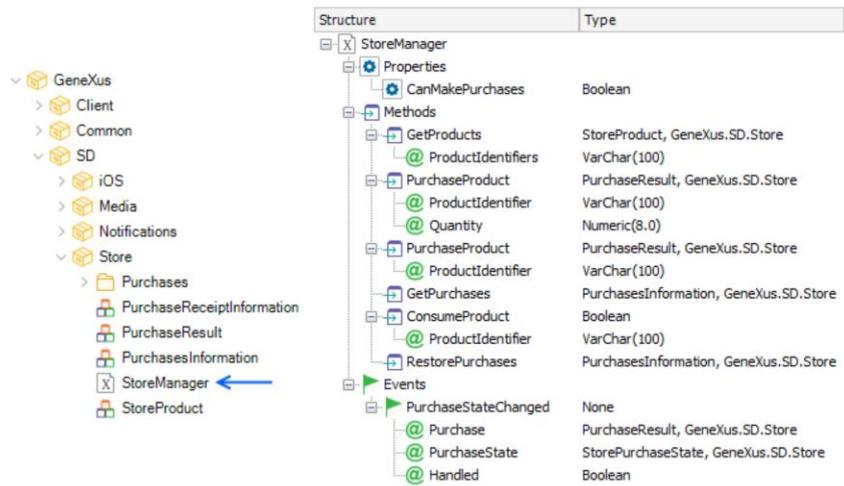
Structure	Type
Share	
Properties	
Methods	
ShareText	None
@ text	VarChar(200)
@ url	Uri, GeneXus
@ title	VarChar(200)
ShareImage	None
@ image	Image
@ text	VarChar(200)
@ url	Uri, GeneXus
@ title	VarChar(200)
Events	



```
Event 'Share'  
  Share.ShareImage(CompanyLogo, MeetingTitle,!"http://www.genexus.com",MeetingNote)  
EndEvent
```

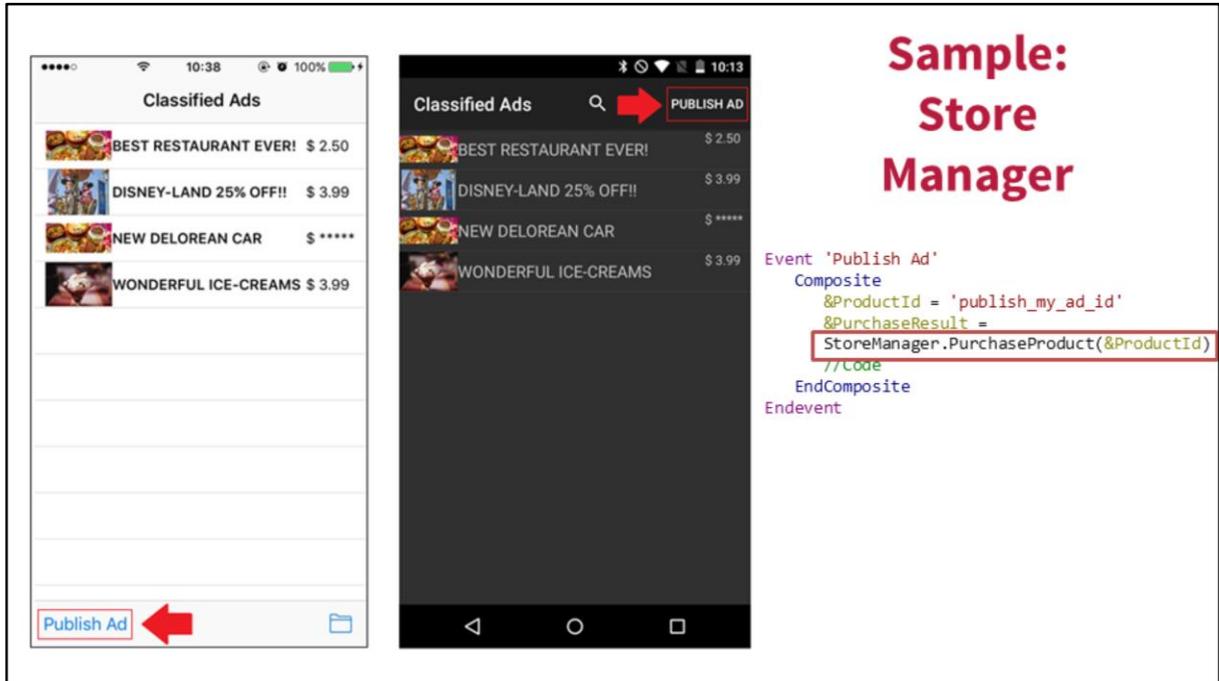
Share is an external Object that allows sharing Texts or Images with an external application that makes it possible to do so.

Store Manager



Store Manager is an External Object for managing purchase applications.

Sample: Store Manager

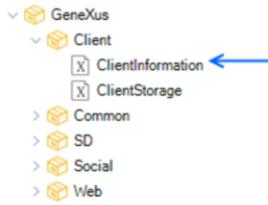


```
Event 'Publish Ad'  
  Composite  
    &ProductId = 'publish_my_ad_id'  
    &PurchaseResult =  
    StoreManager.PurchaseProduct(&ProductId)  
  //Code  
EndComposite  
Endevent
```

In this example we can see that when you tap the Publish Ad button, the option to complete the purchase is displayed.

Client Information

Main Object SD property
Include Network Id in Client Information = TRUE



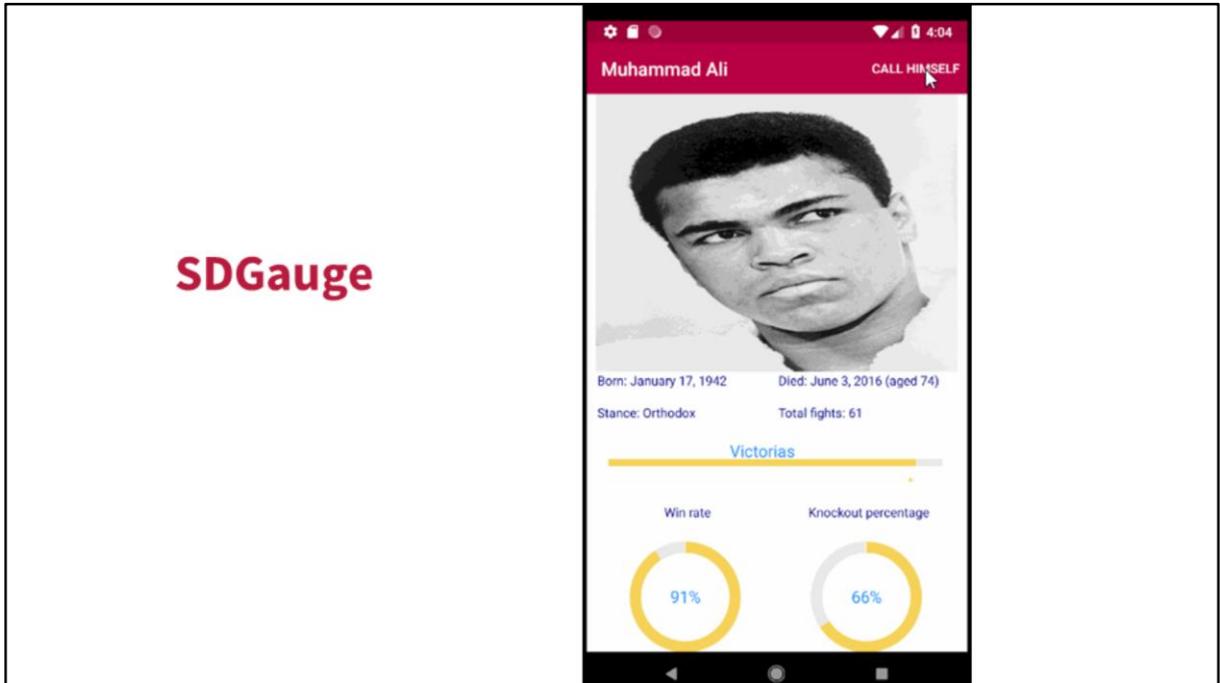
Structure	Type
ClientInformation	
Properties	
Id	VarChar(128)
OSName	VarChar(40)
OSVersion	VarChar(40)
NetworkID	VarChar(128)
Language	Character(20)
DeviceType	SmartDeviceType, GeneXus
PlatformName	VarChar(128)
AppVersionCode	VarChar(40)
AppVersionName	VarChar(40)
ApplicationId	VarChar(128)
Methods	
Events	

IMEI
MEID
ESN



Only Android

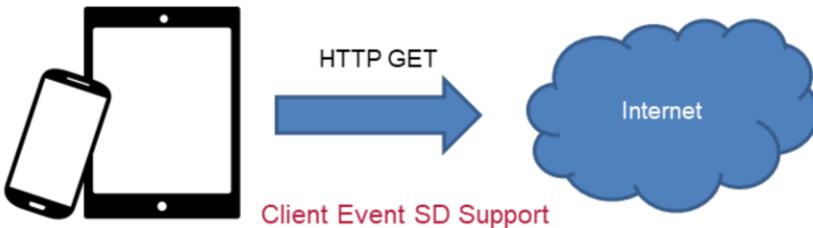
Client Information is an External Object for retrieving the device information. One of its main features is the Network ID property, which together with the Include Network ID property of the main object allows retrieving the IMEI of the device.



SDGauge is a control that allows displaying information as ranges, in linear or circular mode. This version of the control offers animation features to improve the user's experience.

HttpClient

```
Event 'GetInfo'  
composite  
  &httpClient.Host = //your host  
  &httpClient.Secure = //Secure  
  &httpClient.Port = //Port  
  &httpClient.BaseUrl = //Base Url  
  &httpClient.Execute("GET", "<something>")  
endcomposite  
Endevent
```



As from version 15, the HttpClient data type can be used in Smart Device applications, and from upgrade 12, client-side event management is available.

Share session to Webview property

SD object property

Security

Share session to Webview

True

Mobile app



Web Part in Webview



Share session to Webview property allows sharing a session between the mobile app and its web part.

Sample: Share session to Webview



```
Event Start  
    &WebSession.Set('Test', "Hello world!!")  
    &WebView = WebPanel.Link()  
Endevent
```

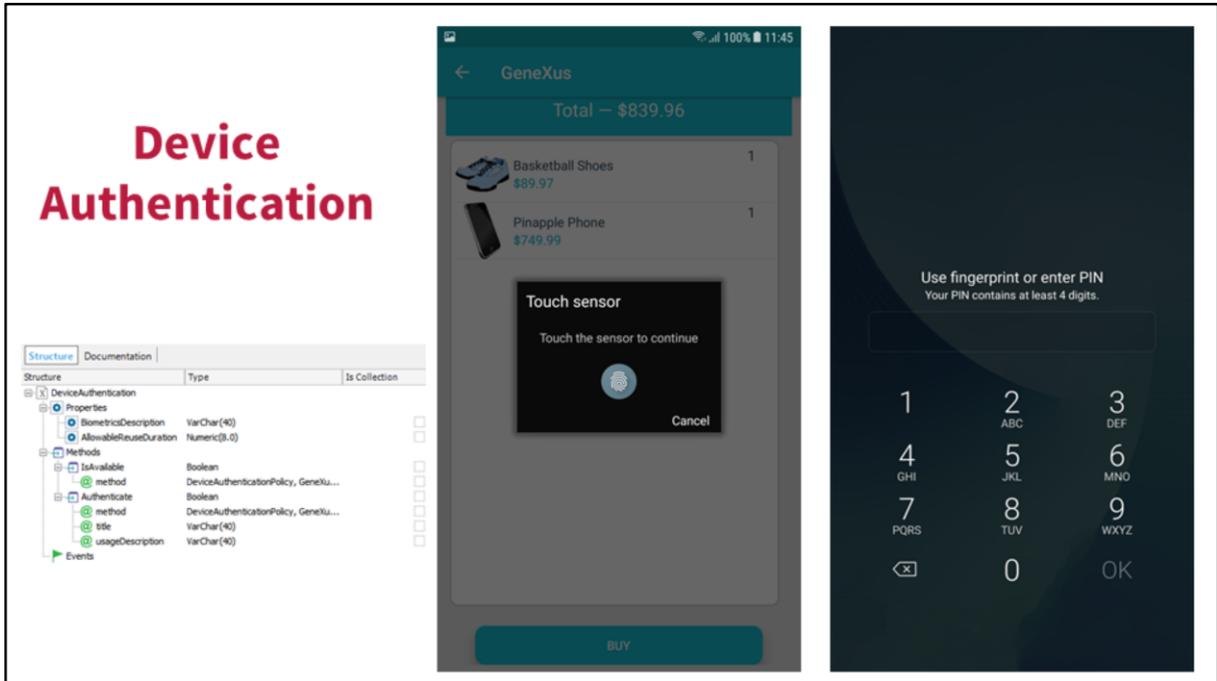


WebPanel1 object

```
Event Start  
    &WebVarchar = &WebSession.Get('Test')  
Endevent
```

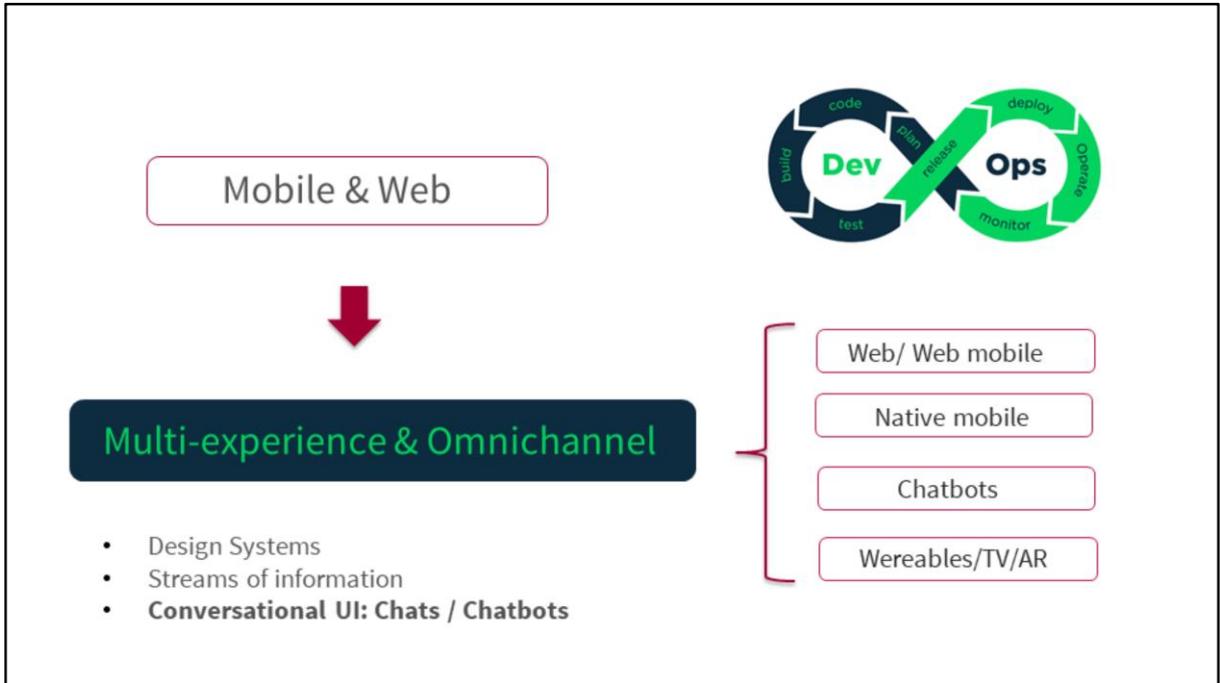
For example, from the SD object the web session is set.

In a web panel we obtain that web session value and open the web panel in a web view.

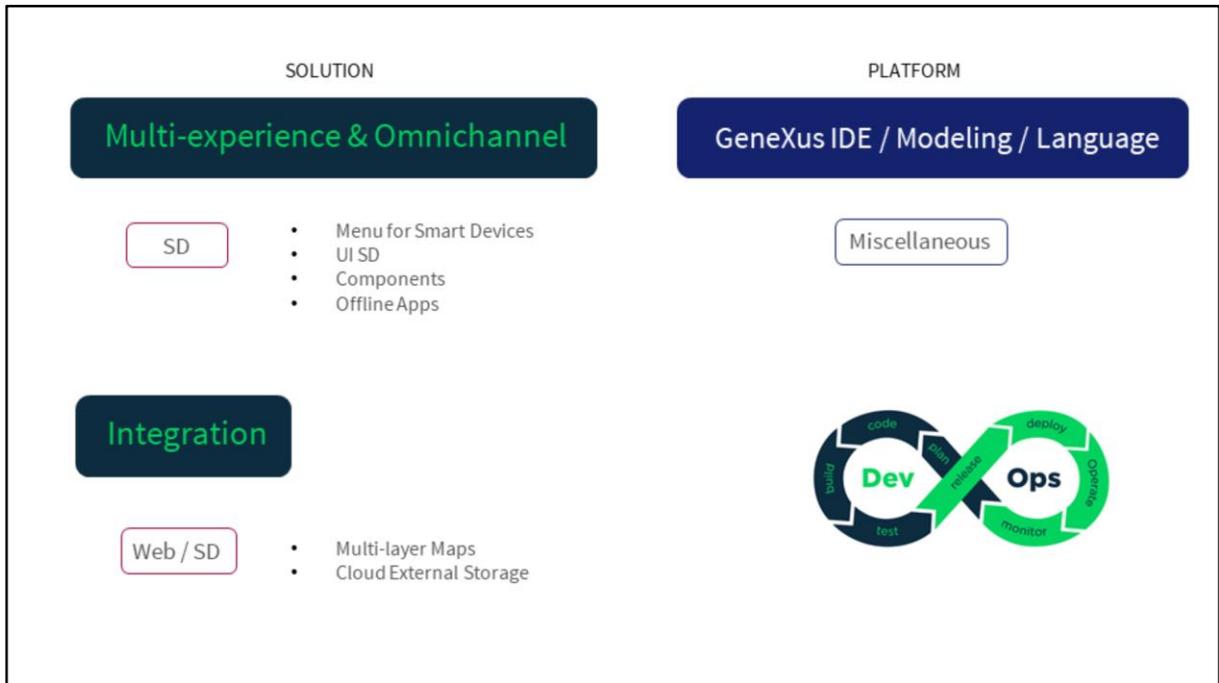


Device Authentication allows the developer to access the local authentication methods of the device.

Each platform manages its own.



We've seen the features available to develop a simple chat and a chatbot, and some aspects about SD.



Next, we'll see more features that have been incorporated in the different upgrades of GeneXus 15 up to version 16.

We'll study the new possibilities offered by maps, and some improvements regarding the external storage of multimedia files in the clouds supported.

Also, we'll see some aspects related to the GeneXus platform.

GeneXus[™]
The power of doing