ORACLE

# Oracle Responsys

## Application Development and Best Practices Guide

04-Mar-2020

CONTENTS

# DEVELOPING APPS FOR THE CX APP FRAMEWORK

## Overview

The Oracle CX App Developer Framework enables AppCloud Partners to build integrations between their systems and Responsys by building apps. These apps help marketers orchestrate their campaigns across platforms in near real time. Developers can develop apps intended for all customers of the Marketplace, or develop custom solutions for just a single Responsys customer.

By building an app, an App Provider makes their services available to Oracle CX Marketing customers. Marketers can use an app to send campaigns to their customers on channels not currently native to Responsys.

## Why Should I Develop Apps on the Framework?

Here are some highlights of developing on the Framework:

- Build your own orchestration stages to Program workflows
- Create seamless user experiences between Responsys and your application
- Post a listing for your app in the Oracle Marketplace

## Getting Started

App providers will need to create a web server application that can send and receive REST API calls to and from the Responsys system via HTTPS. App Developers need to create an App Provider within the application before they can start developing. We assume that app developers are technical and have built web applications before. These topics are not intended to instruct developers on how to develop their app, this guide provides guidance on what Oracle AMS requires from your app to work with the CX App Framework.

App developers must be familiar with:

- REST APIs
- JSON
- JSON Web Tokens (JWT)
- Web application development and deployment
- Responsys Personalization Language (RPL) for Responsys development

## High-Level Steps

The high-level steps for developing an app are as follows:

1. Set up system prerequisites:
    a. Web application server, to receive the requests and send the proper responses to Responsys.
    b. Database for storing the data that the app uses, sent from Responsys at Program runtime.
2. Develop the app, including the Responsys requirements:
    a. App Installation and Configuration: Develop the app's installation and configuration experience. This includes the integration points for which your **app** will interact with Oracle AMS(Application Management Service). Within AMS, these integration points are known as the app's Install URL, Configure URL, Save Configuration URL, and Uninstall URL.
    b. Program Design Time: Develop the Program Activity's configuration experience. This includes the integration points for which your **service** will interact with Oracle AMS. Within AMS, these integration points are known as, the service's Configure URL and Save Configuration URL.
    c. Program Run Time: Develop the Program Activity's runtime capabilities. This includes the integration points for which your **service** will interact with Oracle AMS. Within AMS, these integration points are known as the service's Invoke URL.
    d. Performance: Ensure that the app server can accommodate calls from Responsys, and that the app server does not exceed Responsys' throttling limit for inbound requests.
3. Build and deploy the app on the web server.

## App Developer Fundamentals

Here are some concepts within the App Developer Framework to familiarize yourself with before getting started.

### Apps

Applications are specialized programs that app developers have built to extend the functionality of an Oracle CX Marketing product. Apps are comprised of services, which represent the functionality of the app. An app can contain multiple different services. The apps we walkthrough are defined as apps that have been built to interact with Oracle CX Marketing on the CX App Developer Framework.

App developers host their applications externally outside of the framework. Apps have no executable code in Oracle CX Apps. The App Developer Framework only hosts app URLs, the app hosting is the App developers responsibility. Apps are developed to interact with a set of Oracle CX APIs.

An app is used for management, installation, and configuration. "Services" represent the specific functionality of the app. An app is comprised of services.

Learn more about Apps.

### Services

Services are self-contained components of apps that are used by marketers. There is currently one service type available for the App Developer Framework applicable to Responsys: *Action Services*. Building an action service in Responsys enables marketers to add new Program activities to their orchestration workflows.

The action service is the component of the app that marketers drag onto the canvas.

Most of the development effort will be focused on developing the actual service, as this is the component that contains the Design Time and Run Time elements of the app.

Learn more about Services.

## App Providers

App Providers are the top-level organizational unit for apps. A single App Provider can have multiple apps, and every app on the framework must be associated with a provider.

A provider can have multiple team members that have different roles. Team members of a provider can only manage apps for their provider.

Users who need support for an app will contact the associated App Provider for support. For this reason, we ask App Providers to supply company or support-specific information (such as a support website URL) when App Providers are being created within Oracle AMS.

In Oracle AMS, you must create an App Provider before you can start creating apps.

Learn more about App Providers.

## Oracle AMS

Oracle AMS stands for Oracle Application Management Service. Oracle AMS is the name of the cloud service that interacts with Oracle CX Marketing products, and apps developed for the App Developer Framework.

App providers use Oracle AMS to register their apps with the app framework with the Oracle CX Marketing products the app needs to interact with. Oracle CX Marketing customers then install the apps for their respective accounts.

## Products

A *product* is an entity within the app framework that represents an Oracle CX Marketing product. For example, Oracle Responsys is a product and Oracle Eloqua Sales Tools is a product. Apps are developed with a specific product in mind. Depending on the product you want to develop an app for, the services available for your app will differ.

### Tenants

A *tenant* is the term used to describe an account instance. An account instance can be tied to a marketing account or a developer account. Developers should note that each tenant has an associated `id` which is used when making requests within the app framework.

> **Example**: Let's say for example a marketing company named ABC Company downloaded an app called ZYX app. In this scenario, the tenant is ABC Company. For developer account instances, the tenant may be the account being used to develop apps.

### ID, Key, and Secret

These are details used by AMS for communicating between apps and product. They are sometimes required to be used as private claims for some calls between apps and product. Provided to both the product and app during registration.

There are two different sets of IDs, keys, and secrets:

a. **Application Keys**: Found in AMS under **Application Keys**. App ID, token key, and token secret.

b. **Provider Keys**: Found in AMS under **Provider Keys**. Provider ID, token key, and token secret.

### Record Definitions

Defines the configuration for the service as set by the marketer. You'll learn about record definitions later on, but for more information, see Record Definitions.

## GET STARTED IN ORACLE AMS

Before you start development, you must set up your Oracle AMS account by signing in and creating an App Provider. Creating a provider generates an App Provider Token Key and App Provider Secret which you'll need to authenticate with AMS. Then you can create an app and create a service.

> **Note**: Although you are creating your app now before development starts, you can edit your app details later if any changes have arose after development.

### App Registration and Installation

### Overview

The app provider needs to register the app first in AMS. App Providers are the top-level organizational unit for apps. A single App Provider can have multiple apps, and every app on the framework must be associated with a provider.

A provider can have multiple team members that have different roles. Team members of a provider can only manage apps for their provider. Users who need support for an app will contact the associated App Provider for support.

App Developers host their applications externally outside of the framework. Apps have no executable code in Oracle CX Apps. The App Developer Framework only hosts their app URLs within the framework, the actual app hosting is the App Developers responsibility. Apps are developed to interact with a set of Oracle CX Marketing APIs.

The App Manager (AMS) is used to manage all aspects of registering the app to be used for the framework and managing the app's use within Oracle CX Marketing products.

The App Manager enables providers to:

- Create & manage providers & applications under providers
- Brand their app by using company logos that will appear within the product
- Enter the URLs the App Framework should call out to when Oracle CX Marketing users interact with the App Provider's app for events like app installation, app configuration and so on
- Track app metrics

## Creating an App in the App Manager

Creating an app means "registering" the app that you developed to interact with the Oracle CX Marketing App Manager application. This process ensures that your app, and all of your app's endpoints are registered with the App Manager so that your app can communicate with the app framework.

The app creation process entails you entering the URLs locations for your app's endpoints, as well as specifying logos to visually differentiate your app.

After entering all the details and URLs for the app's endpoints, you must review the information and then continue to view the app details. You'll be directed to the App Details page to manage your app. From the manage apps console, the most important parts are:

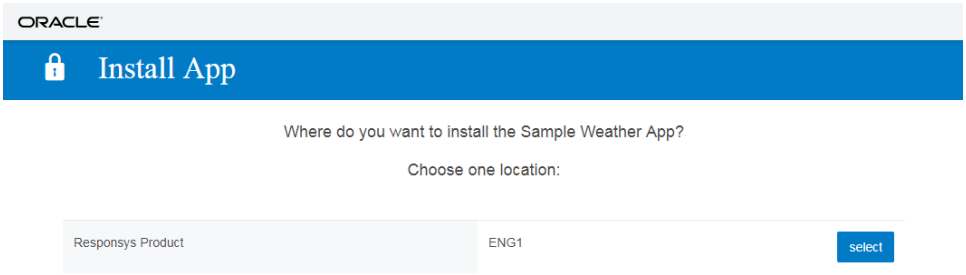
- Product Access: The Oracle CX Marketing products for which the app can be installed.
- Updated: The date the app was last updated.
- Install Link: The app's install URL. Provide this URL to users to install your app when your app is ready for publication. Until you change your app's publication status to CA or GA, the Install Link will display as " Install Link (testing only)".
- Application Keys: The application keys used to sign requests for your JSON Web Tokens for the app to interact with AMS and the Oracle CX Marketing product.

- ID (App UUID): A unique identifier to identify the app.
- Token Key: App Token Key. Your public key, which you will include in the JWT token. This token is tied to your app.
- Token Secret: App Token Secret. Your private key which you will use to generate a signature for your JWT token. On authorization, AMS will also use this secret to verify the sender of the JWT.

The app developer needs to copy the ID, Key, and Secret provided by AMS for communication between the app and AMS as well as app and product.

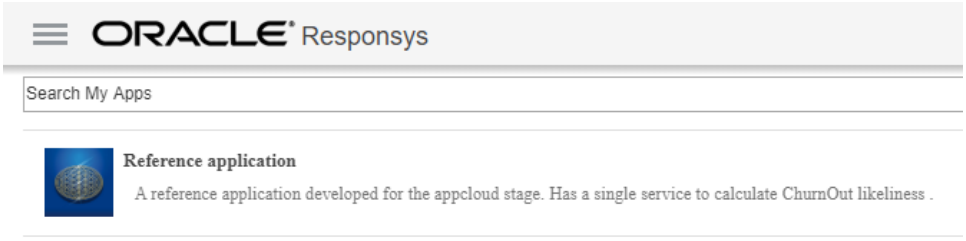
## App Installation by Marketer

After an app is created, an app **Install Link** is automatically created for that app. You can find that Install Link in the App Details page by navigating to an app. Once an app is ready to be published and distributed, this Install Link should be distributed to users. When a user navigates to an app's Install Link, the user is prompted to sign into the App Manager and then prompted through the installation. After selecting the POD, the user is redirected to the login page for the Oracle CX Marketing product.




After installation is complete, the App Manager calls out to the app's install URL endpoint. If the app replies with any 200 status, the app and all its services are made available to the account for which the app is installed.

Post installation, if the marketer navigates to the App Management page, they would see the list of apps installed for their account.

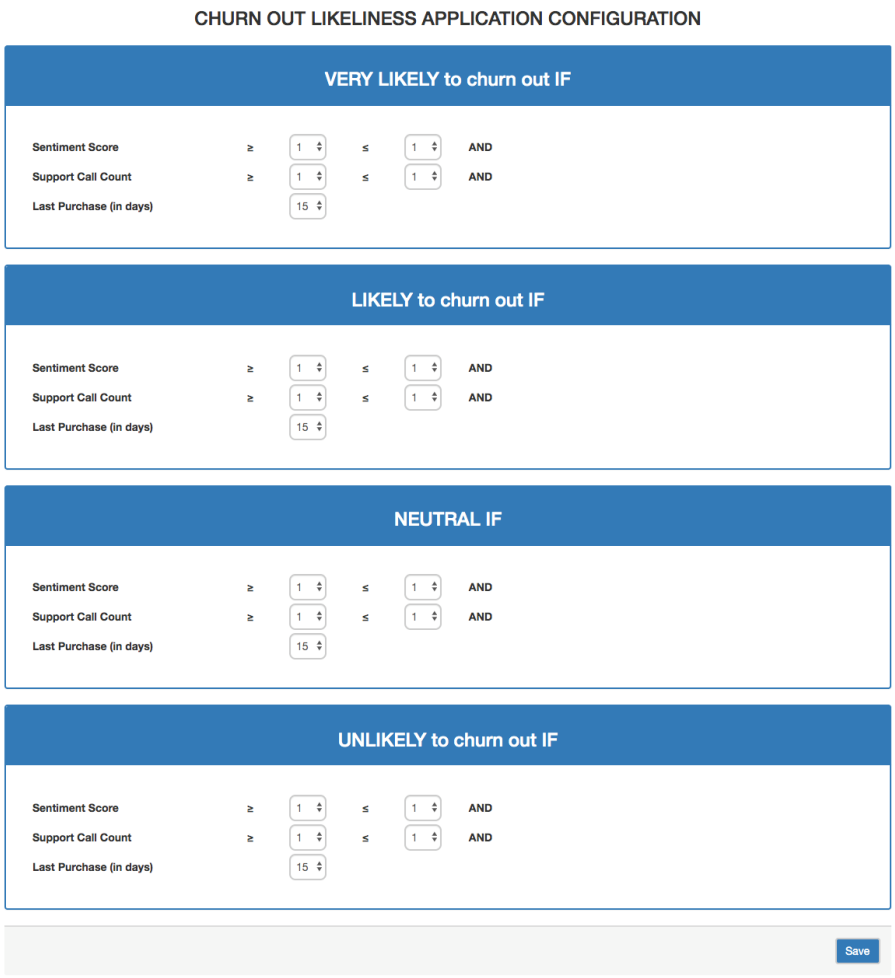



Clicking on a particular app would take the marketer to the app manager's iframe where the marketer can view all the app's services, choose to uninstall an app, or check for dependencies on the app in product's assets (example - Responsys Programs).

The marketer can configure the app post installation. App configuration is the step after an app has been installed and now needs to be configured for use. The two app endpoint URLs involved are the app **Configuration URL** and **Save Configuration URL**. App configuration generally refer to settings that the app developer wants the users to configure which may be applied on an app level i.e to all services provided by the app. But it depends entirely on the app's requirement and there is no specific pattern or rules to specify the fields that the app displays for configuration.

Example app configuration

Here is the app configuration for the reference app:




AMS calls the app's **Save Configuration URL** with the content of this page after the configuration is saved. After the configuration is saved successfully by the app, it responds with a `CONFIGURED` status back to AMS which then notifies the configuration UI about the status. If the status is `CONFIGURED`, a green label appears below the footer notifying the marketer that the configuration has been saved successfully.

This app asks users to set rules for assigning likeliness values of individual end recipients churning out of their product or offering. Based on some data on previous purchase history, support call counts, the sentiment scores and the rules defined in the App Configuration defined above the customers would get the value of likeliness of churning out for each enactment (i.e. recipient). The likeliness values would be one

amongst **LIKELY**, **VERY LIKELY**, **NEUTRAL**, and **UNLIKELY**. Based on these values, the enactments can then be taken along different paths in the program flow (different campaigns can be sent to customers based on the likeliness of churning out to ensure better customer engagement and retention).

## Signing In

To sign in to Oracle AMS, you'll need an Oracle account that has been approved for early access to the App Developer Framework. You can request access to the App Developer Framework for Oracle Responsys at http://alliances.eloqua.com/rsys_early_access.

### To sign in to Oracle AMS:

1. Navigate to https://ams.oraclecloud.com/ams.
2. If you have an Oracle account, click **Sign in with Oracle SSO**. If your Oracle account has been approved for early access, you will be redirected to Oracle'e SSO page to login with your Oracle SSO credentials. If your Oracle account has not been approved yet for early access, you will be redirected to our early access sign up form.

   If you do not have an Oracle account, click **Create New Account**.

After signing in with your Oracle SSO credentials, you will be redirected to Oracle AMS.

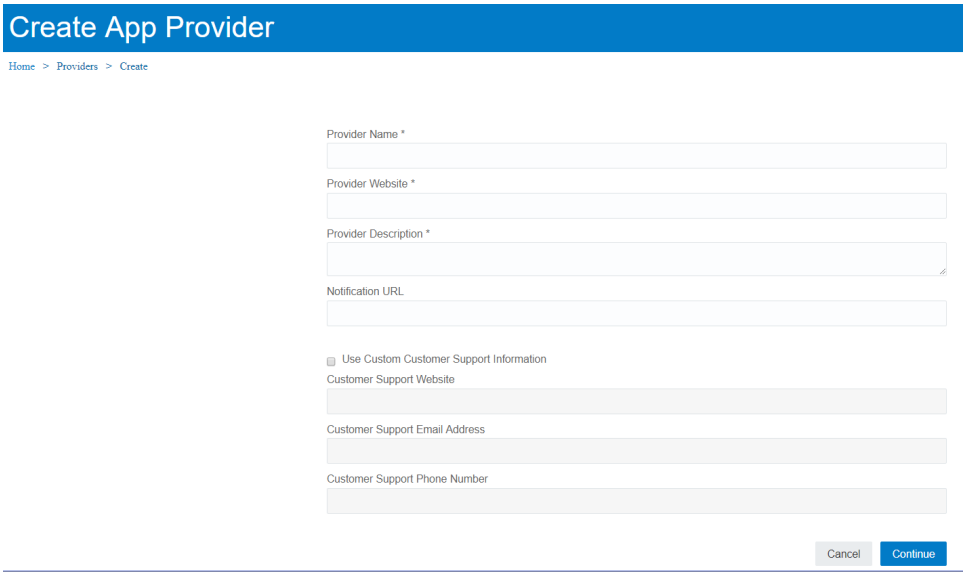
## Creating an App Provider

The first thing you'll need to do to get started in Oracle AMS is to create an App Provider. App Providers are the entity which store all of your company information such as your company website, phone number, and email. Your users using your app will use this information to contact you if they need support.

### To create an App Provider:

1. Click **Create Provider**.

2. Enter your company information into the text boxes:

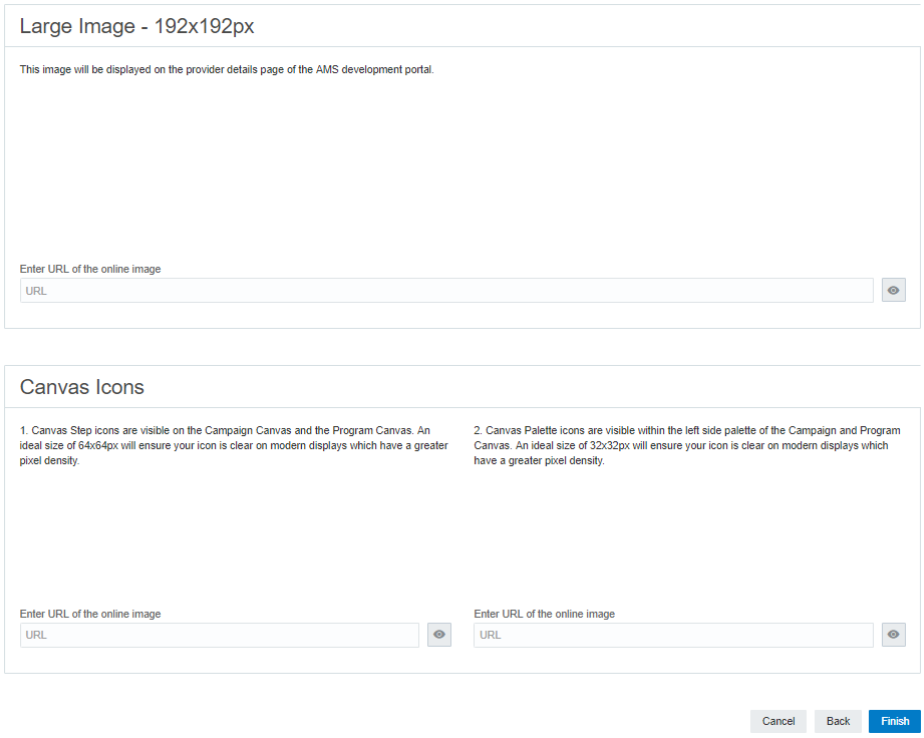



   - **Provider Name**: Your company name, this will also be your App Provider name.
   - **Provider Website**: Your company website URL. This is optional.
   - **Provider Description**: A description of your company.
   - **Notification URL**: Specify a URL endpoint where AMS will send you push notifications if your app's status changes. This is optional.
   - **Use Custom Customer Support Information**: Select to specify customer support information.
   - **Customer Support Website**: The customer support website URL.
   - **Customer Support Phone**: A phone number where you can be contacted if there are problems with your app.
   - **Customer Support Email**: An email address users should use to contact you if there are issues with your app.

3. Click **Continue**.

4. Enter the image URLs to be displayed as your App Provider logo in 3 different sizes. Click the

**Preview** icon to verify your logos display correctly.




5. Click **Finish**.

The details of your app provider are displayed, including your app provider token key and secret. These tokens are used to authenticate with AMS, for more information see Authentication. When you are finished, click **Continue to Provider Details**.

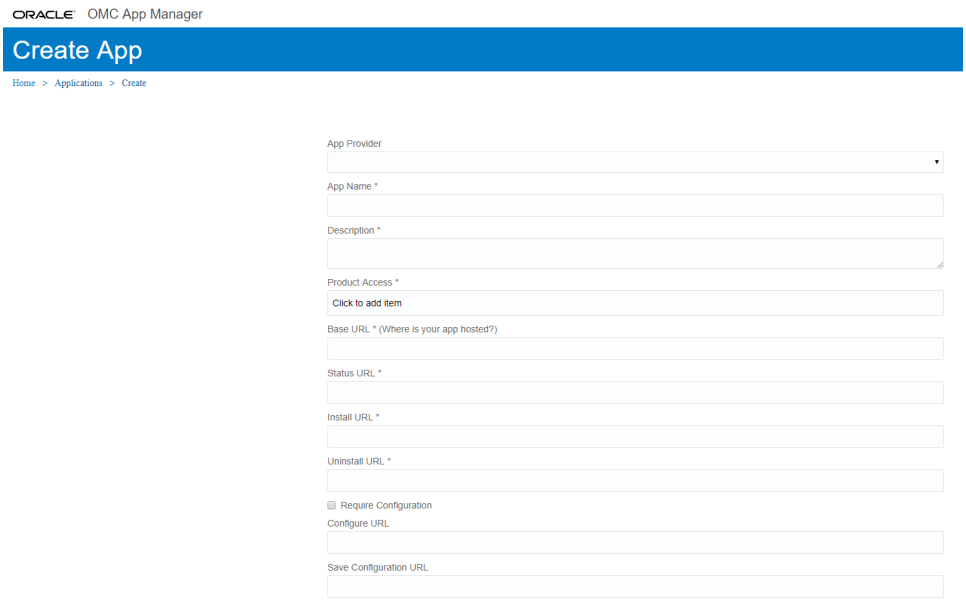
## Creating Apps

Creating an app means "registering" the app that you develop outside of AMS with the Oracle AMS application. This process ensures that your app, and all of your app's endpoints are registered with AMS so that your app can communicate with AMS.

You'll need to enter the URLs locations for your app's endpoints, as well as specify logos to differentiate your app visually.

### To create an app:

1. From the home page, click **Create a New App**.
2. Select the provider for which you want to create the app. See Creating an App Provider if you do not have any providers yet.
3. Enter your app details.




The following table provides information about the fields used to create apps.

| Field | Required | Max Length | Description |
|---|---|---|---|
| App Name | Y | 256 | Name of the app. |
| Description | Y | 2048 | Description of the app. |
| Product Access | Y | | Select the Oracle CX Marketing product your app should be available to. |
| Base URL | Y | 1024 | The location where your app is hosted. This is the base URL for all URLs AMS will call out to. Must be a fully qualified URL including the `https://`. |

| Field | Required | Max Length | Description |
| --- | --- | --- | --- |
| | | | **Example**: The syntax is https://awesomeapp.com. |
| Status URL | Y | 1024 | The endpoint to call to check the app's status. URL can be relative or absolute.<br><br>**Example**: If the Base URL is https://awsomeapp.com, a relative URL could resemble https://awesomeapp.com/status. |
| Install URL | Y | 1024 | The endpoint to call upon app installation. URL can be relative or absolute. |
| Uninstall URL | Y | 1024 | The endpoint to call to uninstall the app. URL can be relative or absolute. |
| Require Configuration | N | NA | Whether or not the app requires configuration before it can be used. Not currently used. |
| Configure URL | Y | 1024 | The endpoint called when the Configure button in the Oracle CX Marketing product is clicked. This endpoint should return HTML. |
| Save Configuration URL | N | 1024 | The endpoint to call when a marketer saves their configuration changes in the app configuration user interface. URL can be relative or absolute. |

4. Click **Continue**.

5. Enter the URL locations for your app's logos. Three sizes are required.

**Important**: Your logos must be linked images. Base64 encoded data is not supported.

- Large: Displayed in Responsys in the app catalog. This image is 192x192 pixels when displayed. The recommended size is 192x192 pixels. Must be absolute URL.
- Canvas Icons (Medium): Displayed in Responsys when the app is dragged onto the canvas. This image is 32x32 pixels when displayed. The recommended size is 32x32 pixels. Must be absolute URL.
- Canvas Icons (Small): Displayed in Responsys when selecting the app from the Program palette. This image is 16x16 pixels when displayed. The recommended size is 16x16 pixels. Must be absolute URL.

Click **Preview** to ensure the images are displayed properly.

6. Click **Finish**.

Your newly created app is displayed. Review the information and then click **Continue to App Details**. You'll be directed to the App Details page to manage your app.

## Creating Action Services

Create a service to power your app. Services are self-contained components of apps that are used by marketers. Creating a service is essentially registering the service you developed with the App Framework by entering your service's URL endpoints.

### To create an Action service:

1. From the AMS home page, select the app you want to create a service for.
2. Click **Add a Service**.
3. Enter your service details and then select **Action** for Service Type.

- **Service Name**: Enter the name of your service. The maximum length is 256 characters.
- **Service Description**: Enter a description of your service. The maximum length is 2048 characters.
- **Service Type**: Select **Action**.
- Service URLs: Enter your service's URLs endpoints. All URLs have a maximum length of 1024 characters unless specified otherwise. All endpoint URLs can be absolute URLs, or be relative URLs to the Base URL unless indicated otherwise. For example, if your Base URL is `https://awesomeapp.com` a relative Create URL would be `create`. When the Create URL is called, the full URL will be `https://awesomeapp.com/create`.
    - **Base URL**: URL location where your app is hosted. The Base URL is the root of your app's web hosting address that stays consistent. Must be a fully qualified URL including the `https://`. For example: `https://awesomeapp.com`.
    - **Configure URL**: URL endpoint for your service that is called when users configure the service. This endpoint should return HTML as an HTTP GET request.
    - **Create URL**: URL endpoint for your service that will be called when services are created.
    - **Delete URL**: URL endpoint for your service that will be called when services are deleted.
    - **Invoke URL**: URL endpoint for your service to call when your app is invoked. App invocation is the process where the App Framework communicates with your app to perform an action.
    - **Status URL**: URL endpoint to call to check the service's status.
    - **Save Configuration URL**: URL endpoint to call when your service's configuration is saved.
    - **Max Batch Size**: The maximum number of records to push per HTTP request. If the number of members exceeds the Max Batch Size, an invoke without data will occur.

4. Click **Continue**.
5. Select the assets your service will support.

   For Responsys:

- **Program**: Enables your service to be used in a Program canvas. Action services are used in programs as program stages.
- **Dashboard**: Enables your service to be used in the Dashboard.

6. Click **Continue**.
7. Enter the URL locations for your service's logos. Three sizes are required.

> **Important**: Your logos must be linked images. Base64 encoded data is not supported.

- Large: Displayed in Responsys in the app catalog. This image is 192x192 pixels when displayed. The recommended size is 192x192 pixels. Must be absolute URL.
- Canvas Icons (Medium): Displayed in Responsys when the app is dragged onto the canvas. This image is 32x32 pixels when displayed. The recommended size is 32x32 pixels. Must be absolute URL.
- Canvas Icons (Small): Displayed in Responsys when selecting the app from the Program palette. This image is 16x16 pixels when displayed. The recommended size is 16x16 pixels. Must be absolute URL.

Click **Preview** to ensure the images are displayed properly.

8. Click **Finish**.

Your service details will be displayed. You'll be sent an email confirmation for your new service.

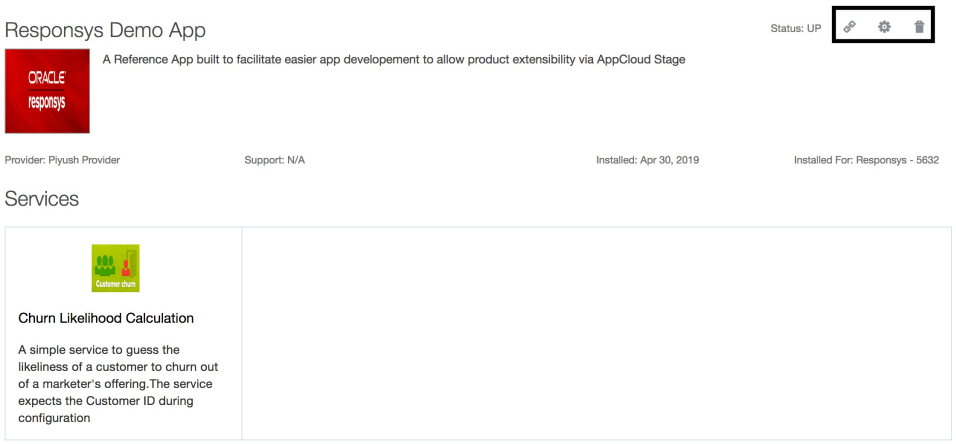
## COMMUNICATION

AMS uses the app's endpoint URLs to notify about different sorts of events and user activities (like installation, instance creation, deletion etc). Similarly, when a program is published in Responsys (a program can have one or more App Stages with each pointing to separate instances of one or more services), the **Invocation URL** of that service is provided to the product by AMS to initiate the service's action.

### App and AMS Communication

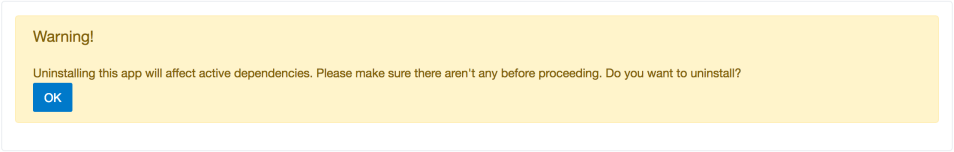
AMS provides a rich set of APIs for application providers for information regarding the app provider and other app installation details. The Reference App doesn't call any of those endpoints. For this app, all communication is from AMS to the App. AMS calls out to the URLs specified during application registration. The endpoints which AMS calls out to on an app level are:

- **Install URL**: Called to notify the app about an installation request (typically from a marketer). The installation succeeds for that account/tenant only if the app responds with a 200 status. The details of the payload and the Authorization tokens are discussed later.
- **Configuration URL**: This endpoint is called out to render the configuration UI after an app installation. Typically, a user logs into their product account and goes to the App Management section. Clicking on that would take the marketer to the catalog of installed apps. Clicking on a particular installed app would open up the AMS console which has the actual app management buttons (deleting, configuring, and dependencies).




Clicking the Gear icon (the middle icon in the image above) triggers a call to the app's configuration endpoint.

- **Save Configuration URL**: Upon saving the configuration settings in the UI, the app's UI does a postMessage to the parent iframe (in this case AMS) which takes the payload and does a POST call to the app's Save Configuration URL. The Response, if successful, has a "CONFIGURED" status, which AMS again notifies to the app's UI. The application can then render the success message based on this flag.
- **Uninstall URL**: This endpoint is called by AMS when a marketer tries to uninstall an App.

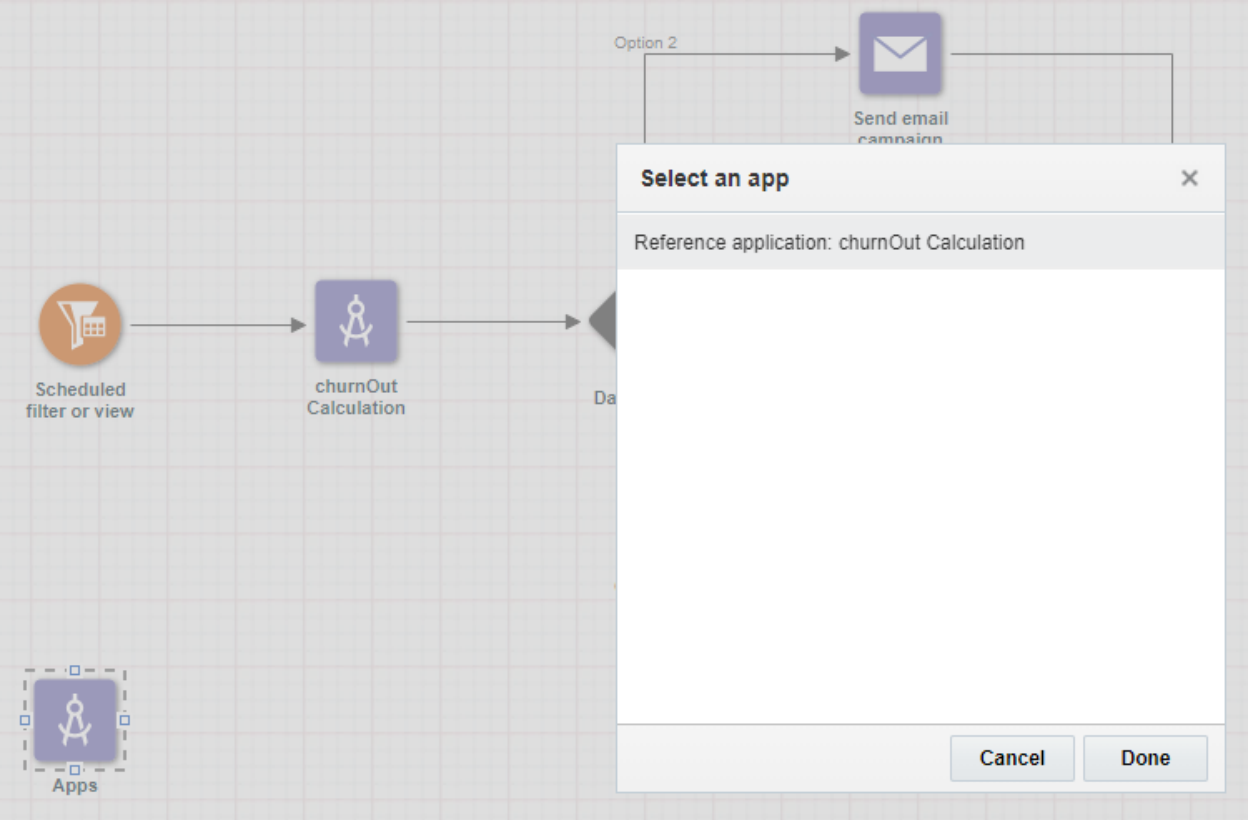



After the user clicks **OK**, AMS calls the App's Uninstall endpoint. But in this case, an uninstall will succeed regardless of what the app sends back the rationale being that we shouldn't prevent uninstalls even if the app fails to execute on an uninstall.

- **Status URL**: This endpoint is expected to be invoked by AMS or a product to inquire application status. The app sends back the service names and any associated warnings or errors.

The URLs mentioned above are on the **app** level. URLs are registered for individual services as well. Service level URLs called by AMS include:

- **Create instance URL**: This is called by AMS when a user chooses a specific service in the Program canvas. Instance IDs are generated by AMS. The detailed payloads and authorization are discussed later.



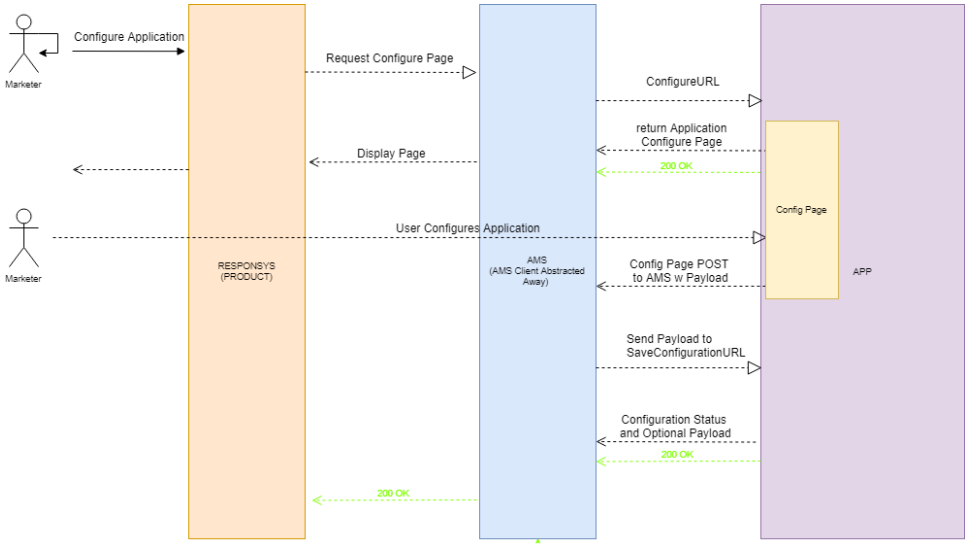

Clicking the service and then **Done** triggers a create instance call from the product to AMS and then from AMS to the app. Instance ID is generated by the AMS.

- **Delete instance URL**: Similar to the uninstall App URL . The notification though, doesn't come immediately to the app when an user removes a service instance from the program canvas. The product notifies the instance IDs of the removed instances on a daily basis to AMS which then calls the app's delete instance endpoint to notify.

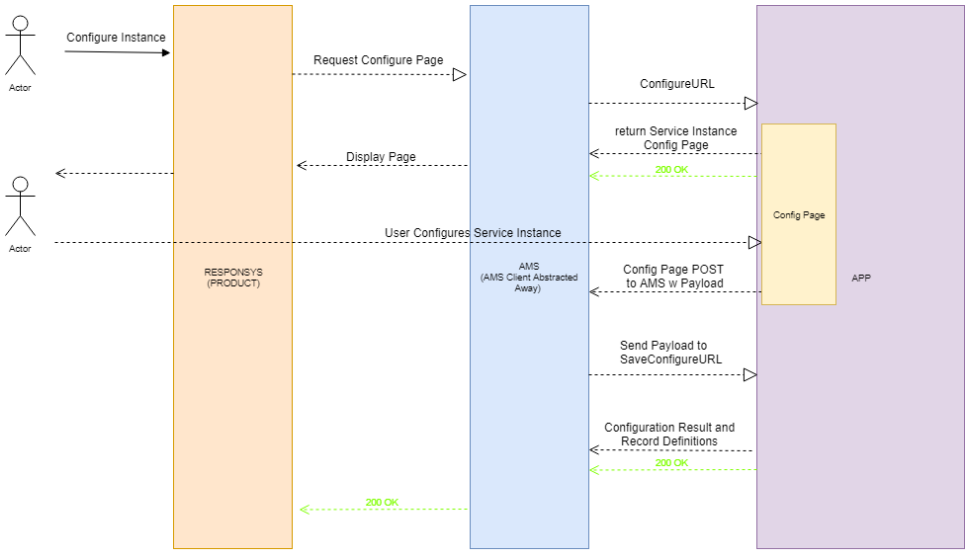
- **Configuration URL**: Similar to the app's configuration URL, but on the service level. The product provides an iframe, which AMS creates its own iframe to load the App service's configuration page.
- **Save Service Configuration URL**: After the user saves the service's configuration, the UI postMessages the payload to the parent iframe (AMS ) which then posts the configuration payload to the service's save configuration URL. The response contains a `status` , which would be `CONFIGURED` , if the configuration is saved successfully. AMS postMessages this back to the UI which can then render a success label based on this flag.

The payloads, authentication and details of the service will be explained in the Reference App section. The communication flow in the case of app install configuration and service configuration are shown below:

## App Configuration Flow



## Service Configuration Flow




Refer to Service Lifecycle for more information.

### App and Product Communication

Oracle CX Marketing products provide a rich set of APIs which apps can call to get more information on the customer's profile lists, campaigns, events, PETs etc. These web services don't depend on any App cloud stage or service instance to be created.

We limit our discussion to APIs which are associated only with the Apps Stage in the product.

The Product can call the following Application URLs:

- **Application Status URL**: This can be called by the product to check the App's status. The application responds with service names, app name, and a list of warnings or errors if any (For example if a service ID is missing).
- **Service Status URL**: Similar to the Application Status URL, but only on an individual service level. The product can query if all service names and IDs have been set, and whether any errors or warning are present.

- **Service Invocation URL**: This is the most important URL from a product's perspective. This URL is provided by AMS to the product during runtime (After a program is successfully published and the flow reaches the Apps stage). The product calls this endpoint with some additional details like the dataset size, install ID, and instance ID, as well as URLs for the app to download (in case of a large batch), upload processed data and the completion callback.

The App can call the following product URLs. All these are provided by the Product during invocation.

- **Product Export Endpoint**: This won't be provided (i.e. null in the invocation payload) in case of small batches. Small batches are decided based on the minimum of product's max push size and the app's max batch size and the record count. If the group size is small, the dataset will be passed along the invocation payload. In case of a large batch, the App would download the dataset in batches by setting offsets and limit parameters as per the product's maximum push size.

- **Product Import Endpoint**: Provided by the product during invocation to enable the app to send back the processed values (called ETVs) for each enactment in the batch back to the product. Three fields are mandatory to be sent back to the product. These are
  - `appcloud_row_correlation_id`
  - `appcloud_row_status`
  - `appcloud_row_errormessage`

  The correlation ID is present in the input parameter (when dataSet is downloaded). Status can be either **success** or **failure** and the error message can be **null** in case of successful processing or an exception reason message in case of failure.

- **Completion Callback Endpoint**: This is provided by the product to the app to allow the app to notify the product about the successful processing of the entire dataset explicitly. Calling this event would raise an event on the product to release the entire group to the next stage in the program orchestration. Otherwise, the enactments would be passed on to the next stage based on the default stage expiration setting.

Details on the app and product's data exchange payloads, authentication is discussed in the next section.

## DEVELOPING RESPONSYS APPS

The following sections explain what you'll need to know to start developing apps for Responsys.

1. Responsys concepts you'll need to know.
    - Responsys Concepts for App Developers
2. The stages of an app's lifecycle in Responsys.
    - Responsys App Installation and Setup Overview
    - Responsys App Configuration Flow
    - Program Design Time Overview
    - Responsys Action App Design Time Flow
    - Program Runtime Overview
3. The URL endpoints you must develop.
    - Reference App Details

### Responsys Concepts for App Developers

If you are not familiar with Responsys and its Program orchestration model, review this section before you begin developing for Responsys. If you are already familar with Responsys, you can skip to Responsys App Installation and Setup Overview.

#### Program Workflow and Activity Stages

A Responsys **Program workflow** enables marketers to coordinate complex marketing campaigns across different channels in a simple, visual manner. On the Program canvas, marketers create workflows that provide highly personalized digital experiences for their customers. For example, the marketer may wish to send customers a message by email or SMS, depending on how the customer has opted in to receive communication from the company.

A Program Workflow comprises Events, Activities, and Switches, each of which are represented by a workflow stage icon available from the Program palette.

- **Event**: Something that happens, such as a customer completing a purchase. Displayed on the Program canvas as orange circle icons.
- **Activity**: Something that gets done, such as an email or SMS message being sent, or an app action sending or receiving enactment data. Displayed on the Program canvas as purple square icons.
- **Switch**: A decision point in the process, such as determining a customer's Opt-in status in order to send them a campaign through the proper channel. Displayed on the Program canvas as grey diamond icons.

### Enactments

An enactment is a representation of a contact within a Responsys Program workflow. For example, when a customer makes a purchase, the customer's contact information is pulled into the program and is represented as an enactment. This enactment moves through the Program workflow. Enactments can be considered in-memory and are not committed to the Responsys database until explicitly instructed. A contact can have multiple enactments within a single orchestration workflow. Though each enactment represents the same contact, each enactment is unique within the workflow.

### Entry Tracking Variables

Entry Tracking Variables (sometimes referred to as *Program Variables* or *Enactment Variables*) enable marketers to update information about an enactment while it is in the Program workflow. Entry Tracking Variables can be used in workflows so that Data Switches can make decisions and route the enactments accordingly. Entry Tracking Variables are local to the specific Program canvas. Another program cannot access the Entry Tracking Variables for an enactment in a different workflow. Entry Tracking Variables are local to the specific enactment in the program workflow. There can be

multiple enactments for the same contact, but each will have its own set of values for their Entry Tracking Variables.

These variables are different from contact record fields in the Profile Table or the Profile Extension Table (PET), because Program Variables only exist in the program workflow. An Entry Tracking Variable is defined by the marketer when the marketer designs the program; for example, the marketer may define a Product SKU variable from a cart abandonment. It is populated at program runtime when the enactment enters the workflow. Once an enactment has completed the workflow, the Program Variable is thrown away.

## Responsys App Installation and Setup Overview

To install apps, marketers navigate to an install URL. The install URL will prompt the app installation process. App Developers do not need to develop the installation process, this process is handled by Oracle AMS. After a developer has created their app within Oracle AMS, an install URL is created automatically.

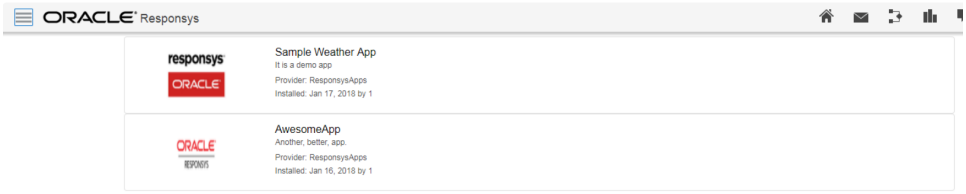
The install URL directs marketers to a page where they will be asked to provide the install location of the app.

After a marketer has installed an app, it can be seen in the App Manager in the Account Settings section.

### To view all installed apps:

1. In Responsys, navigate to **Account** > **Manage Apps**.

Marketers will see all of the apps they have installed. In the example below, there are two apps installed.

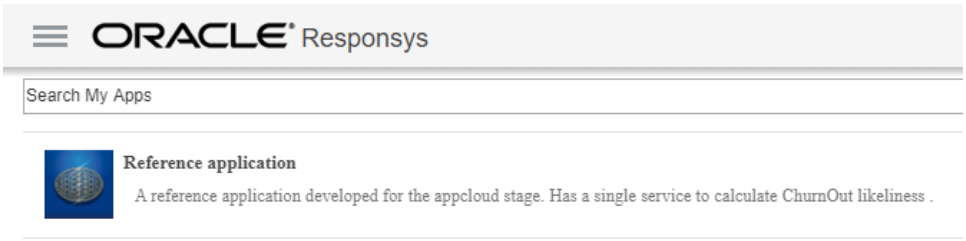



## Responsys App Configuration Flow

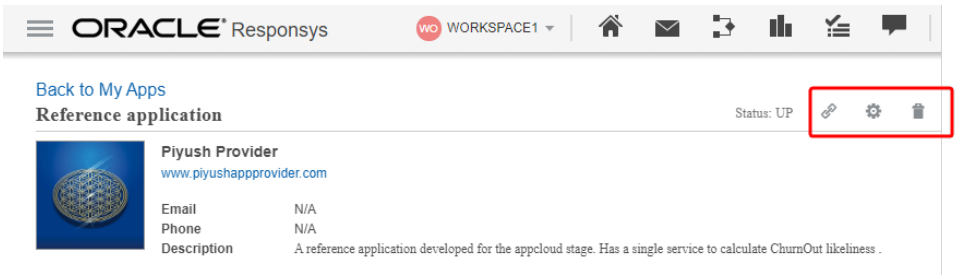
After an app has been installed, the next step for the marketer is to configure that app. App developers must develop the app configuration user experience by ensuring the **Configuration URL** endpoint for their app returns HTML within an iframe. Next, the app developer must ensure the **Save Configuration URL** saves the marketer's configuration changes. App configuration comprises two interactions: view configuration and save configuration.

### Viewing App Configuration

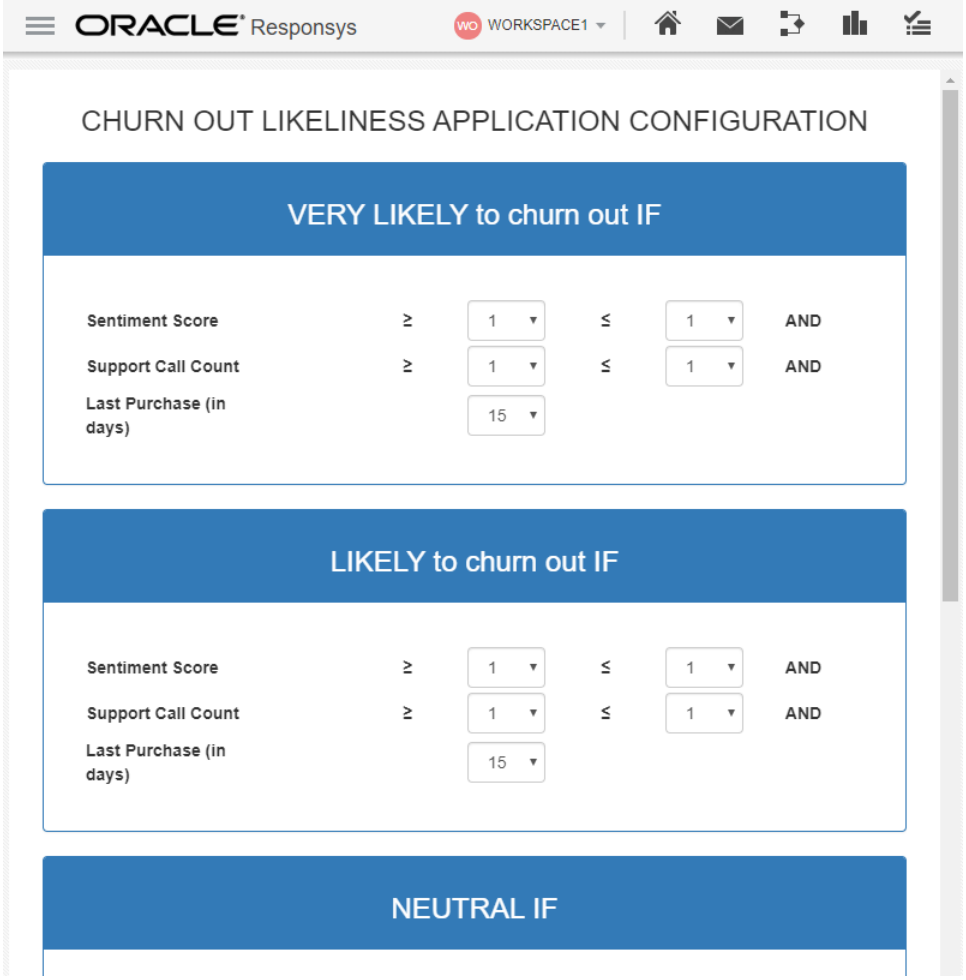
After apps are installed, they will appear in Responsys under **Account** > **App management**.




Selecting an app will open the app's details. Marketers will then click the **Configure App** button to configure the app for use.

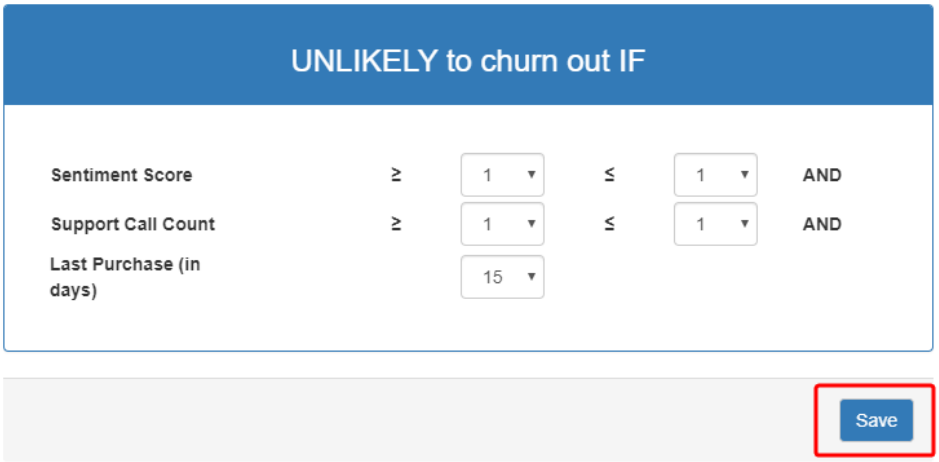


When the **Configure App** button is clicked, the **Configure URL** you entered when creating your app is called. The configuration options for the app appear.




## Saving App Configuration

Your app configuration UI should provide the marketer the option to save their app configuration changes.



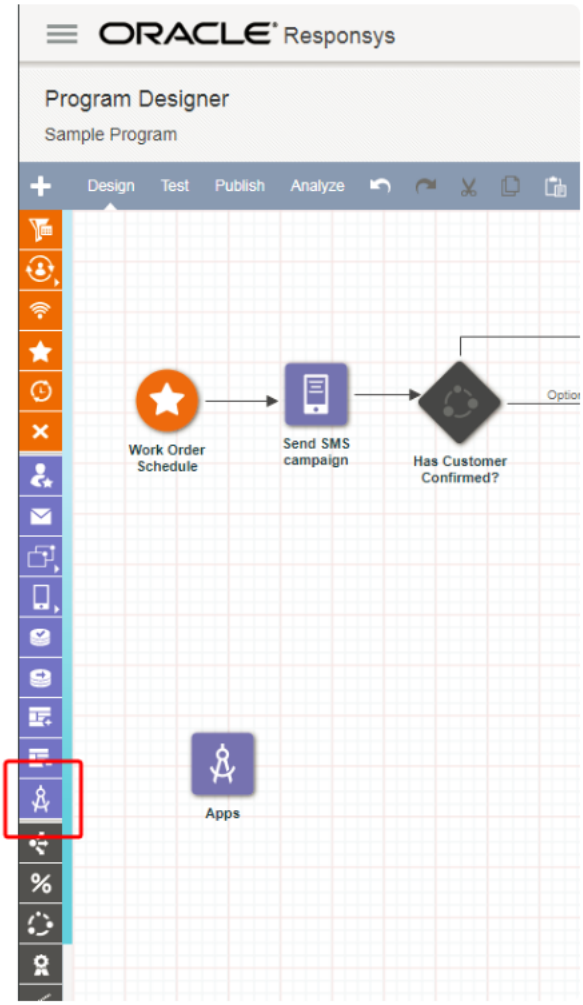

Upon clicking **Save**, the app's **Save Configuration URL** is called.

## Program Design Time Overview

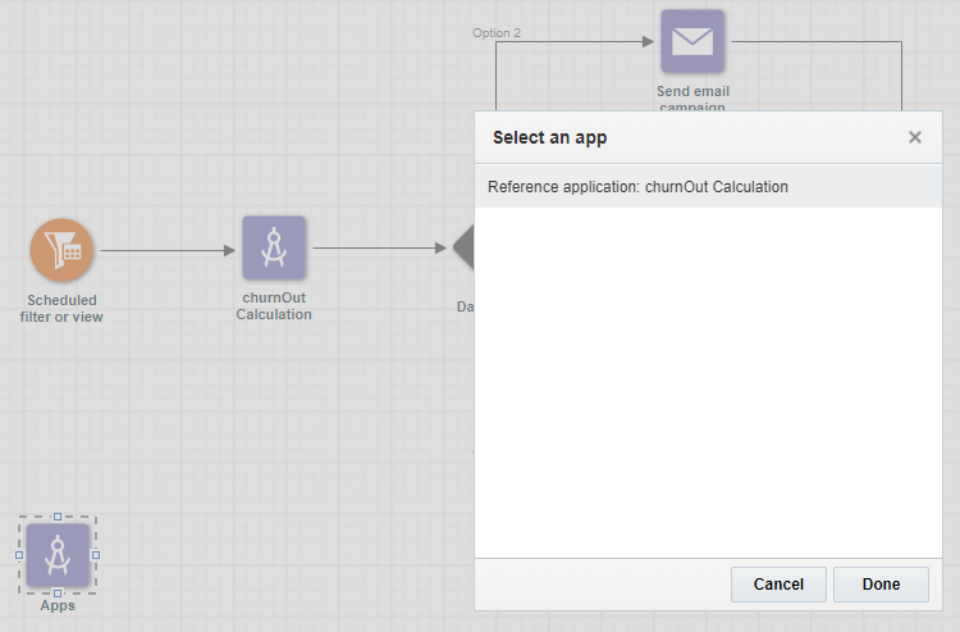
The Design Time phase consists of a marketer designing their program workflow with an app stage. After apps are installed, marketers begin to design their Program workflow. This involves configuring the activity stage for the Program.

In a Program workflow, a marketer adds your app's service as an activity stage on the Program canvas, configures the service with the values to send and to receive. Let's walkthrough how marketers design programs.

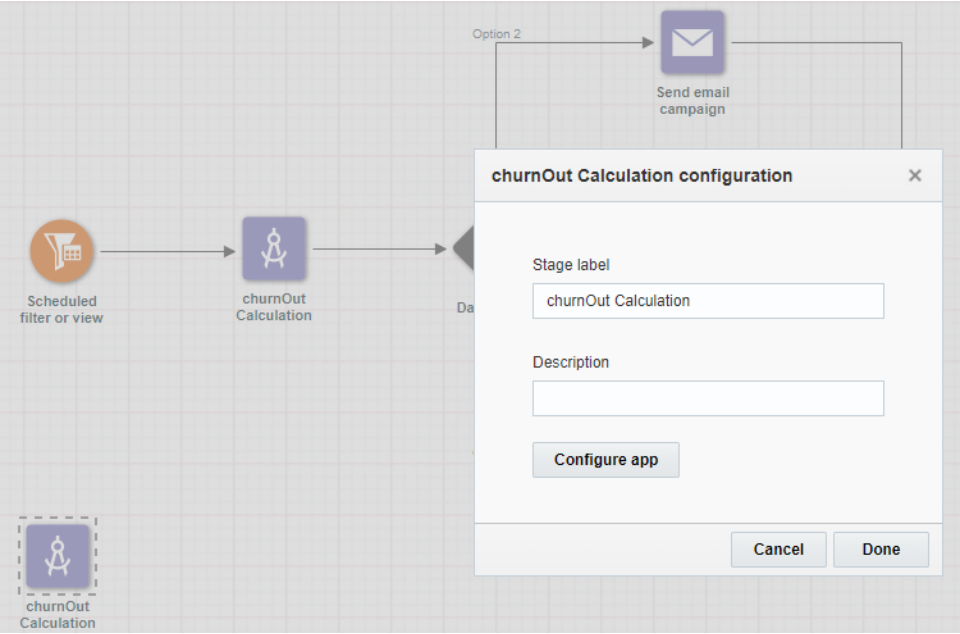
From the program palette, marketers drag an app onto a canvas.




The marketer will then double-click the stage to select an app. The marketer is shown a list of all installed apps and their available services.




After a service is selected, the marketer enters a Stage label and Description to display on the canvas.




If the service requires configuration, the marketer will click Configure app to start configuration. The configuration user interface that appears will differ based on the app.

Refer to Responsys Action App Design Time Flow for more information on how to develop this experience for users.

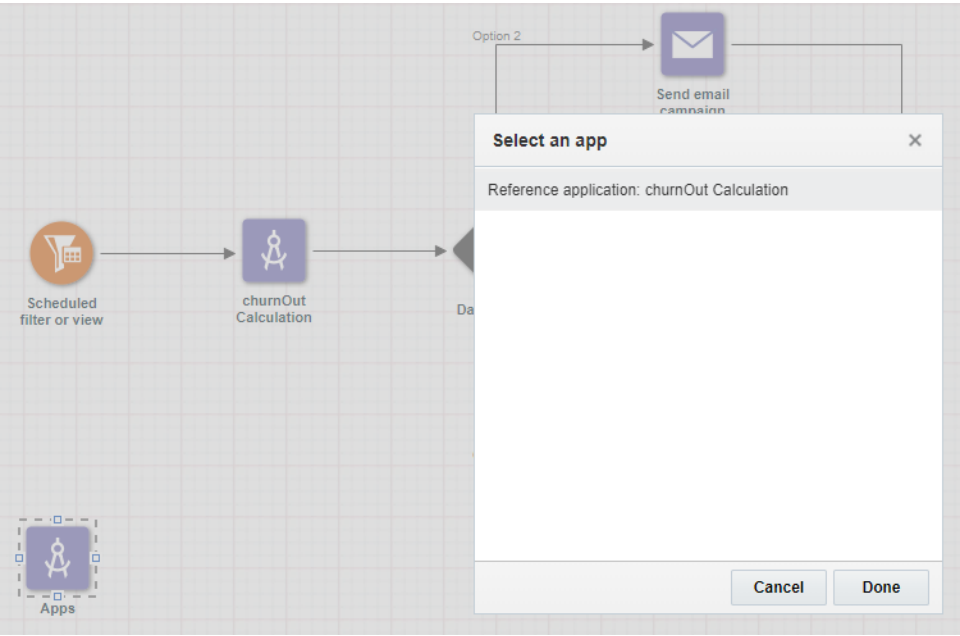
## Responsys Action App Design Time Flow

After apps are configured, the next step is to create a service within a program. Let's walkthrough what users will see during the Design Time phase so that you can develop the experience accordingly. The Design Time phase consists of:

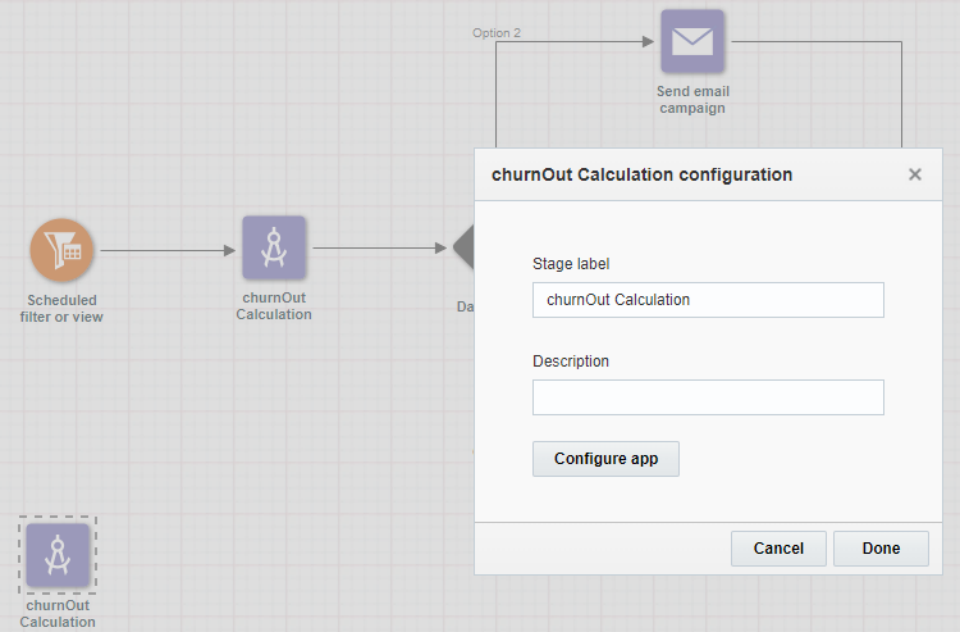
- Marketer creates a service within a Program
- Marketer configures a service
- Marketer saves their configuration changes

### Service creation

When marketers drag an App stage onto a Program canvas, Responsys will ask the marketer to select a service.




After the marketer selects a service, they must enter details about how it will appear on the canvas.



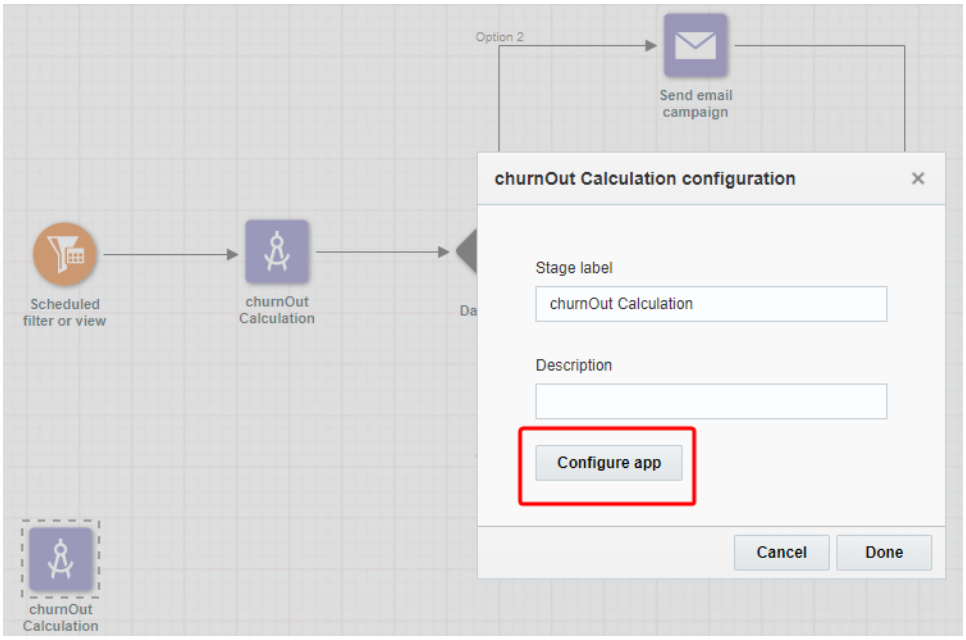

After these details are entered and the marketer clicks **Done**, the service is created on the canvas. This is when your service is called to be created.

At this point, Responsys calls AMS using the service's **Create URL** to create the service on that canvas.

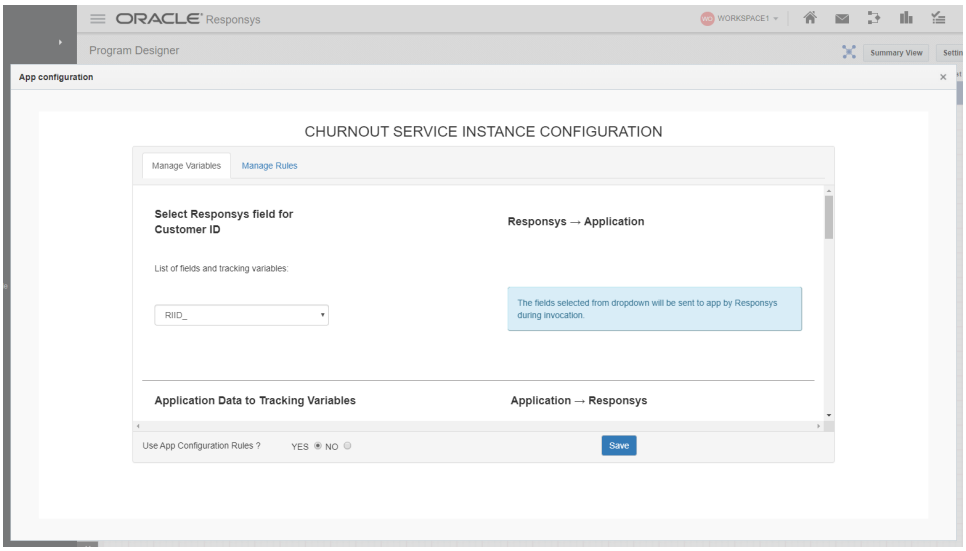
Service view configuration

After the service has been created within a Program, marketers must be able to view configuration options for the service. As a developer you must ensure the configuration user experience enables marketers to configure the service appropriately for your service. This interaction is where the service's **Configuration URL** is called.

Marketer clicks **Configure app** to configure a service.




Responsys returns a page that contains an iframe.




The iframe calls a public AMS endpoint requesting the service configuration page.

### Service save configuration

After the marketer has completed their service configuration, they must save the configuration, this is where the interaction with the service's **Save Configuration URL** happens.

### Program Runtime Overview

The Program Run Time phase occurs after the program is published.

When an enactment reaches the app's activity stage, Responsys sends the personalized payload using HTTP POST. Enactment is released and continues in Program.

After a service has been configured, the next step for the marketer is to publish the Program. Let's walkthrough how to develop a service during the Program Run Time phase. The Run Time phase consists of:

- Marketer publishes Program
- Enactments reach the app's activity stage, Responsys then sends the personalized payload using HTTP POST to the app. Enactment is released and continues in Program.

### Reference App Details

## App Endpoints

A brief overview on the App and service URLs called by AMS and the product has been discussed above in the Communication section. Here we discuss the request payloads, JWTs required for authorization, and the responses sent by the App.

The endpoints exposed on an app level are:

### Install URL

Called by AMS to notify about a new installation request from an Oracle CX Marketing customer.

**JWT for installation**

```
{
  "iss": "4701447f-20f1-4a25-875f-52e36d6a93ae",
       -- The App's Token Key.
  "sub": "4af33f82-25c3-44ec-8594-7c39ef2079d4",
       -- Set to the UUID of the product.
```

```
  "aud": "4701447f-20f1-4a25-875f-52e36d6a93ae",
       -- The App's Token Key.
  "exp": 1539915991,
       -- Set to 60 seconds after the JWT was created.
  "iat": 1539915932,
       -- Set to the time at the instant the JWT is being
created.
  "jti": "88062cca-9b58-4391-a713-4817662526af",
       -- Set to a random UUID
  "o.a.p.ctenantId": "37807",
       -- Set to the id of the tenant as identified by the
product
  "o.a.p.cproductUUID": "4af33f82-25c3-44ec-8594-
7c39ef2079d4",     -- Set to the UUID of the product.
  "o.a.p.cproductName": "Mock Product",
       -- Set to the name of the product.
  "o.a.p.csourceRequest": "4af33f82-25c3-44ec-8594-
7c39ef2079d4",   -- Set to the UUID of the product.
  "o.a.p.cdestinationId": "c1fce274-09c5-457a-a916-
141de834b4cd",   -- Set to the App's UUID.
  "o.a.p.cdestinationUrl":
"https://rsp.eng1.responsys.net/ReferenceApp/rest/1.0/app/in
stall"
}
```

The JWT is signed with the app's token secret. (Also called app secret)

The payload sent by AMS for the installation request is:

#### Request payload for installation

```
{
  "applicationInstall": {
    "uuid": "edcc4c92-2842-46c2-88b9-138f51319aeb",
    "deleted": 0,
    "secret": null,
    "application": {
      "baseUrl":
"https://piyushprovider.localhost.run/ReferenceApp",
      "configureUrl": "/app/configuration.html",
      "description": "A Reference App built to facilitate
```

```
easier app development to allow product extensibility via
AppCloud Stage",
      "deleted": 0,
      "installUrl": "/rest/1.0/app/install",
      "largeLogo": "http://partners.salecycle.com/wp-
content/uploads/2017/02/partner-responsys.jpg",
      "mediumLogo": "http://partners.salecycle.com/wp-
content/uploads/2017/02/partner-responsys.jpg",
      "name": "Responsys Demo App",
      "saveConfigurationUrl":
"/rest/1.0/app/install/configuration",
      "smallLogo": "http://partners.salecycle.com/wp-
content/uploads/2017/02/partner-responsys.jpg",
      "status": "UP",
      "publicationStatus": "GENERAL_AVAILABILITY",
      "statusMessage": null,
      "statusUrl": "/rest/1.0/app/status",
      "uninstallUrl": "/rest/1.0/app/uninstall",
      "uuid": "ff28d366-44ff-4830-9bc3-8b8466e2d6a8",
      "applicationProductApi": [],
      "providerUuid": "da6dad99-4d2f-436f-9418-32a4f4abc2da"
    },
    "providerUuid": "da6dad99-4d2f-436f-9418-32a4f4abc2da",
    "providerName": "Piyush Provider",
    "productManaged": false,
    "tenantRestUrl": "https://api-004.qa1.responsys.net",
    "status": "UP"
  },
  "secret": "91f459e7-562f-402b-a838-3d6736167f19-4d6dd0af-
b0f3-4218-9b04-2398ab901930"
}
```


The installation succeeds if **any 200 status code** response is sent by the app. The app can use the `tenantRestUrl` to make calls to Responsys APIs.

Configuration URL

For the sample application, we bypass the JWT validation and the request payloads since it is a simple HTML page. Though, Application developers can use the Model

View Controller (MVC) design pattern to return the HTML (JSP) by using the installation ID present in the request payload to present customized configuration pages as well.

Save Configuration URL

Invoked by AMS when a user saves the configuration in Responsys's app management interface.

**JWT for saving app configuration**

```
{
  "iss": "f197ac33-162c-4176-b692-375d8328b9af",
                                    -- App's Token Key
  "sub": "AMS-INSTALL-CONFIGURATION",
                                    -- A subject assigned by
AMS
  "aud": "f197ac33-162c-4176-b692-375d8328b9af",
                                    -- App's Token key.
  "exp": 1557000002,
                                    -- Set to 60 seconds
after the JWT creation.
  "iat": 1556999942,
                                    -- The instant at which
the JWT is created.
  "jti": "776e63d0-9bb0-414f-b8b9-ba8d84b46fd6",
                                    -- Set to a random UUID
  "o.a.p.ctenantId": "5632",
                                    -- Set to the tenant's id
as identified by the product.
  "o.a.p.capplicationUUID": "ff28d366-44ff-4830-9bc3-
8b8466e2d6a8",                              -- Set to the
App's token UUID.
  "o.a.p.cinstallUUID": "9f4cc19f-f37e-4aa0-9b45-
8453e8423589",                                  -- Set to the
install UUID corresponding to the config.
  "o.a.p.cproductUUID": "f5c93036-1df1-4f05-8c9d-
17569e63c016",                                  -- Set to the
product's UUID.
  "o.a.p.csourceRequest": "AMS-INSTALL-CONFIGURATION",
                                    -- Same as the subject.
  "o.a.p.cdestinationId": "9f4cc19f-f37e-4aa0-9b45-
```

```
8453e8423589",                                    -- Set to the
install UUID.
  "o.a.p.cdestinationUrl":
"https://rsp.eng1.responsys.net/ReferenceApp/rest/1.0/app/in
stall/configuration"
}
```

The JWT is signed with the **app's token secret**.

#### Request payload for save configuration

```
{
    "installUuid": "9f4cc19f-f37e-4aa0-9b45-8453e8423589",
    "payload": {
        // Defined by developer. Whatever is sent in
postMessage from Configure Page.
            // Depends on the content of the configuration UI.
    }
}
```

#### Response

The app's Save Configuration URL endpoint should respond with a `configurationStatus`, which would be one of the following: `UNCONFIGURED`, `CONFIGURED`, or `ERROR`. Optionally, a `payload` can also be sent back to and handled by the application configuration UI. For example:

```
{
  "configurationStatus": "CONFIGURED"
}
```

### Uninstall URL

Called by AMS when a user clicks on the delete icon in app management section in the product.

**JWT for uninstallation**

```
{
  "iss": "AMS",
                          -- Issuer is set to AMS
  "sub": "1556569300374_b6fece9a-8b66-4288-82a8-
8ad028b7dc47",                            -- Set to the app's
associated product uuid.
  "aud": "f197ac33-162c-4176-b692-375d8328b9af",
                          -- App's token key
  "exp": 1556569389,
                          -- Set to 60 seconds from the
token creation time.
  "iat": 1556569329,
                          -- The instant when the token is
created
  "jti": "50b6400d-e35a-49d6-bbfe-2faf0cfdefaa",
                          -- A random UUID
  "o.a.p.ctenantId": "5632",
                          -- Tenant's Id for which the app
is uninstalled.
  "o.a.p.cinstallUUID": "a9adcff9-8c25-4ff2-ba94-
0eacb993cdab",                          -- Set to the install
UUID.
  "o.a.p.cproductUUID": "f5c93036-1df1-4f05-8c9d-
17569e63c016",                          -- Set to the
Product's UUID.
  "o.a.p.cproductName": "Responsys",
                          -- Set to the product name.
  "o.a.p.csourceRequest": "1556569300374_b6fece9a-8b66-4288-
82a8-8ad028b7dc47",         -- Set to the app's associated
product uuid.
  "o.a.p.cdestinationId": "ff28d366-44ff-4830-9bc3-
8b8466e2d6a8",                       -- Set to the app's
UUID
  "o.a.p.cdestinationUrl":
"https://rsp.eng1.responsys.net/ReferenceApp/rest/1.0/app/un
install"
}
```

Signed with the **app's install secret**.

**Request payload for uninstallation**


```
{
   "uuid":"a9adcff9-8c25-4ff2-ba94-0eacb993cdab",
   "deleted":0,
   "application":{
      "baseUrl":"http:\/\/localhost:8090",

"configureUrl":"http:\/\/localhost:8090\/configuration.jsp",
      "description":"Sample App",
      "deleted":0,
      "installUrl":"appInstalls\/install",

"largeLogo":"https:\/\/images.freeimages.com\/images\/large-
previews\/2fe\/butterfly-1390152.jpg",

"mediumLogo":"https:\/\/images.freeimages.com\/images\/larg
e-previews\/2fe\/butterfly-1390152.jpg",
      "name":"Sample App",
      "saveConfigurationUrl":"appInstalls\/configuration",

"smallLogo":"https:\/\/images.freeimages.com\/images\/large-
previews\/2fe\/butterfly-1390152.jpg",
      "status":"UP",
      "publicationStatus":"DEVELOPMENT",
      "statusMessage":{
         "body":"O",
         "eta":978309000000,
         "reason":"Someone"
      },
      "statusUrl":"status",
      "uninstallUrl":"appInstalls\/uninstall",
      "uuid":"7e1da507-f54a-4a21-8ac2-cf7a8e9ae7ee",
      "providerUuid":"15e2caf9-b8f1-422b-8f8c-a2936a395e0e"
   },
   "status":"UP"
}
```


The uninstall succeeds even if the app doesn't reply with a 200 status.

Status URL

This endpoint is expected to be invoked by AMS or a product to inquire about application status. Application status refers to whether the service's names have been set, any warnings, or errors related to the app settings and so on.

**JWT during app status call**

```
{
 "iss": "AMS",                                          --
Issuer is set to AMS
 "sub": null,                                           --
Subject can be Null
 "aud": "f197ac33-162c-4176-b692-375d8328b9af",     -- Set
to the app's token key
 "exp": 1507072485000,                                  -- Set
to 60s after the token is created
 "iat": 1506964485000,                                  -- Set
to the instant when the token is created
 "jti": "50b6400d-e35a-49d6-bbfe-2faf0cfdefaa"      -- A
random UUID.
}
```

Signed with the **app's token secret**.

**App response to status call**

```
{
  "info": {
    "name": "Demo Application"
  },
  "warnings": [],
  "errors": [],
  "services": [
    {
      "info": {
        "name": "Churn Likelihood Calculation"
      },
      "warnings": [],
      "errors": []
```

```
    }
  ]
}
```

These are the endpoints supported by the sample application on the app level. The service has it's own set of endpoints as well.

## Service Endpoints

### Status URL

Similar to the app's status endpoint. Used by AMS or the product to check for the service's status.

### Instance Create URL

This endpoint is called by AMS when a user selects a service from the Program canvas. The app is expected to save the instance ID provided by AMS with this call, to later save the configuration settings corresponding to the instance ID.

**JWT for Instance Creation URL:**

```
{
  "iss": "AMS",
                                                    -- Issuer
is AMS
  "sub": "f5c93036-1df1-4f05-8c9d-17569e63c016",
                                                    --
subject is set to productUUID
  "aud": "f197ac33-162c-4176-b692-375d8328b9af",
                                                    -- Set to
App Token Key.
  "exp": 1558335244,
                                                    -- Set to
60s from creation instant
  "iat": 1558335184,
                                                    -- The
instant when JWT is created.
```

```
  "jti": "5cf15fd2-993b-4872-bf19-5dbff12185b6",
                                                        -- Set to
a random UUID
  "o.a.p.ctenantId": "5632",
                                                        -- Set to
the account ID.

  "o.a.p.cproductUUID": "f5c93036-1df1-4f05-8c9d-
17569e63c016",
  -- Set to the product's UUID.
  "o.a.p.cproductName": "Responsys",
                                                        -- Set to
the product Name.
  "o.a.p.csourceRequest": "f5c93036-1df1-4f05-8c9d-
17569e63c016",
-- Set to the product's UUID.
  "o.a.p.cdestinationId": "ff28d366-44ff-4830-9bc3-
8b8466e2d6a8",
-- Set to the app's token key.
  "o.a.p.cdestinationUrl":
"https://rsp.eng1.responsys.net/ReferenceApp/rest/1.0/servic
es/1/instance/create" .  -- Set to the instance creation
URL.
}
```

The JWT is signed with the **app's token secret**.

#### Request payload for service instance creation

```
{
  "status": "CREATED",
  "assetId": null,
  "assetType": null,
  "uuid": "4420fa19-e8f6-4cdf-9754-d876dea3002f",
  "secret": "c4321e9f-19a7-48b2-9796-c21142c709c9-
fb24667dde63-4b6a-99af-10b23122a6d0",
  "recordDefinition": null,
  "applicationServiceInstall": {
    "uuid": "40fe3760-a487-4e89-8cf5-2d3e06977623",
    "name": "Demo transform service for developer's test",
    "description": "Demo transform service for developer's
```

```
test",
    "invokeUrl": "https://b3a3ba42.ngrok.io/omc-app-demo-
1.0/rest/1.0/services/1/invoke",
    "smallLogo": "https://www.iconexperience.com/_img/o_
collection_png/green_dark_grey/48x48/plain/delivery_
truck.png",
    "mediumLogo": "https://www.iconexperience.com/_img/o_
collection_png/green_dark_grey/128x128/plain/delivery_
truck.png",
    "largeLogo": "https://www.iconexperience.com/_img/o_
collection_png/green_dark_grey/256x256/plain/delivery_
truck.png",
    "maxBatchSize": 200,
    "application": {
      "uuid": "b95d7bea-0154-47b8-81ca-ea4e35e784a9",
      "name": "Demo App",
      "description": "It is a demo app",
      "baseUrl": "https://oap.p01.elqqa01.com/awesome-app",
      "statusUrl": "/status",
      "installUrl": "/install",
      "configureUrl": "/configure",
      "uninstallUrl": "/uninstall",
      "saveConfigurationUrl": "/save",
      "smallLogo":
"https://images.martechadvisor.com/images/uploads/product_
logos/Oracle%20Responsys.jpeg",
      "mediumLogo":
"https://images.martechadvisor.com/images/uploads/product_
logos/Oracle%20Responsys.jpeg",
      "largeLogo":
"https://images.martechadvisor.com/images/uploads/product_
logos/Oracle%20Responsys.jpeg",
      "status": "UP",
      "providerUuid": "9dbda2d4-e9ac-4aab-8886-be0402a662cb"
    },
    "serviceType": {
      "name": "ACTIVITY",
      "externalName": "Activity",
      "description": "This service will ingest data and send
it to an external service or process it to return back a
status or modified data"
    },
```

```
    "installUuid": "05860261-2f24-4639-ae38-3616306e3f2d",
    "productName": "Responsys",
    "providerName": "ResponsysApps",
    "status": "UP"
  }
}
```

Responsys expects to receive a data transfer object (DTO) describing the record definition. If the app responds back to AMS with a 2xx response status code, the action service is created.

**Note**: If the app is intending to set a `status` and provide an error message per record, the app is expected to include at a minimum these parameters as they are required for setting a record's `status` and providing error detail when needed:

- `inputParameters`
  - `appcloud_row_correlation_id`
- `outputParameters`
  - `appcloud_row_correlation_id`
  - `appcloud_row_status`
  - `appcloud_row_errormessage`

**Sample response sent by the app to Create Service Instance URL (with a record Definition with the parameters above)**

```
{
  "successful": true,
  "content": {
    "inputParameters": [
      {
        "name": "appcloud_row_correlation_id",
        "dataType": "Text",
```

```
      "width": 40,
      "unique": true,
      "required": true,
      "readOnly": null,
      "minimumValue": null,
      "maximumValue": null,
      "possibleValues": null,
      "format": null,
      "resources": null
    }
  ],
  "outputParameters": [
    {
      "name": "appcloud_row_correlation_id",
      "dataType": "Text",
      "width": 40,
      "unique": true,
      "required": true,
      "readOnly": null,
      "minimumValue": null,
      "maximumValue": null,
      "possibleValues": null,
      "format": null,
      "resources": null
    },
    {
      "name": "appcloud_row_status",
      "dataType": "Text",
      "width": 10,
      "unique": null,
      "required": true,
      "readOnly": null,
      "minimumValue": null,
      "maximumValue": null,
      "possibleValues": [
        "success",
        "warning",
        "failure"
      ],
      "format": null,
      "resources": null
    },
```

```
      {
        "name": "appcloud_row_errormessage",
        "dataType": "Text",
        "width": 5120,
        "unique": null,
        "required": null,
        "readOnly": null,
        "minimumValue": null,
        "maximumValue": null,
        "possibleValues": null,
        "format": null,
        "resources": null
      }
    ]
  },
  "errorMessage": null
}
```

Save Instance Configuration URL

This endpoint is invoked by AMS to provide the configuration payload set by the user to the app. The app saves the configuration associated with the instance ID and provides the updated Record Definition back to AMS.

#### JWT for Save Instance Configuration URL

```
{
  "iss": "f197ac33-162c-4176-b692-375d8328b9af",
                -- App token secret
  "sub": "AMS-INSTANCE-CONFIGURATION",
                -- Subject set by AMS
  "aud": "f197ac33-162c-4176-b692-375d8328b9af",
                -- Set to App token secret
  "exp": 1558335305,
                -- Set to 60s from token creation.
  "iat": 1558335245,
                -- Set to the instant during creation
  "jti": "e97b6d0b-612c-4d13-ac7e-7ba83eea0f03",
                -- Set to a random URL
  "o.a.p.ctenantId": "5632",
```

```
                -- Set to the account ID.
  "o.a.p.capplicationUUID": "ff28d366-44ff-4830-9bc3-
8b8466e2d6a8",           -- Set to the app's token ID.
  "o.a.p.cinstallUUID": "9f4cc19f-f37e-4aa0-9b45-
8453e8423589",              -- Set to the installation
UUID.
  "o.a.p.cproductUUID": "f5c93036-1df1-4f05-8c9d-
17569e63c016",              -- Set to the product's UUID.
  "o.a.p.csourceRequest": "AMS-INSTANCE-CONFIGURATION",
                -- Same as the subject.
  "o.a.p.cdestinationId": "8552376b-143e-4b5f-8965-
2cf85d5608e3",             -- Set to the instance UUID.
  "o.a.p.cdestinationUrl":
"https://rsp.eng1.responsys.net/ReferenceApp/rest/1.0/servic
es/1/instance/configuration"
}
```

Signed with the **app's token secret**.

#### Request payload for saving instance configuration

```
{
 "instanceUuid": "6ea036bb-8cfb-46c5-a826-d001d3a0349b",
 "payload": { // Defined by developer. Whatever is sent in
postMessage from Service Configure Page. }
}
```

The content of the response should include a record definition and the `configurationStatus` which could be either `ERROR` or `CONFIGURED`. Optionally, a payload can be sent back to the application service instance configuration UI.

#### Response sent by app to Save Instance Configuration URL

```
{
  "httpStatusCode": 200,
  "payload": null,
  "recordDefinition": {
    "inputParameters": [
      {
```

```
      "name": "appcloud_row_correlation_id",
      "dataType": "Text",
      "width": 40,
      "unique": true,
      "required": true,
      "readOnly": null,
      "minimumValue": null,
      "maximumValue": null,
      "possibleValues": null,
      "format": null,
      "resources": null
    },
    {
      "name": "CUSTOMER_ID_",
      "dataType": "Text",
      "width": 255,
      "unique": true,
      "required": true,
      "readOnly": null,
      "minimumValue": null,
      "maximumValue": null,
      "possibleValues": null,
      "format": null,
      "resources": null
    }
  ],
  "outputParameters": [
    {
      "name": "appcloud_row_correlation_id",
      "dataType": "Text",
      "width": 40,
      "unique": true,
      "required": true,
      "readOnly": null,
      "minimumValue": null,
      "maximumValue": null,
      "possibleValues": null,
      "format": null,
      "resources": null
    },
    {
      "name": "appcloud_row_status",
```

```
      "dataType": "Text",
      "width": 10,
      "unique": null,
      "required": true,
      "readOnly": null,
      "minimumValue": null,
      "maximumValue": null,
      "possibleValues": [
        "success",
        "warning",
        "failure"
      ],
      "format": null,
      "resources": null
    },
    {
      "name": "appcloud_row_errormessage",
      "dataType": "Text",
      "width": 5120,
      "unique": null,
      "required": null,
      "readOnly": null,
      "minimumValue": null,
      "maximumValue": null,
      "possibleValues": null,
      "format": null,
      "resources": null
    },
    {
      "name": "CHURN_ETV",
      "dataType": "Text",
      "width": 500,
      "unique": null,
      "required": null,
      "readOnly": null,
      "minimumValue": null,
      "maximumValue": null,
      "possibleValues": null,
      "format": null,
      "resources": null
    }
  ]
```

```
  },
  "requestId": "save",
  "configurationStatus": "CONFIGURED"
}
```

Delete Service Instance URL

This endpoint is called by AMS to notify the product about the instance IDs corresponding to a service which have been removed by the user from the product's asset (i.e. Programs in Responsys). The app is not notified immediately about an instance's deletion. Rather, a periodic job in the product notifies AMS about the removed instances, which then calls this endpoint to provide the app about the removed instances. The app can then mark the instance as deleted.

#### JWT for Delete Service Instance URL

```
{
  "iss": "AMS",
                  -- Set to AMS
  "sub": "f5c93036-1df1-4f05-8c9d-17569e63c016",
                  -- Set to the Product's UUID.
  "aud": "1bb1c81f-9472-407a-974c-c1483f64aaa5",
                  -- Set to the app's token key
  "exp": 1558350072,
                  -- Set to 60 seconds from the token
creation.
  "iat": 1558350012,
                  -- The Instant when the token is created.
  "jti": "4f3bf5d6-e78a-4c40-9343-a919455c7cf2",
                  -- Set to a random UUID.
  "o.a.p.csourceRequest": "f5c93036-1df1-4f05-8c9d-
17569e63c016",              -- Set to the Product's UUID.

  "o.a.p.cdestinationId": "7a5179c1-4815-4a33-86dc-
f3748e20fe0b",              -- Set to the app's UUID.
  "o.a.p.cdestinationUrl":
"https://rsp.eng1.responsys.net/DemoApplication/rest/1.0/ser
vices/1/delete"
```

```
}
```

Signed with the **app's token secret**.

#### Request payload for instance deletion

```
{
  "uuid": "2128d91f-7248-4361-94b8-38a414d5cbb1",
  "assetType": "campaign",
  "assetId": null,
  "recordDefinition": {
    "inputParameters": [
      {
        "dataType": "Double",
        "unique": true,
        "name": "input",
        "width": 10
      }
    ],
    "outputParameters": [
      {
        "dataType": "Double",
        "unique": true,
        "name": "output",
        "width": 10
      }
    ]
  },
  "applicationService": {
    "uuid": "c2d3a932-2e00-45e5-9899-17d8edda00c7",
    "name": "service",
    "description": null,
    "baseUrl": "https://somefakeappbaseurl.com",
    "configureUrl": "/configure",
    "saveConfigurationUrl": null,
    "deleteUrl": "/delete",
    "createUrl": "/create",
    "invokeUrl": "/invoke",
    "statusUrl": null,
    "operationalUrl": null,
    "smallLogo": "https://smalllogourl",
```

```
    "mediumLogo": null,
    "largeLogo": null,
    "maxBatchSize": 0,
    "application": {
      "uuid": "d6862ebc-8c5e-4905-8da1-28e3166c978c",
      "name": "appname",
      "description": "description",
      "baseUrl": "https://www.oracle.com",
      "statusUrl": "status",
      "installUrl": "/installurl",
      "configureUrl": "/configure",
      "uninstallUrl": "/uninstall",
      "saveConfigurationUrl": "/save",
      "smallLogo": "smalllogo",
      "mediumLogo": "mediumlogo",
      "largeLogo": "largelogo",
      "status": "UP",
      "providerUuid": "53aba933-a031-46ce-ab69-4ad0f4cc3335"
    },
    "serviceType": {
      "name": "activity",
      "externalName": "activityext",
      "description": "activitytypeofservice"
    },
    "status": "UP"
  },
  "secret": null,
  "status": "CREATED"
}
```


**Sample Response in case of success**

```
200 OK
```

The app does not need to respond to this call. If the app chooses responds to this call, it will be ignored by AMS.

These are the URLs called by AMS. The service URL which the product invokes directly is the **Service Invocation URL**.

Service Invocation URL

Invocation is the process where an Oracle CX Marketing product invokes an app to perform a task. The product will retrieve the app's instance context from AMS, which includes the app's Invoke URL endpoint. Then the product will invoke the app based on the app's Invoke URL endpoint. The app should directly respond to the product to relay the result of the invocation back to the product.

Effectively, invocation is how products can use apps using the app's endpoints stored in AMS.

#### Requirements

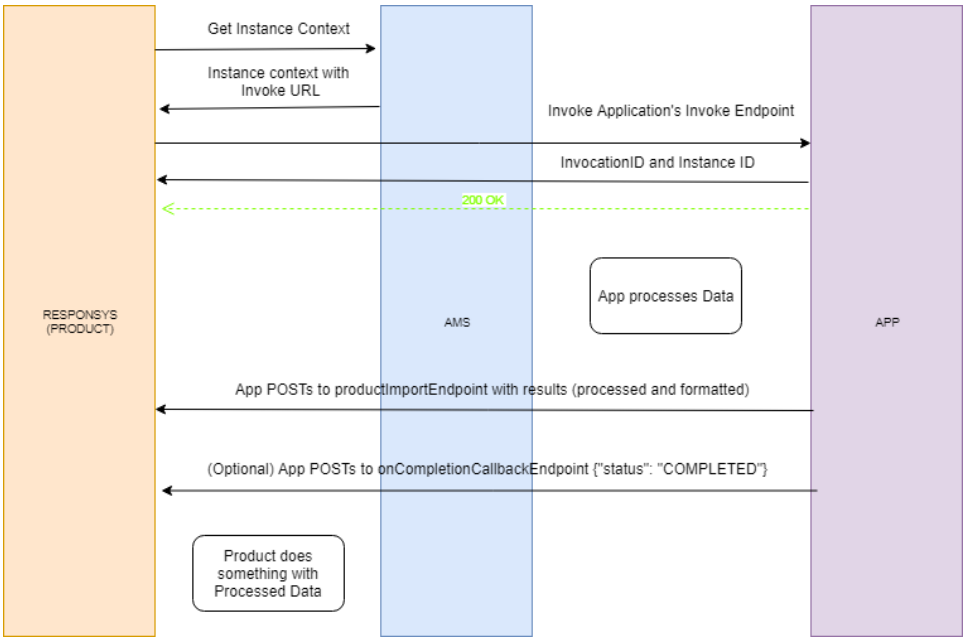
Services must be configured before invocation. Service configuration allows the app to establish record definition field mappings so that the app will know how the product will invoke it.

There are two different types of invocations. Apps should be developed for both invocation types.

- **Invoke with data**. Apps need to parse data and then perform an action.
- **Invoke without data**. Apps need to pull data.

The invocation type is based on the `dataSet` parameter. For **Invokes with data**, the DataRow(s) sent by the product will be populated. For **Invokes without data**, the DataRow(s) will be `null`, but the `DataSetsize` will be set, so the app can know how many `rows` to import.

### Invoke With Data




### JWT sent during invocation

```
{
  "aud": "88bea6ed-0460-427c-9567-1f7ed05d47cc",
   -- Set to the service instance UUID
  "iss": "f5c93036-1df1-4f05-8c9d-17569e63c016",
   -- Set to the Product's token UUID
  "exp": 1558292803,
   -- Set to 60s from JWT creation.
  "iat": 1558292503,
   -- Set to the instant when JWT is created.
  "o.a.p.ctenantId": "5632"
   -- A private claim set to the account ID.
}
```

Signed with the **app install secret** (shared secret between app and product). Besides the Authorization header, another header named **OMC-ID** (value set to the Instance ID) is sent to the app.

**Request payload for Invoke with data**

```
{
   "instanceContext":{
      "appId":"38281836-4bb4-2cdb-6006-592a98d02da1",
      "appVersion":null,
      "installId":"a28b7df0-2a16-26e1-08e4-a302199208d9",
      "instanceId":"6ea036bb-8cfb-46c5-a826-d001d3a0349b",
      "serviceId":"13023bae-8350-f75f-d953-f7a96d6928b6",
      "tenantId":"6607",
      "productId":"78ffbba2-cf29-a607-c611-3f05e9199a39",
      "maxPushBatchSize":1000,
      "secret":"c4321e9f-19a7-48b2-9796-c21142c709c9-
fb24667dde63-4b6a-99af-10b23122a6d0",
      "recordDefinition":{
         "inputParameters":[
            {
               "name":"appcloud_row_correlation_id",
               "dataType":"Text",
               "width":40,
               "unique":true,
               "required":true,
               "readOnly":null,
               "minimumValue":null,
               "maximumValue":null,
               "possibleValues":null,
               "format":null,
               "resources":null
            },
            {
               "name":"CUSTOMER_ID_",
               "dataType":"Text",
               "width":255,
               "unique":null,
               "required":null,
               "readOnly":true,
               "minimumValue":null,
               "maximumValue":null,
               "possibleValues":null,
               "format":null,
               "resources":null
            }
```

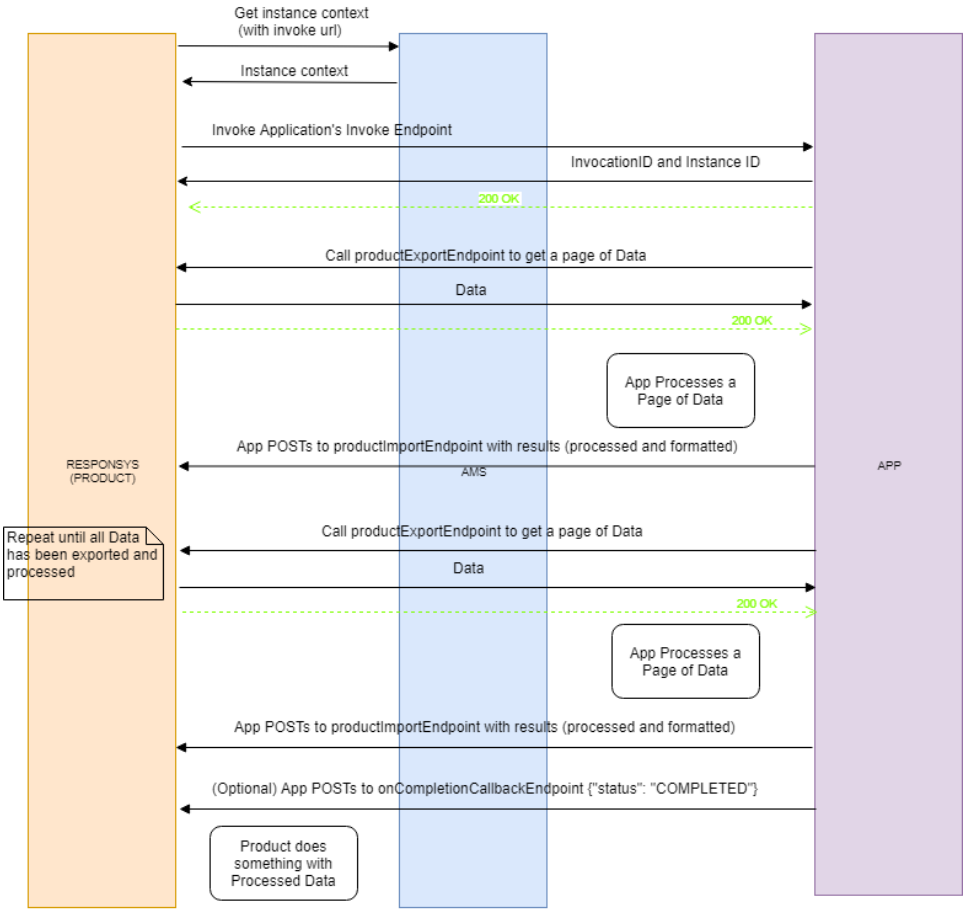
```
        ],
        "outputParameters":[
           {
              "name":"appcloud_row_correlation_id",
              "dataType":"Text",
              "width":40,
              "unique":true,
              "required":true,
              "readOnly":null,
              "minimumValue":null,
              "maximumValue":null,
              "possibleValues":null,
              "format":null,
              "resources":null
           },
           {
              "name":"appcloud_row_status",
              "dataType":"Text",
              "width":10,
              "unique":null,
              "required":true,
              "readOnly":null,
              "minimumValue":null,
              "maximumValue":null,
              "possibleValues":[
                 "success",
                 "warning",
                 "failure"
              ],
              "format":null,
              "resources":null
           },
           {
              "name":"appcloud_row_errormessage",
              "dataType":"Text",
              "width":5120,
              "unique":null,
              "required":null,
              "readOnly":null,
              "minimumValue":null,
              "maximumValue":null,
              "possibleValues":null,
```

```
            "format":null,
            "resources":null
         },
         {
            "name":"CHURN_ETV",
            "dataType":"Text",
            "width":500,
            "unique":null,
            "required":null,
            "readOnly":false,
            "minimumValue":null,
            "maximumValue":null,
            "possibleValues":null,
            "format":null,
            "resources":null
         }
      ]
   },
   "maxBatchSize":1000
},
"dataSet":{
   "id":"my-test-data-set",
   "rows":[
      [
         "72a50e37-4dbc-4c97-bb4e-366dd4dcce6d",
         "1001"
      ]
   ],
   "size":null
},
"productExportEndpoint":null,
"productImportEndpoint":{
   "url":"http://product.com/import/123/data",
   "method":"POST",
   "headers":{

   }
},
"onCompletionCallbackEndpoint":{
   "url":"http://product.com/feedback",
   "method":"POST",
   "headers":{
```

```
        }
    },
    "maxPullPageSize":5,
    "maxPushBatchSize":5
}
```

### Invoke without Data




The JWT remains the same as Invoke with data.

### Request Payload for invoke without data

```
{
    "instanceContext":{
        "appId":"38281836-4bb4-2cdb-6006-592a98d02da1",
        "appVersion":null,
```

```
      "installId":"a28b7df0-2a16-26e1-08e4-a302199208d9",
      "instanceId":"6ea036bb-8cfb-46c5-a826-d001d3a0349b",
      "serviceId":"13023bae-8350-f75f-d953-f7a96d6928b6",
      "tenantId":"6607",
      "productId":"78ffbba2-cf29-a607-c611-3f05e9199a39",
      "maxPushBatchSize":1000,
      "secret":"c4321e9f-19a7-48b2-9796-c21142c709c9-
fb24667dde63-4b6a-99af-10b23122a6d0",
      "recordDefinition":{
         "inputParameters":[
            {
               "name":"appcloud_row_correlation_id",
               "dataType":"Text",
               "width":40,
               "unique":true,
               "required":true,
               "readOnly":null,
               "minimumValue":null,
               "maximumValue":null,
               "possibleValues":null,
               "format":null,
               "resources":null
            },
            {
               "name":"CUSTOMER_ID_",
               "dataType":"Text",
               "width":255,
               "unique":null,
               "required":null,
               "readOnly":true,
               "minimumValue":null,
               "maximumValue":null,
               "possibleValues":null,
               "format":null,
               "resources":null
            }
         ],
         "outputParameters":[
            {
               "name":"appcloud_row_correlation_id",
               "dataType":"Text",
               "width":40,
```

```
            "unique":true,
            "required":true,
            "readOnly":null,
            "minimumValue":null,
            "maximumValue":null,
            "possibleValues":null,
            "format":null,
            "resources":null
        },
        {
            "name":"appcloud_row_status",
            "dataType":"Text",
            "width":10,
            "unique":null,
            "required":true,
            "readOnly":null,
            "minimumValue":null,
            "maximumValue":null,
            "possibleValues":[
                "success",
                "warning",
                "failure"
            ],
            "format":null,
            "resources":null
        },
        {
            "name":"appcloud_row_errormessage",
            "dataType":"Text",
            "width":5120,
            "unique":null,
            "required":null,
            "readOnly":null,
            "minimumValue":null,
            "maximumValue":null,
            "possibleValues":null,
            "format":null,
            "resources":null
        },
        {
            "name":"CHURN_ETV",
            "dataType":"Text",
```

```
                "width":500,
                "unique":null,
                "required":null,
                "readOnly":false,
                "minimumValue":null,
                "maximumValue":null,
                "possibleValues":null,
                "format":null,
                "resources":null
             }
          ]
       },
       "maxBatchSize":1000
    },
    "dataSet":{
       "id":"my-test-data-set",
       "rows":null,
       "size":6
    },
    "productExportEndpoint":{
       "url":"http://product.com/export/123/data",
       "method":"GET",
       "headers":{

       }
    },
    "productImportEndpoint":{
       "url":"http://product.com/import/123/data",
       "method":"POST",
       "headers":{

       }
    },
    "onCompletionCallbackEndpoint":{
       "url":"http://product.com/feedback",
       "method":"PATCH",
       "headers":{

       }
    },
    "maxPullPageSize":5000,
    "maxPushBatchSize":5000
```

```
}
```

The app responds back to the product with a 2xx with the invocation ID. The invocation ID is included so that the app can determine which result comes from which invocation ID. The actual data for the invocation result is sent directly from the app to the product's `productImportEndpoint`. This happens asynchronously and independent of AMS.

**Sample Response to invocation call**

```
200 OK
{
  "invocationId": "bb876080-7655-4d17-bfbd-80350ace9ce4",
  "instanceId": "6ea036bb-8cfb-46c5-a826-d001d3a0349b"
}
```

The app needs to send this as response to the invocation call immediately. The product endpoints can be called asynchronously via other worker threads.

Notes:

- The `recordDefinition` indicates the structure of the `DataSet` rows. This also indicates the structure (record definition) that the app should return to the product. To learn more about record definitions, see the online help.
- `dataSet` contains the data that the app is being invoked with in case of small batch (i.e Invoke with Data). Will be null in case of Invoke without data, but would contain the dataset size.
- `productExportEndpoint` is the endpoint on the product where the invocation data resides. The app needs to call this endpoint to get the invocation data with the correct REST call specified in the `productExportEndpoint`. Should be a GET. Used to fetch

pages of data from a product and into an app based on the `maxPullPageSize`. The `maxPullPageSize` parameter determines the largest pagination size provided by the `productExportEndpoint`. Is omitted if the invoke is done with data. Would be null in case of Invoke with Data.

- `maxPullPageSize` is the largest pagination size provided by the `productExportEndpoint`.
- `productImportEndpoint` is the endpoint the product expects the app to respond to with the processed `DataRows`. Used to push results back into a product by the app based on the `maxPushBatchSize`. The `maxPushBatchSize` determines the largest amount of records to push per batch to the `productImportEndpoint`. Should be a POST or a PUT. Is omitted if product is not interested in the results.
- `onCompletionCallbackEndpoint` (optional) should be called upon invocation completion by the app to notify the app to pass on the enactments to the next stage in program orchestration. Is a PATCH call for now.

Now, the app needs to callout to the `productExportEndpoint` to get a page of data. There are two important query parameters in the URL that need to be specified: `offset` and `limit`.

- `offset` indicates the offset within the Products entire `dataSet` to begin exporting a page of data. It is important that this offset is correctly populated as it can result in duplication of data if the offsets are not correctly calculated. The `maxPullPageSize` in the payload is used as the `exportPageSize` as the sample app attempts to grab as much data as it can per export. For example, if there are 100 contacts in the product and the `maxPullPageSize` is 10, the initial offset should be 0 so the first 10 entries are exported, the second callout should have an `offset` of 10 so that the 10-19 entries are exported, and so on. The default is 0 (no offset).

For example: `?offset=10`.

- `limit` indicates the size of the export that the app wants. So if the limit is set to 10, the product will export a maximum of 10 contacts to the app. The app should not try to export more than the `maxPullPageSize` specified by the product as that is undefined behaviour. The default limit value is 1000.

  For example: `?limit=10`.

**JWT sent by app to product:**

```
{
  "iss": "ff28d366-44ff-4830-9bc3-8b8466e2d6a8",        --
Set to the app's token UUID
  "aud": "88bea6ed-0460-427c-9567-1f7ed05d47cc",        --
Set to the instance UUID
  "exp": 1558292540,                                    --
Set to 30 seconds from i.a.t (shouldn't be too large)
  "iat": 1558292510,                                    --
Set to the Instant at which token is created.
  "o.a.p.ctenantId": "5632"                             --
Private claim set to the account ID.
}
```

Signed with the **app install secret**.

The same JWT claimset is used by app for all calls to the Product. (to `productExportEndpoint`, `productImportEndpoint`, and `onCompletionCallbackEndpoint`).

Apart from the Authorization Bearer token, additional headers sent are:

```
Content-Type - "application/json"
OMC-ID - instanceContext.getInstanceId()
INVOCATION-ID - invocation.getUuid()
DATASET-ID - invocation.getDataSet().getId()
```

## ChurnOut Likeliness Service

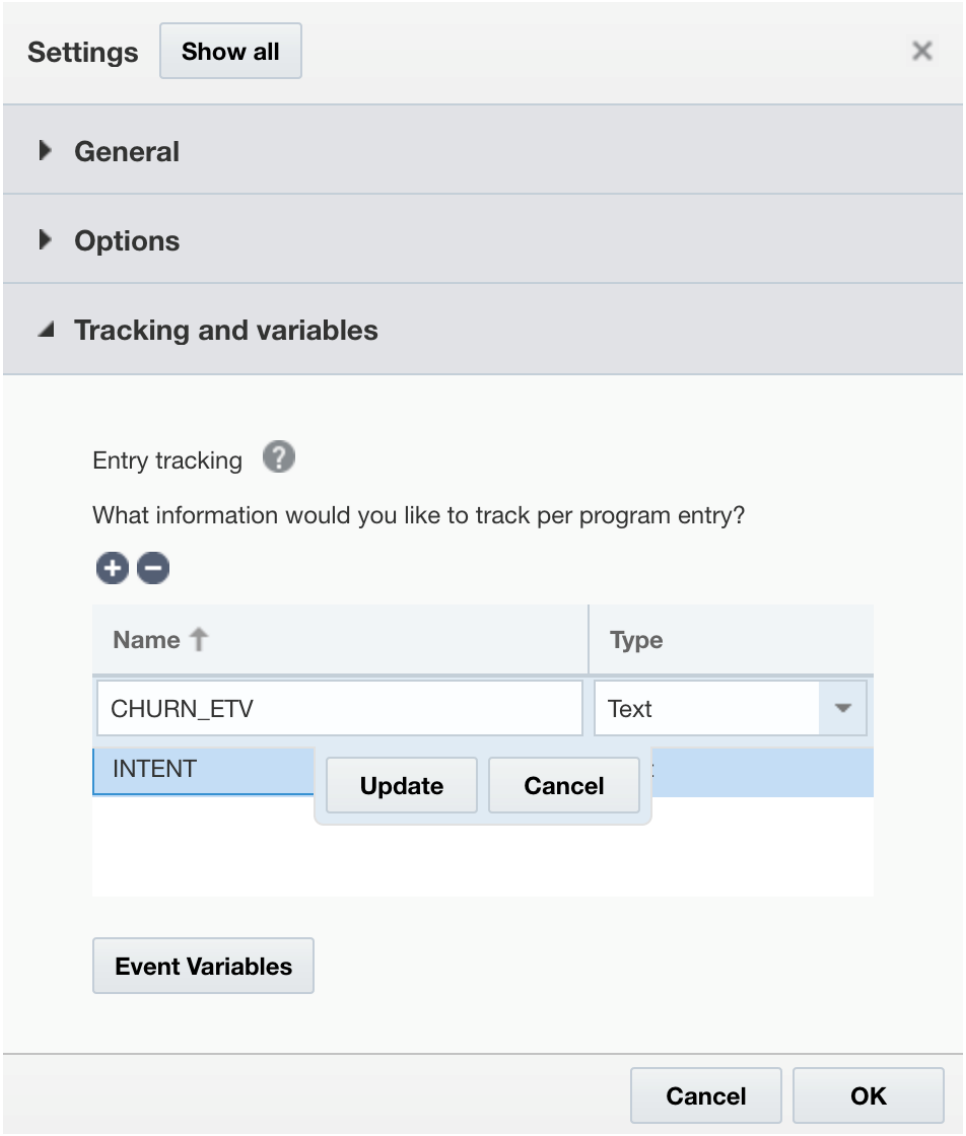
The ReferenceApp provides a single service called ChurnOut Likeliness Service. The service is a simple use case considering the fact that the app uses pre-populated data to calculate customer churnOut likelihood. The service uses the configuration saved by the marketer during the service configuration to calculate the churn likeliness. The pre-populated data (in the metrics and purchase_history tables described in the next section) is for each CUSTOMER_ID_ present in the profile List.

The service depends on two parameters:

CHURN_ETV

ETVs (also called Enactment Tracking variables) are variables defined in a program's settings around which the appcloud framework is based. If any service has additional data or processed information available to be sent back to Responsys, it has to do so through ETVs. Our service calculates the value of churn likeliness and sets it as the ETV value for that customer.

The marketer has to first add the program variable by clicking on the Settings tab in the program.




Adding this will help the marketer to configure the service as this field will now be part of the fieldMetaDataEndpoint's payload. This endpoint is invoked by AMS to retrieve field definitions. This is currently only applicable to Responsys. The endpoint responds with an object that contains a collection of objects. For invoking this endpoint, the service UI needs to invoke a call to it's parent iframe (i.e AMS iframe ) with an amsAction: "metadata" . & requestId: "metadata". After this, AMS calls the productEndpoint to fetch the fields (which contain both the profile list fields as well as the ETVs). AMS then postMessages the payload to the app's UI ,which must have handlers in place to capture this event and populate the UI accordingly.

Sample payload sent by the product to AMS from fieldMetadataEndpoint:

Payload sent by Responsys from schema endpoint


```
{
  "objects": [
    {
      "name": "pk_list",
      "type": "Standard",
      "fields": [
        {
          "name": "RIID_",
          "dataType": "Text",
          "width": 0,
          "readOnly": true,
          "format": null
        },
        {
          "name": "CREATED_SOURCE_IP_",
          "dataType": "Text",
          "width": 255,
          "readOnly": true,
          "format": null
        },
        {
          "name": "CUSTOMER_ID_",
          "dataType": "Text",
          "width": 255,
          "readOnly": true,
          "format": null
        },
        {
          "name": "EMAIL_ADDRESS_",
          "dataType": "Text",
          "width": 500,
          "readOnly": true,
          "format": null
        },
        {
          "name": "EMAIL_DOMAIN_",
          "dataType": "Text",
```

```
        "width": 255,
        "readOnly": true,
        "format": null
      },
      {
        "name": "EMAIL_ISP_",
        "dataType": "Text",
        "width": 255,
        "readOnly": true,
        "format": null
      },
      {
        "name": "EMAIL_FORMAT_",
        "dataType": "Text",
        "width": 1,
        "readOnly": true,
        "format": null
      },
      {
        "name": "EMAIL_PERMISSION_STATUS_",
        "dataType": "Text",
        "width": 1,
        "readOnly": true,
        "format": null
      },
      {
        "name": "EMAIL_DELIVERABILITY_STATUS_",
        "dataType": "Text",
        "width": 1,
        "readOnly": true,
        "format": null
      },
      {
        "name": "EMAIL_PERMISSION_REASON_",
        "dataType": "Text",
        "width": 255,
        "readOnly": true,
        "format": null
      },
      {
        "name": "EMAIL_MD5_HASH_",
        "dataType": "Text",
```

```
        "width": 50,
        "readOnly": true,
        "format": null
      },
      {
        "name": "EMAIL_SHA256_HASH_",
        "dataType": "Text",
        "width": 100,
        "readOnly": true,
        "format": null
      },
      {
        "name": "MOBILE_NUMBER_",
        "dataType": "Text",
        "width": 50,
        "readOnly": true,
        "format": null
      },
      {
        "name": "MOBILE_COUNTRY_",
        "dataType": "Text",
        "width": 25,
        "readOnly": true,
        "format": null
      },
      {
        "name": "MOBILE_PERMISSION_STATUS_",
        "dataType": "Text",
        "width": 1,
        "readOnly": true,
        "format": null
      },
      {
        "name": "MOBILE_DELIVERABILITY_STATUS_",
        "dataType": "Text",
        "width": 1,
        "readOnly": true,
        "format": null
      },
      {
        "name": "MOBILE_PERMISSION_REASON_",
        "dataType": "Text",
```

```
        "width": 255,
        "readOnly": true,
        "format": null
      },
      {
        "name": "POSTAL_STREET_1_",
        "dataType": "Text",
        "width": 255,
        "readOnly": true,
        "format": null
      },
      {
        "name": "POSTAL_STREET_2_",
        "dataType": "Text",
        "width": 255,
        "readOnly": true,
        "format": null
      },
      {
        "name": "CITY_",
        "dataType": "Text",
        "width": 50,
        "readOnly": true,
        "format": null
      },
      {
        "name": "STATE_",
        "dataType": "Text",
        "width": 50,
        "readOnly": true,
        "format": null
      },
      {
        "name": "POSTAL_CODE_",
        "dataType": "Text",
        "width": 25,
        "readOnly": true,
        "format": null
      },
      {
        "name": "COUNTRY_",
        "dataType": "Text",
```

```
        "width": 50,
        "readOnly": true,
        "format": null
      },
      {
        "name": "POSTAL_PERMISSION_STATUS_",
        "dataType": "Text",
        "width": 1,
        "readOnly": true,
        "format": null
      },
      {
        "name": "POSTAL_DELIVERABILITY_STATUS_",
        "dataType": "Text",
        "width": 1,
        "readOnly": true,
        "format": null
      },
      {
        "name": "POSTAL_PERMISSION_REASON_",
        "dataType": "Text",
        "width": 255,
        "readOnly": true,
        "format": null
      },
      {
        "name": "CREATED_DATE_",
        "dataType": "DateTime",
        "width": 0,
        "readOnly": true,
        "format": "yyyy-MM-dd'T'HH:mm:ss'Z'"
      },
      {
        "name": "MODIFIED_DATE_",
        "dataType": "DateTime",
        "width": 0,
        "readOnly": true,
        "format": "yyyy-MM-dd'T'HH:mm:ss'Z'"
      }
    ]
  },
  {
```

```
      "name": "AppCloudSampleProgram",
      "type": "Standard",
      "fields": [
        {
          "name": "CHURN_ETV",
          "dataType": "Text",
          "width": 500,
          "readOnly": false,
          "format": null
        }
      ]
    }
  ]
}
```

**Note**: Objects here can be any Responsys object. For this application , the objects will be the profile list and the Program associated with the service Instance.

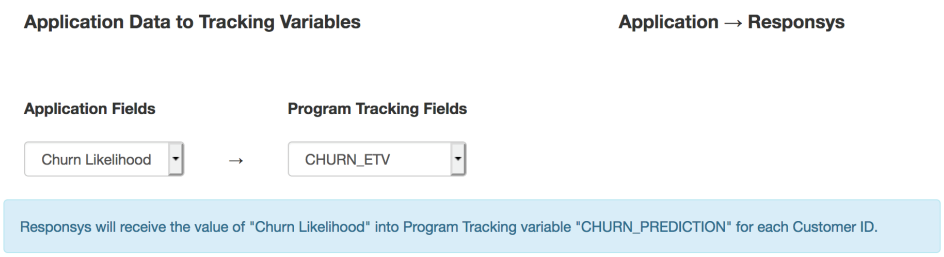
The app's config UI throws an error message if ETVs are not declared in the program.




After everything has been set properly, the service's configuration page finally comes up. We can break down the UI and analyze:

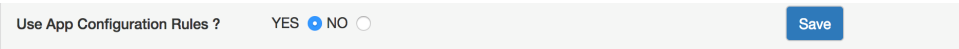

This is the field that the service requires: **CUSTOMER_ID_**.

The dropdown is populated using the values obtained from the product's `fieldMetadataEndpoint`. The service fetches this parameter's value for each customer that flows to the App stage during invocation.

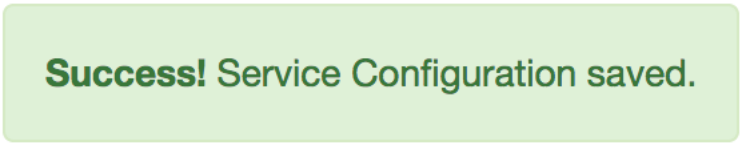



This section maps the app's field to the program ETV defined by the marketer. This is the field that will be populated by the service for each CUSTOMER_ID_ that it receives.

Next, the service has a Manage Rules section which is basically the same as the app install configuration page (Which we access from App Management section under **Account**). The rules can be overriden by the marketer here if they want. But if they want to modify the values, they must make sure to choose **NO** in the **Use App Configuration Rules** option.




The default is set to YES. Clicking on save would trigger a postMessage to AMS with an amsAction : "save" which would prompt AMS to POST the entire config page's payload to the save instance Configuration URL. Those aspects have been discussed earlier. The service returns a status back to AMS ( saying if the instance was saved correctly, in which case it would send the status as "CONFIGURED"). AMS posts this to the service UI which can then give a confirmation to the marketer based on this flag.




That's all about the service configuration. Based on the configuration rules , the service sends the value of the CHURN_ETV for each enactment it receives using the `productImportEndpoint`.

A thing to note here is , the apps won't be sending the CHURN_ETV or any ETV value for that matter corresponding to any field (Ex- CUSTOMER_ID_) .Rather, the ETV values should correspond to the "appcloud_row_correlation_id" which is an identity that the Product uses . It's value is provided in the input Parameters i.e the dataSet payload would contain the values of both correlation ID and the CUSTOMER_ID_ for our service.

Sample data sent by app to the product post processing a dataSet

**Data Sent to productImportEndpoint**

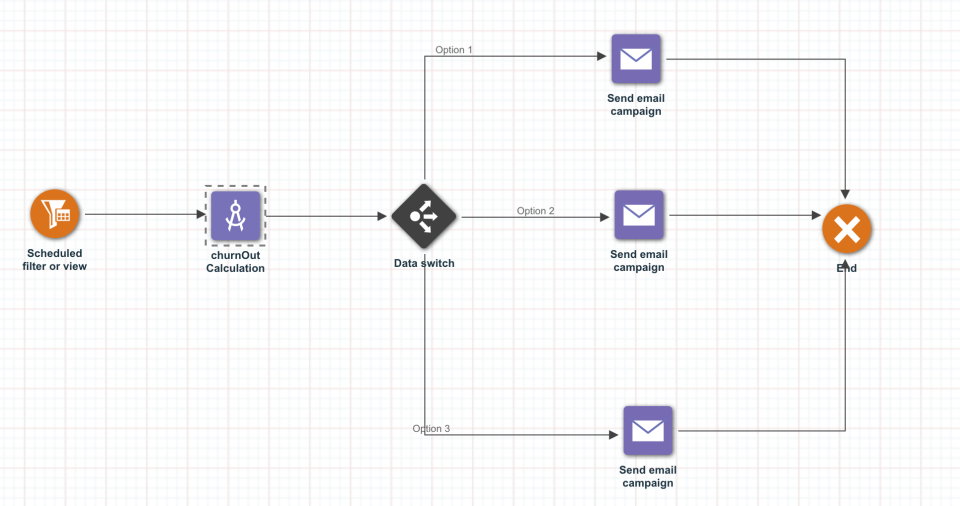
```
{
  "fieldDefinitions": [
    {
      "name": "APPCLOUD_ROW_CORRELATION_ID",
      "dataType": "Text",
      "width": 40,
      "readOnly": false
    },
    {
      "name": "APPCLOUD_ROW_STATUS",
      "dataType": "Text",
      "width": 10,
      "readOnly": false
    },
    {
      "name": "APPCLOUD_ROW_ERRORMESSAGE",
      "dataType": "Text",
```

```
      "width": 5120,
      "readOnly": false
    },
    {
      "name": "CHURN_ETV",
      "dataType": "Text",
      "width": 20,
      "readOnly": false
    }
  ],
  "dataSet": {
    "id": "608569cb-531c-4c49-a822-16f9f39f0275",
    "size": "5",
    "rows": [
      [
        "18286211;3",
        "success",
        null,
        "NEUTRAL"
      ],
      [
        "18286212;23",
        "success",
        null,
        "LIKELY"
      ],
      [
        "18286213;43",
        "success",
        null,
        "VERY LIKELY"
      ],
      [
        "18286214;63",
        "success",
        null,
        "UNLIKELY"
      ],
      [
        "18286215;83",
        "success",
        null,
```

```
          "INVALID"
        ]
      ]
    }
}
```

Finally, after the app posts the data to the Product, the marketer can take decisions to take recipients along different paths (example - Send different kinds of Emails to customers based on the churnOut likelihood provided by the app).

Example of a sample program would be:



## RESPONSYS APP DEVELOPMENT BEST PRACTICES

Best practices to follow when developing apps for Responsys.

- Handling dataSets
- JWT Creation and Usage
- Retry and Failure Scenarios
- Release Call

### Handling dataSets

A `dataSet` can be sent as part of an invocation payload by the product (in case of small batches or an Invoke with Data) or required to be downloaded in batches using the `productExportEndpoint`.

It's important from an app perspective to understand how a `dataSet` is categorized as small or large by the product.

### Determining Batch Size

App providers can configure individual services to have a **Max Batch Size** in the App Manager UI by editing the service, but the default value is 5000. The product has an upper limit on the maximum records that can be part of a `dataSet`, so an app now needs to consider the following scenarios:

1. **Triggered**. All triggered enactments are considered **small batches**. A single record is passed to app during invocation.
2. **Small Batch**. Separate from the Triggered scenario, small batches are decided based on the following logic:

```
Find minimum of Max Batch Size set by Service and the
```

```
product's batch limit. Define this as MaxDatasetSize. Thus
MaxDatasetSize = min(App's Max Batch Size and Product's batch
limit)
Get the number of records to be sent to app. Let it be defined
as recordCount.
Check if recordCount <= MaxDatasetSize
```

If this condition is true, the `dataSet` is considered a small batch, and is sent via Invoke with Data.

3. **Large Batch**. If the above criteria isn't met.

Thus, the app should set an optimal value for fetching the entire `dataSet`. Setting a relatively low **Max Batch Size** would increase the number of HTTP GET calls to download using the **productExportEndpoint**.

## Multi-Threading Approach

Alternatively, apps can opt to download and process rows and send back the `resultSet` back to Responsys in a sequential fashion (by incrementing `offsets` after a particular `dataSet` has been processed). However it would hamper the app's performance in the event of a very large `dataSet`.

A better way would be to spawn multiple threads to download in parallel, process, and send back the records to the product. Or they can also opt to download all records at once using multiple threads and then start processing on the downloaded threads. It depends on how the app providers design their architecture. However, a multi-threaded approach is the recommended approach.

### JWT Creation and Usage

Apps should ensure they generate a **new** JWT token every time they want to communicate with the product.

It is not best practice to create a single JWT with an arbitrarily large expiration time, and reuse this JWT for every call to the product associated with an invocation. Not only will this pose serious security concerns (in the event malicious entities acquire a token, they can call the download endpoint unlimited times if the expiration time is too long), but the JWT itself can expire if the `dataSet` is very large and the app takes longer than expected to download all the entire `dataSet`.

### Retry and Failure Scenarios

Apps can encounter issues while downloading and uploading dataSets. These issues will most often arise when downloading larger dataSets. The product can throw status codes besides a 200 OK for many reasons, ranging from throttling limits, to downtime, and so on. Apps must be prepared for such scenarios.

For failed downloads, apps can **save the offsets** for which the downloads failed, and retry the calls to the **productExportEndpoint** at a later time. For upload failures, calls to the **productImportEndpoint**, apps can implement a flag to maintain the status for the calls. Failed imports/exports can be retried later.

### Release Call

Calling the `onCompletionCallbackEndpoint` is not mandatory. This endpoint **should not be called for each batch**. Only when the app is sure of completion of all the processing for the entire `dataSet`, it can opt to call this endpoint. Currently, Responsys doesn't support batch level release in programs. So calling this endpoint would release all enactments to the next stage. The app can, of course, choose not to call this endpoint explicitly in which case the enactments would stay at the Apps stage until the stage

expires (set to 3 days by default). This endpoint **should not be called for each batch**. A single call to this endpoint would move enactments to the next stage.

In summary, apps can keep retrying any failed batches for up to 3 days, at which point the app can call this endpoint after all processing is finished and imports have succeeded.

# ORACLE APPLICATION MANAGEMENT SERVICE GLOSSARY

## A

### AMS

Oracle AMS stands for Oracle Application Management Service. Oracle AMS is the name of the cloud service that interacts with Oracle CX Marketing products, and apps developed for the App Developer Framework.

### Apps

Applications are specialized programs that app developers have built to extend the functionality of an Oracle CX Marketing product. Apps are comprised of services, which represent the functionality of the app. An app can contain multiple different services.

### App Providers

An App Provider is the name of a company or organization who creates apps for the App Framework.

## E

### Enactments

An enactment is a representation of a contact within a Responsys Program workflow. For example, when a customer makes a purchase, the customer's contact information is pulled into the program and is represented as an enactment. This enactment moves through the Program workflow. Enactments can be considered in-memory and are not committed to the Responsys database until explicitly instructed. A contact can have

multiple enactments within a single orchestration workflow. Though each enactment represents the same contact, each enactment is unique within the workflow.

### Entry Tracking Variables

Entry Tracking Variables (sometimes referred to as *Program Variables* or *Enactment Variables*) enable marketers to update information about an enactment while it is in the Program workflow. Entry Tracking Variables can be used in workflows so that Data Switches can make decisions and route the enactments accordingly. Entry Tracking Variables are local to the specific Program canvas. Another program cannot access the Entry Tracking Variables for an enactment in a different workflow. Entry Tracking Variables are local to the specific enactment in the program workflow. There can be multiple enactments for the same contact, but each will have its own set of values for their Entry Tracking Variables.

## P

### Products

A *product* is an entity within the app framework that represents an Oracle CX Marketing product. For example, Oracle Responsys is a product and Oracle Eloqua Sales Tools is a product. Apps are developed with a specific product in mind. Depending on the product you want to develop an app for, the services available for your app will differ.

## R

### Record Definition

The record definition is essentially a set of input and output parameters. For each input, the app will return a specific output. These inputs and outputs defines the configuration for the service as set by the marketer.

For example, when configuring a service, the configuration UI must present the user with options to map product fields to app fields. This way, the app knows what to do with the data. In example, a `CITY_` field in the product must be mapped to something like a `City` variable in the app so that the app can perform some operation with it.

See Record definitions to learn more.

## S

### Service

Services are self-contained components of apps that are used by marketers.

## T

### Tenants

A *tenant* is the term used to describe an account instance. An account instance can be tied to a marketing account or a developer account. Developers should note that each tenant has an associated `id` which is used when making requests within the app framework.