ELEVATE 2024

How to allow customers to send mass emails effectively

Gian Luca De Bonis, Enable Development, CEO/CTO
December 2-3, 2024

APPEON ©2024 Appeon and its subsidiaries. All rights reserved.

DISCLAIMER

This presentation was authored by volunteer(s) in the Appeon community. This is not a work for hire by Appeon. The views and opinions expressed in this presentation are those of the author(s).

Its contents are protected by US copyright law and may not be reproduced, distributed, transmitted, displayed, published or broadcast without the prior written permission of Appeon. All rights belong to their respective owners.

Any reference to third-party materials, including but not limited to Websites, content, services, or software, has not been reviewed or endorsed by Appeon. YOUR USE OF THIRD-PARTY MATERIALS SHALL BE AT YOUR OWN RISK.

Appeon makes no warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. Appeon assumes no responsibility for errors or omissions.

Session Agenda

- Email messaging: the use case
- The “simple” way
- Problems with the “simple” way
- Implementation approaches
- The Fidelius ISV approach
- Implementation steps with Fidelius ISV – for email
- Implementation steps for other channels

Presenter Profile

Gian Luca De Bonis

linkedin.com/in/gldb

Recent Projects

- From 2021 – CTS Eventim: Scrum consulting and coaching, Agile Project Management, Symphony Product Owner
- From 2021 – Fidelius: cloud solution for messaging (WhatsApp, SMS, emails, …) from PB applications
- From 2021 – tens of PowerBuilder consulting missions for PB version upgrade, architecture and UI modernization, Scrum
- From 2018 – PB Open
- From 2004 – Enable Multilingual

Key Skills

- PowerBuilder
- C#
- DBMS
- API
- Scrum, Agility, SDLC
- Project Management
- PB Modernization

Company Profile

Enable Development

- Enable Development, Appeon Consulting Partner, helps companies to enhance the productivity and the quality of their development, by providing consulting, training, mentoring and software development services – including managed out-staffing.
- With more than one hundred and fifty customers all over the world, Enable Development organizes a network of consultants (freelancers and employees) and provides development services on projects it manages directly, backed by a team of about 500 developers with all technologies, including PB.
- The company specializes in: PowerBuilder, C#, DBMS, API, PB Modernization, Scrum and Agility, development tools, localization, Agile project management, strategic IT consulting, PowerBuilder-centered migration to C# (PB Open).

Email messaging: the use case

- Sending emails is one of the most common use cases for data-centric applications, where PowerBuilder is undiscussed king
- Our applications usually send:
  - Transactional emails – linked to a specific event or process
  - Mass emails – linked to a marketing campaign
- Emails are in different languages (for different customers), most of the times in HTML format, sometimes with attachments (in different languages)
- Emails often contain links to resources (images), and action links/buttons for specific actions

The “simple” way

- Sending emails? It’s so simple!
- We just need an SMTP server and an SMTP client object!
- Every customer has an email address, so we could use their SMTP server, and configure the details in the application

Problems with the “simple” way

- Well, the world changed since a while ago…

- SMTP Server Management
- Sender with certification, DKIM, SPF, DMARC
- Spam, reputation, IP management
- Server availability
- Hard to get a feedback (delivered, read, bounced), and un-subscription
- Action links require specific solutions
- and… it’s slow!!!

Implementation approaches

1. Historical approach: Customers manage their own servers
2. Historical approach with service: we manage the customers' SMTP servers
3. Centralized solution
4. Use a cloud provider

Implementation approaches

Historical approach: Customer manages their own server

- do they have advanced technical knowledge?
- where do the action links point?
- how to get the feedback and manage un-subscription?
- it is slow

Implementation approaches

Historical approach with service: we manage the customer's SMTP server

- can we get the security clearance?
- where do the action links point?
- how to get the feedback and manage un-subscription?
- it is slow

Implementation approaches

Centralized solution

- We have one or more SMTP servers
- This includes spam management and reputation, IP management
- A big problem is also the sender certification, and allowing customers to use their own address
- the action links could point to a high-available service, in the infrastructure
- the service also needs to be ready for the webhooks to manage feedback and un-subscription
- and … it’s slow!!!

Implementation approaches

Use a cloud provider

- Several cloud providers allow ISVs to send emails, in a modern and efficient way
- All we need is an integration, API Keys management, sender certification, account billing and so on
- A highly-available service needs to be setup in our infrastructure for processing the action links
- the service also needs to be ready for the webhooks to manage feedback and un-subscription
- So, we can have one provider for emails, several for SMS (geographically organized), one for WhatsApp and so on
- Basically, dedicating development and operations resources to a non-core business

The Fidelius ISV approach

- Fidelius ISV is a fully managed, multi-modal, unified REST API for messaging (SMS, WhatsApp, Email)
- It features a simplified and easy to use API (plus a PowerBuilder class) to send SMS, WhatsApp, transactional emails, mass emails
- It’s independent from the implementation of the providers it uses, and isolates developers from providers’ breaking API changes
- Totally white-labeled, allows flexible management of commercial policies (i.e.: reselling to customers)
- No infrastructure needed for our applications – it’s either pooled or private infrastructure – and it is fully managed
- Gives the feedback for delivered, read, bounced messages
- Allows action links that can be used easily by the application
- Fast! 10,000 emails can be sent in less than a minute

Implementation steps with Fidelius ISV – for email

- get an account as a partner and load it with credits
- onboard each sender, and freely allocate credits
- a few lines of code in the application:
  - service initialization
  - sending a message
  - sending mass messages
  - getting message statuses and link actions

Implementation steps for other channels

- Sending a message via Email, SMS, WhatsApp is very similar – just specify the channel and the needed parameters
- Getting the status is exactly the same as for emails (it’s based on the message ID, regardless of the channel used)
- WhatsApp also supports chatting, that can be implemented in the application or in a Web Page / Mobile application
- Messages can include a link that points to InfoPages: powerful HTML pages that allow the recipients to interact with the message

Code snippets

Service initialization

```
n_fidelius = CREATE n_fidelius
.SetServer("https://api.fidelius.online/api/v1")
.SetAPIKeys(<partnerAPIKey>,<customerAPIKey>)
```

Sending a message

```
.PostMessage(<channel>,<text>,<recipient>,…)
.GetMessage(…)
.GetModifiedMessages(…)
```

Sending transactional or mass emails

```
.PostEmailMessage(…,<subject>,<content>,…,<ds>…)
.GetEmailMessageStatus(…)
.GetEmailRecipients(…)
```

Code snippets

Sending a message (SMS/WhatsApp)

```
.PostMessage(ls_channel, ls_text, ls_recipient,
lb_includelink, ls_infopagetext, ldt_programmed, ref
lll_msgid, ref ls_error)

.GetMessage(lll_msgid, ref ls_json)

.GetModifiedMessages(ldt_last, ref ls_json)
```

WhatsApp Chat

```
.GetChatMessages(ls_channel, lll_lastid, ls_recipient,
ls_json, ref lds_chat)

.GetChatRecipients(lds_chat, ref lds_recipients)

.PostChatMessage(ls_channel, as_text, as_recipient, ref
lll_msgid, ref ls_error)
```

Code snippets

Sending a message with InfoPages

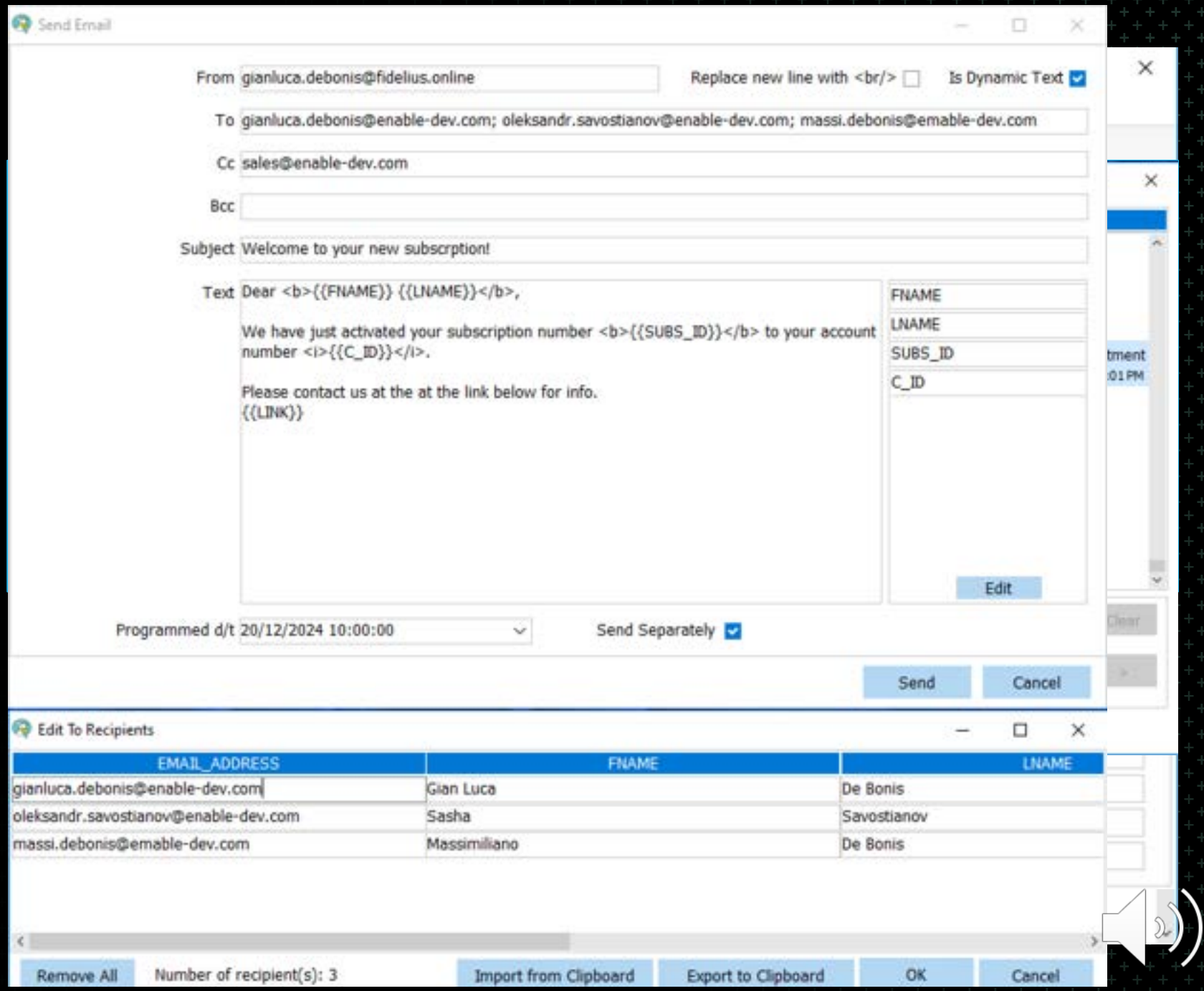
```
.PostMessage(ls_channel, ls_text, ls_recipient,
lb_includelink, ls_infopagetext, ldt_programmed, ref
lll_msgid, ref ls_error)

.GetMessage(lll_msgid, ref ls_json)
```

InfoPages endpoints

```
GET /InfoPage/MessageDetails/{messagecode}
GET /InfoPage/Customer/{messagecode}
POST /InfoPage/Confirm?
      messagecode=<msgcode>&
      confirmationstatus=<action>
```

Fidelius Control Panel



Connect with Us

community.appeon.com

Discussions, tech articles and videos, free online training, and more.

facebook.com/AppeonPB

Encourage us with a “like”, see cool pics, and get notified of upcoming events.

twitter.com/AppeonPB

Follow Appeon and community members to get the latest tech news.

linkedin.com/company/appeon

Build up your career profile, and stay in contact with other professionals.

youtube.com/Appeon

Share important Appeon videos with others; no account registration required.

Thank you

Q&A Time