Paper 4096-2020

Cloud, On-premises, and DevOps: Implementing SAS® Customer Intelligence 360 in a Banking Environment

Andrea Defronzo, ING; Roel Van Assche, SAS

ABSTRACT

SAS® Customer Intelligence 360 is a SaaS platform that integrates with your on-premises customer data and marketing processes.

This hybrid setup might raise a few questions when applied in a corporate environment, especially in a tightly regulated one such as a bank:

- How do we authenticate users?
- How do we make the system secure?
- How do cloud and on-premises components communicate? Which data is persisted in the cloud?

On the other hand, keeping your data in-house while benefiting from the advantages that cloud-based software offers, like continuous development of new features, elastic scalability and reduced maintenance, is wonderful.

This paper answers those questions with a real-world example: the implementation of SAS 360 Engage: Direct at ING BE and ING DBNL.

You get insights into the data flows of SAS Customer Intelligence 360, how APIs are used to integrate in ING's IT landscape for user management and four-eyes principle, and how DevOps software tools and methodology can be used to automate the setup and configuration of SAS. Also, given that this implementation used an early release of SAS 360 Engage: Direct, this paper also shows how the flexibility of SAS software allows for workarounds for features not yet available.

INTRODUCTION

This paper is for anybody with appetite for Customer Intelligence or cloud technology as it provides a technical understanding of SAS® Customer Intelligence 360 Engage Direct and how it was implemented at ING BE and ING DBNL, the Belgium and Netherlands branches of ING.

By using ING's implementation as a reference, it shows the technical challenges that a company can face when implementing the platform and how they can be overcome while ensuring maintainability and security.

Note that, in the following discussion, the term "ING" will refer to ING BE and ING Domestic Bank NL, for brevity.

ING

ING is a global bank with a strong European base. It has 53,000 employees serving around 38.4 million customers, corporate clients and financial institutions in over 40 countries.

ING's products include savings, payments, investments, loans and mortgages in most of our retail markets. For Wholesale Banking clients it provides specialised lending, tailored corporate finance, debt and equity market solutions, payments & cash management and trade and treasury services.

ING as a strong focus on customer experience and it is continuously innovating to improve it.

SAS® CUSTOMER INTELLIGENCE 360 ENGAGE: DIRECT

SAS® Customer Intelligence 360, or short, SAS 360, is a SaaS marketing platform to orchestrate marketing processes.

SAS 360 Engage Direct is the direct marketing or database marketing component of that platform.

Engage Direct forms the link between the on-premises customer data and traditional marketing processes with the online, digital marketing processes. SAS 360 Engage Direct makes SAS 360 a true hybrid marketing solution: (1) it unites digital and database marketing and (2) the interface and the digital data are hosted in the cloud infrastructure managed by SAS, while customer data is hosted on your infrastructure of choice.

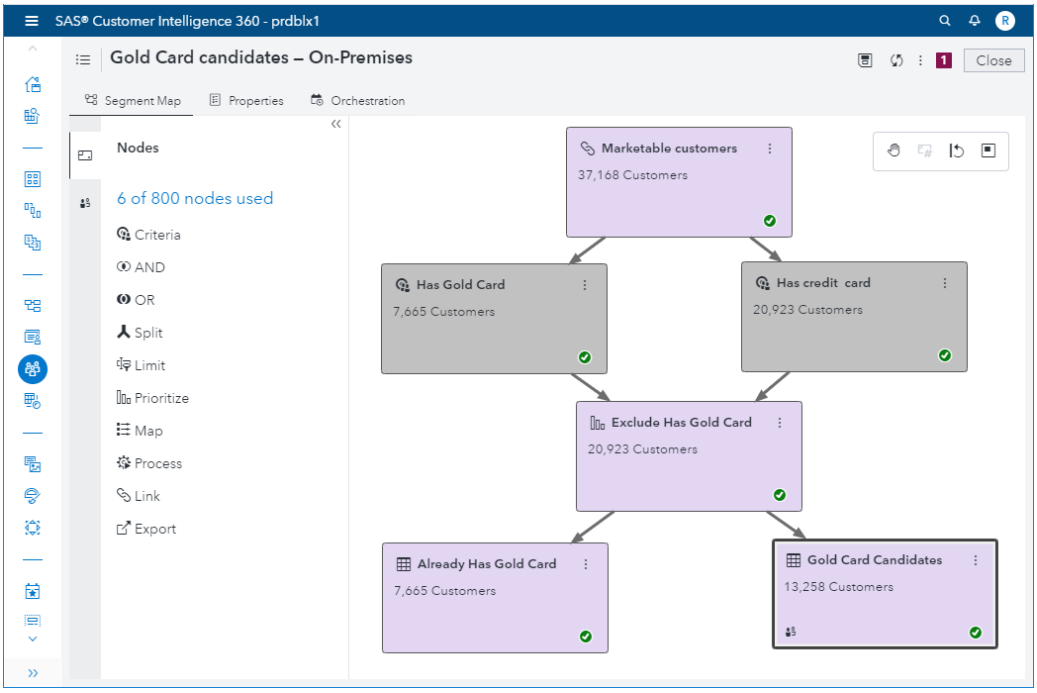
ING AND SAS® 360 ENGAGE DIRECT

ING chose SAS 360 Engage Direct for database marketing to allow the marketing team to create very targeted audiences or segments based on customer profile information, transaction history, preferences either shared by customers or derived using analytical models, etc.

The marketing team uses Engage Direct Task objects to combine these segments with messages, export layouts and timings to make them ready for scheduling and execution. At ING the execution feeds data for each selected individual into a prioritization process that eliminates overlap based on commercial pressure rules and pushes out the communication through email, paper, voice, web or other channels.

The marketing database to support the targeting activity is a tailored to ING's data and it resides in ING's private cloud. SAS 360 does not store a copy of that data, but it does keep track of:

- who is a member of which segment and
- what task objects were used to contact who and when.



Display 1. A SAS® Customer Intelligence 360 segment on on-premises data

DIRECT MARKETING AT ING

As mentioned above, ING uses SAS 360 to select groups of customers to be targeted with certain communications. These communications can be of commercial or legal nature and can be sent through a variety of channels (Email, Letters, SMS etc.), depending on the nature of the communication and the preference of the customer.

While SAS 360 could handle this process end-to-end, from the selection to the actual delivery of the message, ING decided to use it only for the selection part, since it already has other software component to handle the rest of the process.

Therefore, the high-level business requirements for the platform in ING are:

- Be able to create a selection of customers based on several hundreds of possible attributes, all stored in ING's analytical database (based on IBM PDA);
- Define several attributes for each selection (Channel, Type of Communication, etc.);
- Enforce validation of a second user for sensitive or large selections (4-eyes principle);
- Submit the selection to the software component responsible for routing it to the requested channel(s).

These requirements were mapped to the following functionalities of SAS 360 Engage Direct:

- Segment Maps: the selection diagram where users can define their selection criteria. A segment map is a sequence of nodes, each applying a different refinement (for example, filtering based on a certain customer attribute, combining two different sub-selection) to build up targeting;
- Segments: the outcome of the selection diagram. A segment simply represents a group of customers to be targeted with communications;
- Direct Marketing Tasks: the action to be performed on one or more segment. For ING, the action is always to export the segment to a specific database table(s), to be picked up by the routing component. Users can also define several properties of the task to refine the behavior of the routing component.

SAS® 360 HYBRID ARCHITECTURE

The SAS 360 Engage Direct has a hybrid system architecture because one part of the infrastructure is on AWS and a second part is on an infrastructure of your choice, which can be on-premises or at a cloud infrastructure service provider. For simplicity let's refer to this second part of the system the on-premises components.

SAS® 360 APPLICATION ARCHITECTURE

Likewise, the application architecture of SAS 360 Engage Direct is divided into a cloud and an on-premises part. The cloud application components are built up of micro services to support authentication, authorization, data identity management, scheduling, execution, analytics etc. The design center, which supports the user interface and consists of design services, is also hosted in the AWS cloud.

The on-premises components are based on a SAS 9.4 Maintenance 5 or higher and a relational database of your choice to store the campaign management or customer data mart. The SAS 9.4 server communicates with the 360 cloud via the Direct Marketing Agent, which is a component that you download from your SAS 360 cloud instance, as a part of the

initial setup. This information exchange happens via a secure web socket, opened from the on-premises SAS 9.4 server to SAS 360.

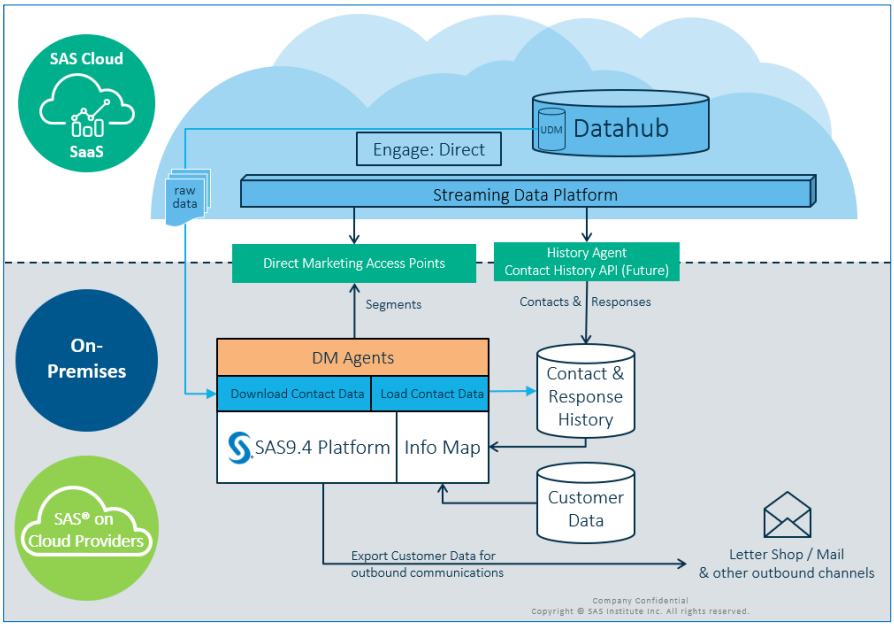


Figure 1. SAS® 360 Hybrid Architecture

As your customer data resides on-premises and the users design the segmentation logic the SAS 360 cloud UI pushes the logic down to the on-premises SAS 9.4. In the on-premises environment, the Information Map defines how to generate the SQL code to extract customer information from the relational database. The resulting count and list of subject IDs are returned to SAS 360. These subject IDs are the surrogate keys that refer to each customer.

If you or your users choose to push customer data to a communication channel, the on-premises SAS 9.4 writes the data out in the format of your choice, to the location of your choice. If your communication channel needs personally identifiable information, like names, phone numbers and addresses, SAS 360 Engage Direct does not need to push this information to the cloud, it remains on-premises.

SAS® 360 DATA ARCHITECTURE

The on-premises part of the data architecture consists of your customer data and a download of the contact history data, who is contacted when. This contact history data is typically enriched with metadata of the direct marketing tasks and segments related to the contact data.

The SAS 360 Data Hub is the data sub-system of the SAS 360 digital marketing platform. It is core to the integration of the different SAS 360 components. The identity management service of the data hub links Engage Direct with the remainder of the SAS 360 platform. By uploading the on-premises subject ID and how they link to online IDs used in the digital channels, the SAS 360 identity management service can map the on-premises and on-line data. Uploading subject IDs to 360 is a two-step process, where first the data is uploaded to an AWS S3 bucket via a signed URL, before the data from the S3 bucket is imported into the data hub.

The UDM, Unified Data Model, is the structured data layer in the SAS 360 data hub that contains the data available for download. If you have SAS 360 Engage Direct licensed, you can download contact history data and metadata of the direct marketing tasks and on-premises segments.

The streaming data platform is a fault-tolerant durable data facility that can publish and subscribe to streams of data, like a messaging system. SAS 360 Engage Direct uses the streaming data platform to exchange information with on-premises agents.

ING CAMPAIGN MANAGEMENT TOOL: A CONCRETE IMPLEMENTATION OF SAS 360

The following chapter details how the hybrid architecture described above was implemented in ING and how it was customized to suit ING specific needs.

ON-PREMISES ARCHITECTURE

The on-premises component of SAS 360 is a standard SAS 9.4 installation, with an additional component called On-Premises agent.

In ING, the installation is a multi-tier one that uses four different servers:

- 1 server for the Compute Tier;
- 2 servers for the Middle Tier and the SAS 360 agent, load balanced;
- 1 server for the SAS clients (SAS Enterprise Guide, SAS Management Console, SAS Information Map Studio), remotely accessible by users through Citrix.

The following diagram illustrates the complete architecture.

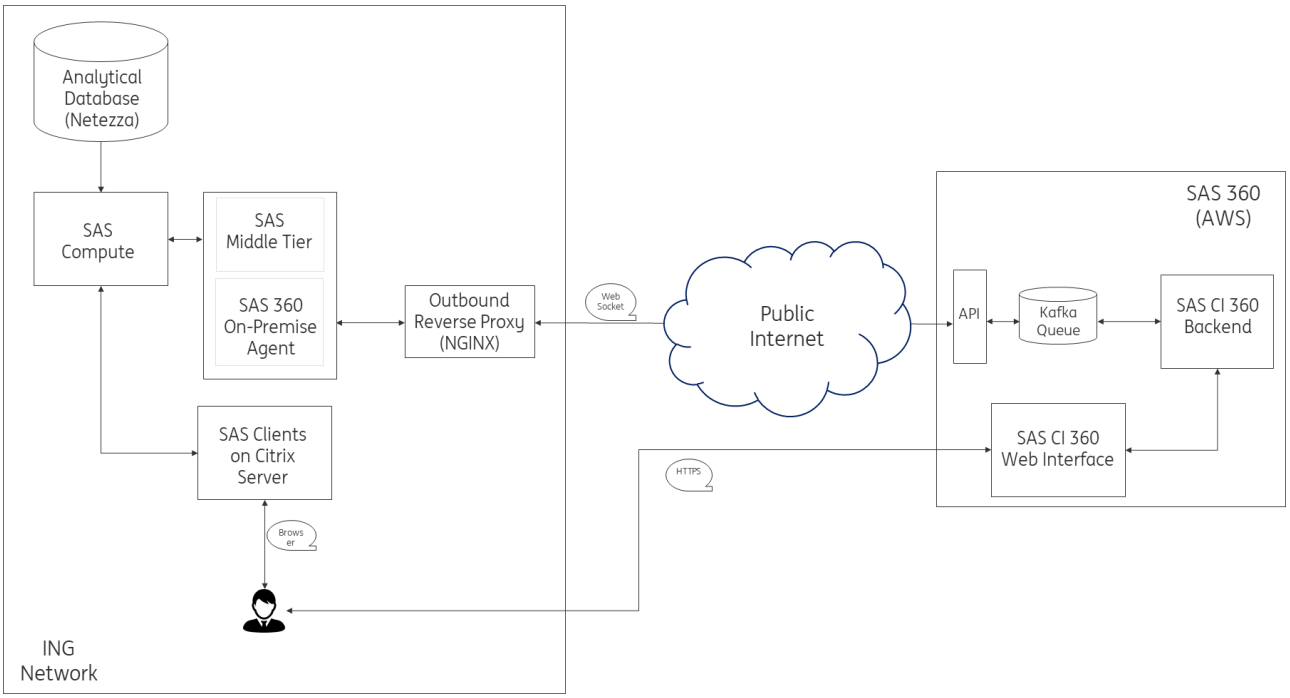


Figure 2. Architecture of the SAS 360 implementation at ING

All the servers are virtual machines, deployed on a virtualization platform called ING Private Cloud, abbreviated as IPC. The IPC is an application similar in concept to AWS or GCP, in that it allows engineers to create and manage new virtual servers from a web-based interface and have them ready to use in a short time. However, since it's completely hosted on ING's datacenters, it doesn't incur the usually risk-related challenges that companies have when trying to use the public cloud.

The IPC is very effective; while traditionally in a large enterprise it can take weeks or even months to set up a new machine ready for deploying applications, with the IPC that's no longer the case.

The SAS 360 agent is the component that connects the on-premises installation to the cloud-based portion of SAS CI 360. As mentioned above, the communication between the two components happens through a WebSocket; when the agent starts, it establishes the connection to the cloud and keeps it alive as long as it is active.

Websocket connections are by definition full duplex, which means that the agent and the cloud can send new messages on their own initiative.

Usually, the communication flow is the following:

- The cloud portion sends new commands to the agent in response of a user action in the web interface, for example the execution of a Segment Map
- The agent executes the command in asynchronous mode and sends the result back to the cloud.

The fact that the connection is initiated by the on-premises agent makes the deployment much easier, since the only thing needed is an access to the public Internet from the server the agent is deployed on.

Of course, in most enterprises servers are not allowed to connect directly to the Internet, for security reasons; outgoing connections are usually mediated by a proxy. This brings up another advantage of using WebSockets: since the protocol is designed to work over HTTP ports 80 and 443 as well as to support HTTP proxies and intermediaries, it can be proxied much more easily than other types of TCP protocols, since most enterprises already have effective forward or reverse HTTP proxies in place.

At ING, a reverse proxy is used to mediate all outgoing HTTP connections from servers; this reverse proxy is based on the very popular and powerful open source web server NGINX. The main considerations that led to the decision of using a reverse proxy instead of a forward proxy are:

- Possibility of load balancing the outgoing connections across multiple servers;
- Possibility of enabling HTTP compression.

A full discussion on the differences between forward and reverse proxies is not in the scope of this paper; for more information on the subject, see the reference section.

Note also that the on-premises agent supports both type of proxies.

The WebSocket channel is encrypted using industry standard protocol TLS 1.2, with AES 256 in GCM mode as cipher and SHA-384 for hashing.

To ensure that only an authorized agent is allowed to connect to the cloud component, JSON Web Tokens are used, another industry standard.

A full discussion of TLS and JWT is out of scope for this paper; a link with more information is provided in the reference section.

USER ACCESS MANAGEMENT

The management of user access in SAS 360 (and in basically every software application out there) can be divided in two aspects: authentication and authorization.

Authentication is the process of verifying that a certain user is really who he claims to be. The most common way of authenticating a user is through a userid/password challenge, eg. if the password submitted by the user for its userid matches the password stored in the user

database of the application for that userid, then we assume the user is really the owner of that userid.

Authorization is the process of defining what operations a certain user can carry out in the application. The most common way to manage authorizations is to use User Groups and User Roles, eg. if the user has the role "Enterprise Guide: Advanced" or is a member of a group that has that role, then he can access Enterprise Guide and use all the tasks defined for that particular role.

In an enterprise environment, users are usually managed through a centralized facility such as Active Directory or LDAP. Allowing people to access SAS using these centralized credentials (user/password) is usually a very good idea for several reasons:

- It's more user friendly instead of having to manage a dedicated password;
- It allows to enforce the same password policy used at enterprise level;
- It allows to use the approval process already in place to confirm that a specific person should have access to the SAS platform;
- If a user is removed or deactivated for any reason (eg. change of department, leaves the enterprise), the user will automatically lose the rights to access the SAS platform as well.

To allow users to use their company credentials to access SAS applications, two steps are necessary:

- User IDs and user groups have to be defined in SAS internal user database (Metadata Server), to ensure proper authorizations;
- SAS Metadata Server has to be connected to the LDAP/AD server to be able to offload the authentication process. Note that copying over the passwords in the Metadata Server is usually impossible, because for security purposes authentication systems usually don't allow retrieval of plain text passwords.

With SAS 360 there is an additional complication in the fact that the cloud portion and on-premises portion each maintain its own database of users and user groups, which are completely distinct from each other. This means that each database has to be synchronized separately.

ING uses Microsoft Active Directory to manage users and user groups.

For the authorization part, user synchronization is implemented in the following way:

- For the on-premises component, users and groups are loaded in the SAS Metadata server by the AdSync job shipped with every SAS installation, customized to assign parents groups and roles to user groups programmatically;
- For the cloud component, users are loaded using a custom-made component that simulates the operation of a human user loading users in the interface by hand, since at the time of this writing there is no API available to create users in AppCentral.

For the authentication part, the connection from SAS to ING AD is implemented in the following way:

- The SAS Metadata Server of the on-premises component is connected to Active Directory by specifying the appropriate configuration options, eg:

```
-set AD_HOST AD.ING.NET
-set AD_PORT 636
-set AD_TLSMODE 1
-authpd ADIR:AD
```

- For the cloud component, namely the SAS 360 web interface, Single Sign-on (SSO) is enabled.

SSO means that, when a user tries to access the web interface of SAS 360 using an ing.com user id, he is redirected to the authentication to ING's already existing authentication page. Once the authentication has been carried out, the user is redirected back to SAS 360 web application.

For the SSO to work, ING AD and the SAS 360 identity platform have to be linked to each other and have to be able to communicate with each other securely. This was accomplished by creating a Federated Trust between the identity platform of SAS 360 (managed by Okta) and the Active Directory Federation Services of ING (ADFS). ADFS is a software component created by Microsoft that sits on top of Active Directory and provides SSO access to applications outside an organization's network.

Creating the federated trust is quite simple, it only requires applying proper configuration on both sides (OKTA and ADFS); for more details about how to create the federation see the reference section.

Once the federation is created, the two parties communicate with each other using the SAML 2.0 protocol. SAML 2.0 is a XML-based protocol that uses security tokens to pass information (called "assertions") about an end user between the SAML authority ("Identity Provider" or IdP for short) and a SAML consumer ("Service Provider" or SP for short).

In this case, ING's ADFS is the IdP and SAS Okta is the SP. When a user tries to log in to SAS 360, Okta sends a SAML request to ADFS, requesting to authenticate the user. ADFS does that and then sends a SAML assertion back to Okta, containing the outcome of the authentication process (successful or denied) and some information about the user, for example the full name; Okta passes the SAML assertion back to SAS 360, which uses it to check for the authorizations.

The following diagram describes in more detail the flow of the authentication process:

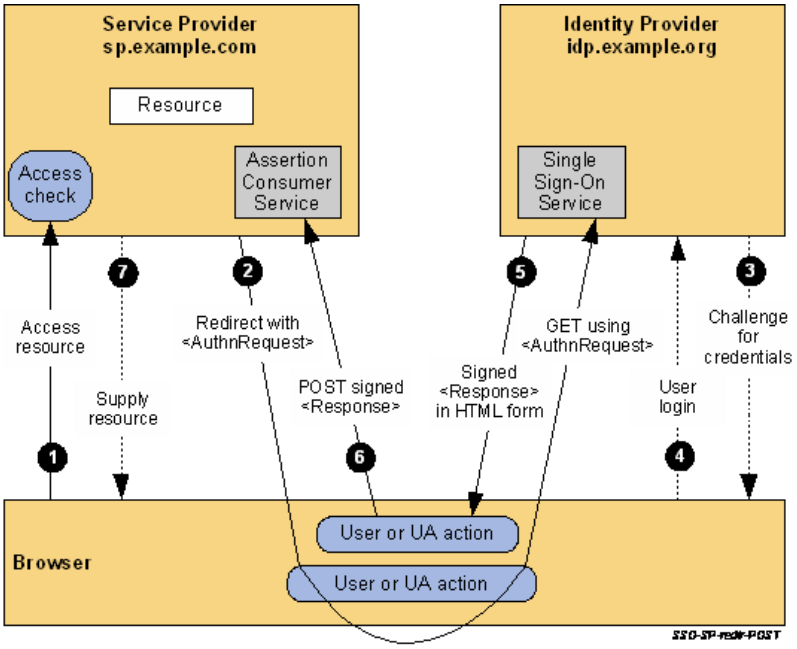


Figure 3. SSO authentication flow

As mentioned above, one of the advantages of SSO is that the authentication is performed by the already existing User Access Management facility of ING, which applies all the security policies required by company.

Among these policies, of the most important is the enforcement of Multi-Factor Authentication (MFA for short). MFA means that, when a user tries to log in, she is asked not only for her password, but also to confirm the login through a second, distinct check performed through another channel (for example, an email link). In ING, the second check consists of confirming the login through a smartphone app.

MFA greatly enhances the security of the system, which is of particular importance since SAS 360 is accessible from the public internet.

DEPLOYMENT AUTOMATION

Deployment automation refers to the process of automatizing all the steps necessary to install and configure the on-premises component of the platform on the physical servers. These steps include:

- Installing all prerequisite OS packages required by SAS 9.4;
- Performing the SAS 9.4 installation of the Compute and Middle Tier components;
- Performing the installation and configuration of the 360 Agent;
- Create all the necessary SAS Metadata objects, such as Libraries, Information Maps, User Groups, Internal Accounts etc.
- Deploy all the custom code necessary to run the day-to-day activities, for example the code needed to fetch Task custom properties from SAS 360 API;

Traditionally, many of these steps are carried out manually by Ops engineers; however, this approach has several disadvantages:

- It can be very time consuming;
- It is extremely error prone;
- In a multi-environment implementation, it is very difficult to replicate the installation in the exact same way on every environment, which makes testing more difficult.

To solve these issues, several IT automation tools have been created in the past, such as Puppet, Chef, Terraform etc.

The IT automation tool used by ING for deploying SAS is Ansible, developed by RedHat.

Ansible is, to quote its official whitepaper:

"…a powerful open source automation language. Uniquely, it's also a deployment and orchestration tool. While Ansible provides more productive drop-in replacements for many core capabilities in other automation solutions, it also seeks to solve other major unsolved IT challenges, such as CI/CD"

In practical terms, Ansible offers a simple language, based on YAML, to define actions to be carried out on the target machines. These actions include but are not limited to the following:

- Creating directories with appropriate permissions;
- Create configuration files based on templates;
- Modifying existing configuration files using simple text substitution or regular expression;
- Installing OS packages from yum, apt, Windows Software Center etc.;
- Executing operating system commands and checking for their outcome;

Each action in Ansible is called a Task; a sequence of such actions is called a Playbook. The idea is that, to perform a deployment, a DevOps engineer only needs to run a playbook

from the Ansible Controller machine. The controller will connect to all the target machines via SSH or Windows RM and execute the tasks defined in the playbook on every machine.

The fact that playbooks are YAML files also allows them to be placed in a SCM tool such as Git or SVN, which makes very easy to monitor changes applied to the infrastructure; this is what is commonly referred to as "infrastructure as a language".

One of the most important principle of Ansible is that Tasks are idempotent; this means that, if the target machine is already in the state required by the task, no actions will actually be carried out. This ensures that a playbook can be run multiple times on the same machines without breaking anything. It also ensures that, if a new machine is added to the target and the playbook is run, that machine will be brought in the same state as the other machines, while the other machines will not be changed.

Of course, idempotency is enforced by the implementation of the tasks. While task execution is defined in playbooks using YAML, behind the scene the Ansible engine uses Python to actually carry out the instructions required by the task.

Each type of task has its own implementation in Python. Many types of tasks are offered out-of-the-box by a default installation of Ansible, for example the ones mentioned above, while it also possible to develop custom types using Python.

This is an example of a task that creates the Config directory of a SAS installation:

```
- name: "Create/update SAS config directory"
  become_user: root
  become: true
  file:
    path: /sas/config/Lev1
    recurse: no
    mode: 0770
    state: directory
    owner: sas
    group: sas
```

This instead is an example of a task that modifies the autoexec of SASApp to add a sasautos statement:

```
- name: Modify autoexec for SASApp
  become: true
  lineinfile:
    path: "/sas/config/Lev1/SASApp/appserver_autoexec_usermods.sas"
    line: "{{ item }}"
  with_items:
    - "%let CMT_ROOT_PATH = {{ cmtRoot }};"
    - "%let CMT_DATA_ROOT_PATH = {{ sasdataRoot }};"
    - 'options append=(sasautos "&CMT_ROOT_PATH./_shared/macro/fsutil");'
    - 'options append=(sasautos "&CMT_ROOT_PATH./_shared/macro/util");'
```

For deploying SAS on-premise, ING used standard Ansible tasks to automatically:

- Create all required directories with the appropriate permissions;
- Install all required yum prerequisite packages;
- Set required OS limits, such as maximum number of open file descriptors;
- Run the SAS Deployment Wizard to install the Compute Tier, Midtiers, Client Tier;
- Run the SAS Deployment Manager to create several SAS Application Servers on the Compute tier, each with its own SAS General Server user and lockdown options;

- Modify several SAS configuration files, such as autoexec_usermods.sas, sasv9_usermods.cfg, sasenv_usermods to setup environment variables, autoexec code, memory limits, AD authentication etc.

In addition, to automate the creation of SAS metadata, a custom Ansible module was developed; the task is called sas_metadata and offers several benefits over manual creation of metadata in SAS Management Console:

- It allows to automate the creation of several metadata objects (user&usergroups, libraries, directory etc.) that usually have to be created by hand, a very tedious and error-prone process, especially if it has to be done on multiple environments;
- It is idempotent, meaning that an already existing object with the required structure will not be recreated, so the code works correctly even if it is ran multiple times;
- It has a three-layered architecture:
    1. The first layer is a SAS macro that lets the caller create arbitrary objects and their relationship by describing their structure using a json-like syntax;
    2. The second layer is made of several SAS macros that act as a wrapper around the first layer to abstract the complexity of creating several common type of objects (base libraries, oracle libraries, users, directory etc.)
    3. The third layer is the Ansible module that calls the macros in layer 1 and 2 inside Ansible, with all the advantages Ansible offers, described above.

The layered structure means that it is also possible to use this inside base SAS, for example for departments who do not have Ansible; also, it means that is relatively easy to add automation for new type of objects by simply adding a new macro in layer 2.

For example, this is how to use it to define a SAS Base library:

```
- name: Create base libraries
  become: true
  become_user: "{{ sasInstaller.login }}"
  sas_metadata:
    object_type: base_library
    object_params:
        library_name: "{{ item.name }}"
        library_libref: "{{ item.libref }}"
        library_metadata_path: "/Shared Data/Campaign
Management/Resources/Libraries/{{ item.name }}"
        library_physical_path: "{{ item.path }}"
        is_preassigned: "1"
        library_appservers:
            - SASApp-DAExperts
            - SASApp-DA
            - SASApp
  with_items:
     - { name: "Exports - CI360", libref: ci360exp, path: "{{ sasRoot
}}/user_data/ci360_exports"}
```

And this is an example of the creation of a Metadata directory with specific permissions:

```
- name: Create SAS Metadata main directory
  become: true
  become_user: "{{ sasInstaller.login }}"
  sas_metadata:
    object_type: directory
    object_params:
```

```
        dir_to_create_full_path: "/Shared Data/Campaign Management"
    object_permissions:
      group:
        - group_name : "CMT Technical Users"
          permission_name :
"/^(WriteMetadata|WriteMemberMetadata|CheckInMetadata|ReadMetadata|Read|Wri
te|Delete|Create)\b/"
       authorization : "G"
```

This approach to IT automation is extremely powerful; using these playbooks, it's possible to have a new SAS installation fully up and running in about two hours.

4-EYES IMPLEMENTATION FOR TASK APPROVAL

To make sure that targets selected by Tasks in SAS 360 do not contain incorrect customers, either by mistake or by a deliberate malicious action, ING implemented a 4-eyes mechanism in the platform.

The mechanism of 4-eyes makes sure that no user can create and send communications to a target without a second user validating her selection. This validation considerably decreases the chance that an incorrect selection is used.

At the time of this writing, SAS 360 doesn't offer a validation functionality; to work around this, ING implemented it as a dedicated web application.

The following diagram illustrates the general architecture of the implementation:

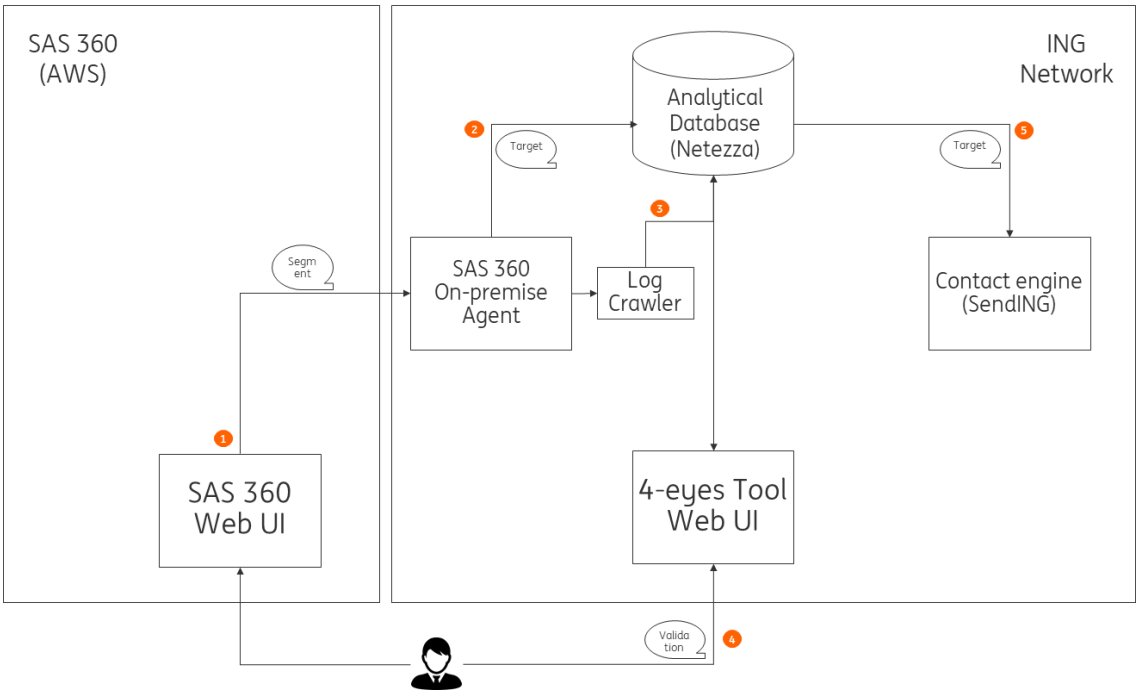


Figure 4. General Architecture of the 4-eyes validation

The general steps are:

- User A executes a Direct Marketing Task in SAS 360 to export the content to the On-Premises database;
- A crawler program running continuously reads information about the execution of the Task from the On-Premises agent log and writes it to the 4-eyes tool database;

- User B logs in to the 4-eyes web application, checks the information about the Task that the webapp is retrieving from the database and validates it by pushing on the Validate button;
- The validation information is stored in the 4-eyes tool database;
- On its daily run, the contact engine (that we call SendING) checks that the target has been validated and sends the communication to the customer in that target. If the target has not been validated, the engine ignores it.

Figure 5. 4-eyes tool task list

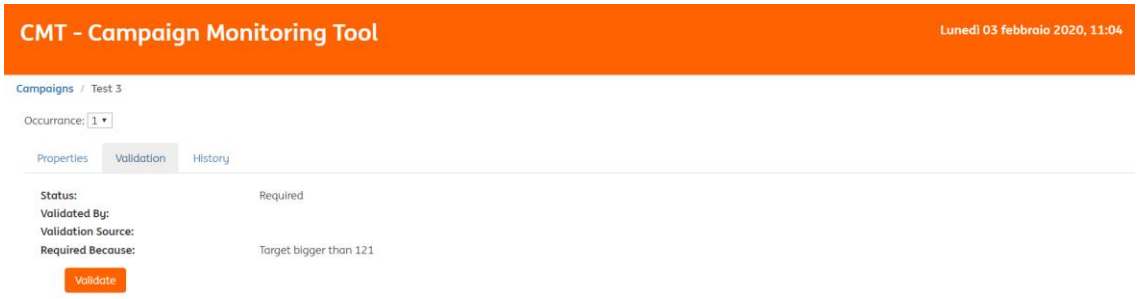


Figure 6. 4-eyes tool validation page

As mentioned above, the 4-eyes tool is a web application. This webapp is a web 2.0 application hosted on the on-premises compute tier that uses SAS Stored Process web application to act as a webapp server.

Specifically, SAS Stored Processes can be called through a dedicated web application that is part of every installation of SAS Web Infrastructure Framework; moreover, SAS code invoked in this way has the capability to output HTML, JSON or Javascript through a special fileref called _WEBOUT.

By combining these two capabilities, you can create a full web application by:

- Creating a Stored Processes that outputs the entire code of the frontend, essentially the bundle of javascript+html+css that make up the user interface of the application;
- Creating several Stored Processes, each one representing one of the REST API that the frontend calls to retrieve the data it needs.

Using this type of deployment instead of a dedicated web application server (like Tomcat, JBoss, etc.) offers several advantages:

- It is not necessary to install a maintain a separate application, since it uses the infrastructure that is already in place as part of the standard SAS installation;
- All users, user groups and authorizations defined in SAS Metadata Server can be used for the web application as well, there is no need to create a dedicated UAM functionality;
- All database connections defined in the SAS metadata can be used to access the underlying data, without the need of configuring and maintaining a separate set of configuration files;
- All the backend code can be written in SAS language, which the DevOps engineers are most likely already familiar with.

Of course, using this technique for big applications is not advisable; however, for applications that only have a few hundred users it works very well.

FETCHING TASK PROPERTIES ON-PREMISE

As mentioned in the business requirements section, ING uses the custom properties of the tasks to define how communications are sent to the target. Since the component that sends communications is on-premise, these custom properties must be available in the on-premises database.

At the time of this writing, SAS 360 doesn't offer an out-of-the-box functionality to retrieve custom properties on-premises. However, the GUI of SAS 360 uses REST APIs to retrieve this information from the SAS 360 backend and display it. Therefore, it's possible to call the same APIs from on-premises and store the results in whatever database you desire.

At ING, this was implemented by adding a custom piece of SAS code to the stored process that writes the content of a task to its final destination. This piece of code:

- Retrieves the task ID from the content of the export;
- Calls the appropriate API to retrieve the details of that task as a JSON file, using proc http;
- Uses the JSON libname engine to read the task properties as a SAS dataset, extract only the custom properties from it and write them back to the database.

Here is a sample of the code:

```
%let api_url =
&api_base_url./SASWebMarketingMid/rest/tasks?ticket=&api_auth_ticket.;

%httputil_send_get_request(&api_url., response, dummy,
content_type=application/json, request_headers="Referer"%str(=)"ticket_url",
output_as_file=1, expected_status=200)
%if &SYSCC. > 4 %then %return;

libname response json "&response.";

proc sql noprint;
      create table all_tasks as
      select a.ID
              ,c.href as GET_URL
      from response.items a
            ,response.items_links b
            ,response.links_links(where=(method = 'GET')) c
```

```
      where a.ordinal_items = b.ordinal_items
                        and b.ordinal_links = c.ordinal_links;
quit;
```

httputil_send_get_request is a wrapper around proc http that handles several common operations, for example:

- Retrieving HTTP headers from the response;
- Checking for connection errors and unexpected HTTP status codes;
- Writing the JSON response into a file, so that the JSON libname engine can read it.

In particular, you should take special care when reading the response, because it often breaks the 32767 row length limit for SAS file statement; therefore, to avoid losing data you must read it byte by byte using the "N" record format, for example:

```
      infile resp;
            file "&&&return_var_response.." recfm=N;

            length byte $1;

            input byte $char1.;
            put byte $char1. @;
      run;
```

THE FUTURE FOR SAS 360

SAS 360's continuous software delivery results in the release of new capabilities almost every four weeks. ING can immediately benefit from these new features. In some cases ING may need to adapt some of the abovementioned customizations.

Besides new functionality there is also existing functionality that the SAS 360 Engage Direct license gives access to, that ING could use in the future. 360 is known for its capability to integrate with other systems. Some examples are:

- Send next best actions to sales automation platform
- Register an inbound communication
- SMS an offer based on a transaction at the Point of Sale (POS) system

In these examples SAS 360 Engage Direct act as control tower and integrates both with inbound and outbound channels. In the third example:

- A SAS 360 External Event receives a transaction event from the POS system.
- The SAS 360 External System Task evaluates the event, checks if the customer fits the target audience and pushes the offer text message on to the SAS 360 Agent
- The SAS 360 Agent, which is an on-premises component, has a plug-in developed, in this case to communicate with the SMS API to send out the message.

Figure 7. Trigger a SMS from a POS system using a SAS® 360 External System Task

CONCLUSION

This paper has shown how it is possible to use SAS 360 to implement a very effective marketing platform; while its cloud-based nature poses some challenges, they can be overcome even in a tightly regulated enterprise environment.

It has also shown how SAS 360 and the SAS software in general are flexible enough that with enough expertise, such as the one ING has, it possible to customize it to adapt to every needs. Of particular importance is the fact that modern DevOps concepts and tools, such as IT automation and Continuous Deployment, can be applied to greatly speed up the setup and simplify the maintenance of SAS platforms.

REFERENCES

OASIS, 2008. "Security Assertion Markup Language (SAML) V2.0 Technical Overview" Accessed February 18, 2020. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.html

Microsoft, 2017. "Create a Relying Party Trust" Accessed February 18, 2020. https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/operations/create-a-relying-party-trust

IETF, 2015. "JSON Web Tokens, RFC7519" Accessed February 16, 2020. https://tools.ietf.org/html/rfc7519

IETF. 2008. "The Transport Layer Security (TLS) Protocol Version 1.2, RFC5246" Accessed February 16, 2020. https://tools.ietf.org/html/rfc5246

Red Hat. "Ansible in Depth" Accessed February 4, 2020. https://www.ansible.com/resources/whitepapers/ansible-in-depth

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Andrea Defronzo
Development Engineer
1-1 Analytics
ING Belgium NV
andrea.defronzo@ing.com

Roel Van Assche
Principal Consultant
SAS Institute NV
roel.van.assche@sas.com

Other brand and product names are trademarks of their respective companies.