MDS GLOBAL

CMP 8.21

Communications Functionality and Configuration Guide

Version 1.0

Classification: **Customer Confidential**

Copyright

Confidentiality

Disclaimer

Trademarks

Contact Details

Please visit www.mdsglobal.com for further information on MDS Global products, solutions and services.

ISO 22301 standard is applicable to MDS Global Business Operations.

Table of Contents

Version Control

Version	Issue Date	Author	Comments
Version 1.0	16 January 2026	MDS	CMP 8.21 Release - No changes since the last release.

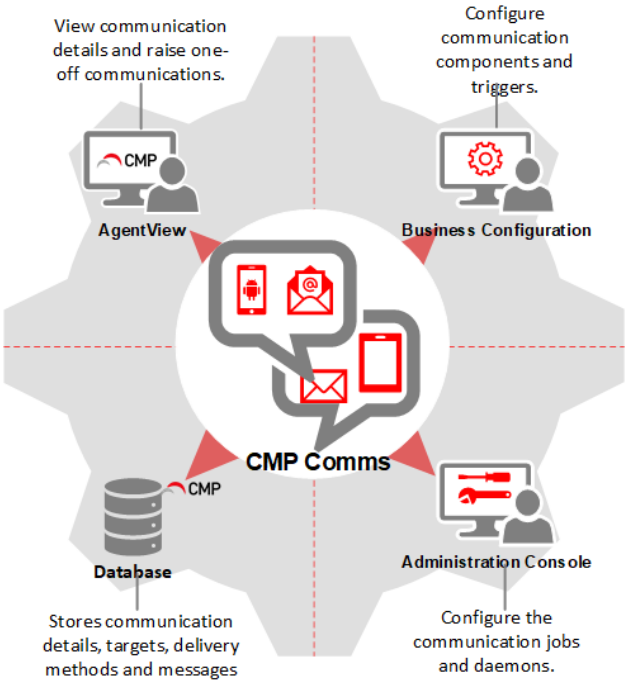
Terms Used in this Document

For definitions and explanations of the terms, abbreviations and acronyms used in this document, please see the *CMP Glossary* document.

1.0 About Communications

CMP supports outbound communication to the customer. The CMP Communications module handles all types of delivery methods for communications: external systems, emails, letters, SMS, and Android/Apple push notifications. It can also support multiple languages.

Different CMP components contribute to the communications process:



- Database

  The main avenue for communications in CMP is the Comms Request, which is an entry in the CommsRequestHeader data table. Many CMP database tables are involved in the communications process. See "Communications Database Tables" on page 59 for more information.

- Business Configuration

  In the Business Configuration console, you can configure the communications themselves, by creating and editing communication components, triggers, preferences, time exclusions, default fields and aliases. See " Comms Configuration: Business Configuration" on page 3 for more information.

- Administration Console

  In the Administration Console you can monitor and maintain the jobs and daemons associated with the communications process. This includes configuring schedules,

enabling triggers or defining storage locations in the daemon properties. See "Comms Configuration - Administration Console" on page 8.

- AgentView

  In AgentView, CSAs can raise the workflows that automatically trigger communications, add one-off communications to a workflow event, view and configure communication preferences, and view the communications associated with an account or subscription. See "Communications in AgentView" on page 75 and the online help in the AgentView application for more information.

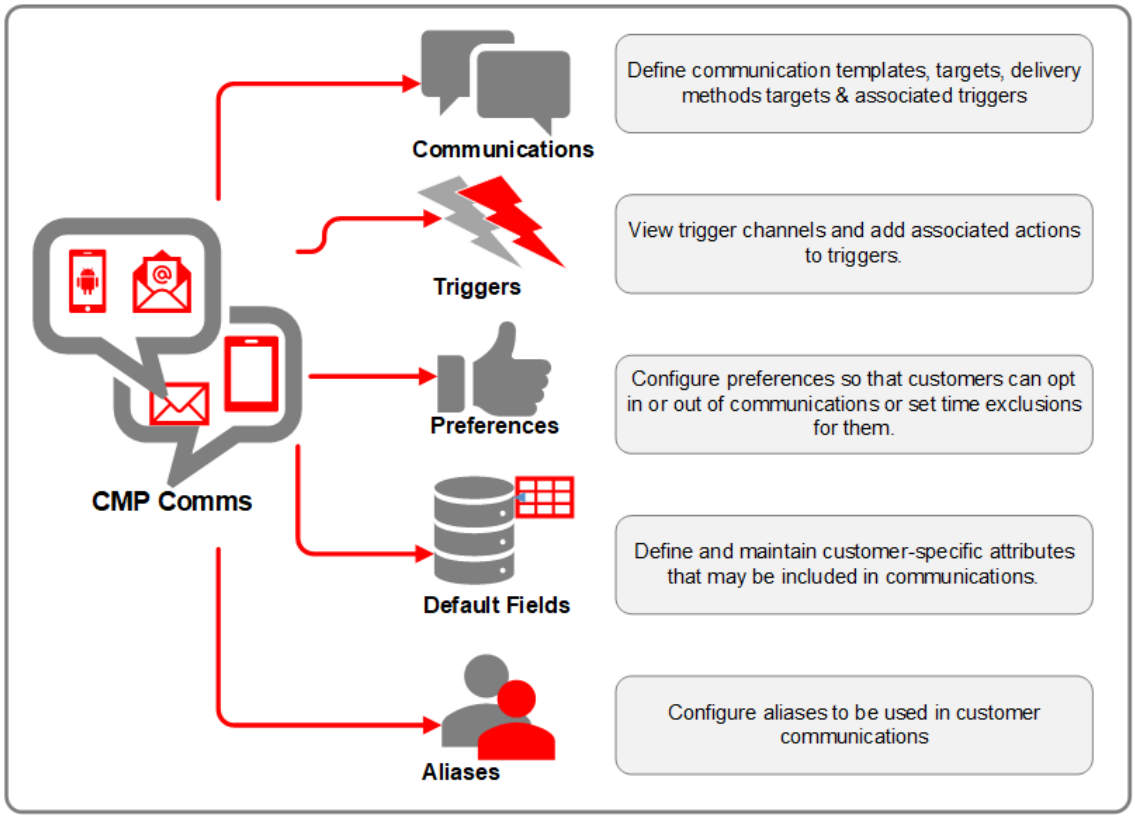
2.0 Communications Configuration

To set up communications in CMP, you need to:

- Configure communication templates, triggers, preferences, aliases and default fields in the Business Configuration console. See " Comms Configuration: Business Configuration" below.
- Configure the `sabre-comms` module. This module allows you to configure a range of properties needed for CMP 8 communications, for example SMTP server details, server directory locations needed for files used in the communications process, and enabling/disabling of communications triggers. You can configure the module in the Administration Console. See "Comms Configuration - Administration Console" on page 8
- Ensure you have set up the required software for creating email and letter templates - see "Working with Email and Letter Templates" on page 61.

2. 1 Comms Configuration: Business Configuration

The following diagram shows the components that you can configure for communications in the Business Configuration console:



The sections that follow describe the components in more detail.

2. 2 Communications Templates (Comms Codes)

To define communications to customers in CMP, you create communication templates (comms codes) via communications configuration. A communication template defines the following:

- The code and description for the communication.
- The CMP hierarchy level at which the communication can be invoked.
- The delivery method - for example email, SMS, letter, or external system. Once you configure a delivery method, you can configure the form and content of the message by specifying the static text and default fields to include in the message. Depending on the delivery method, you can also specify targets, external systems, email subjects and languages for the communication.
- The trigger - whether internally from CMP or from an external source.
- Preferences - for example, marketing material sent via email.
- Whether the communication is visible and selectable in AgentView.
- Whether the communication is Active, that is available to be used for new subscribers.
- The sender and the recipient(s) - or target.

  When the console is operating in B2B mode (system property CONSUMER.MODE = false) and an email, SMS, or push communication is defined at agreement level, the target for a communication can be the agreement administrator that is defined for the relevant agreement. For example a company could have an employee who handles all financial matters for mobile phones, so all communications about invoices and payments could be sent to them as the agreement administrator.

For more information on creating a communication, see the online help for the Business Configuration console.

Some communication formats, for example letters and emails, must be associated with an additional external template that determines the format, style and layout of the correspondence. Templates contain elements such as static text, images, headers and footers, and dynamic text or "merge fields", which can be populated with customer-specific data from the CMP default fields to create personalised communications.

For information on configuring the date format for SMS and push notifications, see "Configuring Date Formats - Administration Console" on page 9.

For more information, see "Communication Email and Letter Templates" on page 60.

2. 3 Communications Triggers

A trigger is an event from CMP or an external source that results in a communication being sent to a customer.

Typically, triggers are pre-configured based on the specific CMP deployment. In Business Configuration, you can view trigger channels[1] and configure associated actions.

Trigger actions can be:

- Communications (Comms) - a communication such as an email, push notification or letter.
- Workflow events - such as an account movement notification.
- Other - for example, a communications request from an external system, such as a provisioning notification.

When the console is operating in B2B mode (system property CONSUMER.MODE = false) and the trigger action is defined at subscription level, you can define a trigger action that will send a communication to the agreement administrator that is defined for the relevant agreement.

When you configure a trigger action, you:

- Supply a trigger ID and description.
- Indicate the level at which the comm is raised is either a subscriber or account.
- Set a priority for the trigger action.
- Select the associated action.

  Actions can be either comms or workflow events. If the trigger action is at Subscription level, you can configure both an action and an administrator action.

For more information on creating a communication, see the online help for the Business Configuration console.

2. 4 Communications Preferences

In Business Configuration, you can define preferences that allow you to create profiles for whether and when a customer wants to receive communications. You can configure a preference for a communication that enables a customer to:

- Opt in to receiving the communication by default.
- Opt out of receiving the communication.
- Request a time exclusion for the communication - a time period in which they will not receive the communication.

You can also configure a preference as the default preference and whether it is available.

[1] Trigger channels are the sources for triggers. They can be internal (CMP) or external.

Communication preferences are available at account and subscription level. Preferences such as language, special needs requirements, preferred delivery method, and time exclusions are visible on the same AgentView screen as the preferences for each communications preference.

Opt In and Opt Out are based on the customer's communication preference at the level in the CMP hierarchy at which the communication was raised. However, if no preference exists at subscription level, Opt In and Opt Out will go up from subscription to account level. Opt In/Out is not available at corporate or group level.

Time exclusion is based on the level in the CMP hierarchy to which the communication is being sent. However it will go up from subscription to account if no subscription preference exists.

For more information on configuring a preference, see the online help for the Business Configuration console.

2. 5 Communications Aliases

You can define aliases to be used in email, push, and SMS communications to end users. For example, instead of displaying a full email address, you can configure an alias that is your company name. To configure an alias, you:

- Provide the alias.
- Select the delivery method.
- Depending on the delivery method, provide one the following:
  - For push methods, the identity of the sender.
  - For email notifications, the email address.
  - For SMS notifications, the telephone number.
- Specify whether the alias is the default to be used.
- Specify whether the alias is to be available to subscriptions and accounts, i.e. active.

2. 6 Communications Default Fields

Default fields contain customer-specific information from the CMP database that can be included in communications with end users. Business Configuration allows you to define and maintain these default fields.

Default fields are associated with beans, which are the software objects that retrieve the data from CMP. When CMP processes a communication, it checks which field needs to be populated for that particular communication and then invokes the associated beans, which fetch the data from the CMP database according to the configured parameters. The bean parameter identifies the exact piece of information that the bean should retrieve. For example, the parameter SubscriptionUserName to the *Subscription* bean is used to retrieve the SubscriptionUserName field from the Subscription table. Parameters

can be any customer-specific attribute supported by the bean, such as a field, static text, a value, or a date. For example the WORKFLOWBEAN supports the following parameters:

- Event
- Total SL Adjustments
- Resolved Date
- Resolution Required By Date.

When CMP is being used in conjunction with an Online Charging System (OCS), the OCS sends notifications to subscribers when they have exceeded their allowance threshold. The CMP OCSNOTIFICATIONATTRIBUTES bean allows the attributes provided within the Openet notification to be used to enrich the outbound communication with subscriber information held within CMP. The following CMP attributes can be used:

- Allowance Description
- Allowance Type (Data, Voice, Text or Cash)
- Spend Cap Description
- Formatted values for entitlement size, usage amount and remaining quota. Formatting is based on the type of allowance.

Default fields can be configured with beans that have multiple parameters. When you supply multiple parameters for a bean, they must be separated with commas.

Configuration of a default field requires the following:

- Supplying a code and description for the default field.
- Choosing a category for the information in the field, for example static text, account or financial.
- Selecting the levels in the customer hierarchy at which the field is supported.
- Selecting the bean (the software object) that will retrieve the field information from CMP.
- Indicating whether the data source is internal or external.
- Specifying a parameter that tells the bean what information to retrieve. Beans can have multiple parameters.
- Specifying a maximum length for the field. This is used to calculate whether an SMS message with fields will be longer than the 140 characters valid for a single SMS.
- Specifying whether the field is active.

For more information, see the CMP Business Configuration Overview. For more information on configuring default fields, see the online help for the Business Configuration console.

Some communication formats, for example letters and emails, must be associated with an additional external template that determines the format, style and layout of the correspondence. Templates contain elements such as static text, images, headers and footers, and dynamic text or *merge fields*. The merge fields are populated with the attributes from the CMP default fields. When you compose an external template, you must

reference the default fields correctly. For information on how to do this, see "Working with Email and Letter Templates" on page 61.

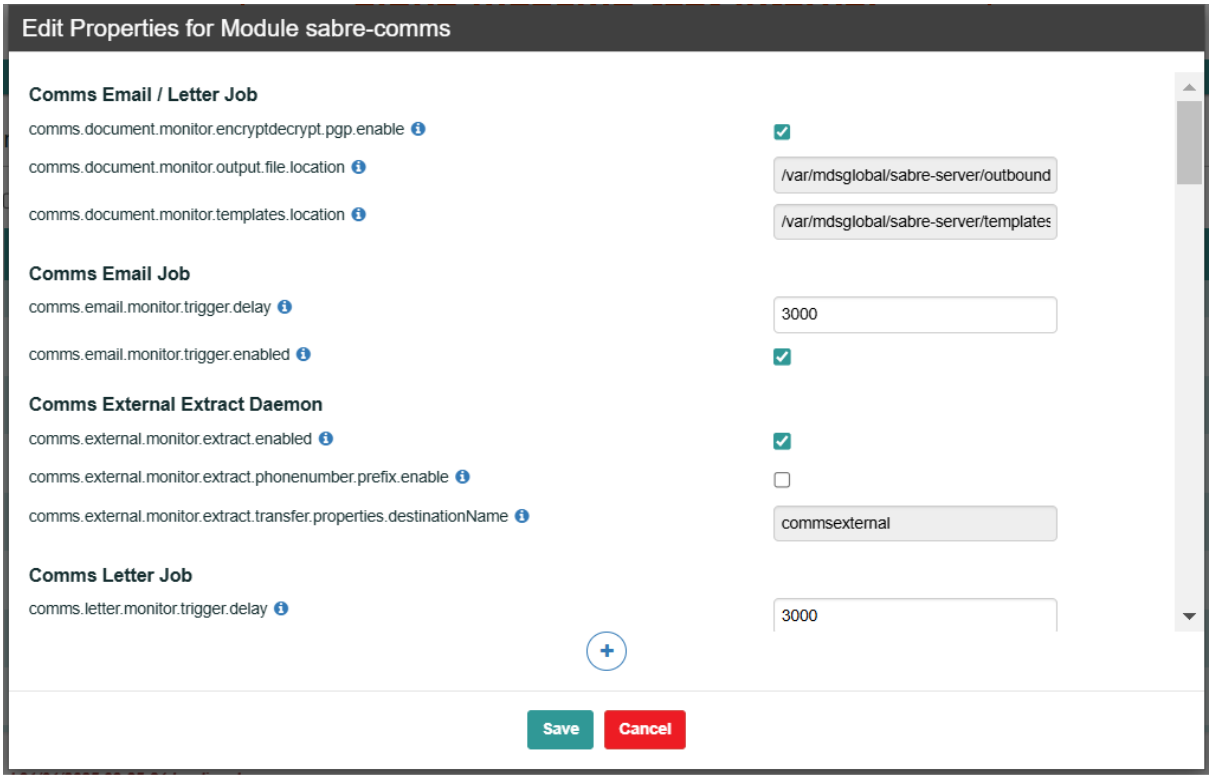
2. 7 Comms Configuration - Administration Console

You will need to configure the `sabre-comms` module. This module allows you to configure a range of properties needed for CMP 8 communications, for example SMTP server details, server directory locations needed for files used in the communications process, and the enabling/disabling of communications triggers.

A list of the sabre-comms properties is available in the section sabre-comms Module Properties.

The sabre-comms module includes the properties for the batch jobs and daemons that handle communications. A list of the jobs and daemons, plus their descriptions and properties, is available in the section "Communications Jobs and Daemons" on page 10.

You can configure the job and daemon properties in the appropriate module (sabre-comms and others) in **System Configuration** in the Administration Console:

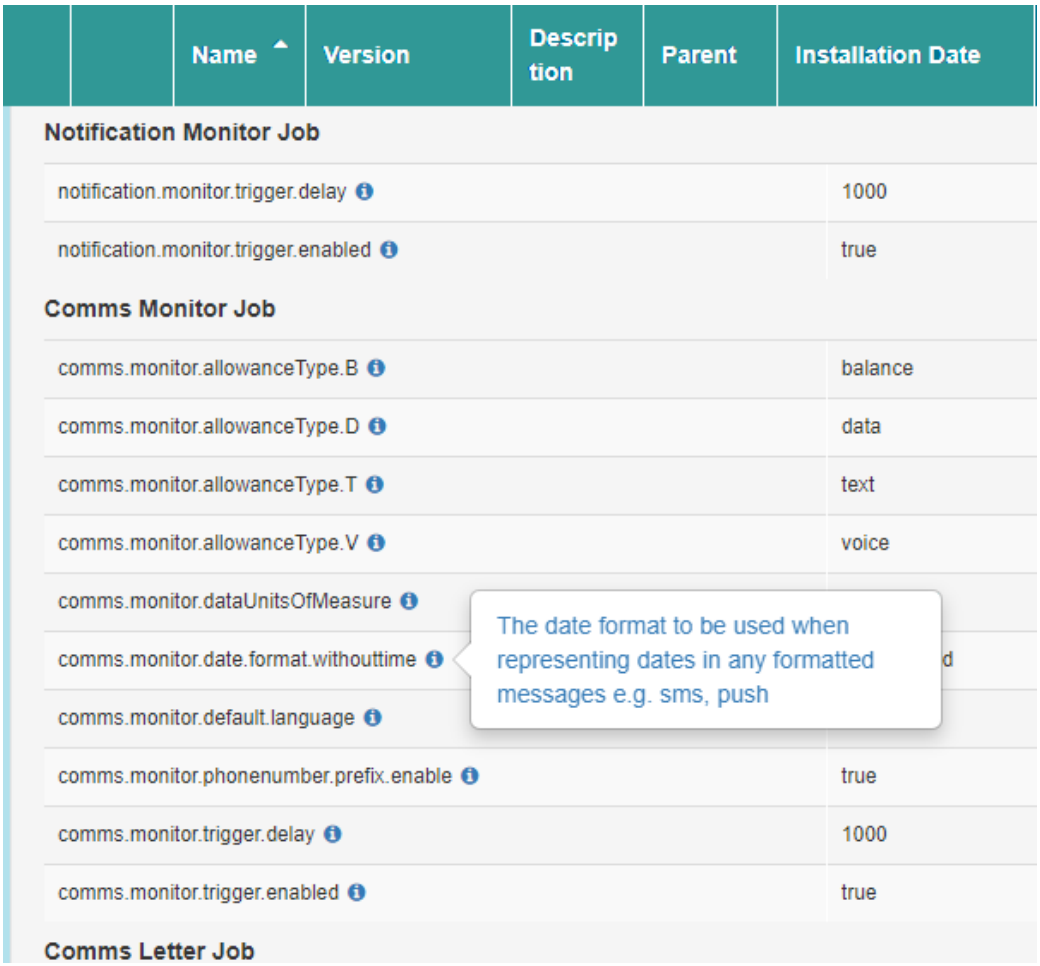


For more information, consult the online help for the console.

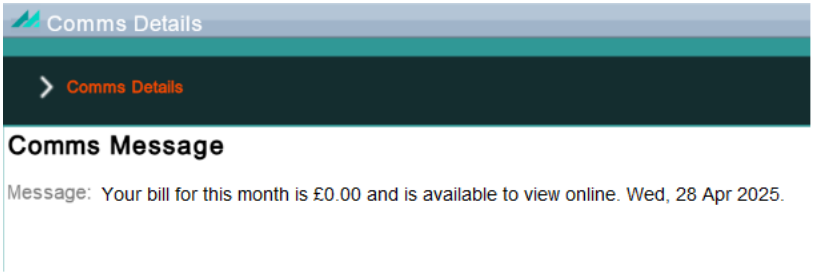
2. 8 Configuring Date Formats - Administration Console

In the **Modules** screen of the Administration Console, you can set the format for dates that appear in SMS and push notifications. This format will apply to all dates in SMS messages and notifications; you cannot set individual formats.

To set the date format, adjust the setting in the `sabre-comms` module for the `comms.monitor.date.format.withouttime` parameter for the Comms Monitor job:



For example, a setting of `EEE, d MMM yyyy`, produces the date format shown here:



The supported date formats are listed here.

2. 8. 1 Communications Jobs and Daemons

Every job and daemon in the Administration Console has a Details page that provides information on parameters, properties, errors, exceptions and configuration. Consult this information for help configuring the jobs and daemons that handle CMP communications.

The batch jobs that control communications are as follows:

2.8.1.1 Comms Monitor

CMP supports sending communication directly to customers (by letter, email, text, SMS, or push notifications) and indirectly. This job gathers the details to be included in each individual communication and determines the delivery mechanism, destination and delivery time.

2.8.1.2 Comms Email Monitor

Email communications require additional processing to merge data gathered from CMP with document templates to generate the required document type. The Comms Monitor job gathers the data; the Comms Email Monitor job merges the data.

2.8.1.3 Comms Letter Monitor

Letter communications require additional processing to merge data gathered from CMP with document templates to generate the required document type. The Comms Monitor job gathers the data; the Comms Letter Monitor job merges the data.

2.8.1.4 Notifications Monitor

Third parties, for example an Online Charging System (OCS), can generate notifications that may be of interest to CMP and cause it to send a communication (Comms) to a subscription. This job converts notifications received from external systems into Comms Requests. Depending on configuration, this job may also raise a workflow instead of or in addition to the Comms Request. This job is triggered automatically when a notification is created.

2.8.1.5 Advance Notification

This job issues comms to subscriptions relating to certain events that are due to happen which are of interest to a subscription. At the moment, the job supports the issue of

comms relating to upcoming expiry of credit/debit cards and contracts and advance notification of when a payment will be taken from a bank account or credit/debit card.

The daemons that handle comms are as follows:

2.8.1.6 Load Comms From Generic Format

This daemon is responsible for the collection and decryption of generic CMP files and messages and the creation of CMP batches which are available for processing into CMP by the appropriate batch job.

2.8.1.7 Transmission Comms to Document Storage

The daemon is responsible for placing Letters and Emails in long term storage to be accessible via AgentView.

2.8.1.8 Transmission Comms Push To Handset

This daemon picks up entries that are ready to be pushed to a handset. These messages will contain the full text to be sent, and sent within an encrypted JSON message.

2.8.1.9 Acknowledge Comms Receipt Of

This daemon takes in a JSON message/file and then updates various values of a particular comms e.g. an update on the status of a comm from the third party that actually sent it.

2.8.1.10 Transmission Comms to Print Bureau

This daemon zips up a directory of letters and sends it to a target location.

2.8.1.11 Extract Comms to Generic Format

This daemon creates and encrypts generic CMP file that are available for conversion to third party format.

2.8.1.12 Transmission Comms SMS To Handset

This daemon picks up entries that are ready to be sent via SMS. These messages will contain the full text to be sent, and sent within an encrypted JSON message.

2.8.1.13 Transmission Comms To Email

This daemon picks up documents that have to be emailed and sends them to the specified email recipient.

2.8.1.14 Transmission Comms to SMPP

This adapter interrogates the queue where SMS messages are placed, and sends them to a Short Message Peer-to-Peer Protocol (SMPP) Server.

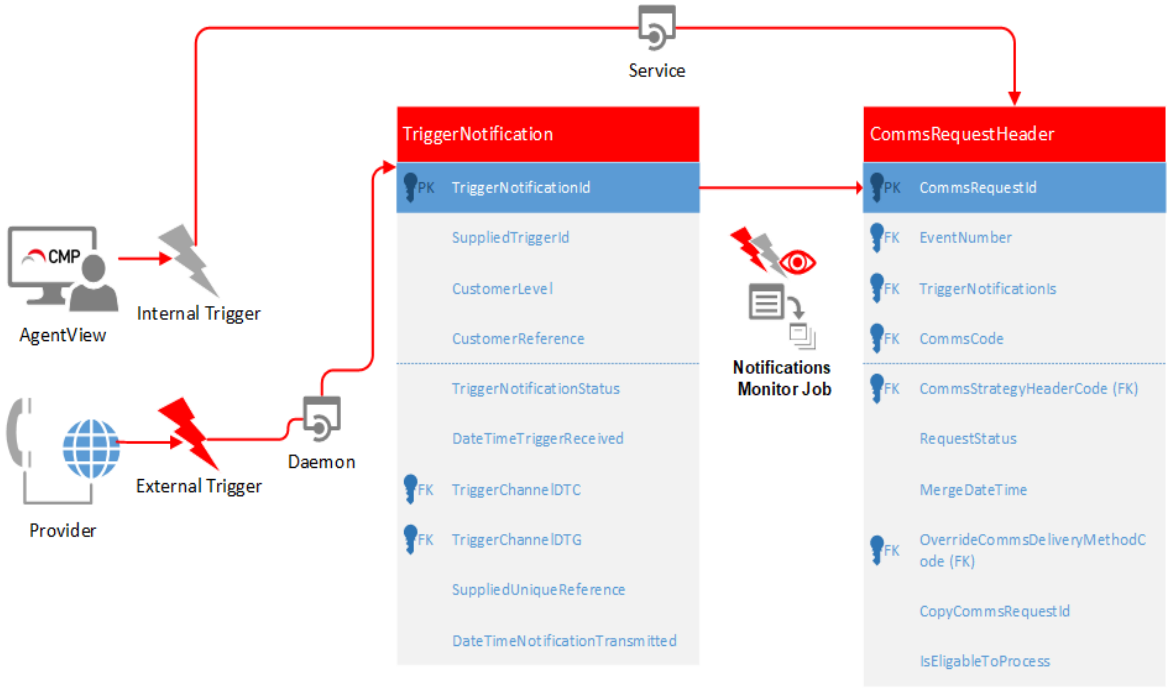
3.0 Communications Process Flow

Each communication results in the communication being sent to one target. Multiple targets, such as sending an email to all subscriptions on an account, is not currently supported. The main avenue for creating communciations in CMP are Comms Requests, which are entries in the CommsRequestHeader table.

The following is an overview of the communications process. Later sections in this chapter go into more detail.

The communications process involves a number of steps:

Step 1: A communication request is triggered



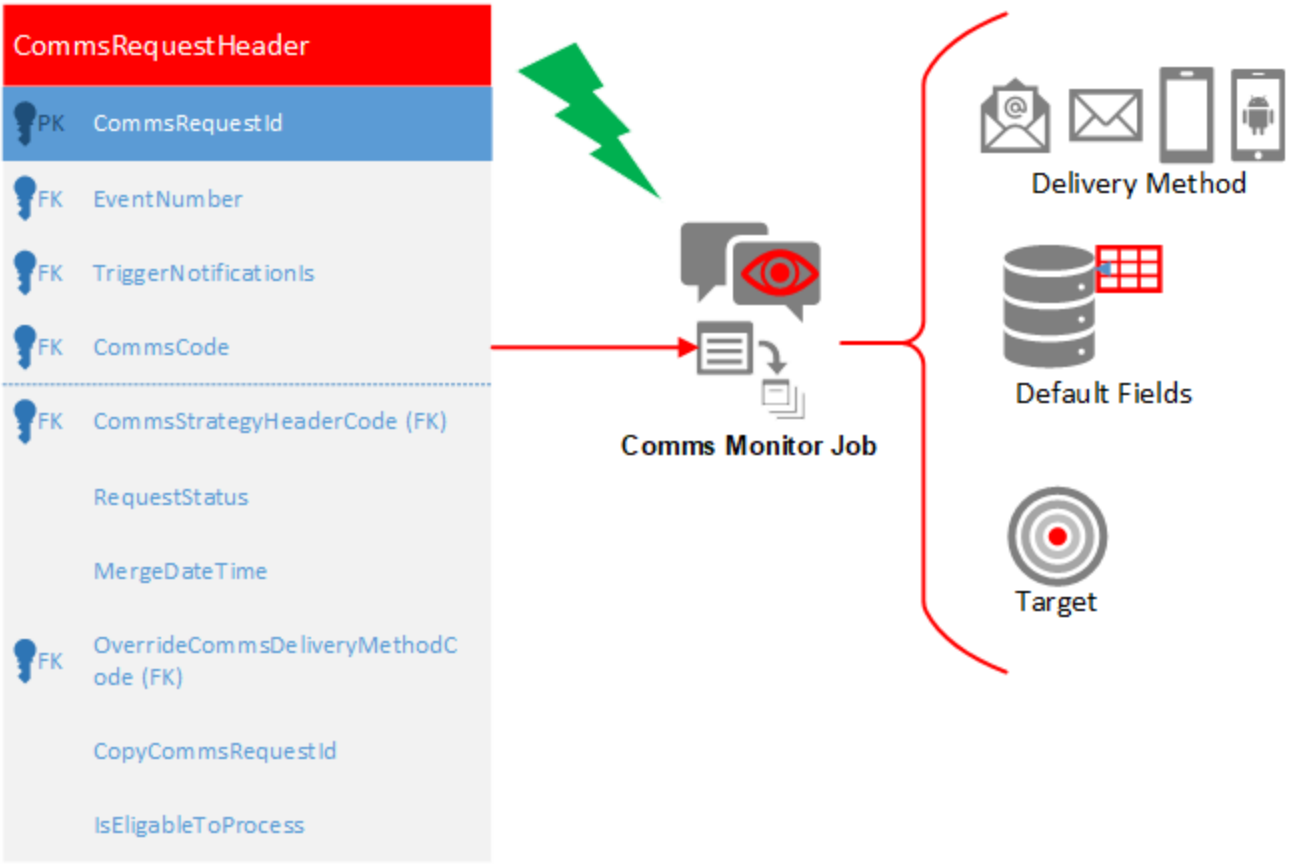
The Communications process starts when a communication request is triggered. A trigger can be:

- Internal - for example a CMP workflow event, raised either manually or automatically
- External - a request from an external system. A communication request can be created automatically on receipt of a notification from an external system.

When a trigger is invoked, CMP creates an entry in the core table in Communications, the CommsRequestHeader table.

When a Workflow Event is created, irrespective on whether the WhenToCreate is set to OnCreate or OnResolve, an entry in created in the Comms Request Header.

Step 2: Communication details are gathered



When an entry is created in the CommsRequestHeader table, the Comms Monitor Job is triggered to process the CommsRequestHeader and produce a CommsRequestDetail entry. This job:

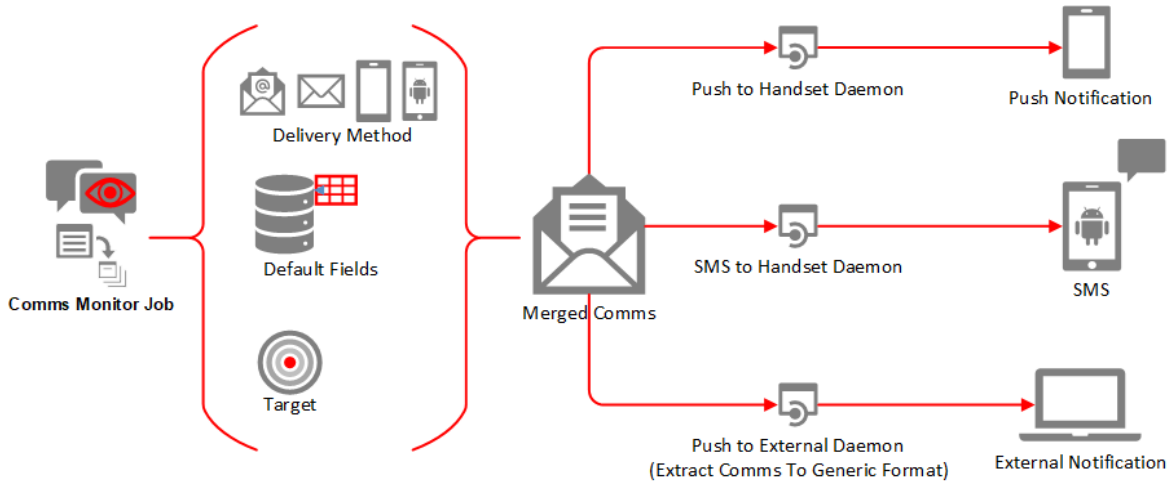
- Works out the appropriate delivery method and target, for example a mobile number, email address etc.
- Derives the values of the default fields[1] that are associated with the communication.
- In the case of push notifications, online notifications and SMS delivery methods, the job also merges the data of the derived fields with the message template to produce

[1]Default fields are contain customer-specific attributes stored in the CMP database. This customer data can be merged into communications. For example, a customer account number and account balance can be sent in a bill reminder SMS.

the actual message. For letters and email, this merging is carried out by separate jobs.

For more information, see "Comms Process Flow: Gather Details" on page 21.

Step 3: SMS, push and external notifications are sent

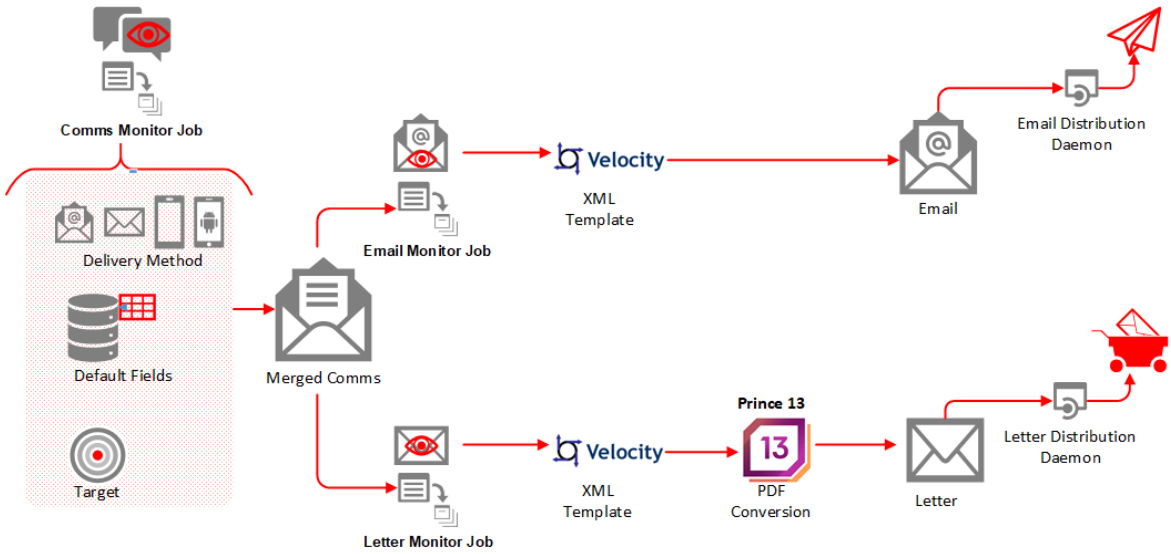


After the Comms Monitor Job has completed, dedicated daemons pick up the requests to distribute them to the destination:

- Comms SMS Daemon
- Comms Push To Handset Daemon
- Comms Push To External Daemon

For more information, see "Comms Process Flow: Send SMS, Push and External Notifications" on page 42.

Step 4: Letters and emails are created and sent



Email and letter communications require additional processing to merge data from default fields gathered from CMP with document templates to generate the required document type. The Comms Monitor job gathers the data; the Comms Email Monitor and the Comms Letter Monitor jobs merge the data.

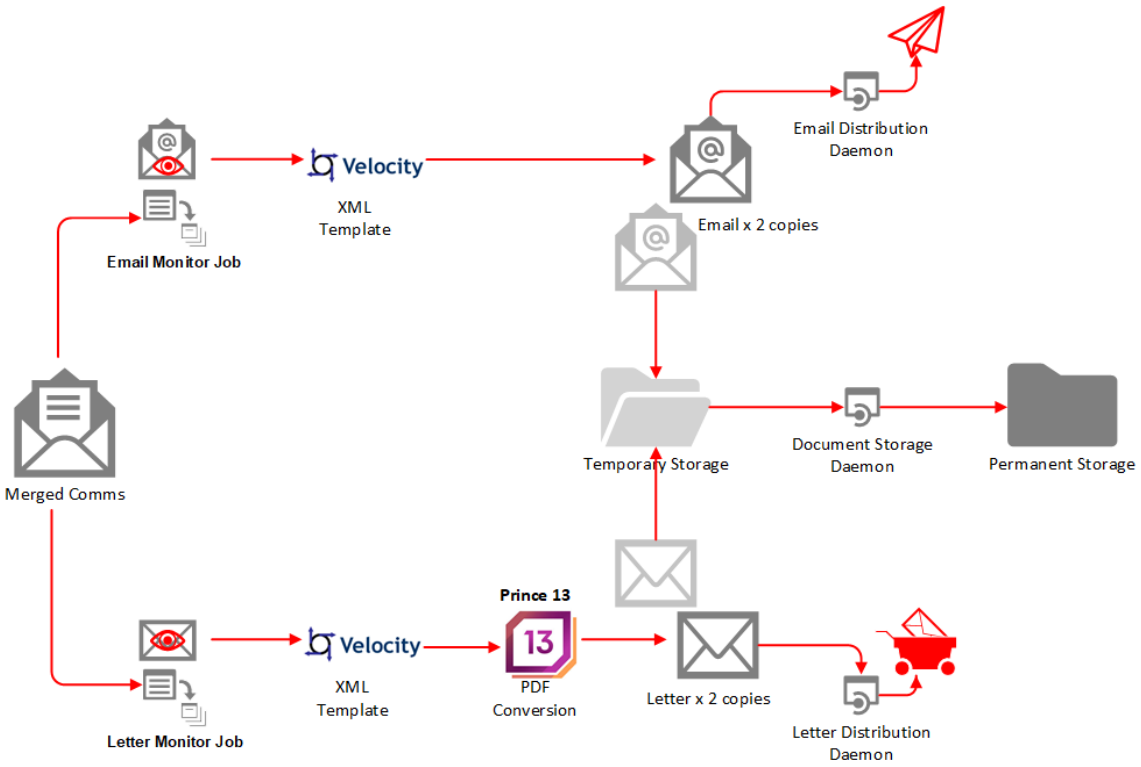
The Comms Email Monitor and Comms Letter Monitor jobs are triggered automatically when the Comms Monitor job has finished gathering data for Comms Requests relating to emails or letters. For more information, see "Comms Process Flow: Create and Send Letters and Emails" on page 46

These jobs use external frameworks - for example, Velocity and Prince XML - to generate documents. Generated documents are subsequently detected by the following downstream daemons for transmission to the customer:

- Comms Email Daemon
- Comms Letter Daemon

For more information, see About Email and Letter Templates.

Step 5: Letters and Emails are Stored



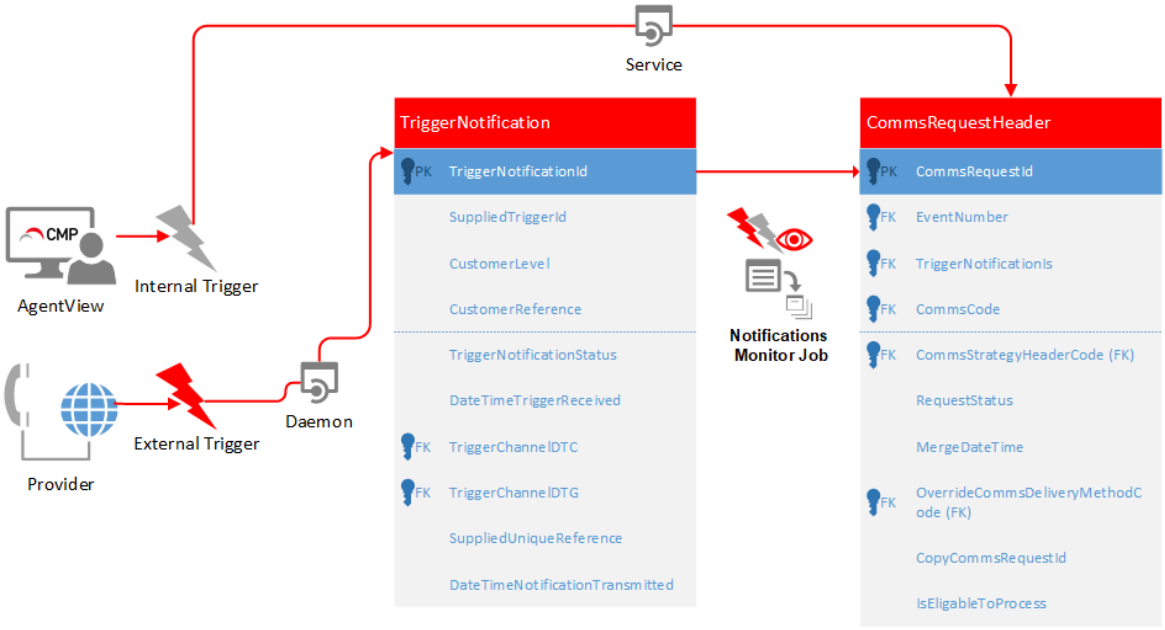
This is a distinct step carried out by the Comms Transmission to Document Storage Daemon. This allows the documents to be placed into storage on any machine, for example in systems such as Amazon S3 or on a machine via SFTP. Storing the documents in a long-term storage directory allows them to be viewed in AgentView.

For more information see "Comms Process Flow: Store Letters and Emails" on page 56.

External Systems Update Communications Status

A daemon allows external systems to update the status in CMP of communications that were sent externally for processing and/or distribution. For more information, see "External Comms Status Updates" on page 57.

3. 1 Comms Process Flow: Triggers



The Communications (Comms) process starts when a communication request is triggered. When a trigger is invoked, CMP creates an entry in the core table in Comms, the CommsRequestHeader table. A trigger can be:

- Internal - for example a CMP workflow event, raised either manually or automatically
- External - a request from an external system.

The following sections describe the process for internal and external triggers in more detail.

3. 1. 1 Process Flow for External Comms Triggers

Comms requests can be created in CMP on receipt of notifications from external systems, for example an Online Charging System (OCS) can alert a customer when they are close to consuming all their data allowance. The notifications are formatted according to a JSON schema, `commsExternalTrigger.schema`. Communications from external systems are handled by the Load Comms from Generic Format daemon (also known the External Trigger daemon) and the Notification Monitor batch job.

> For more information on daemons and batch jobs see "Communications Jobs and Daemons" on page 10 and the *CMP Batch Jobs and JSON Schemas Guide*.

The Load Comms from Generic Format daemon is subscribed to the inbound message queue and is notified when a message arrived. The URL for the message queue is set by the daemon properties:

Property	Description	Example
comms.external.monitor.load.transfer.properties.brokerUrl	The URL of the ActiveMQ instance onto which inbound external comm triggers are placed e.g. module-properties-help.camel.ACTIVEMQ.brokerurl.	tcp://127.0.0.1:6161-6
comms.external.-monitor.load.transfer.properties.destinationName	The name of the queue onto which inbound external comm triggers are placed.	inboundCommsExternal

> For more information see "Load Comms From Generic Format" on page 11 in *Communications Jobs and Daemons*.

When the daemon receives a notification, a record is written into the TriggerNotification table.

Once the triggerNotification table is updated, the Notification Monitor job can either be triggered automatically or a user can run the job manually.

> See the *CMP Operations Overview* and the *Administration Console Help* for more information on the job.

The job cross-references the record (SuppliedTriggerId value) from the TriggerNotification table to the TriggerID value in the triggerAction table to see what comms need to be sent. The job can either send a comms request, raise a workflow event, or do both. The request depends on the configuration of the comms code.

> For more information on comms codes, see "Communications Templates (Comms Codes)" on page 4.

If CommsCode is populated in the triggerAction table, the job creates an entry in the CommsRequestHeader table.

If a workflow event is configured against the TriggerID in the TriggerAction table, the workflow will be raised against the supplied customer reference (Account or Subscription) supplied in the original JSON file, using the Event Type and Event Code from the TriggerAction entry.

Once the request has been completed, the job updates the TriggerNotificationStatus in the TriggerNotification table to P (Processed).

If the SuppliedTriggerID value does not match any TriggerID between the TriggerNotification and TriggerAction tables, an error is raised. The job creates an error in the TriggerNotificationError table.

Finally, the job updates the TriggerNotificationStatus in the TriggerNotification table to E (Error).

3. 1. 2 Process Flow for Internal Comms Triggers

In CMP, much communication is currently driven by workflows:

- Comms can be automatically raised when a workflow is created or resolved, depending on the configuration of the workflow event.
- A one-off comms can be manually added to a workflow event via AgentView.

3.1.2.1 Process Flow for Comms Raised when A Workflow Event is Created or Resolved

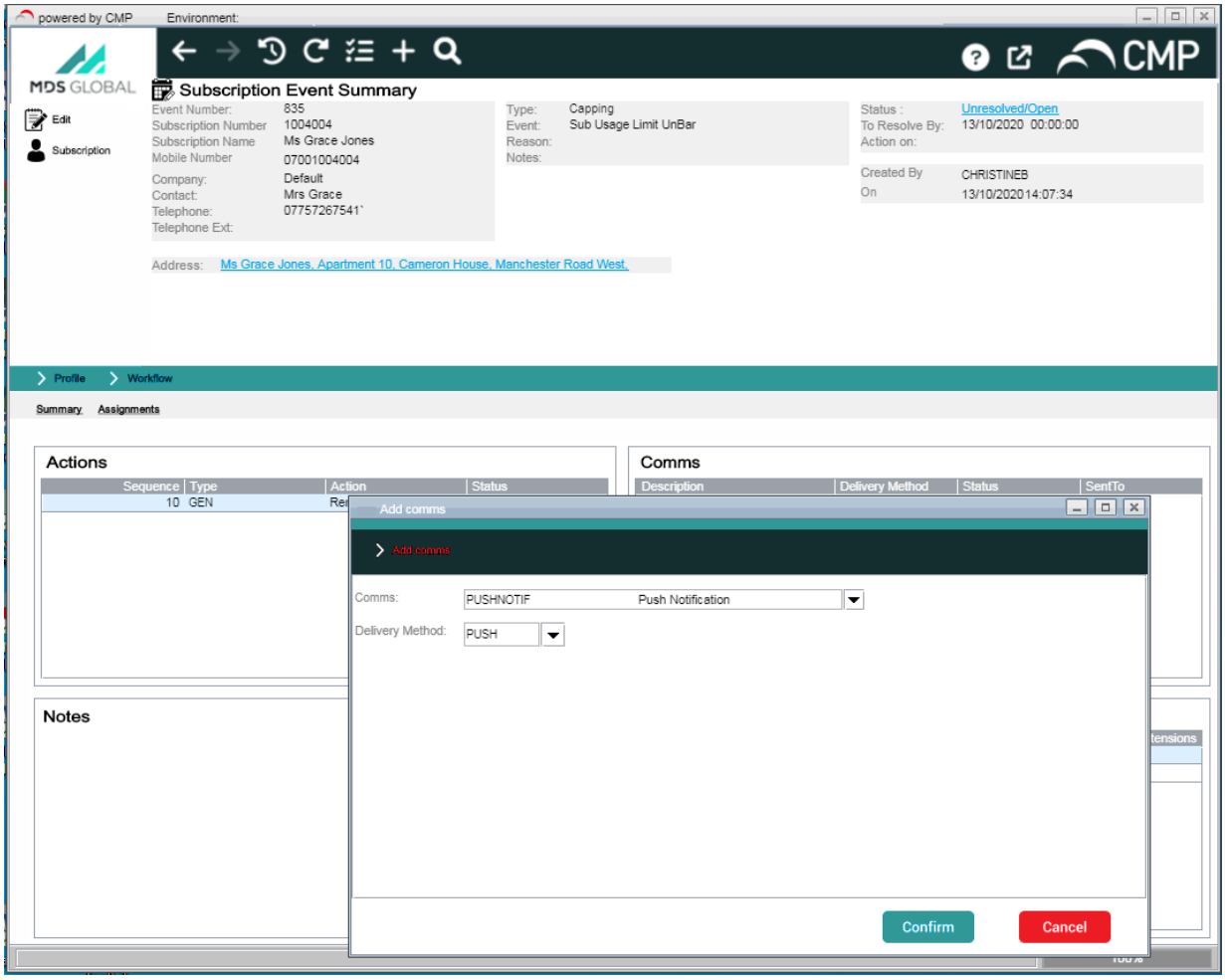
When a workflow event that is configured to raise a comm is created, the Workflow Monitor job checks the following tables:

- DiaryEventCodeComms
- CommsCode

For each entry that has a comms code populated for the Workflows Event Type and Event Code in the DiaryEventCodeComms table, the job creates an entry in the CommsRequestHeader table. The comms request is created when the workflow event is created regardless of whether the comms is to be sent on creation or resolution. However the status of the comms request is different.

3.1.2.2 Process Flow when a Comm is Manually Added via AgentView

A user can add a one-off comms to a workflow event via AgentView. To do this, in the lower panel of the **Event Summary** screen, the user right-clicks in the **Comms** pane and clicks **Add**. This opens **Add Comms** pop-up.



Event Summary screen and the Add Comms pop-up

In the Add Comms pop-up, the user selects a comms from the Comms drop-down. The drop-down is populated with a list of comms taken from the CommsCode table that meet the following criteria - the comms is:

- Available and selectable.
- Applicable to the hierarchy at which the workflow event was raised.
- At a security level greater than or equal to that of the user.

Once the user selects a comms, CMP identifies the list of delivery methods from the CommsCodeCommsDeliveryMethodCode table and populates the Delivery Method drop-down.

If the user selects EMAIL as a delivery method, CMP checks whether an email address exists at the level where the comms is being raised. If no email address exists, an error message is displayed to the user informing them that it is missing and it needs to be added.

Once the comms is created, CMP creates a record in the CommsRequestHeader table.

*Because the user is manually adding the comms at the customer's request, the choice of comms and delivery method is assumed to be deliberate. So when the Comms Monitor job comes to work out the comms and delivery method it can ignore Opt Out logic and Comms Preferences, although time exclusions are taken into consideration.

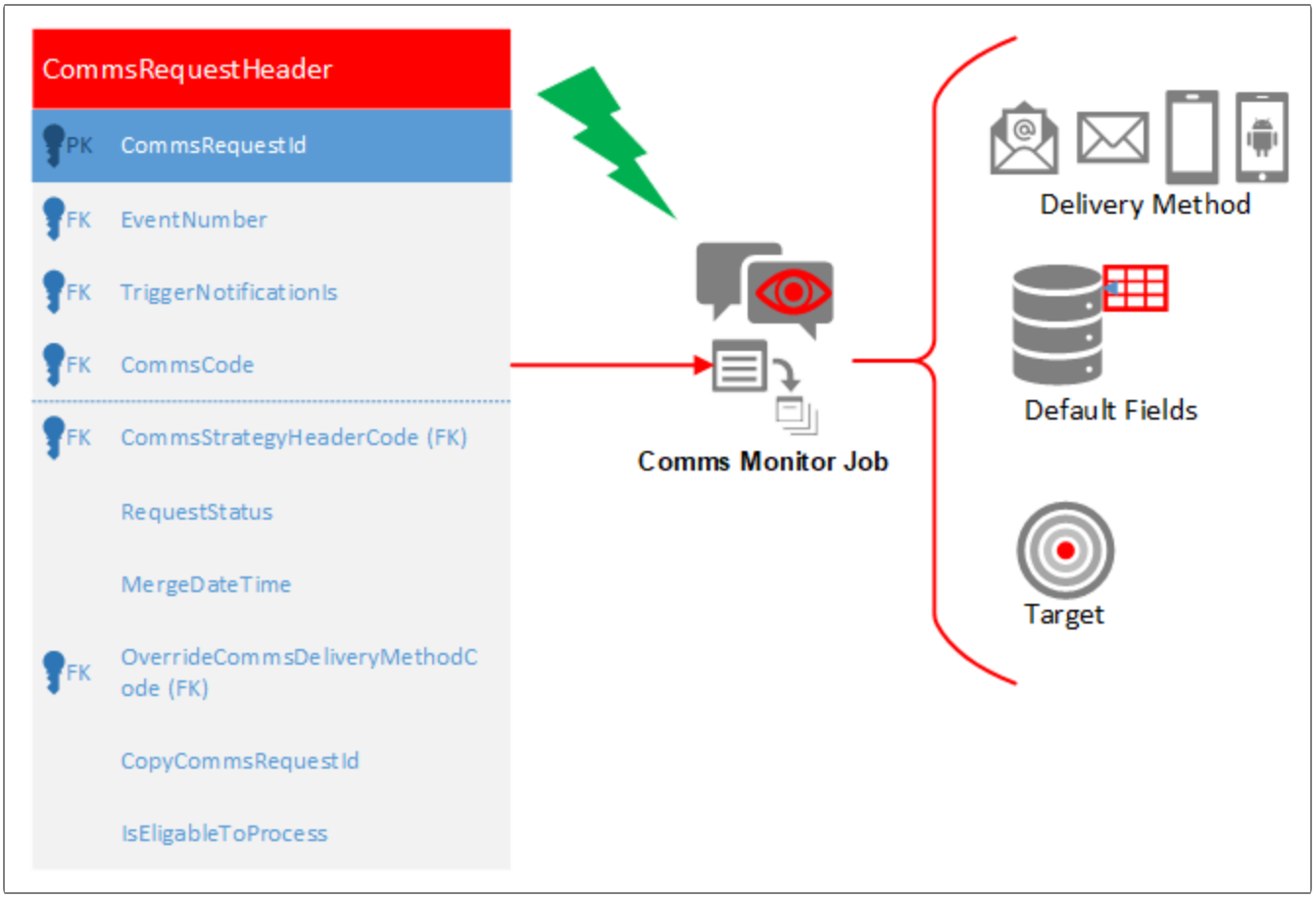
3. 2 Comms Process Flow: Gather Details

When an entry is created in the CommsRequestHeader table, the Comms Monitor Job is triggered to process the comms request and gather and record comms request details. This job:

- Works out the appropriate delivery method. The Delivery Method chosen depends on:
  - The delivery methods that are associated with the comms. So if only one delivery method is configured then that is the chosen delivery method.
  - Whether the customer has opted in to receive comms.
  - The communications preference selected by the customer. This is relevant if there are multiple delivery methods associated with a communication. For example, if SMS and Email are configured as delivery methods, and the customer preference is SMS, then they will receive an SMS.
- Determines the target, for example a mobile number, email address etc.
- Derives the values of the default fields[1] that are associated with the communication.
- Once the details are determined, creates an entry in the CommsRequestTargetDistribution table with the status A (Awaiting Merge).
- In the case of push notifications, online notifications and SMS delivery methods, the job also merges the data of the derived fields with the message template to produce the actual message. For letters and email, this merging is carried out by separate jobs.

The sections that follow describe these steps in more detail.

[1]Default fields contain customer-specific attributes stored in the CMP database. This customer data can be merged into communications. For example, a customer account number and account balance can be sent in a bill reminder SMS.



The Comms Request job gathers details such as the delivery method, targets and default field values.

3. 2. 1 Determine Delivery Method

To determine the delivery method, the Comms Monitor job:

1. Determines if there is a override delivery method.
2. Checks if the customer has opted in or out of comms.
3. Applies the customers Comms Preferences, if any.

3.2.1.1 Determine if Delivery Method is Overridden

First, the job checks whether the OverrideDeliveryMethodCode field in the CommsHeaderRequest table is populated. If so, it uses that delivery method. This happens, for example, with one-off comms added to workflow events.

For more information see "Process Flow when a Comm is Manually Added via AgentView " on page 19.

3.2.1.2 Check if Customer has opted into Comms

Now the jobs checks if the customer has opted in to the Comms. To do this, it checks whether the CommsCode field is populated in the CommsRequestHeader table and then cross-checks the tables:

- CommsCodeCommsDeliveryMethodCode
- CustomerCommsPreference
- CommsPreference

The result is also affected by the level at which the comms is raised - account, agreement or subscription. Corporate and group levels do not have comms preferences associated with them.

If the comms is raised at account level

For comms raised at account level, the job looks for an IsOptIn entry in the CustomerCommsPreference table for this account and Comms PreferenceCode (that is associated with the comms code).

- If there is no entry, the job cross-references the CommsPreferenceCode in the CommsPreference table to check the value for isDefaultOptedIn, that is, whether the default preference for this comms is for the customer to be opted in or out of receiving it.
    - If DefaultOptedIn = FALSE then the job updates the RequestStatus on the CommsRequestHeader table to indicate O (Opted Out) and processing of this comms request stops.
    - isDefaultOptedIn = TRUE then normal processing continues.
- If there is an IsOptIn value in the CustomerCommsPreference table:
    - If IsOptIn = TRUE, normal processing continues.
    - If IsOptIn = FALSE, the job updates the RequestStatus on the CommsRequestHeader table to O (Opted Out) and the processing of this comms request stops.

If the comms is raised at subscription level

For comms raised at subscription level, the job looks for an entry in the CustomerCommsPreference table at subscription level first and if it finds one, bases the Opt In logic on that.

If it does not find an entry at subscription level, the job looks for a CustomerCommsPreference entry at account level. If there is, it uses that.

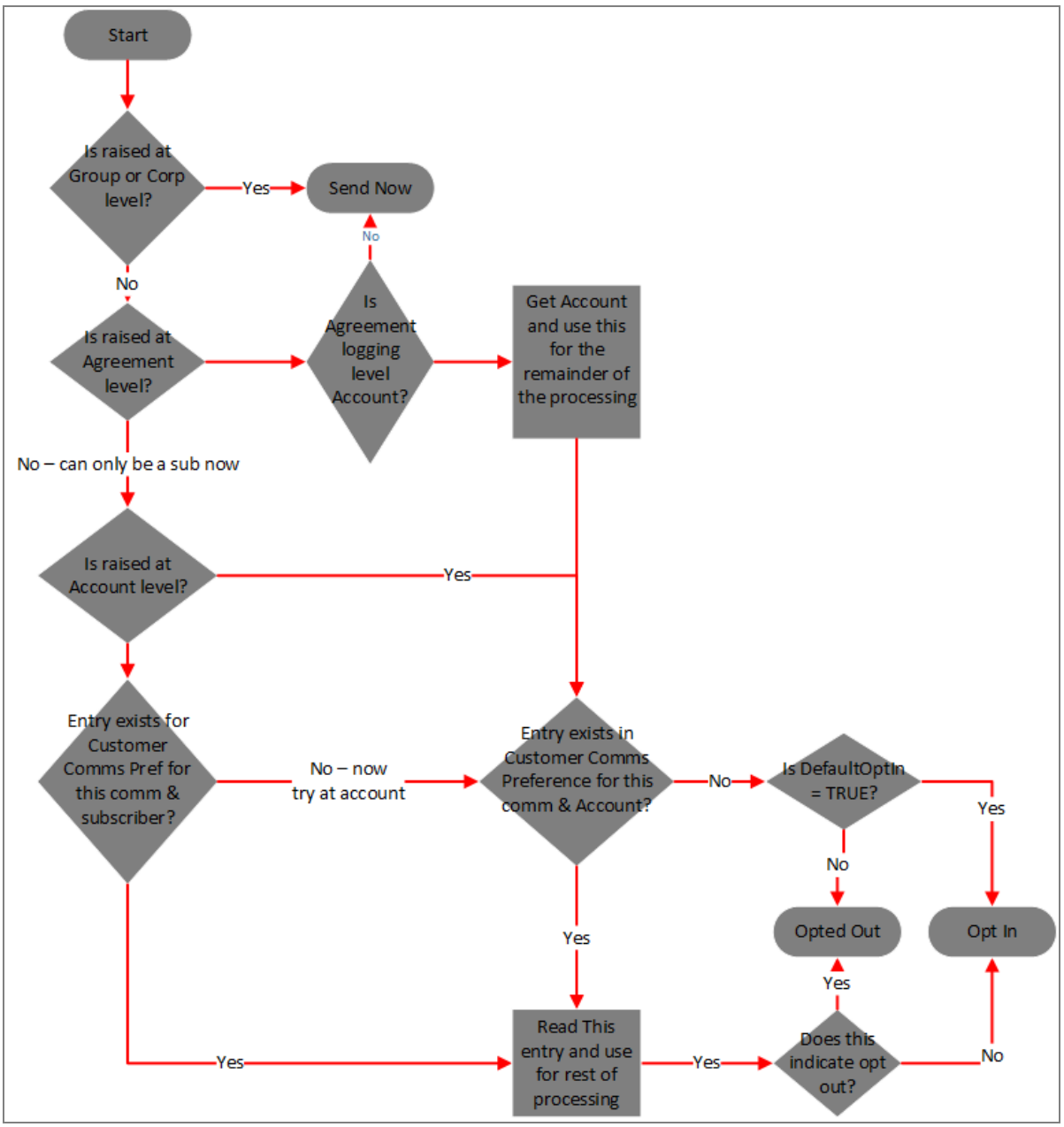
Otherwise, the proceeds is for a comms raised at account level.

If the comms is raised at agreement level

When the comms is raised at agreement level, the job looks up the comms preference based on the agreement logging level. When the agreement logging level is account, the

job uses this account number to perform the Opt In logic as for a comms raised at account level. If the logging level is corporate or group, the job skips the Opt In logic and the comms is sent.

This flowchart describes the logic for determining the Opt In status:



Example - Opt in/Opt Out at Different Logging Levels

Comms are raised at agreement-level for the following:

- Agreement 1 - logged at account level. Account 1 has no customer comms preferences set in the CustomerCommsPreference table. The comms has a comms preference configured so that customers are opted in to receive the comms by default. This means that for this comms, a record is written to the CommsPreference table with IsDefaultedOptedIn value set to 1 (TRUE).
- Agreement 2 - logged at account level. Account 2 which has no customer comms preferences set in the CustomerCommsPreference table. The comms has a comms preference configured so that customers are opted out of receiving the comms by default. This means that for this comms, a record is written to the CommsPreference table with IsDefaultedOptedIn value set to 0 (FALSE).
- Agreement 3 - logged at account level. The comms raised against Account 3. In the CustomerCommsPreference table entry for this account, IsOptedIn = 1 (TRUE), so this account as opted in to receive the comms.
- Agreement 4 - logged at account level. The comms raised against Account 4. In the CustomerCommsPreference table entry for this account, IsOptedIn = 0 (FALSE), so this account as opted in to receive the comms
- Agreement 5 - logged at corporate level.
- Agreement 6 - logged at group level.

Results:

- Agreements 1 and 3: CMP will send the comms to the customer. The account level preferences will be used.
- Agreements 2 and 4: Based on the account preferences, CMP should not send the comms. Therefore, the RequestStatus is updated on the CommsRequestHeader table to O (Opted Out) and the processing of this comms request stops.
- For Agreements 5 and 6: No comms preferences can be set at corporate or group level. Therefore, the comms request will be sent as normal.

3.2.1.3 Get the Delivery Method

Once the Comms Monitor job has determined whether there is an override delivery method and worked out the Opt In status, it gets a list of delivery methods associated with the comm and identifies out the highest priority method to populate the CommsRequestTargetDetail table. To do this, CMP needs to take into account:

Customer preferences for delivery methods

The preference for a particular delivery method is specified in the CommsDeliveryMethodCode in the CustomerCommsProfile table. Currently, CMP supports customer preferences for delivery methods at subscription and account levels. The preference is just that - a preference - and does not guarantee that the customer will get the comms delivered to them via the preferred mechanism. A comms can often have a delivery method that is not the customers preferred delivery method. For example credit control comms are usually sent as letters or SMS for legal reasons. However a customer may have PUSH as their preferred delivery method. In this case, CMP will decide based on a pre-configured global priority to send via either SMS or LETTER.

Delivery methods configured against a comms

The Comms Monitor job checks the CommsCodeCommsDeliveryMethodCode table to get the list of supported comms delivery methods for the particular comm.

Preconfigured priority of the delivery methods

The list of supported comms delivery methods for a particular comm are processed in order of priority as specified on the CommsDeliveryMethod table. The highest priority deliveryMethodCode is stored in the CommsRequestTargetDetail table.

The Comms Monitor job works out the order in which delivery methods are returned as follows:

If the comm is raised at account level

The job determines whether account-level preferences are present. If so, it gets the account customer comms preference. Then the job checks whether the method is supported for the comm. If so, it uses that delivery method. If not, it uses the delivery method with the highest priority for that comm. If there is a customer comms preference set for the account, the job also checks if there is a language set in CustomerCommsProfile. If there is, the job populates the language fields on the CommsRequestTargetDetail table. Otherwise, the job leaves the fields empty. They will be populated later in the process using the default language.

If the comm is raised at subscription level

The Comms Monitor job determines whether subscription-level preferences are present. If so, it gets the subscription customer comms preference, and if that method is supported for the comms, uses it. If it is not supported, the job uses the delivery method with the highest priority for that comm. If the comm is raised at subscription level and there is not a subscription-level preference, the job looks for an account-level preference. If one exists and it is supported by the comm, the job uses the account-level delivery method. If there isn't an account-level comms preference, the job uses the highest priority delivery method for the comm. If the language is set at subscription level, the job populates the language fields on the CommsRequestTargetDetail table with that language. Otherwise, the job uses the language set at account level. If no language is set at either account or subscription level, the job leaves the language fields empty at this point.

If the comms is raised at agreement level

CMP does not have the ability to store comms preference at the agreement level. So if the comms is raised at agreement level, the job determines whether the agreement is logged at the account level. If it is, the job proceeds as if for a comm raised at account level. Otherwise, the job returns the supported delivery method with the highest priority. If a preferred delivery method is set against the account, the job also checks if there is a language set in CustomerCommsProfile. If there is, the job populates the language fields on the CommsRequestTargetDetail table. Otherwise, the job leaves the fields empty. They will be populated later in the process using the default language.

When writing to the CommsRequestTargetDetail table, later on, the job populates the CustomerLevel and CustomerReference with either A (Account) or S (Subscription) and Account Number or Subscription Number respectively, depending on whether it used the CustomerCommsProfile table at subscription or account level.

3.2.1.4 Error Handling

If CMP is unable to send a comm because no suitable target/delivery method was identifiable, then CMP looks to see if a NotSent Event Type and Code are configured against the job as a property. If it is, then CMP raises that event against the customer level at which the comms or notification trigger was raised.

> It is common practice that these events are assigned to a group or a particular CSA in the call centre that handles such scenarios.

3. 2. 2 Get the Delivery Method Target

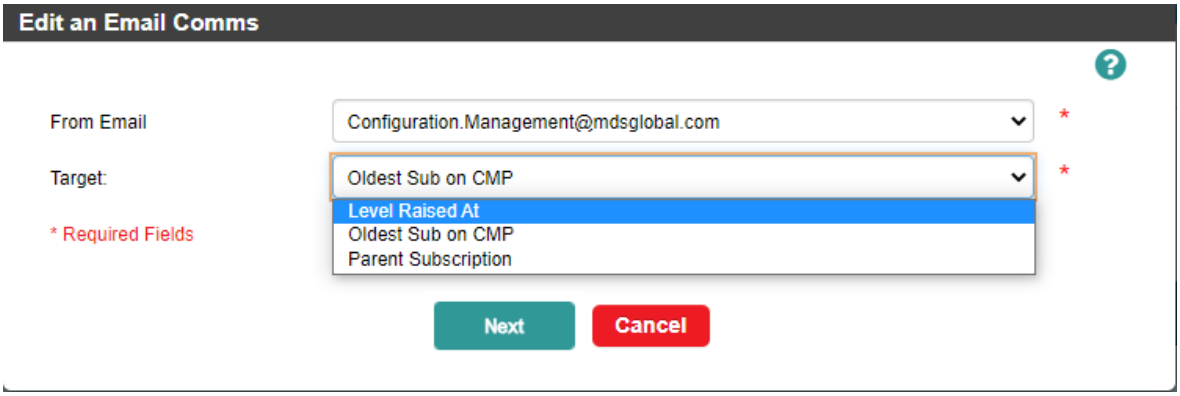
Once the Comms Monitor job has determined the delivery method for the comms, it identifies the target for that delivery method, for example, a phone number, email address, postal address or external system. The Comms Monitor job determines targets for delivery methods in the following order:

1. EMAIL
2. EXTERNAL
3. PUSH or SMS
4. ONLINE
5. LETTER.

3.2.2.1 Delivery Method = Email

When an EMAIL comms is configured in the Business Configuration console, the target set for the comms can be the email address for:

- The level the comms is raised at
- The oldest subscription for that customer
- The parent subscription.



Configuring the target for an EMAIL comms in Business Configuration

Email Target = Raised at Level

The Comms Monitor job checks the CommsCodeCommsDeliveryMethodCode table for an InternalTarget entry. If InternalTarget = L (Raised at Level), the job checks the CommsCode table, which records the level at which the comm was raised - account, subscription, group or corporate.

The logic and workflow is as follows:

1. Is email raised at account level? If so, get account-level email address.
2. Is the email at subscription level? If so, get subscription-level email.
3. Is email raised at corporate level? If so, get corporate-level email.
4. Is email is raised at group level? If so, get group-level email.
5. Is email is raised at agreement level? If so, get the level at which the agreement is logged – Account, Corporate or Group – and get the email for that entity.

The job retrieves the email address from the CustomerEMailAddress table using the appropriate Customer Level and Customer Reference. The email address that it uses is the primary email address (it has the entry isPrimaryEMail = true). If there is only one email address present, this is the primary email.

The job then sets the ToEmail entry in the CommsRequestTargetDetail table to the email address derived above.

Email Target = Oldest on Sub

The Comms Monitor job checks the CommsCodeCommsDeliveryMethodCode table for an InternalTarget entry. If InternalTarget = O (Oldest Sub on CMP), the job checks the CommsCode table, which records the level at which the comm was raised - account, subscription, group or corporate.

The logic and workflow is as follows:

1. Is the workflow at account level? If so get the lowest active subscription number on account and get its email.

   The lowest active subscription is the subscription that meets these criteria:

- Has the oldest connection date and is active.
- Is not a prepaid subscription.

2. Is the workflow at agreement level? If so get the lowest active subscription number on agreement and get its email.
3. This option is not available for group or corporate level.

The job retrieves the email address from the CustomerEMailAddress table using the appropriate Customer Level and Customer Reference. The email address that it uses is the primary email address (it has the entry isPrimaryEMail = true). If there is only one email address present, this is the primary email.

The job then sets the ToEmail entry in the CommsTargetRequestDetail table to the email address derived above.

Email Target = Parent Subscription

The Comms Monitor job checks the CommsCodeCommsDeliveryMethodCode table for an InternalTarget entry. If InternalTarget = P (Parent Subscription), the job checks the CommsCode table, which records the level at which the comm was raised - account, subscription, group or corporate.

If the workflow is at account level, the job identifies the parent subscription on the account and gets its email address. Any other level returns an error, as it should not be possible to configure this.

The job retrieves the email address from the CustomerEMailAddress table using the appropriate Customer Level and Customer Reference. The email address that it uses is the primary email address (it has the entry isPrimaryEMail = true). If there is only one email address present, this is the primary email.

The job then sets the ToEmail entry in the CommsTargetRequestDetail table to the email address derived above.

Set Email Alias and Request Status

Once the Comms Monitor job has identified and set the email target, it populates the following fields on the CommsRequestTargetDistribution table:

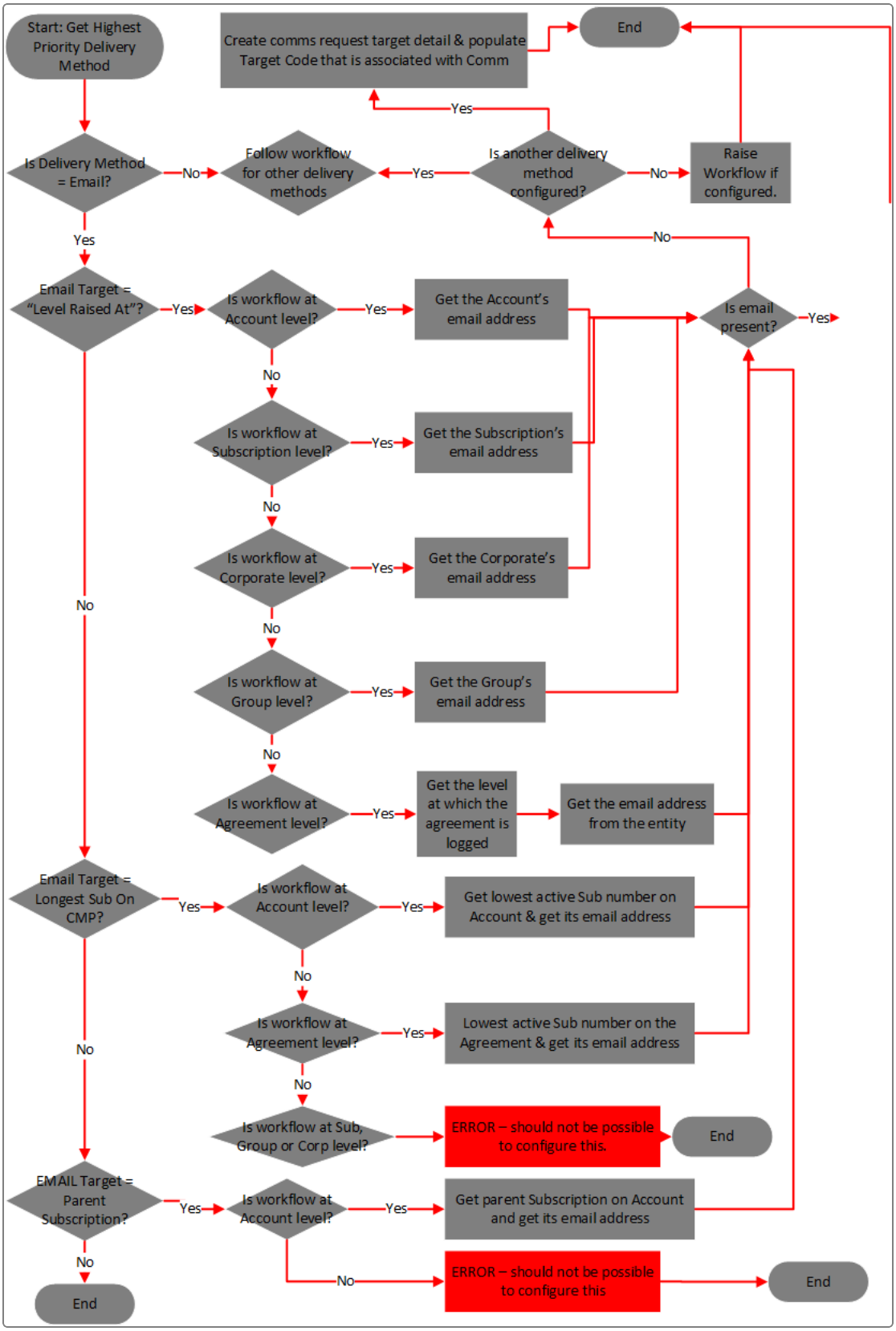
- AliasValue field with the *From email address* from the AliasValue field on the CommsAlias table, which is associated with the record on the CommsCodeCommsDeliveryMethodCode table.
- EMailSubject field.
- DistributionStatus set to A (Awaiting Merge).

Error Handling

If the email address is invalid or not present then the job looks to use an alternative delivery method. If there is no alternative delivery method then CMP cannot send the comms

and the job updates the RequestStatus in the CommsRequestHeader table to N (Not Sent).

The following diagram shows the workflow to determine the email target:


Start: Get Highest Priority Delivery Method
Create comms request target detail & populate Target Code that is associated with Comm
End
Yes
Is Delivery Method = Email?
No
Follow workflow for other delivery methods
Yes
Is another delivery method configured?
No
Raise Workflow if configured.
No
Yes
Email Target = "Level Raised At"?
Yes
Is workflow at Account level?
Yes
Get the Account's email address
Is email present?
Yes
No
Is workflow at Subscription level?
Yes
Get the Subscription's email address
No
Is workflow at Corporate level?
Yes
Get the Corporate's email address
No
Is workflow at Group level?
Yes
Get the Group's email address
No
Is workflow at Agreement level?
Yes
Get the level at which the agreement is logged
Get the email address from the entity
No
Email Target = Longest Sub On CMP?
Yes
Is workflow at Account level?
Yes
Get lowest active Sub number on Account & get its email address
No
Is workflow at Agreement level?
Yes
Lowest active Sub number on the Agreement & get its email address
No
Is workflow at Sub, Group or Corp level?
ERROR – should not be possible to configure this.
End
No
EMAIL Target = Parent Subscription?
Yes
Is workflow at Account level?
Yes
Get parent Subscription on Account and get its email address
No
End
No
ERROR – should not be possible to configure this
End

3.2.2.2 Delivery Method = EXTERNAL

If the delivery method is EXTERNAL, the Comms Monitor job creates an entry in the CommsRequestTargetDetail table and populates it with the ExternalTargetCode associated with the comm, which is retrieved from the CommsCodeCommsDeliveryMethod table.

3.2.2.3 Delivery Method = Push or SMS

The target for a PUSH or SMS delivery method can be:

- Oldest Sub on CMP (account only)

  When the entry for the CommsInternalTarget on the CommsCodeCommsDeliveryMethodCode table for the selected comms is O (Oldest Subscription), the Comms Monitor job retrieves the current mobile number of the subscription on the account with the oldest ConnectedDate that is still active. The subscription must also meet these criteria:

  - The subscription is postpaid.
  - The subscription's network allows SMS (this means the *isSMSAllowed flag* set to true on its Network Type).

  If the subscription is found, the job gets the mobile number and updates the PrimarySerialNumber field on the CommsRequestTargetDetail table.
  The PrimarySerialNumber is written in E164 format.

- Parent Subscription (account only)

  Push comms requests raised at account level can be sent to the parent subscription of the account. A record must exist in the DefaultSubscriptionLink view that specifies the parent subscription for an account. The job queries the table using the account number and linkType = PC.

  When the request is processed the job populates the CommsRequestTargetDetail table as follows:

  - CustomerLevel column = S (Subscription)
  - CustomerReference = Parent subscription number for the account
  - PrimarySerialNumber = mobile number of parent subscription, in E164 format

- Subscription

  If neither of the above targets apply, then the target is the subscription mobile number. The jobs checks whether the subscription network will accept SMS (IsSMSAllowed = true on the Subs Network Type). If it does, the job retrieves the mobile number of the subscription to which the comms will be sent and updates the PrimarySerialNumber field on the CommsRequestTargetDetail table.
  The PrimarySerialNumber is written in E164 format.

Error handling

If no subscription is available, the Comms Monitor job looks for an alternative delivery method to send the comms request. The comms will be sent to the alternative if one is available. If no other delivery methods are available, the comms request will go to error and produce a RequestStatus set to N (Not Sent) in the CommsRequestHeader table.

Example - SMS Comms Requests

An SMS comms has target is set to O (Oldest Sub on CMP).

The comm is raised at account level for the following:

- Account 1 has two post-paid subscriptions that are on a Network Type that has the isSMSAllowed flag set to True.
- Account 2 has one post-paid subscription that is on a Network Type that has the isSMSAllowed flag to True.
- Account 3 has two pre-paid subscriptions that are on a Network Type that has isSMSAllowed flag to True.
- Account 4 and one post-paid subscription and one-prepaid subscription. Both subscriptions are on a Network Type that has isSMSAllowed flag to True.

Results:

For Accounts 1, 2 and 4:

The Comms Monitor job obtains the mobile number of the oldest subscription of the Account and populates the PrimarySerialNumber field on the CommsRequestTargetDistribution table with the number is E164 format. For Account 4, only the subscription number for the post-paid subscription is entered.

For Account 3:

Due to the subscriptions being pre-paid, the Comms Monitor job will not send SMS or Push notifications to the subscriptions.

The job should look to see if there is an alternative delivery method. It will use the highest priority based on the CommsDeliveryMethod table configuration.

If no alternative delivery method is found an error will be produced.

3.2.2.4 Delivery Method = ONLINE

The Comms Monitor job checks the entry for the CommsInternalTarget on the CommsCodeCommsDeliveryMethodCode table to determine the target for the ONLINE delivery method:

Target = Oldest Sub on CMP

The job gets the subscription number of oldest active subscription on the account/agreement. If the workflow was raised at subscription level, the job gets the subscription account to work out the longest subscription on CMP. It then creates an entry in the CommsRequestDetailTarget table and sets the app (online) delivery method as appropriate. The DistributionStatus is set to P (Published), as there is no actual sending of a record to do.

Target = Parent Subscription.

This concerns comms requests raised at account level where the target is Parent Subscription. To identify the parent subscription on an account, the job queries the DefaultSubscriptionLink table using account number and linkType = PC. When this record is identified, the job gets the subscription number from the DefaultSubscriptionNumber field. If there is a parent subscription, the job creates an entry in the CommsRequestDetailTarget and sets the app delivery method as appropriate. The DistributionStatus is set to P (Published), as there is no actual sending of a record to do. If there is no parent subscription, an error is raised and a workflow, if configured.

Target = Raised at Level

If the target is not Oldest Sub on CMP or Parent Subscription, then it must be Raised At Level:

- Account Level

  If the comm is raised at account level, the Comms Monitor job creates an entry in the CommsRequestDetailTarget table and populates Customer Reference fields with the account number. It sets the CommsDeliveryMethodCode to ONLINE. Once the job has gathered the values for the default fields and merged them with the configured text to create a message, the DistributionStatus is set to P (Published), as there is no actual sending of a record to do.

  > For more information on gathering the values for the default fields and creating the message, see "Get the Values for the Default Fields" on the next page.

- Subscription Level

  If the comm is raised at subscription level, the Comms Monitor job creates an entry in the CommsRequestDetailTarget table and populates Customer Reference fields with the subscription number. It sets the CommsDeliveryMethodCode to ONLINE. Once the job has gathered the values for the default fields and merged them with the configured text to create a message, the DistributionStatus is set to P (Published), as there is no actual sending of a record to do.

- Agreement Level

  If the agreement is logged at account level, the Comms Monitor job creates an entry in the CommsRequestDetailTarget table and populates Customer Reference fields

with the account number. It sets the CommsDeliveryMethodCode to ONLINE. Once the job has gathered the values for the default fields and merged them with the configured text to create a message, the DistributionStatus is set to P (Published), as there is no actual sending of a record to do. If the agreement is not set at account level this raises an error because ONLINE delivery is not supported for comms raised at group and corporate level. A workflow event is raised, if configured.

3.2.2.5 Delivery Method = Letter

A LETTER comms can be sent only to the level at which it is was raised. When the LETTER delivery method is used on a workflow event, the comms is sent at the level at which the workflow event is raised.

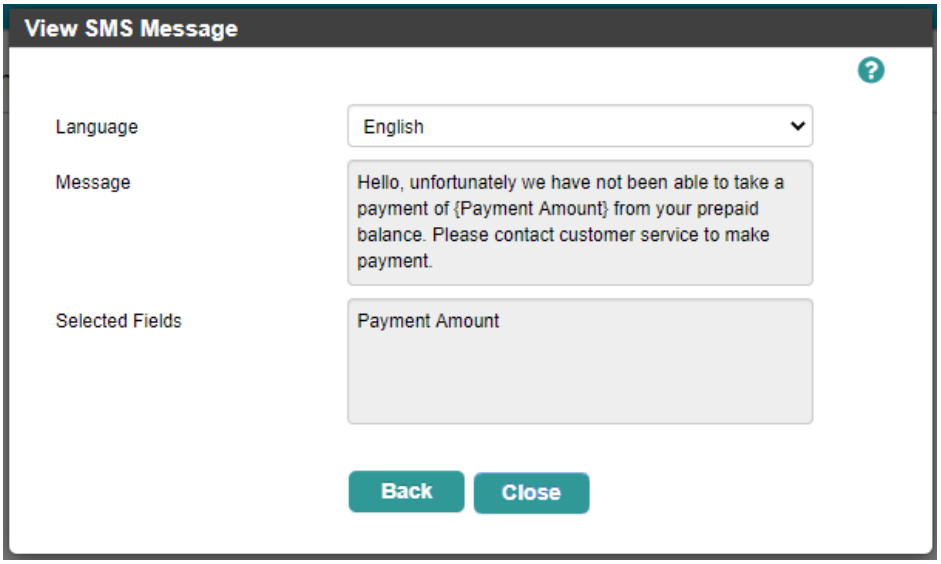
The Comms Monitor job creates an entry in the CommsRequestDetailTarget table and queries the relevant entities to derive the DeliveryAddressNumber, which it populates. It then updates the DistributionStatus in the CommsRequestTargetDistribution table to A (Awaiting Merge).

3. 2. 3 Get the Values for the Default Fields

Recommended reading:

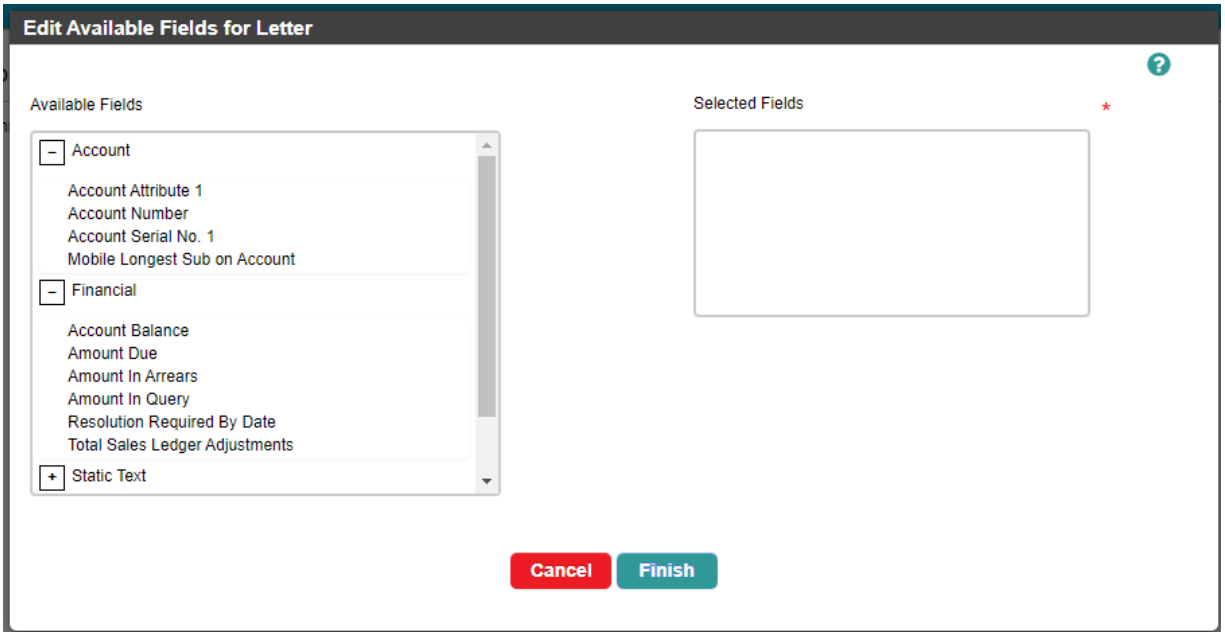
- For information on default fields, see "Communications Default Fields" on page 6
- For lists of default fields and the values they hold, see the *CMP Configuration Overview* guide.

Default fields contain customer-specific attributes stored in the CMP database. This customer data can be merged into communications. For example, a customer account number and account balance can be sent in a bill reminder SMS. For PUSH, SMS and ONLINE communications, the messages and their associated default beans are configured in the Business Configuration console, as shown below:



An SMS message that includes the {Payment Amount} default field

Some communication formats, for example letters and emails, must be associated with an additional external template that determines the format, style and layout of the correspondence. However, the default fields that can be incorporated into these templates are still configured in the Business Configuration console:



Configuring the available default fields for a letter

So the Comms Monitor job must retrieve the values of the default fields for all comms.

Each default field is associated with a bean, which is the software object that retrieves the customer information for the default field from CMP. One or more parameters are configured for the bean, which tell the bean exactly what information to retrieve.

In the case of push notifications, online notifications and SMS delivery methods, the job also merges the data of the derived fields with the message template to produce the actual message. For letters and email, this merging is carried out by separate jobs. The workflow is as follows:

1. The Comms Monitor job writes a record into the CommsRequestFieldDetail table.
2. The job gets a list of fields associated with the comms.
3. The job invokes the relevant beans used by the default fields to populate the CommsRequestFieldDetail table.
4. The job repeats step 3 until the list of fields is complete and then it updates the RequestStatus in the CommsRequestHeader table to P (Processed).
5. Then the job checks for time exclusions (see "Checking Time Exclusion" below).
6. The job checks the CommsCodeLanguage table to identify which Message value should be used for the comm.
7. For EMAIL and LETTER, once the job has processed time exclusions, it creates an entry in the CommsRequestTargetDistribution table with the DistributionStatus of A (Awaiting Merge). Another job and daemons are responsible for merging email and letter templates with default field values and sending the comms.
8. For SMS, PUSH and ONLINE, once the time exclusions are processed, the job sets the Message value in the CommsRequestTargetDistribution table with the correct merged field (CommsFieldCode and TextValue from CommsRequestFieldDetail table) and sets the DistributionStatus to D (AwaitingDistribution).

3.2.3.1 Checking Time Exclusion

Customers may want to exclude getting communications during certain hours. To support this, customers are able to set exclusion periods for receiving comms. If they have set up an exclusion period, this can be ignored for certain comms.

The delivery method has a flag called isTimeExcludable. This is used to determine whether a particular delivery method is to be time excludable or not. For example, LETTER is usually not time excludable because it doesn't really matter when it is sent, as it doesn't concern the customer. However, an SMS is usually time excludable because the customer may not want to receive them late at night.

For each record in the CommsRequestFieldDetail table, the Comms Monitor job checks whether a time exclusion has been set.

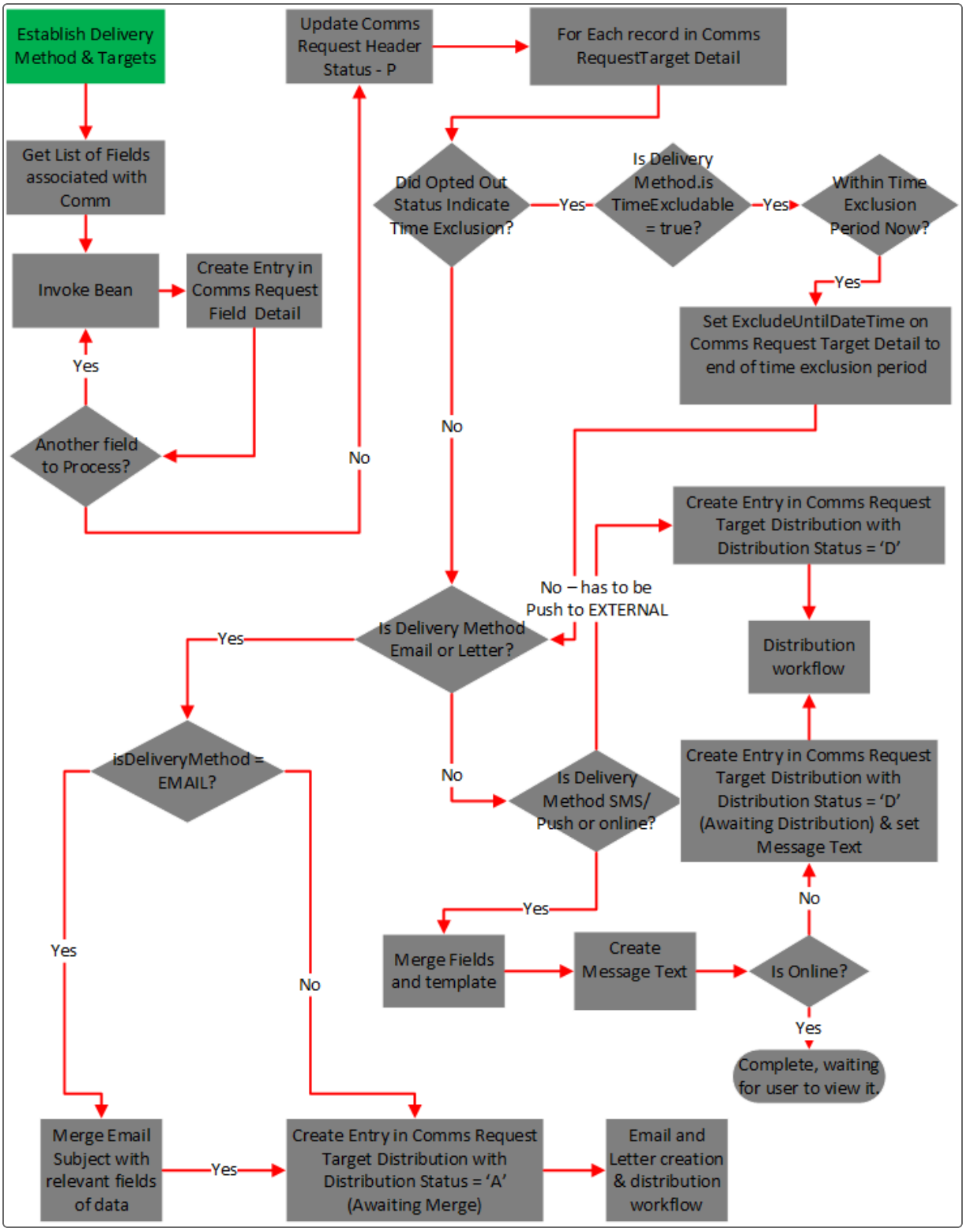
If the isTimeExcludable flag = True on the delivery method that is derived then the job executes the following logic:

- If there is a time exclusion set on the CustomerCommsProfile table, the job checks whether the current time is within that exclusion period and then checks whether

isAllowTimeExclusion is set to True on the CommsPreference table. If it is, then the job sets the ExcludeUntilDateTime on the CommsRequestTargetDetail table to the TimeExclusionEnd value that is on the CustomerCommsProfile table.

- If isTimeExcludable= False on the CommsDeliveryMethod table, or if there is no time exclusion set on the CustomerCommsProfile table, or if isAllowTimeExclusion is set to False on the CommsPreference table, or if the current time is outside of the time exclusion, then the job does not set ExcludeUntilDateTime on the CommsRequestTargetDetail table.
- Any comms requests raised outside of the Exclude Start/End times are processed immediately and the value of the ExcludedUntilDateTime field is set to Null when creating the record in the commsRequestTargetDetail table.

The following diagram shows the workflow and logic for getting the value of the default fields:

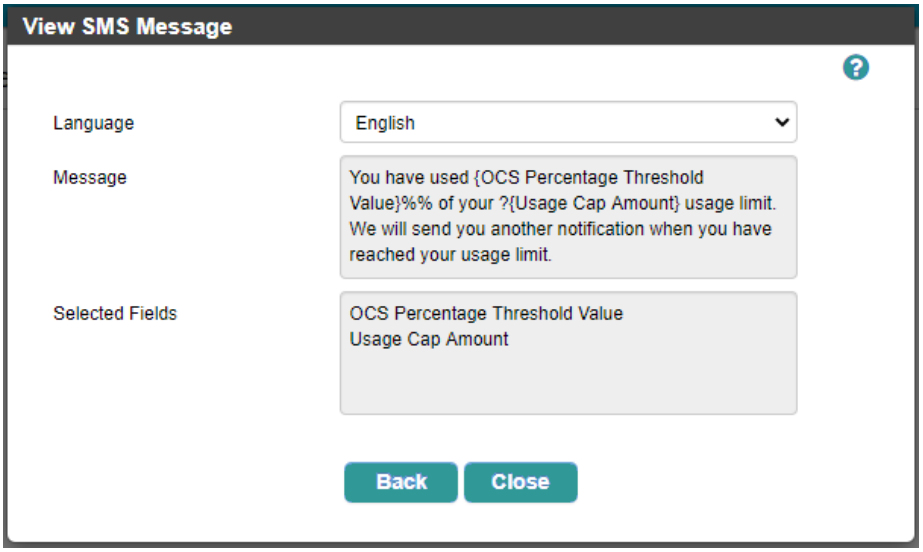

Establish Delivery Method & Targets
Update Comms Request Header Status - P
For Each record in Comms RequestTarget Detail
Get List of Fields associated with Comm
Did Opted Out Status Indicate Time Exclusion?
Yes
Is Delivery Method.is TimeExcludable = true?
Yes
Within Time Exclusion Period Now?
Yes
Invoke Bean
Create Entry in Comms Request Field Detail
Yes
Set ExcludeUntilDateTime on Comms Request Target Detail to end of time exclusion period
Another field to Process?
No
No
Create Entry in Comms Request Target Distribution with Distribution Status = 'D'
No – has to be Push to EXTERNAL
Is Delivery Method Email or Letter?
Yes
Distribution workflow
isDeliveryMethod = EMAIL?
No
Is Delivery Method SMS/ Push or online?
Create Entry in Comms Request Target Distribution with Distribution Status = 'D' (Awaiting Distribution) & set Message Text
Yes
No
Yes
Merge Fields and template
Create Message Text
Is Online?
Yes
Complete, waiting for user to view it.
Yes
No
Merge Email Subject with relevant fields of data
Yes
Create Entry in Comms Request Target Distribution with Distribution Status = 'A' (Awaiting Merge)
Email and Letter creation & distribution workflow

3. 2. 4 Merge Fields and Templates

3.2.4.1 EXTERNAL, SMS, PUSH or ONLINE Comms

Once the Comms Monitor job has derived the values for the default fields associated with the comm, it merges the field values and the comms template to create the message.

For SMS, PUSH or ONLINE comms, the message template is created in the Business Configuration console and may have a number of default fields embedded in it, for example:



An SMS message configured with multiple default fields

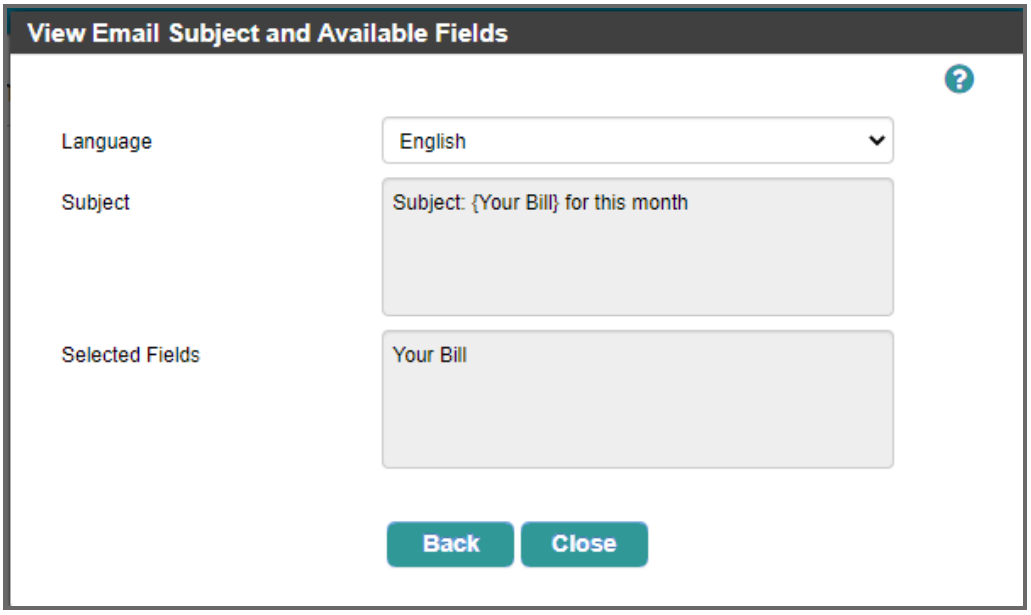
The message template is held in the Message field in the CommsCodeLanguage table and can be identified by the presence of `${commsFieldCode}` where `commsFieldCode` is the identifier of the CommsRequestFieldDetail entry for the default field value. The Comms Monitor job merges values of the default fields with the message template as follows:

1. The job checks the CommsFieldCode in the CommsField table, which relates to a default field record and its parameter held in the CommsFieldDefinition table.
2. When a CommsFieldCode is used in the Message value held in the CommsCodeLanguage table, the job merges the default field value into the message.
3. The job stores the merged message in the CommsRequestTargetDistribution table.
4. For EXTERNAL, SMS and PUSH comms, the job updates the DistributionStatus on the CommsRequestTargetDistribution table to D (Awaiting Distribution).
5. For ONLINE comms, once the merged message is stored, it is ready and waiting for the user to view it, so no further distribution is necessary and the job updates the DistributionStatus to P (Published).

3.2.4.2 EMAIL and LETTER Comms

Email

The template for the body of an email is an HTML template created in a third party application such as Velocity, which is used in CMP 8. The email subject heading is configured in Business Configuration console and can include default field values:



Configuring an email Subject that includes a default field

For emails, the Comms Monitor job follows the process above to merge Email Subject with relevant default field values. After the job has invoked the Java beans, retrieved the default field values and stored them in the CommsRequestFieldDetail table, it merges the EmailSubject held in the CommsCodeLanguage table with the values stored in the CommsRequestFieldDetail table by replacing an instances of `${fieldCode}` with the derived data.

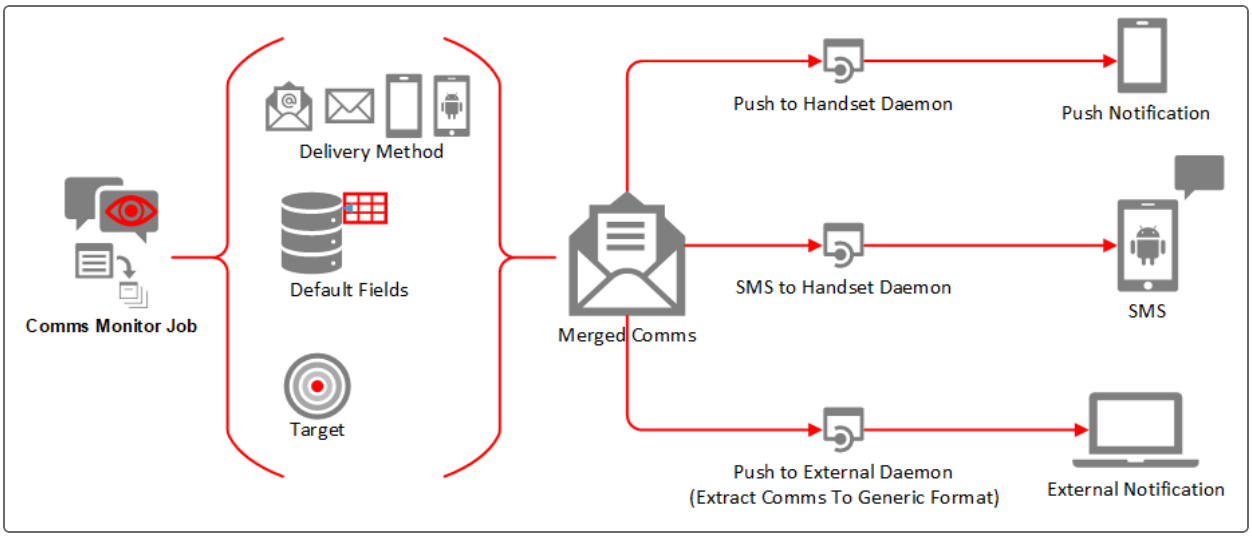
Once the merged subject has been created, the job stores it in the CommsRequestTargetDistribution table and sets the DistributionStatus field on the table to A (Awaiting Merge). Creating and distributing the email is handled by the Email Monitor job; see "Comms Process Flow: Create and Send Letters and Emails" on page 46.

Letter

For letters , the Comms Monitor job creates an entry in the CommsRequestTargetDistribution table and sets the DistributionStatus field on the table to A (Awaiting Merge). Creating and distributing the letter is handled by the Letter Monitor job; see "Comms Process Flow: Create and Send Letters and Emails" on page 46.

3. 3 Comms Process Flow: Send SMS, Push and External Notifications

Once the merge has occurred, the merged message is placed in the CommsRequestTargetDistribution table in the Message field and the DistributionStatus is set to D (Awaiting Distribution). The comms must now be distributed. The Comms Monitor job is responsible for distributing EXTERNAL, SMS, PUSH and ONLINE comms.



Comms Monitor job distributing Push, SMS and External comms

3.3.0.1 ONLINE

For ONLINE comms, once the merged message is stored, it is ready and waiting for the user to view it, so no further distribution is necessary and the job creates an entry in the CommsRequestTargetDistribution table with the DistributionStatus of P (Published).

3.3.0.2 PUSH

Distribution of comms with the PUSH delivery method is handled by the Push to Handset (Transmission Comms Push to Handset) daemon. The push to handset daemon picks up entries that are ready to be pushed to a handset. These messages contain the full text to be sent within either a plain or an encrypted JSON message, depending on the configuration. The daemon queries the CommsRequestTargetDistribution table where:

- Distribution Status = D
- AND DeliveryMethod = PUSH
- AND Time Exclusion >= Current Date and Time

The daemon converts the message into an encrypted JSON format and places it on an outbound ActiveMQ queue for delivery. The queue is set in the following properties when the daemon is configured:

Property	Description
comms.handset.monitor.transmission.to.properties.destinationName	The queue in the ActiveMQ instance onto which Push to Handset messages will be put.
comms.handset.monitor.transmission.to.properties.brokerUrl	The URL of the ActiveMQ instance which will be used to handle outbound Push Notifications.

For more information on the daemon and its configuration see "Transmission Comms Push To Handset" on page 11 in the *Communication Jobs and Daemons* section.

Once the message is placed on the queue, the job updates the DistributionStatus to either:

- S (Sent) and updates the SentDateTime on the CommsRequestTargetDistribution table.
- P (Published) and updates the PublishedDateTime on the CommsRequestTargetDistribution table.

Whether the status is set to Sent or Published is determined by the setting for the `comms.handset.monitor.extract.distribution-status` property for the daemon. Valid values are P (Published) or S (Sent) and this determines the Status text that is displayed to the CSA in AgentView.

3.3.0.3 SMS

The SMS To Handset (Transmission Comms SMS to Handset) daemon picks up entries that are ready to be sent via SMS. These messages contain the full text to be sent and sent within either a plain or an encrypted JSON message, depending on configuration.

The daemon queries the CommsRequestTargetDistribution table where:

- Distribution Status = D
- AND DeliveryMethod = SMS
- AND Time Exclusion >= Current Date and Time

The daemon creates the JSON file and puts it on a file system or a queue for distribution, depending on the configuration.

Once the message is placed on the file system or queue, the job updates the DistributionStatus to either:

- S (Sent) and updates the SentDateTime on the CommsRequestTargetDistribution table.
- P (Published) and updates the PublishedDateTime on the CommsRequestTargetDistribution table.

Whether the status is set to Sent or Published is determined by the setting for the `comms.sms.monitor.extract.distribution-status` property for the daemon. Valid values are P (Published) or S (Sent) and this determines the Status text that is displayed to the CSA in AgentView.

For more information on the daemon and its configuration see "Transmission Comms SMS To Handset" on page 11 in the *Communication Jobs and Daemons* section.

For information on configuring the date format for SMS and push notifications, see "Configuring Date Formats - Administration Console" on page 9.

3.3.0.4 EXTERNAL

The Extract Comms to Generic Format daemon handles communications with the EXTERNAL delivery method. The daemon queries the CommsRequestTargetDistribution table where:

- Distribution Status = D
- AND DeliveryMethod = EXTERNAL
- AND Time Exclusion >= Current Date and Time

The daemon converts the message into either a plain encrypted JSON format, depending on configuration, and places it on an outbound ActiveMQ queue for delivery. The queue is set in the following properties when the daemon is configured:

Property	Description
comms.external.monitor.transmission.to.properties.destinationName	The name of the queue on the ActiveMQ instance onto which the outbound external comms will be placed.
comms.external.monitor.transmission.to.properties.brokerUrl	The URL of the ActiveMQ instance that will be used to handle outbound comms via the external route.

For more information on the daemon and its configuration see "Extract Comms to Generic Format" on page 11 in the *Communication Jobs and Daemons* section.

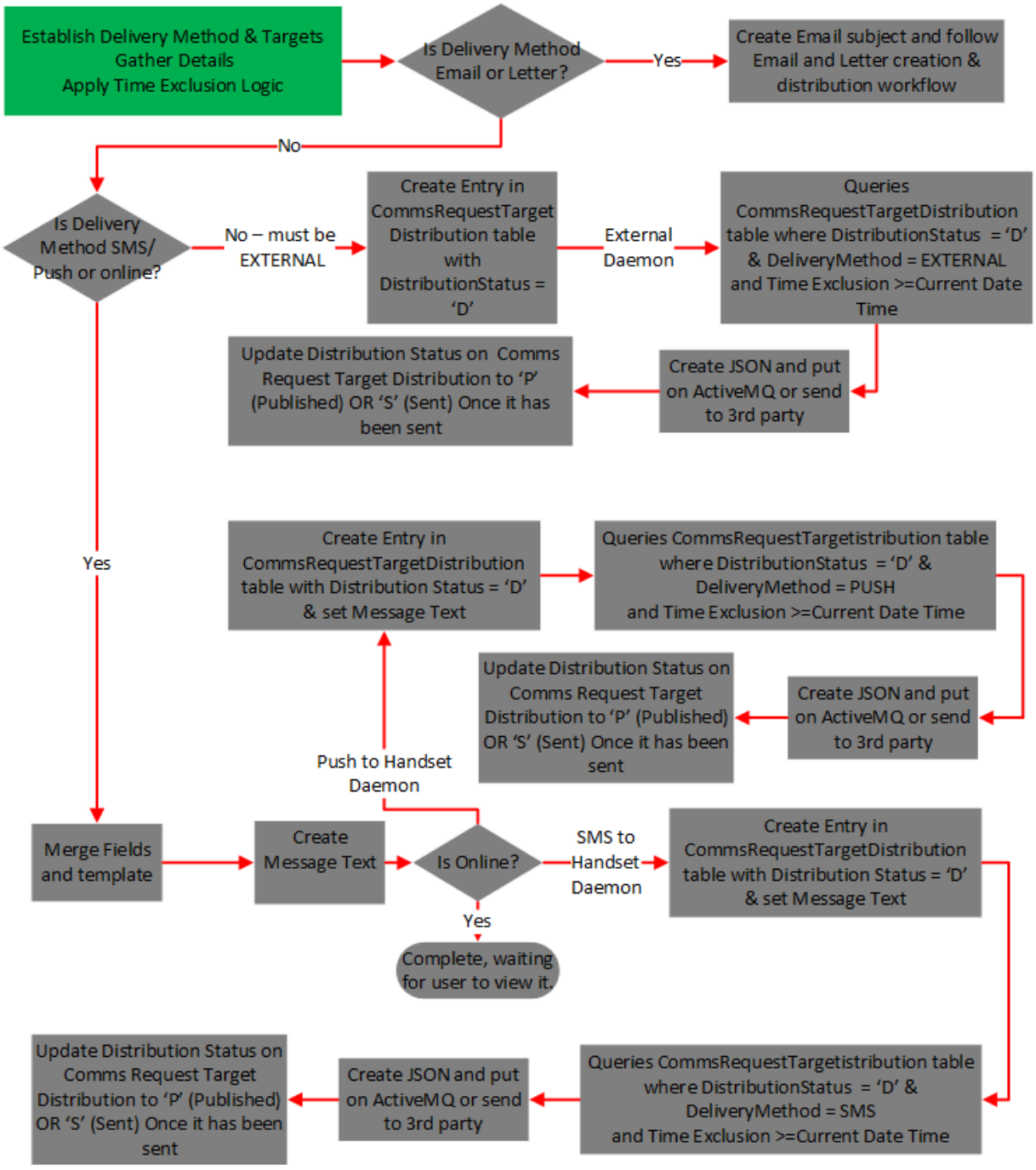
Once the message is placed on the queue, the job updates the DistributionStatus to either:

- S (Sent) and updates the SentDateTime on the CommsRequestTargetDistribution table.
- P (Published) and updates the PublishedDateTime on the CommsRequestTargetDistribution table.

Whether the status is set to Sent or Published is determined by the setting for the `comms.external.monitor.extract.distribution-status` property for the

daemon. Valid values are P (Published) or S (Sent) and this determines the Status text that is displayed to the CSA in AgentView.

The following diagram shows the workflow for sending SMS, PUSH, ONLINE and EXTERNAL comms:



3. 4 Comms Process Flow: Create and Send Letters and Emails

Email and letter communications require additional processing to merge data from default fields gathered from CMP with document templates to generate the required document type. The Comms Monitor job gathers the data; the Comms Email Monitor and the Comms Letter Monitor jobs merge the data.

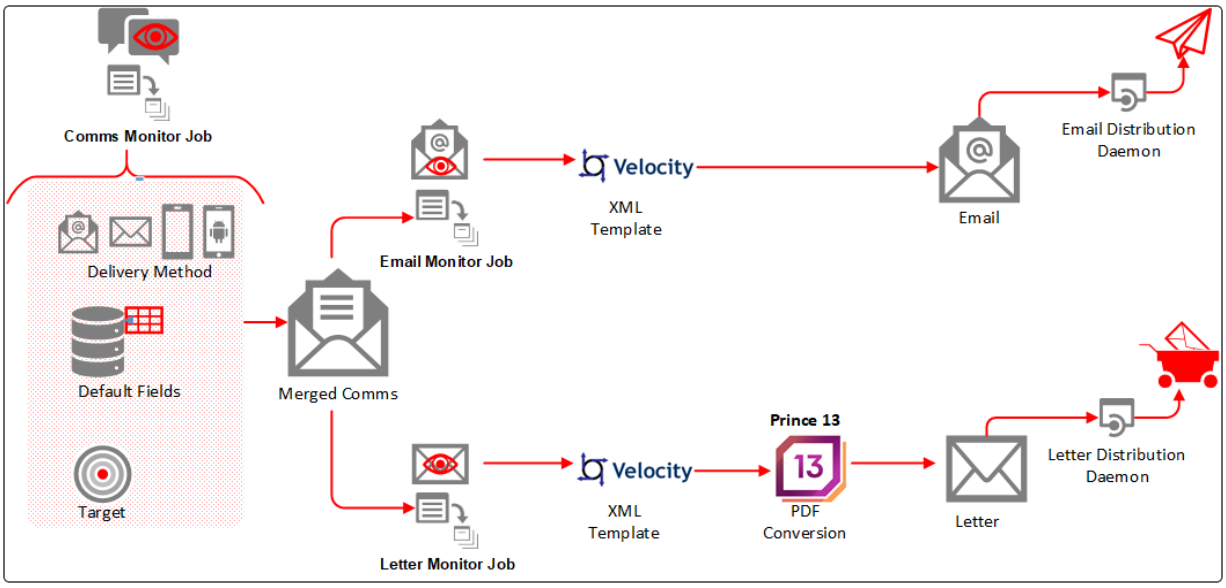
The Comms Email Monitor and Comms Letter Monitor jobs are triggered automatically when the Comms Monitor job has finished gathering data for comms requests relating to emails or letters.

These jobs use external frameworks - for example, Velocity and Prince HTML - to generate documents. Generated documents are subsequently detected by the following downstream daemons for transmission to the customer:

- Transmission Comms to Email (Email daemon)
- Transmission Comms to Print Bureau (Letter daemon)

For more information, see About Email and Letter Templates

Copies of the generated documents are created and placed in long-term storage by the Transmission Comms to Document Storage daemon. These documents can be viewed in AgentView. For more information, see "Comms Process Flow: Store Letters and Emails" on page 56.



3. 4. 1 Comms Email Monitor Job

At this point in the communication process, the Comms Monitor job has:

- Established the delivery method and target for an email communication.
- Gathered details such as the values of default fields and the email subject.
- Applied any time exclusion logic so that emails are not sent during a time period when the customer doesn't want to receive them.

For email communications, the job merges the email subject with the data field values and creates an entry in the Comms RequestTargetDistribution table with a DistributionStatus of A (Awaiting Merge). The Comms Email Monitor job now takes over processing the email communication.

The job can be configured as a triggered job or a scheduled job in the Administration Console, according to requirements. Whether the job is triggered depends on the value for the `comms.email.monitor.trigger.enabled` property for the job.

For more information on job properties, see "Comms Email Monitor" on page 10 in the *Communications Jobs and Daemons* section.

Whether triggered or scheduled, the job picks with entries in the CommsRequestTargetDistribution table for entries where:

- DistributionStatus = A (Awaiting Merge), and
- DeliveryMethod = EMAIL, and
- TimeExclusion >= Current Date/Time

For each entry that meets these criteria, the job produces a document that consists of the details gathered by the Comms Monitor job (default field values) merged with an HTML template that determines the format, style and layout of the correspondence. For CMP 8.0, these are Velocity templates.

For more information on templates, see "Communication Email and Letter Templates" on page 60.

Velocity templates are named according to the format: `<COMMSCODE>-<LANGUAGE>-<DELIVERYMETHOD>-<Format>.vm`, where `<LANGUAGE>` can be `EN` for English or `SP` for Spanish, for example `LETT1-EN-LETTER-pdf.vm or MAIL1-EN-EMAIL-html.vm`.

The Comms Email Monitor job looks in the location where Velocity templates are stored to match a template with the CommsCode and Delivery Method that match those for the email communication that it is processing.

The storage location for the templates is set by the value for the `comms.document.monitor.templates.location` property for the Comms Email Monitor job.

Once the final document is ready, the Comms Email Monitor job names the document according to the format `{commsRequestId}.XXX` where `XXX` is the file format e.g. `html`, `pdf` and stores the job in the location set in the `comms.document.monitor.templates.location` property for the job.

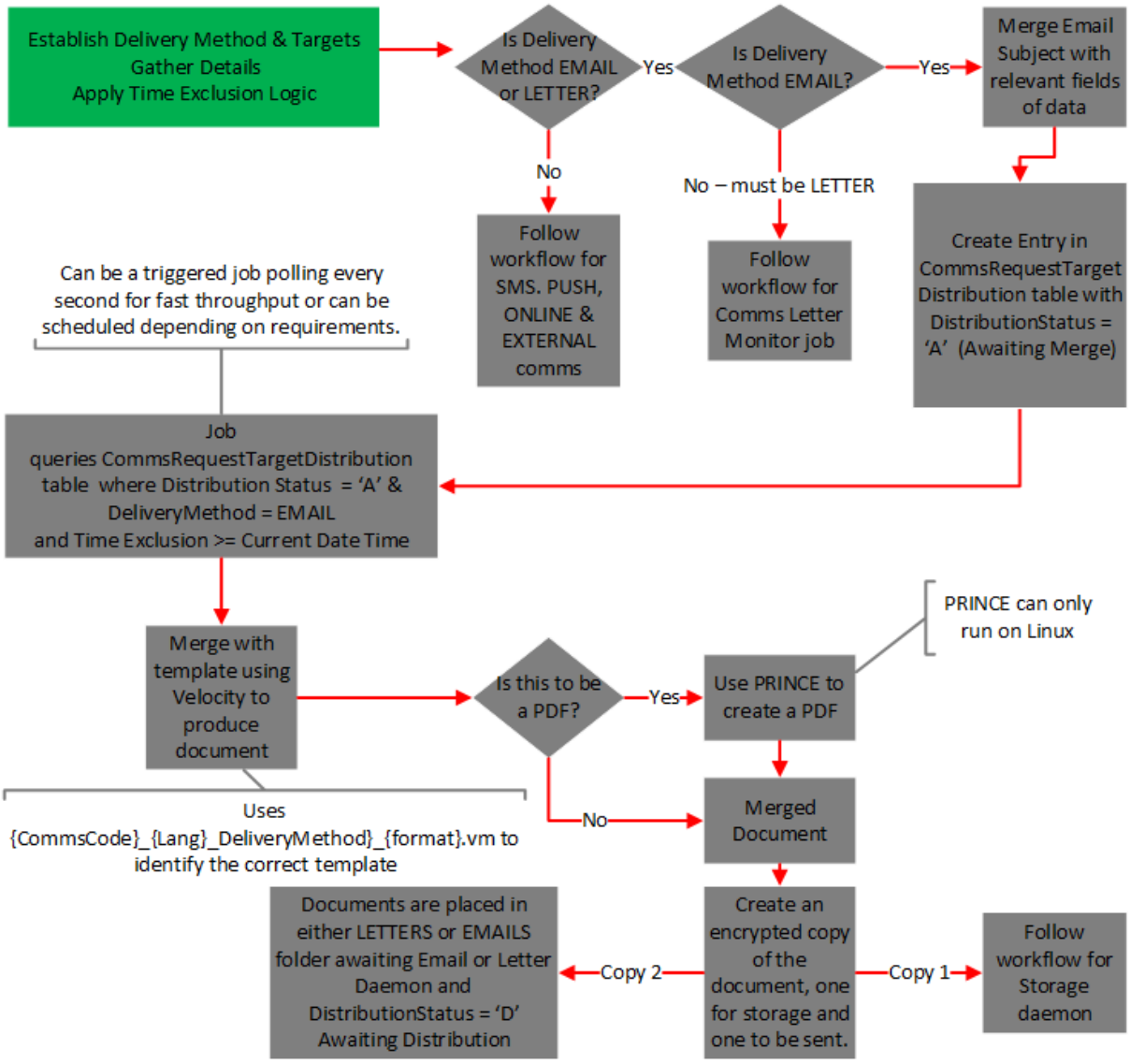
The job then updates the FileLocation field on the CommsRequestTargetDistribution table.

The job then creates two encrypted copies of the document :

- One copy is for long-term storage in CMP, which will allow communications to be viewed using AgentView. This task is handled by the Transmission Comms to Document Storage daemon. For more information, see "Comms Process Flow: Store Letters and Emails" on page 56.
- The other copy is stored in an EMAILs folder, from which it will be collected for distribution by the Transmission Comms to Email daemon. The location of the folder is set by the value for the `comms.document.monitor.output.file.location` property for the sabre-comms module. For more information, see "Transmission Comms To Email" on page 11 in the Communications Jobs and Daemons section.

The job updates the DistributionStatus in CommsRequestTargetDistribution table to D (Awaiting Distribution).

The following diagram illustrates the workflow for the Comms Email Monitor job:



3.4.1.1 Transmission Comms to Email Daemon

The Transmission Comms to Email daemon polls the local emails folder for merged documents to be sent. The location is set in the `comms.e-mail.monitor.transmission.from.properties.dirname` property for the daemon.

The files are named according to the format `{commsRequestId}.XXX` where `XXX` is the file format. For each CommsRequestId found, the daemon constructs and sends an email.

If the format is HTML or TXT, the daemon embeds the file into the main body of the email itself.

Any image files such as JPG or PNG are also added as attachments. These images are stored in the location set by the `comms.email.monitor.transmission.image-folder` property for the daemon.

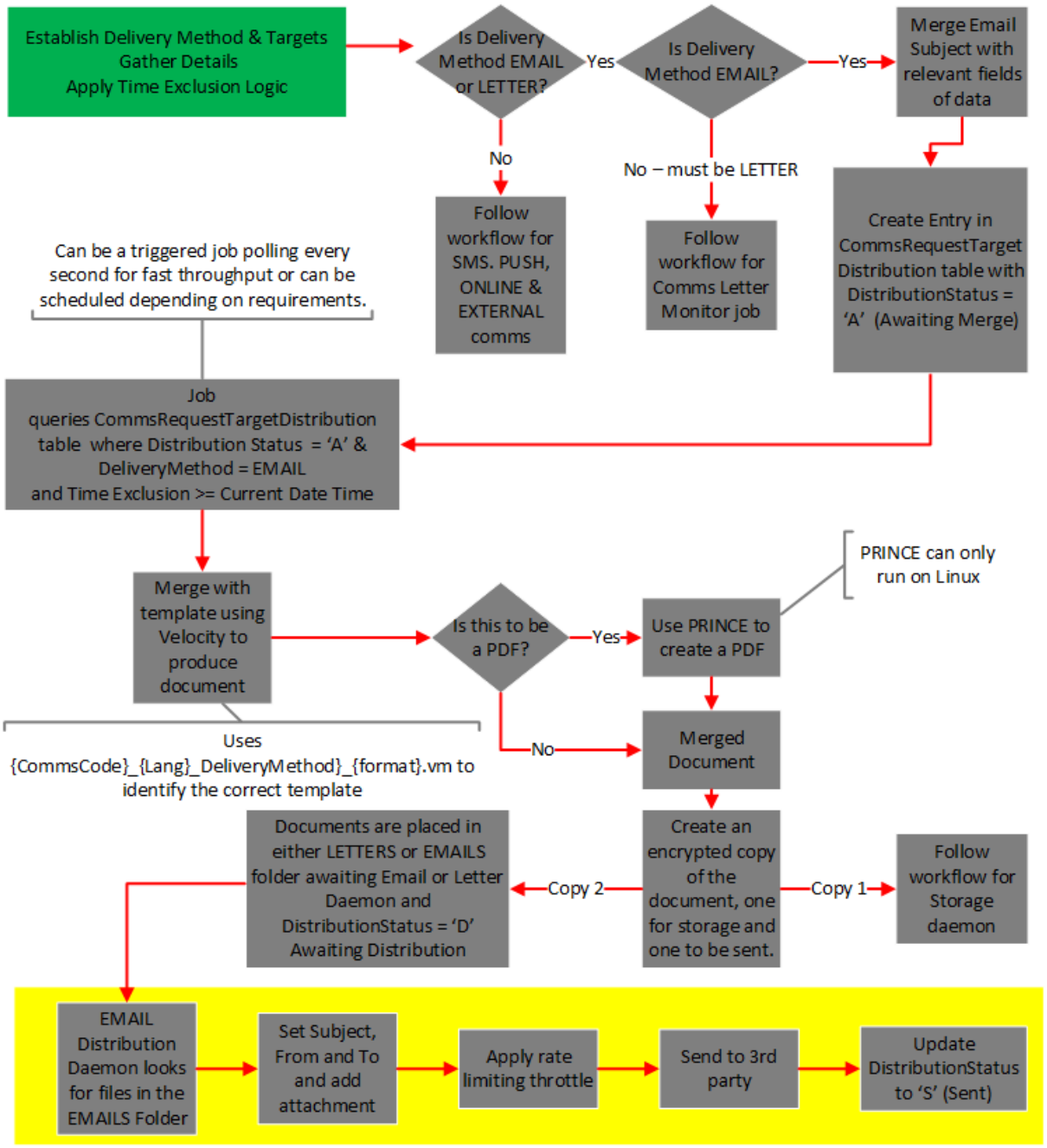
The daemon then sets the following:

- Email Subject - from the EmailSubject field on the CommsRequestTargetDistribution table.
- From - from the AliasValue field on the CommsRequestTargetDistribution table.
- To - from the ToEmail field on the CommsRequestTargetDistribution table.

The daemon then updates the following:

- The DistributionStatus field on the CommsRequestTargetDistribution table to S (Sent).
- The SentDateTime record in the CommsRequestTargetDetail table to the current date and time.

The highlighted portion of the following diagram shows the workflow for distributing email communications:



3. 4. 2 Comms Letter Monitor Job

At this point in the communication process, the Comms Monitor job has:

- Established the delivery method and target for a letter communication.
- Gathered details such as the values of default fields.
- Applied time exclusion logic, if any.

> Time exclusions are usually not set for letter communications because when letters are sent does not affect the customer.

For letter communications, the job creates an entry in the Comms RequestTargetDistribution table with a DistributionStatus of A (Awaiting Merge). The Comms Letter Monitor job now takes over processing the letter communication.

The job can be configured as a triggered job or a scheduled job in the Administration Console, according to requirements. Whether the job is triggered depends on the value for the `comms.letter.monitor.trigger.enabled` property for the job.

For more information on job properties, see "Comms Letter Monitor" on page 10 in the *Communications Jobs and Daemons* section.

Whether triggered or scheduled, the job picks with entries in the CommsRequestTargetDistribution table for entries where:

- DistributionStatus = A (Awaiting Merge), and
- DeliveryMethod = LETTER, and
- TimeExclusion >= Current Date/Time

For each entry that meets these criteria, the job produces a document that consists of the details gathered by the Comms Monitor job (default field values) merged with an HTML template that determines the format, style and layout of the correspondence. For CMP 8.0, these are Velocity templates.

For more information on templates, see "Communication Email and Letter Templates" on page 60.

Velocity templates are named according to the format: `<COMMSCODE>-<LANGUAGE>-<DELIVERYMETHOD>-<Format>.vm`, where `<LANGUAGE>` can be `EN` for English or `SP` for Spanish, for example `LETT1-EN-LETTER-pdf.vm or MAIL1-EN-EMAIL-html.vm`.

The Comms Letter Monitor job looks in the location where Velocity templates are stored to match a template with the CommsCode and Delivery Method that match those for the email communication that it is processing.

The storage location for the templates is the same one as set by the value for the `comms.document.monitor.templates.location` property for the sabre-comms module.

If the communication output format is PDF, the job performs some extra steps. It uses Prince to generate the PDF from the merged document. Prince is a third party product that produces PDF files from HTML input. It only runs on Linux. If the Velocity template specifies a PDF format, the velocity template must be configured in a way that enables an HTML file to be created from it. The job first creates a merged HTML file from the template and stores it temporarily. The storage location is set in the `comms.document.monitor.output.file.tempstorage` property of the Comms Letter Monitor job. Prince creates the PDF.

The location of Prince is set by the value for the `comms.document.monitor.pdf.generator.executable.location` property for the sabre-comms module.

Once the final document is ready, the Comms Letter Monitor job names the document according to the format `{commsRequestId}.XXX` where `XXX` is the file format e.g. `html`, `pdf` etc.and stores the job in the location set in the `comms.document.monitor.templates.location` property for the job.

The job then updates the FileLocation field on the CommsRequestTargetDistribution table.

The job creates two encrypted copies of the document:

- One copy is for long-term storage in CMP, which will allow communications to be viewed using AgentView. This task is handled by the Transmission Comms to Document Storage daemon. For more information, see "Comms Process Flow: Store Letters and Emails" on page 56.
- The other copy is stored in a LETTERS folder, from which it will be collected for distribution by the Transmission Comms to Print Bureau daemon. The location of the folder is set by the value for the `comms.document.monitor.output.file.location` property for the job. For more information, see "Transmission Comms to Print Bureau" on page 11 in the *Communications Jobs and Daemons* section.

The job updates the DistributionStatus in CommsRequestTargetDistribution table to D (Awaiting Distribution).

3.4.2.1 Transmission Comms to Print Bureau Daemon

Once the Comms Letter Monitor has created the merged letter communications and stored them in a LETTERS folder, this daemon takes the contents of that directory and creates a `ZIP` file of letter communications. It sends the file to a print bureau, where the letters can be printed and mailed to recipients.

The location of the folder with the letter communications is set in the `comms.letter.monitor.transmission.from.properties.dirname` property for the daemon.

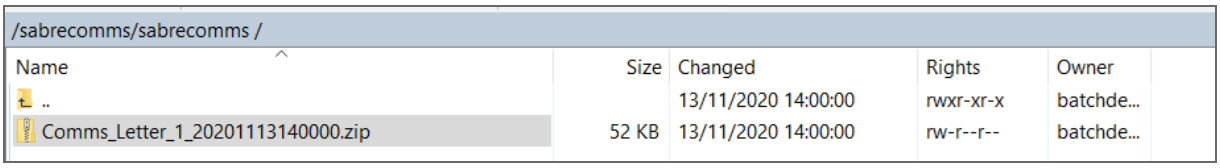
The daemon is a scheduled daemon - not triggered. The schedule is typically once per day and is configured as a value of the daemon property: `comms.letter.monitor.transmission.from.options.[scheduler.cron]`. The value is a `chron` expression. For more information on writing chron expressions, click here.

The daemon generates a unique name for the ZIP file following this convention:

`<prefix>_<sequence>_<date/time>.zip`, where:

- prefix = The prefix that is configurable in the comms.letter.monitor.transmission.filename-prefix property of the daemon, for example Comms_Letter
- sequence = A number that indicates the ZIP's position in the sequence of ZIPs produced by the daemon that day: 1 = first today, 2 = second today etc.
- date/time = the date and time the daemon produced the file.

The following screenshot shows an example of the ZIP file produced:



The name of the ZIP file is the value of the DistributionReference.

For each letter in the ZIP file, the daemon updates the DistributionReference field on the CommsRequestTargetDistribution table with the name of the ZIP file. So all letters in the same ZIP file have the same DistributionReference value.

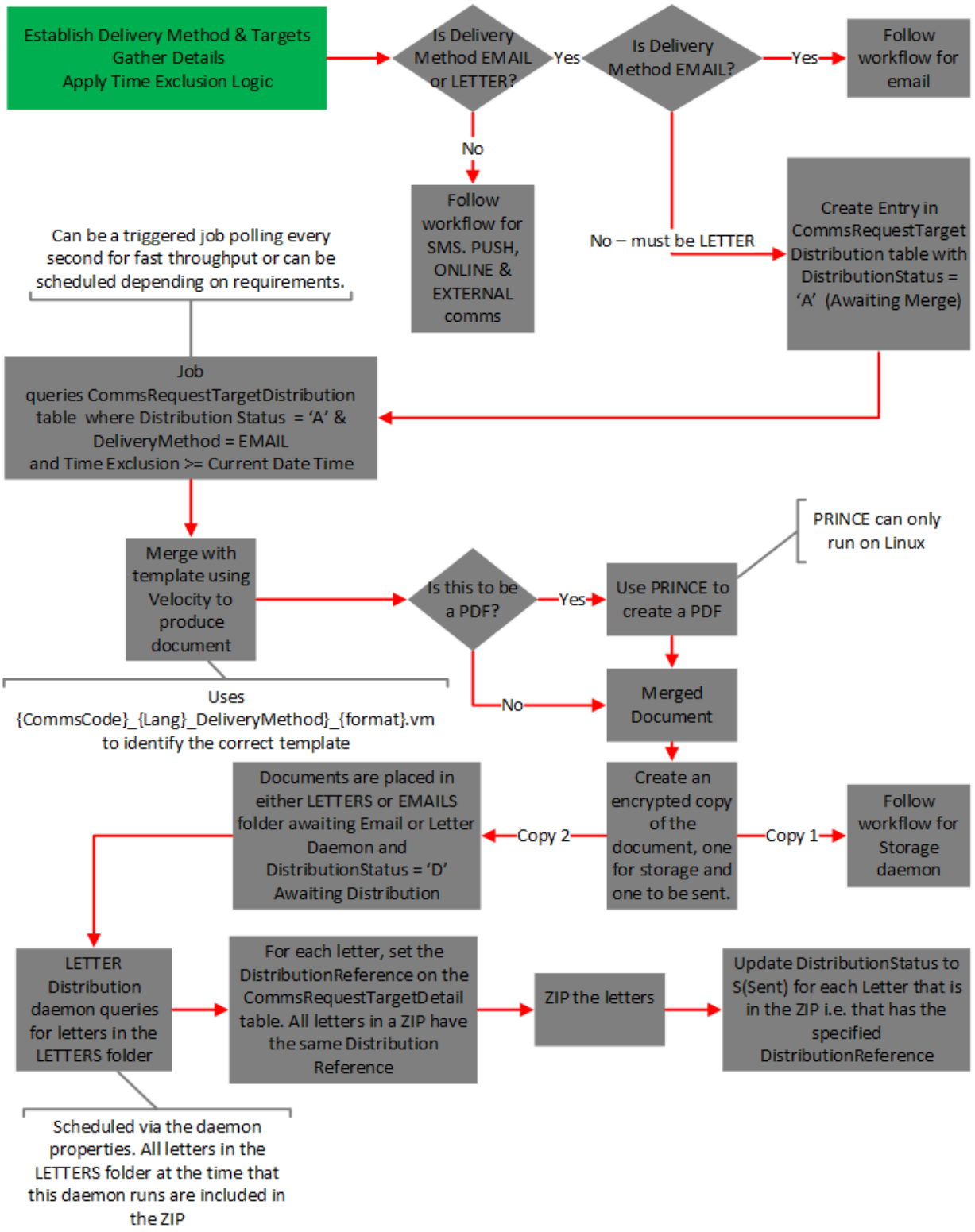
The daemon sends the ZIP file to the print bureau. The file is transferred by SFTP. The IP address of the recipient, the hostname of the SFTP server and the username to be used are all configurable in the daemon properties.

Once the file is sent, the daemon updates the DistributionStatus of each individual letter communication to S (Sent) on the CommsRequestTargetDistribution table.

Letter communications that have been successfully zipped and sent are then transferred to a *done* folder, the location of which is set in the comms.letter.monitor.transmission.from.options.move property of the daemon.

Letter communications that failed to send due to errors are moved to a location specified in the comms.letter.monitor.transmission.from.options.moveFailed property.

The following diagram illustrates the workfow for creation and distribution of letter communications:


Establish Delivery Method & Targets
Gather Details
Apply Time Exclusion Logic
Is Delivery Method EMAIL or LETTER?
Yes
Is Delivery Method EMAIL?
Yes
Follow workflow for email
No
Follow workflow for SMS. PUSH, ONLINE & EXTERNAL comms
No – must be LETTER
Create Entry in CommsRequestTarget Distribution table with DistributionStatus = 'A' (Awaiting Merge)
Can be a triggered job polling every second for fast throughput or can be scheduled depending on requirements.
Job queries CommsRequestTargetDistribution table where Distribution Status = 'A' & DeliveryMethod = EMAIL and Time Exclusion >= Current Date Time
Merge with template using Velocity to produce document
Uses {CommsCode}_{Lang}_DeliveryMethod}_{format}.vm to identify the correct template
Is this to be a PDF?
Yes
Use PRINCE to create a PDF
PRINCE can only run on Linux
No
Merged Document
Create an encrypted copy of the document, one for storage and one to be sent.
Copy 1
Follow workflow for Storage daemon
Copy 2
Documents are placed in either LETTERS or EMAILS folder awaiting Email or Letter Daemon and DistributionStatus = 'D' Awaiting Distribution
LETTER Distribution daemon queries for letters in the LETTERS folder
Scheduled via the daemon properties. All letters in the LETTERS folder at the time that this daemon runs are included in the ZIP
For each letter, set the DistributionReference on the CommsRequestTargetDetail table. All letters in a ZIP have the same Distribution Reference
ZIP the letters
Update DistributionStatus to S(Sent) for each Letter that is in the ZIP i.e. that has the specified DistributionReference

3. 5 Comms Process Flow: Store Letters and Emails

Once the Comms Email Monitor or Comms Letter Monitor jobs have created a merged document from Velocity templates and default field values, they create two encrypted copies of the document. One is to be sent to the customer, the other is to be placed in long-term storage.

Storage is a distinct step carried out by the Transmission Comms to Document Storage daemon. This allows the documents to be placed into storage on any machine, for example in systems such as Amazon S3 or on a machine via SFTP. Storing the documents in a long-term storage directory allows them to be viewed in AgentView. The storage daemon runs in parallel to the distribution daemons; these daemons are independent of each other.

The files to be stored are first placed in a temporary storage location by the job, as configured for the property for the sabre-comms module in the Administration Console: `comms.document.storage.monitor.transmission.from.properties.dirname` The files have a `.done` suffix.

The Transmission Comms to Document Storage daemon polls the temporary storage for `.done` files. The daemon moves these files to the location specified for the property: `comms.document.storage.monitor.transmission.to.properties.dirname`. This can be a remote location. The files are transmitted by SFTP. The hostname, port number, username and password used for transferring files by SFTP are set in the daemon properties. For more information, see "Transmission Comms to Document Storage" on page 11 in the *Communications Jobs and Daemons* section.

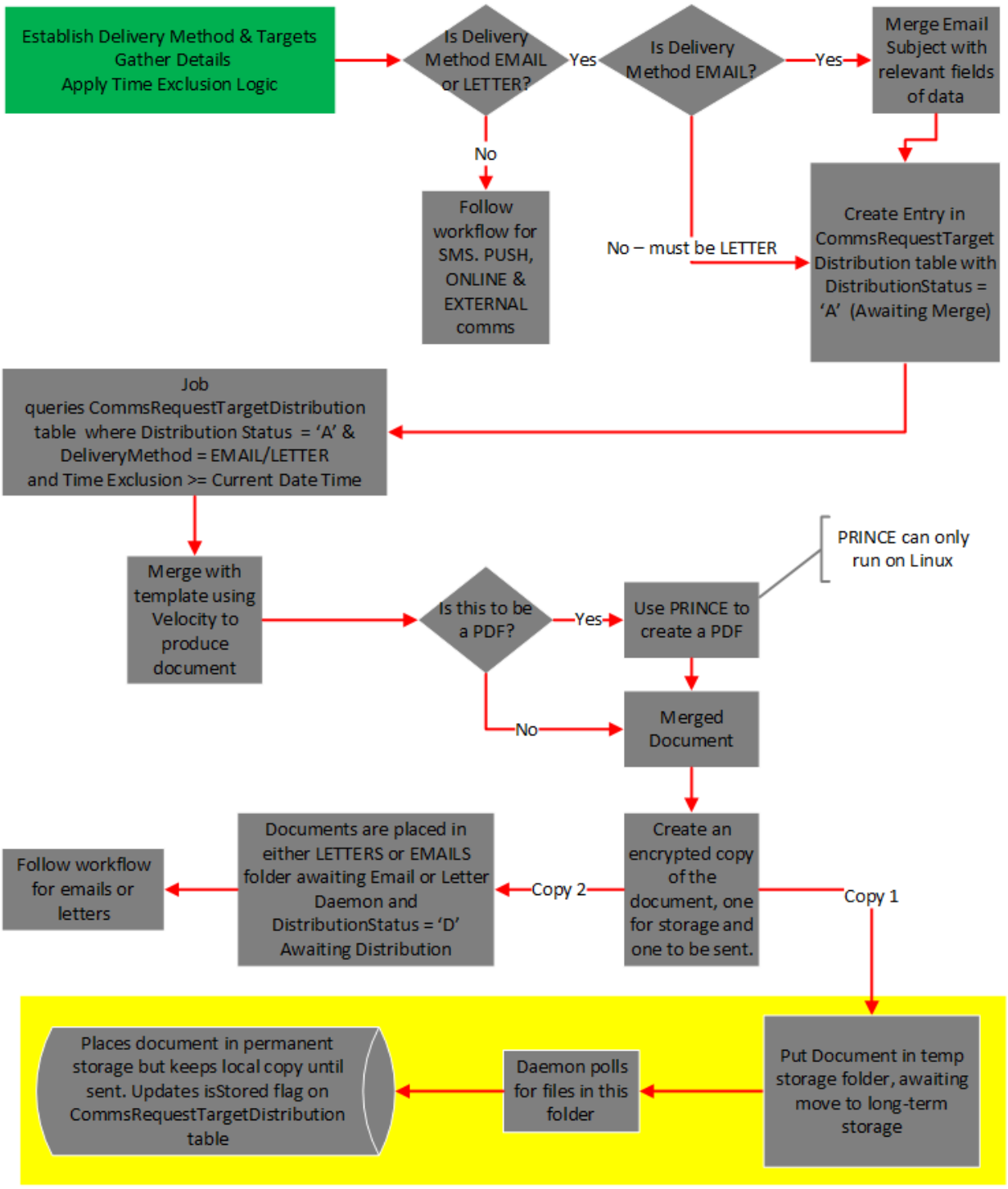
The directory structure in which the documents are stored is automatically created. The naming convention for the directory includes the current date and time. The configured root folder in remote machine follows the convention: `YYYY/MM/dd/HH/<Comms Req Document>`, for example `/permanentstorage/2018/08/02/17/5002761.pdf`.

The daemon then updates the CommsRequestTargetDistribution table with:

- isStored = true
- FileLocation = location where the document is stored

If the transmission of a file fails, the daemon raises an error and stores the affected file in the location configured for the daemon property: `comms.document.storage.monitor.transmission.from.options.moveFailed`.

The highlighted portion of the following diagram illustrates the storage workflow for letter and email communications:



3. 6 External Comms Status Updates

The Acknowledge Comms Receipt Of daemon takes in a JSON message/file and then updates values of a particular communication, for example an update on the status of a communication from the third party that actually sent it.

The daemon updates the DistribustionStatus on the Comms RequestTargetDistribution table accordingly:

- S (Sent)
- E (Error)

For more information, see "Acknowledge Comms Receipt Of" on page 11 in the *Communications Jobs and Daemons* section.

3. 7 Comms Not Sent Workflow

A comm can have multiple delivery methods which may not be compatible with the target's delivery method. If a comms cannot be sent, CMP automatically raises a workflow - the Comms Not Sent workflow event. The reasons why delivery methods could not be implemented are written to a database table, CommsRequestNotSent. The status of the comm is changed to Not Sent. The reasons can be viewed via the **Reasons Not Sent** option in AgentView in the **Comms > Comms** lower panel in the relevant **Subscription Summary** screen or the **Comms** pane in the **Event Summary** screen **Profile >Summary** lower panel.

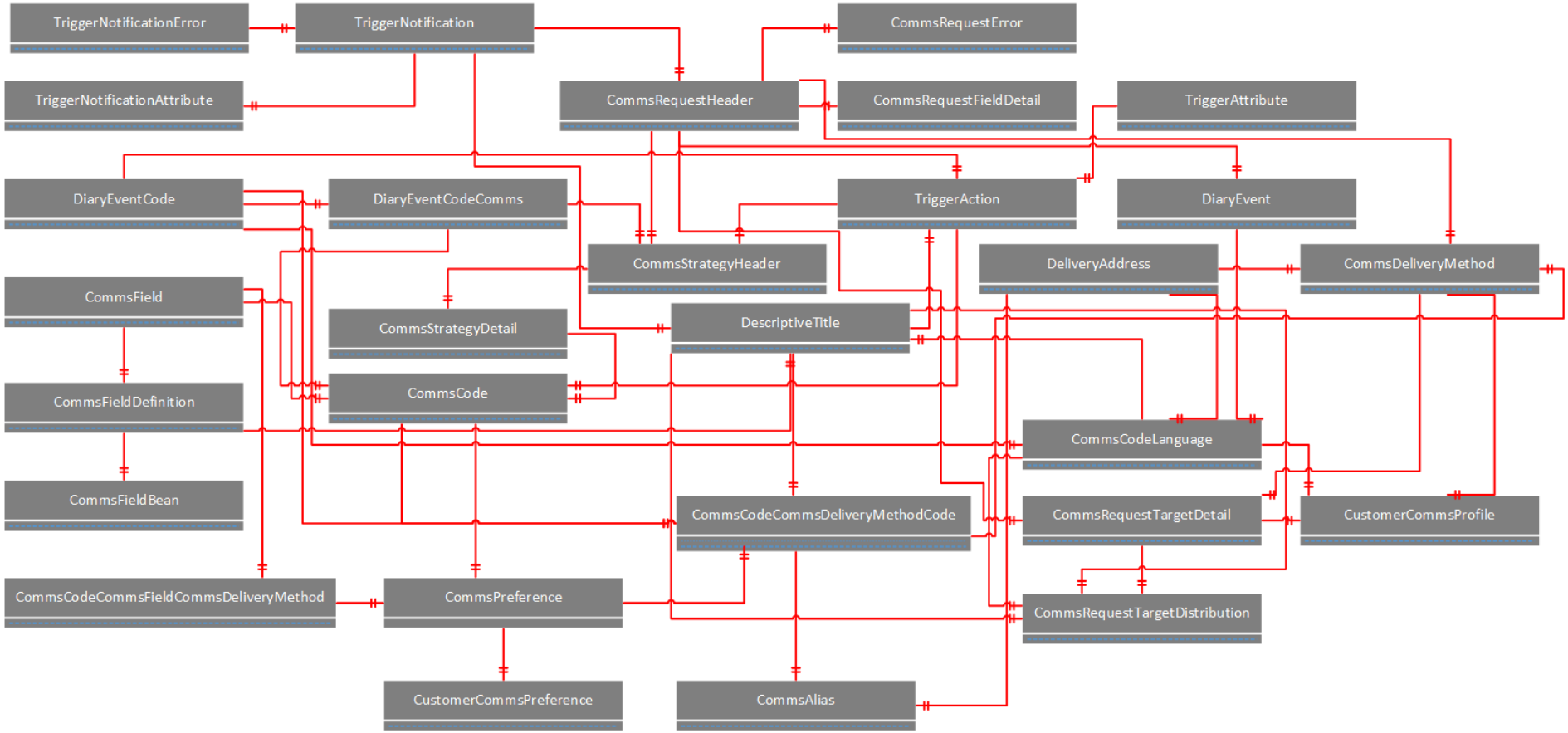
3. 8 Comms Error Workflow

If a comm raises an error, the reasons for it are written to the database and the comms status is set to Error. The error can be viewed using the Error Details option in AgentView in the **Comms > Comms** lower panel in the relevant **Subscription Summary** screen or the **Comms** pane in the **Event Summary** screen **Profile >Summary** lower panel.

4.0 Communications Database Tables

The main avenue for communications in CMP are Comms Requests, which are entries in the CommsRequestHeader table. This and other database tables that are mainly involved in CMP communications are described in the sections that follow.

The following diagram shows the relationship between the main database tables for CMP Comms:



The tables shown here are the major tables involved in communications. The list is not exhaustive.

5.0 Communication Email and Letter Templates

CMP Supports outbound communications to customers using a number of delivery methods, including email and letter. The Communications process merges templates and default field values together to produce letters and emails that are personalised for customers by including customer-specific attributes from the CMP database.

See the "Communications Process Flow" on page 13 topic for more information.

To achieve this, letter and email communications must be associated with an additional external template that determines the format, style and layout of the communication. HTML templates are created using external frameworks such as Velocity produce output such as PDF, HTML and RTF.

In CMP 8.x, templates are created manually using Velocity Template Language. Velocity uses templates to produce the output (HTML for example) and Prince to convert HTML output from Velocity templates to PDF:

- For more information on Velocity, click here.
- For more information on Prince, click here.

Templates contain elements such as static text, images, headers and footers, and dynamic text or *merge fields*. The merge fields are populated with the values from the CMP *default fields*.

Default fields contain customer-specific information from the CMP database that can be included in communications with end users. The graphic below shows a section of a template that includes style settings such as the text font, a logo image and default fields that contain address and subscription information:

```
<html>
    <body style="font-family: LiberationSans">
    <img alt="mds_logo.png" src=
    "$imageStoreMap['mds_logo.png']" style=
    "width:304px;height:228px;"/><BR>
    <BR/>

    $fieldDataMap['ADDRESS1']<BR/>
    $fieldDataMap['ADDRESS2']<BR/>
    $fieldDataMap['ADDRESS3']<BR/>
    $fieldDataMap['ADDRESS4']<BR/>
    $fieldDataMap['ADDRESS5']<BR/>
    $fieldDataMap['POSTCODE']<BR/>
    $fieldDataMap['FIRSTNAME']<BR/>
    $fieldDataMap['SHORTDELIVERYADDRESSNAME']<BR/>


    Subscription Bean<br/>
    Subscription Number: $fieldDataMap['SUBSCRIPTIONNUMBER']<br/>

    Subscription Connected Date: $fieldDataMap['SUBSCONNECTEDDATE']<br/>

    Subscription Disconnected Date: $fieldDataMap['SUBSDISCONNECTEDDATE']<br/>

    Subscription Termination Date: $fieldDataMap['SUBSTERMINATIONDATE']<br/>

    Password: $fieldDataMap['SUBPASSWORD']<br/>

    Subscription Contract Start Date: $fieldDataMap['SUBCONTRACTSTARTDATE']
    <br/>

    Subscription Contract End Date: $fieldDataMap['SUBCONTRACTENDDATE']<br/>
    <br/>
```

Default fields are associated with beans, which are the software objects that retrieve the data from CMP. When CMP processes a communication, it checks which fields needs to be populated for that particular communication and then invokes the associated beans, which fetch the data from the CMP database according to the configured parameters. This is explained in detail in "Comms Process Flow: Gather Details" on page 21 and "Comms Process Flow: Create and Send Letters and Emails" on page 46.

> For more information see Communications Default Fields and the online help for the Business Configuration module of the CMP Administration Console.

Examples of Velocity templates and lists of default fields are available in the "Appendix" on page 85.

5. 1 Working with Email and Letter Templates

In CMP, external templates determine the format and style of customer correspondence.

Software

The template solution is based on Apache Velocity 2.2. Templates are created using Velocity Templating Language (VTL). Velocity supports creating correspondence templates in HTML, RTF and TXT format. Prince is the application used to convert the merged templates into PDF format.

Templates can contain static text, dynamic content (merge fields), images, logos, a header, and a footer.

Storage

Templates are stored in a default location so that CMP can access the templates: `/var/mdsglobal/sabre-server/templates`. The default location for images to be included in the templates is `/var/mdsglobal/sabre-server-/templates/img`.

If customers want to store templates in another location, CMP must be configured with the new location. Once correspondence is processed, a dedicated daemon stores the generated letters and emails in long-term storage so that they can be viewed in AgentView. For more information, see "Comms Process Flow: Store Letters and Emails" on page 56.

Template User Guides

The Velocity user guide is available here: http://velocity.apache.org/engine/devel/user-guide.html

Documentation for Prince is available here: https://www.princexml.com/doc/

Naming Convention for Templates

Templates must be named according to the naming convention `<COMMSCODE>-<LANGUAGE>-<DELIVERYMETHOD>-<Format>.vm` for example `LETT1-EN-LETTER-pdf.vm or MAIL1-EN-EMAIL-html.vm.`, where `<LANGUAGE>` can be:

- `EN` = English
- `SP` = Spanish

Adding Default Fields and Objects to Templates

Adding Default Fields

To add customer data from a default field to a template, add a leading dollar symbol ($) to the `fieldDataMap` reference, then in square brackets - [ ] - supply the default field name, enclosed in single quotes:

`$fieldDataMap['<default field>']`

For example, the following adds account balance information to a communication:

```
<br/>
Total Balance: $fieldDataMap['ACCOUNTBALANCE']<br/>
Total Amount Due: $fieldDataMap['ACCOUNTAMOUNTDUE']<br/>
Total Amount In Query: $fieldDataMap['ACCOUNTAMOUNTINQUERY']<br/>
Total Amount In Arrears: $fieldDataMap
['ACCOUNTAMOUNTINARREARS']<br/>
<br/>
```

For a full list of CMP default fields, see the "Appendix A: CMP Default Fields and Beans" on page 86.

Adding Objects

Objects such as images, logos, headers, and footers must be stored within the same file-sharing system as the template. Templates reference these objects in order to include them in emails and letters.

Reference an image using `imageStoreMap` by providing the filename of the referenced image enclosed in single quotes in square brackets as follows:

```
<img alt="mds_logo.png" src="$imageStoreMap
['mds_logo.png']" style="width:304px;height:228px;"/>
```

Adding External References

You can use externally available objects by referring to a standard URL, for example:

```
<img src="http://www.mdscem.com/sites/default/files/2016-06/Re-
tail-VNO-Image-Banner.png">
```

Formatting Dates for Emails and Letters

You can specify the format for dates in emails and letters by using Velocity's `DateTool`[1]. For example, the following template extract maps to the `RESOLUTIONREQUIREDBYDATE` default field and uses `date.format` to specify how the date data stored in that default field should display:

```
Hi $fieldDataMap['YOURBILL'] <BR>

<br>

Raw date is $fieldDataMap['RESOLUTIONREQUIREDBYDATE']

Formatted date is ${date.format('yyyy-M-d', $fieldDataMap
['RESOLUTIONREQUIREDBYDATE'])} <br>
```

[1]Tool for working with Date and Calendar in Velocity templates. It is useful for accessing and formatting the "current" date as well as for formatting arbitrary Date and Calendar objects.

```
Long formatted date is ${date.format('long', $fieldDataMap
['RESOLUTIONREQUIREDBYDATE'])} <br>

This is the result:
```

Hi Your Bill

Raw date is 2021-04-28 00:00:00.0
Formatted date is 2021-4-28
Long formatted date is 28 April 2021 00:00:00 UTC

For possible formatting options, click here.

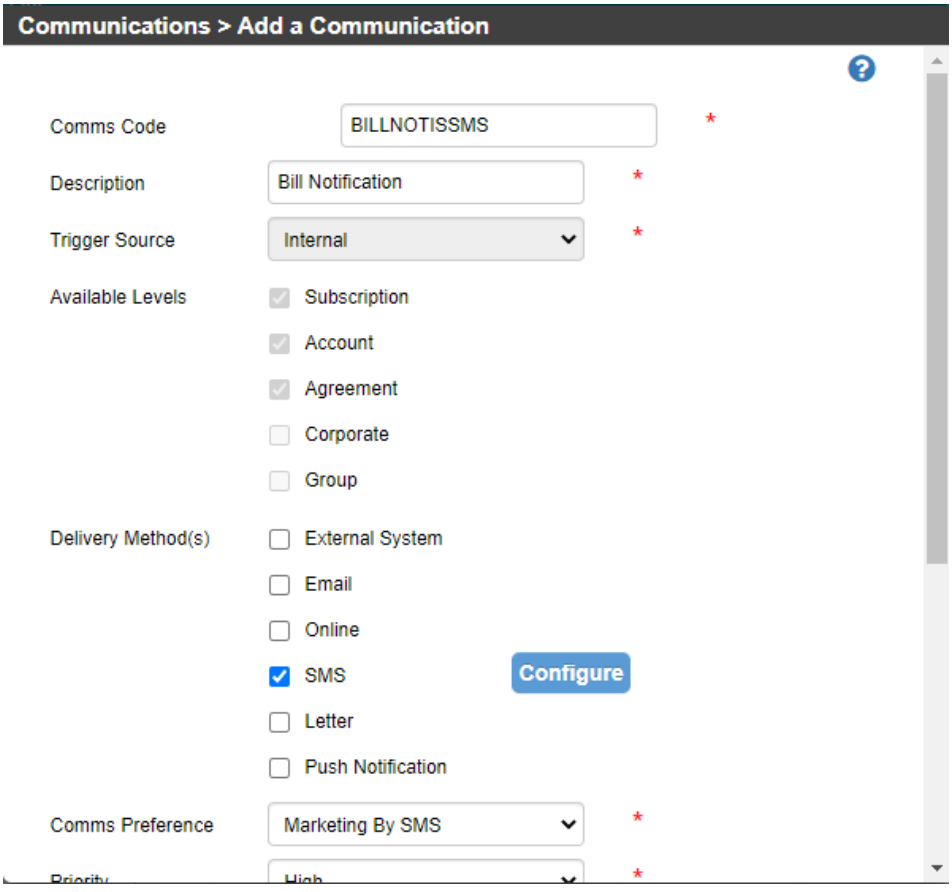
6.0 Example: Configuration and Processing of a Communication

Suppose a provider wants to send customers an SMS or push notification when their bill is ready. This section examines the end-to-end workflow for an individual bill notification communication, from configuration to distribution.

6. 1 Configuration

6. 1. 1 Communication Configuration

To notify customers that their bill is ready by either SMS or push notification, the Business Configuration user creates a communication template in the Business Configuration console in **Communications** > **Communications List**:

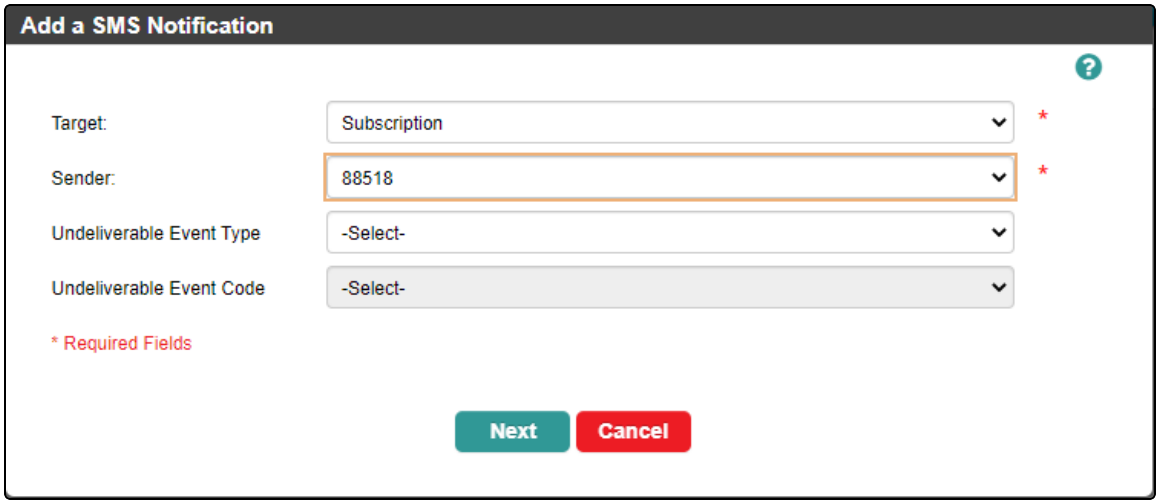


The settings are as follows:

Name	Description
Comms Code	A unique alphanumeric code for the communication. Allowed characters are A - Z and 0 - 9. In this case, the code is BILLNOTISMS.
Description	A brief meaningful descriptive name for the communication, in this case Bill Ready Notification.
Trigger Source	Whether the initiating event for the communication is internal to CMP or from an external system, for example a network. In this case, the trigger source will be a notification from a print bureau in the form of a JSON file.*
Available Levels	The levels in the CMP hierarchy at which the communication can be selected and generated; Subscription in this case.
Delivery Method(s)	SMS*
Comms Preference	The set of preferences that apply to this communication. In this case, the provider has selected the Default Comms Code Preferences.
Priority	Very High - it's important that a customer know that they have a bill to pay.
Security Level	The security levels or roles that have access to the communication, can view it in the system, and select it. In this case the security level is **Everyone**, the lowest level.
Selectable	Whether the communication can be selected in the CMP GUI - it is, in this case.
Active	Whether the communication is available to new subscribers. This communication is active.
Explanation	Additional explanatory text to help the future CMP users understand the purpose and configuration of the communication: "Comms via SMS to advise that the customers next bill is available"

*These settings will be examined in more detail.

To configure the SMS message, the user clicks **Configure** in the **Add a Communication** window and provides:



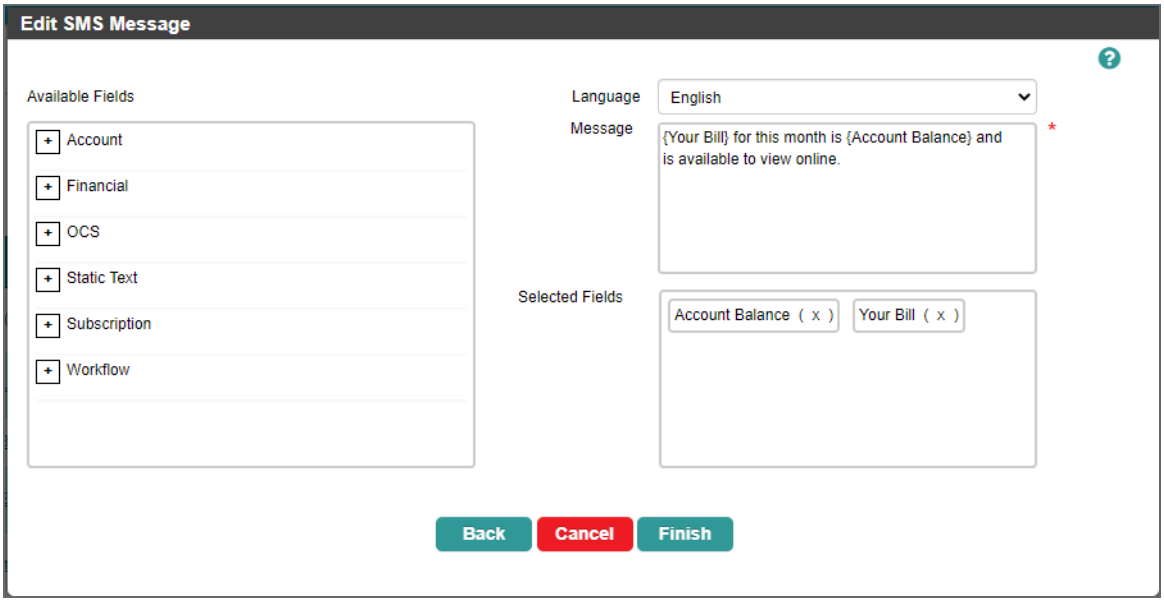
- A target - in this case the mobile number at the subscription level (according to the settings above, the communication can only be raised at subscription level).
- A sender - the mobile number used to send SMS from CMP for the provider. This can be an alias.

Users can also choose to configure a workflow event that will be triggered if the SMS is undeliverable.

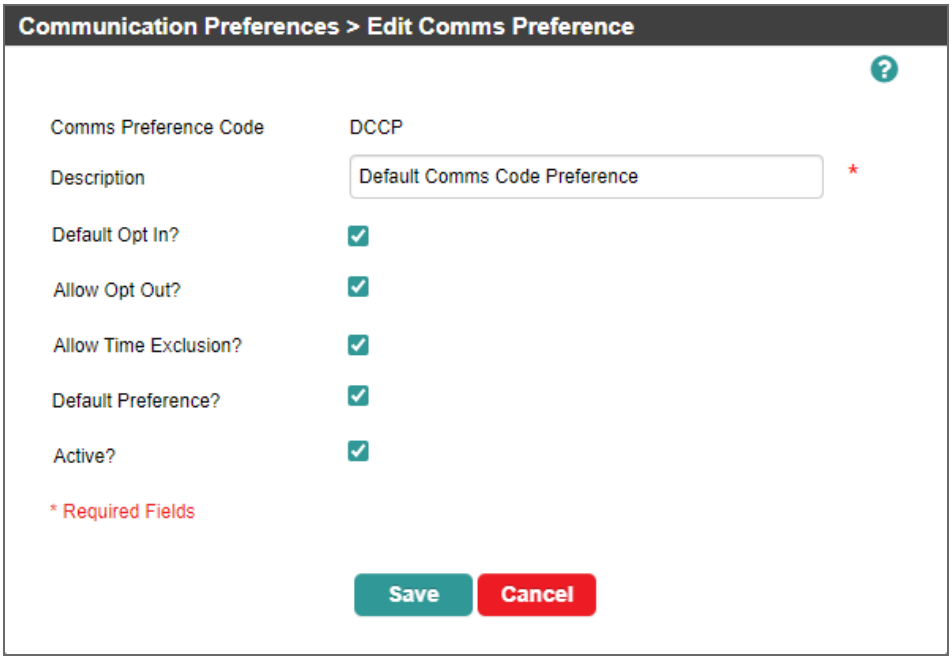
Next the user needs to configure the SMS message itself. The user chooses to send the text in English and selects two default fields to include:

- `{Your Bill}` - a simple text default field that includes the words *Your Bill* in the SMS message.
- `{Account Balance}` - a default field that will enable CMP to retrieve the account balance from the CMP database.

The final message for the SMS will be: `{Your Bill} for this month is {Account Balance} and available to view online.`



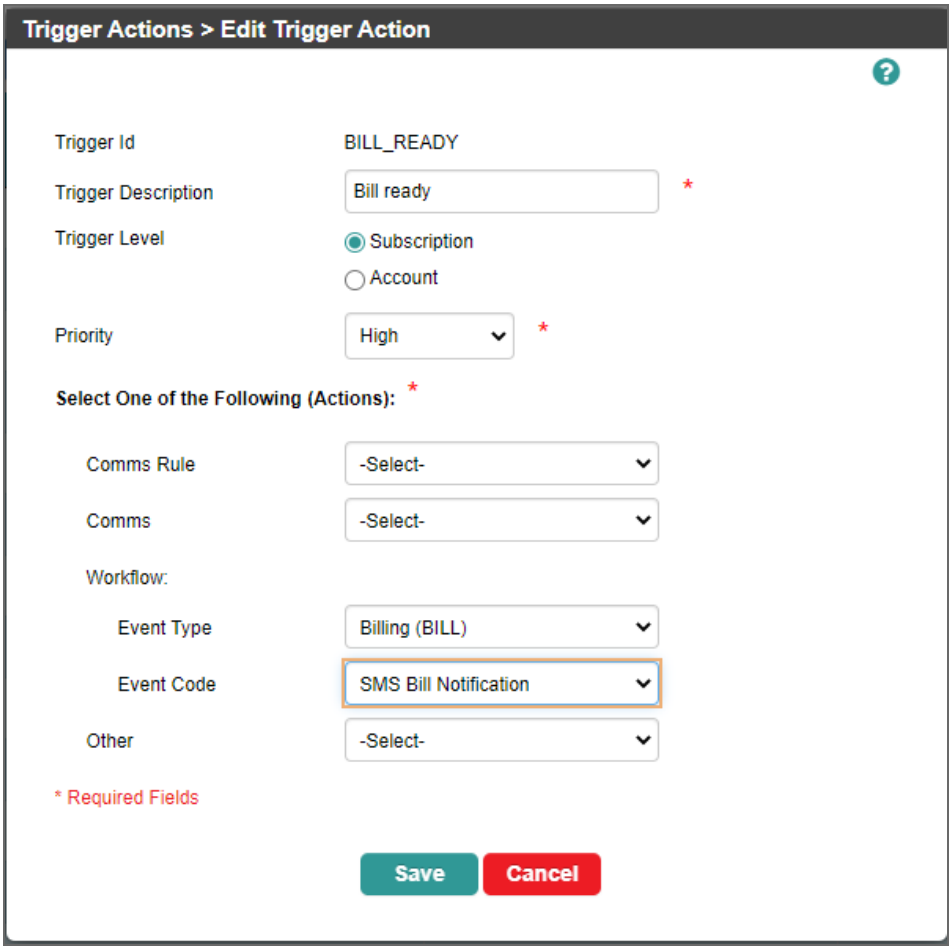
The preference that the user set for the SMS communication was the Default Comms Code Preference. It has been configured as follows:



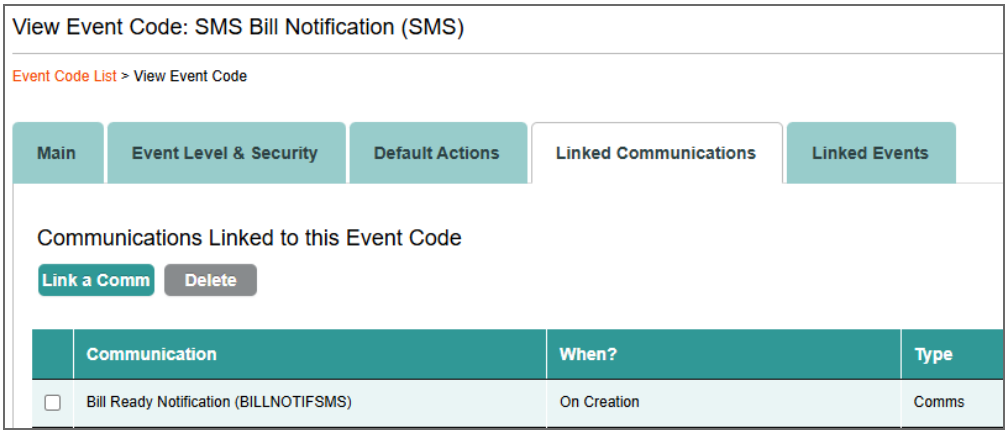
This means that by default, a customer is opted in to receiving a bill notification SMS, but opting out and setting a time exclusion period are allowed. This is expected, as customers may not want to be disturbed by SMS messages at certain times. The Default Comms Code Preference is also available to be applied to new subscriptions.

The trigger source for the communication is External - a notification from the print bureau. The user therefore configures a trigger action for the communication called BILL READY. This trigger raises a workflow event of the Billing type called SMS Bill Notification.

> The comms can also be raised directly by selecting the comms code from the Comms drop-down.



The SMS Bill Notification workflow event is configured to send the Bill Ready Notification SMS communication as soon as the event is created.



So the user has configured the following: A communication in the form of an SMS that alerts a customer that the bill for a subscription is ready. The message will include a subscription's account balance. When CMP receives a notification from an external system -

the print bureau - this will trigger the creation of a workflow event that sends the communication.

6. 1. 2 Jobs and Daemons Configuration

The following jobs and daemons will be involved in processing the Bill Ready Notification communication. Configuration that affects the processing of the Bill Ready Notification communication is as follows:

Load Comms from Generic Format daemon

This daemon is configured to actively poll for JSON messages on the externalComms queue with a specific channel defined. The ActiveMQ queues in which inbound triggers are placed are configured, as are the queues for erroneous triggers.

Notifications Monitor job

The Notifications job has been set up to fire automatically if a new external notification becomes eligible for processing.

Comms Monitor job

The Comms Monitor job will fire automatically if a new comms notification becomes eligible for processing. The job has also been configured with the date format to be used when representing dates in an SMS. The international dialling prefix is not configured to be displayed in SMS communications and the default language code is set to English.

Transmission Comms SMS to Handset daemon

The daemon is configured to actively poll for new messages every 100 milliseconds. It is configured to place messages on the outboundCommsSms queue to be picked up by an external system or by the SMPP adapter. The status to which it will update the communication after it is sent out of CMP is set as S (Sent).

6. 2 Processing

6. 2. 1 Triggering the Communication

The print bureau sends a notification to CMP that a customer's bill is ready. The notification is in the form of a JSON schema, `commsExternalTrigger.schema`.

This JSON file is picked up by the Load Comms from Generic Format daemon, which is continuously polling the Active MQ queue for inbound trigger notifications. The daemon decrypts the JSON file and writes the record in the file to the TriggerNotification database table as follows:

- TriggerNotificationId - Auto-generated by the database
- SuppliedTriggerId - this is from the JSON
- DateTimeNotificationReceived- Current time/date
- TriggerChannelDTC is supplied in the JSON.

- SuppliedUniqueReference - This may or may not be supplied in the JSON and is intended to capture a unique reference for the message from the external system if one is supplied; for informational purposes only. In this case, it is not supplied.
- TriggerNotificationStatus - This is set to A - Awaiting Processing.
- AuditProcess, AuditUser, AuditTimestamp - the current date and time and the user and process responsible.

Once the TriggerNotification table is updated, the Notification Monitor is triggered automatically.

The Notification Monitor job cross-references the record (SuppliedTriggerId value) from the TriggerNotification table to the TriggerID value in the TriggerAction table to see what communication needs to be sent - BILLNOTISMS, in this case. The jobs raises the workflow event configured for BILLNOTISMS, which triggers a Comms Request.

The job creates an entry in CommsRequestHeader table with the following values:

- CommsRequestID - Autogenerated
- Priority - from the Priority on the TriggerAction table. Very High, in this case.
- EventNumber - The number for the configured workflow event (Billing SMS Bill Notification).
- TriggerNotificationId - from the TriggerNotification table
- CommsCode - from the TriggerAction table; BILLNOTISMS, in this case
- CommsStrategyHeaderCode - from the TriggerAction table
- Request Status - A - Awaiting Processing
- MergeDateTime - Null
- OverrideCommsDeliveryMethodCode - Null
- AuditProcess, AuditUser, AuditTimestamp - the the current date and time and the user and process responsible.
- CopyCommsRequestId - This is used if the comms needs to be resent
- IsEligabletoProcess - Yes

Because a workflow event is configured against the TriggerID in the TriggerAction table, the workflow will be raised against the supplied customer reference (the subscription number) supplied in the original JSON file, using the Event Type (Billing) and Event Code (SMS Bill Notification) from the TriggerAction file.

Once the request has been completed, the job updates the TriggerNotificationStatus in the TriggerNotification table to P (Processed).

6. 2. 2 Gathering the Communication Details

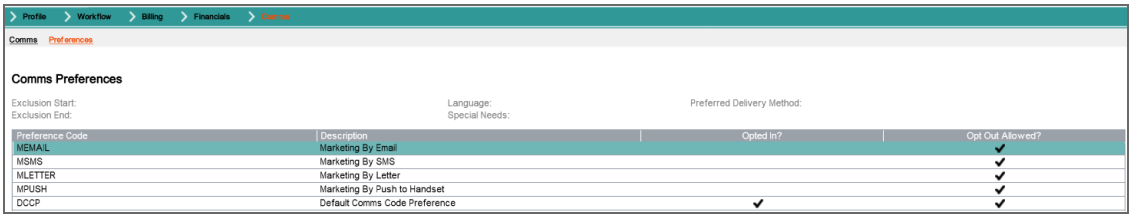
When the new Bill Ready Notification communication is entered in the CommsHeaderRequest table, the Comms Monitor job is automatically fired and takes over the processing of the communication.

First it determines the delivery method. The job checks whether the OverrideDeliveryMethodCode field in the CommsHeaderRequest table is populated. This

communication is not a one-off communication added in AgentView, so no override delivery method is set.

Next the job works out the Opt In/Out logic for the communication. The communication was configured to be raised at subscription level and set up with the Default Comms Code Preference. This means the job looks for an entry in the CustomerCommsPreference table at subscription level first and it finds one, bases the Opt In logic on that. The Default Comms Code Preference is that the customer is opted in to the communication. However, the preference is set up so that the customer can opt out.

In this case, the customer has not altered the settings of the Default Comms Code Preference, meaning they are opted in to the communication and they have not set a time exclusion, as shown here in the **Comms** > **Preferences** panel for the **Subscription Summary** in AgentView:



Once the Comms Monitor now gets a list of delivery methods associated with the comm and identifies out the highest priority method to populate the CommsRequestTargetDetail table. To do this, the Comms Monitor job must take into account any customer preferences, the delivery methods configured against the communication and the priority of those delivery methods.

Because the communication is raised at subscription level, the job first checks for customer preferences at this level. It finds that SMS is supported at this level. SMS is also the only delivery method configured for the communication, so the job need not consider delivery method priorities and use SMS as the delivery method.

Once the job has determined the delivery method is SMS, it must find the target, which is the mobile number at subscription level. The Bill Ready Notification communication was configured with **Subscription** as the target (as opposed to **Longest Sub on CMP** or **Parent Subscription**, which are only available at account level), so the jobs checks whether the subscription network will accept SMS (IsSMSAllowed = true on the Subs Network Type). It does; so the job retrieves the mobile number of the subscription to which the communication will be sent and updates the PrimarySerialNumber field on the CommsRequestTargetDetail table. The PrimarySerialNumber is written in E164 format.

Now that the Comms Monitor job has determined the delivery method and target for the communication, it must gather the details of the message. This means getting the values for any default fields used in the message and merging them with the message text as follows:

1. The Comms Monitor job writes a record into the CommsRequestFieldDetail table.
2. The job gets list of fields associated with the comms - {Your Bill} and {Account Balance} in this case.
3. The job invokes the relevant beans used by the default fields to populate the CommsRequestFieldDetail table - the STATICTEXT and ACCOUNT beans respectively, in this case.
4. The job repeats step 3 until the list of fields is complete and updates the RequestStatus in the CommsRequestHeader table to P(Processed).
5. Then the job checks for time exclusions. In this case, the customer did not set any time exclusions.
6. The job checks the CommsCodeLanguage table to identify which Message value should be used for the comm. In this case, the message is: `Your Bill}` for this month is `{Account Balance}` and available to view online.
7. The job sets the Message value in the CommsRequestTargetDistribution table with the correct merged field (CommsFieldCode and TextValue from CommsRequestFieldDetail table) and sets the DistributionStatus to D (AwaitingDistribution).

The merged message created in step 7 is now ready to be sent.

6. 2. 3 Sending the Communication

The SMS To Handset (Transmission Comms SMS to Handset) daemon picks up entries that are ready to be sent via SMS.

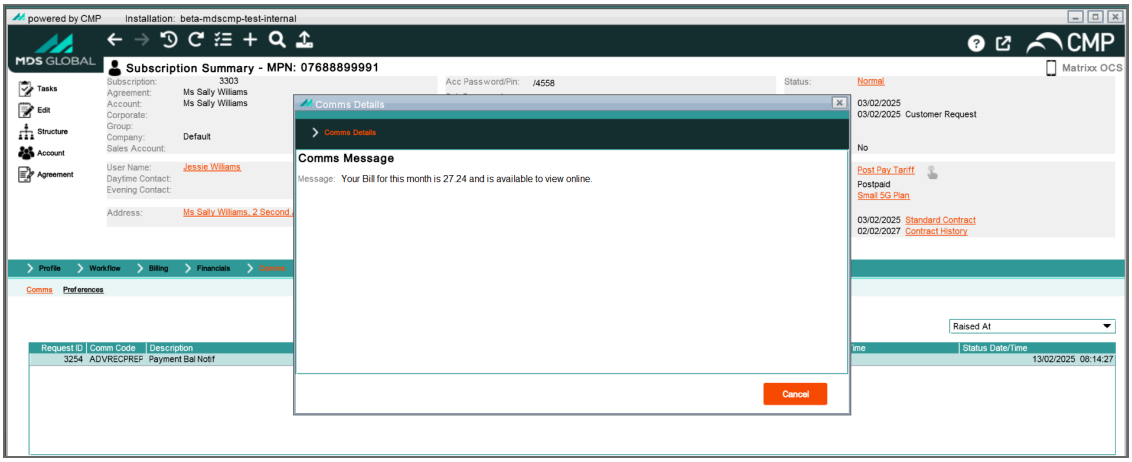
The daemon queries the CommsRequestTargetDistribution table where:

- Distribution Status = D
- AND DeliveryMethod = SMS
- AND Time Exclusion >= Current Date and Time

The daemon creates the JSON file and puts it on the outboundCommSms queue as set for the daemon.

Once the message is placed on the queue, the job updates the DistributionStatus to S (Sent) and updates the SentDateTime on the CommsRequestTargetDistribution table.

The final step is to store the message in long-term storage so that it can be viewed in AgentView by an agent, as shown here:



The section "Communications in AgentView" on page 75 goes into more detail.

7.0 Communications in AgentView

In AgentView, an agent can perform the following tasks related to communications:

- View communications.

  The details for communications for an account or subscription can be viewed in the relevant Summary screen. Agents can also view the stored communications themselves. See "View Communication Details" below and "View Communications" on page 77.

- View and set communications preferences.

  Agents can view communications preferences, change opt in/out settings, set time exclusions and language for communications and configure a preferred delivery method. See "View and Edit Communication Preferences" on page 78.

- Raise a communication request.

  Agents can raise a workflow event that is configured to send a communication or they can add a one-off communication to a workflow event. See "Add a One-Off Communication" on page 80.

- View details why a communication failed.

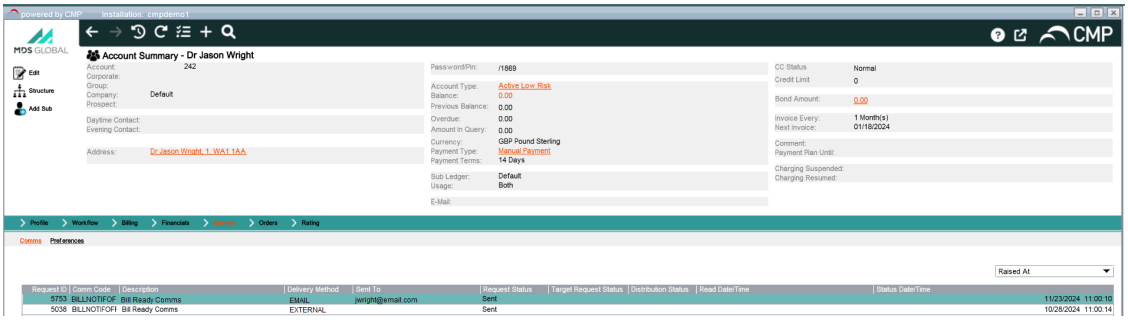
  Agents can view the reasons why a comm was not sent or view comms error details. See "View Reasons for Comms Not Sent and Error Details" on page 82

For more information on communications in AgentView, see the online help for the application.

7. 1 View Communication Details

The communications related to an account or subscription can be viewed in the following areas in AgentView:

- The **Comms** pane in the **Event Summary** lower panel for the relevant workflow event.
- The **Comms** > **Comms** lower panel in the relevant **Summary** , **Account**, or **Agreement** screen.



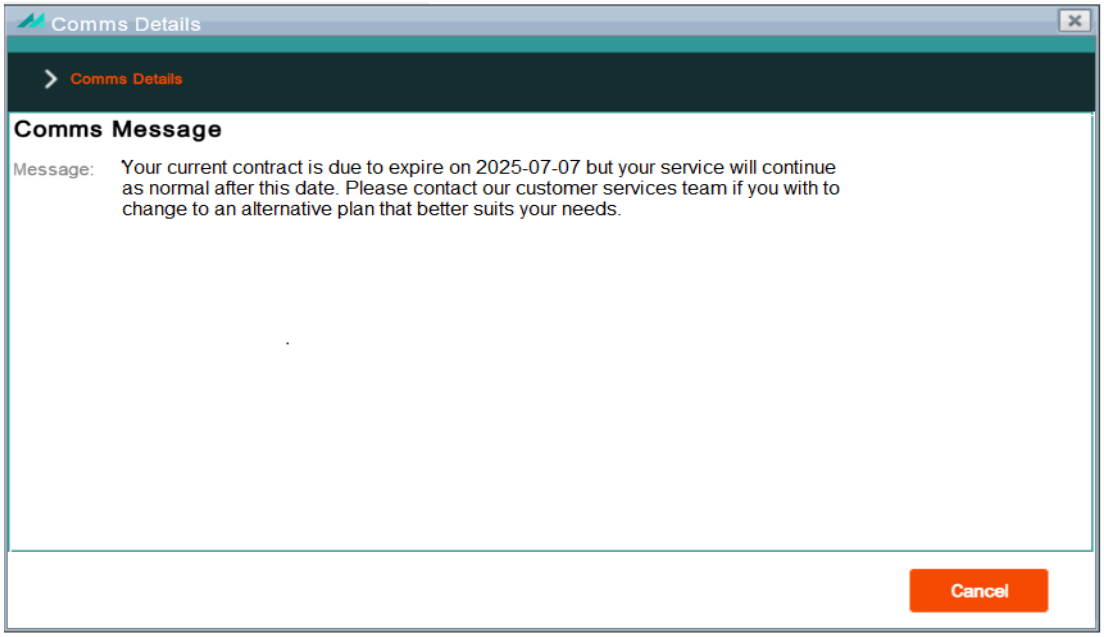
*Account **Summary** screen showing the **Comms** lower panel with communication details*

The communications for the subscription or account are listed with the following column headings:

Field	Description
Request ID	The ID of the request that raised the communication
Comms Code	The alphanumeric code for the communication
Description	The brief description name of the communication, for e.g. Bill Notification, Card Payment SMS
Delivery Method	How the communication was delivered, e.g. email, letter, SMS, online, external system or push notification
Request Status	The status of the communication request, e.g. Awaiting Processing, Processed, Cancelled, Not Sent, Opted Out, Error etc.
Target Request Status	The status of the target of the communication, e.g. Normal, Error
Distribution Status	The status of the communication distribution e.g. Published, Sent, Error etc.
Read Date/Time	The date and time that the communication was read
Status Date/Time	The date and time for the most recent status for the communication

You can filter the list of communications using the list on the right to view communication by when they were **Raised At** (Default) or who they were **Sent To**.

In either the **Comms** pane or **Comms** lower panel, to view the message content for an individual SMS, push or external comms, you can use the **View Details** shortcut menu option to access the **Comms Details** pop-up.

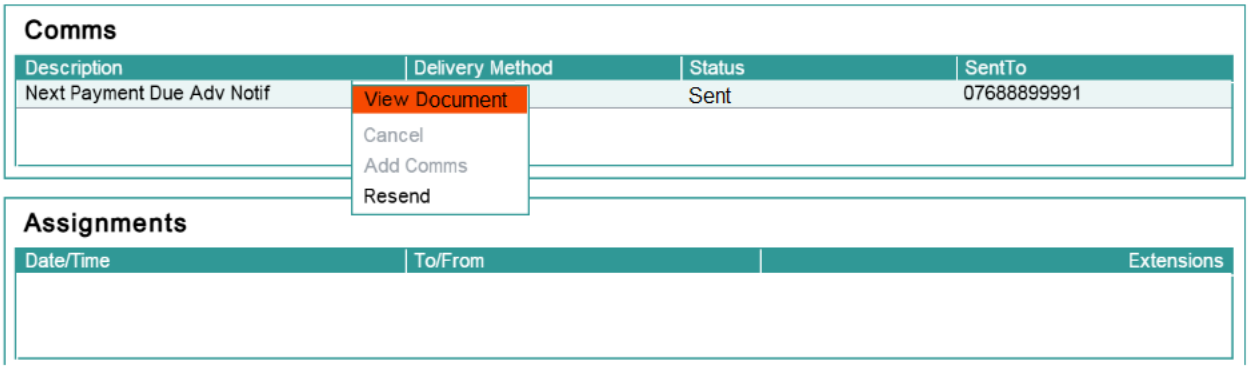


***Comms Details** pop-up showing the message content for a push notification*

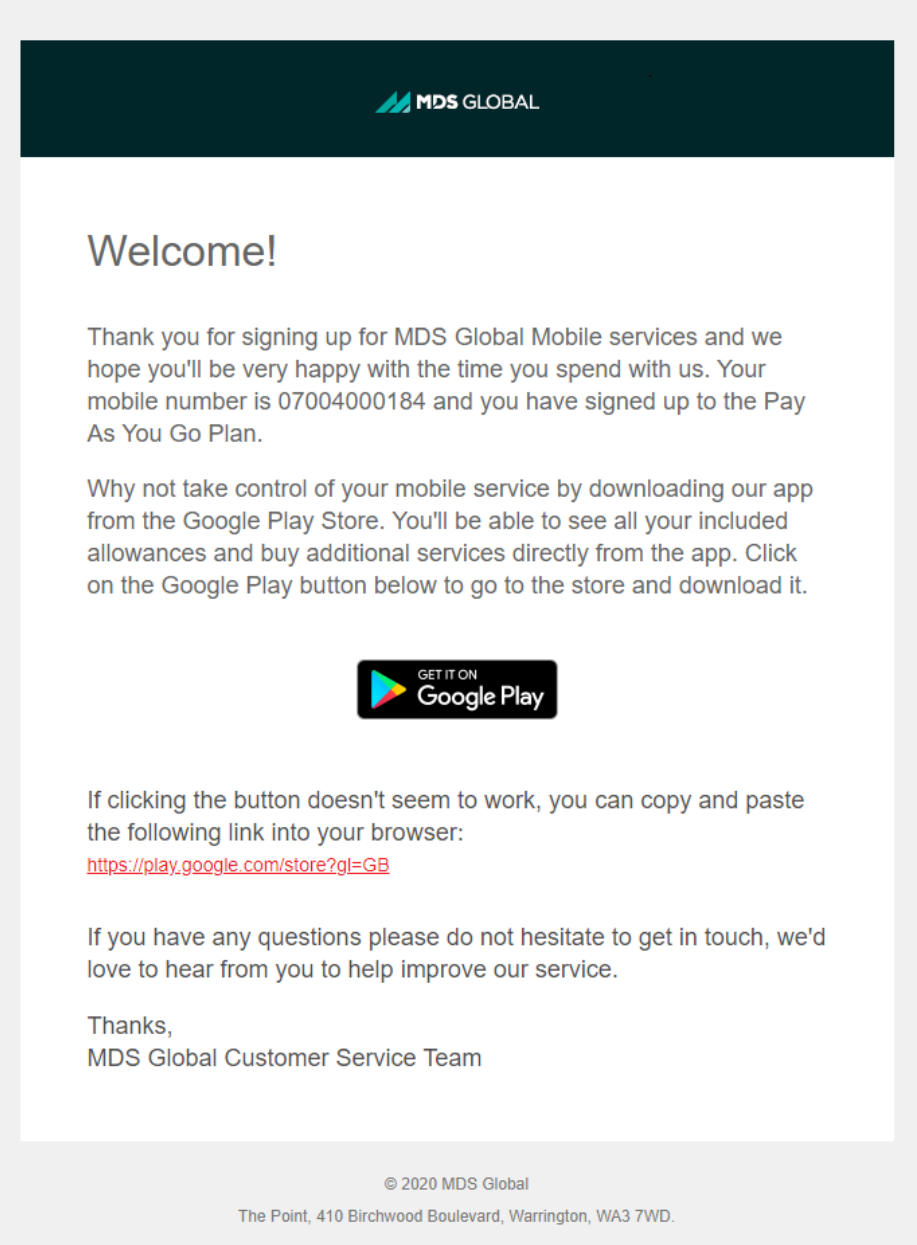
7. 2 View Communications

In AgentView, agents can view stored communications (comms) for letters and emails that have been raised for workflow events as follows:

- In the relevant **Subscription**, **Agreement** or **Account Summary** screen, in the **Comms > Comms** lower panel, by right-clicking the comms and selecting **View Document**.
- In the relevant **Event Summary** screen by right-clicking the comms in the **Comms** panel and selecting **View Document**:



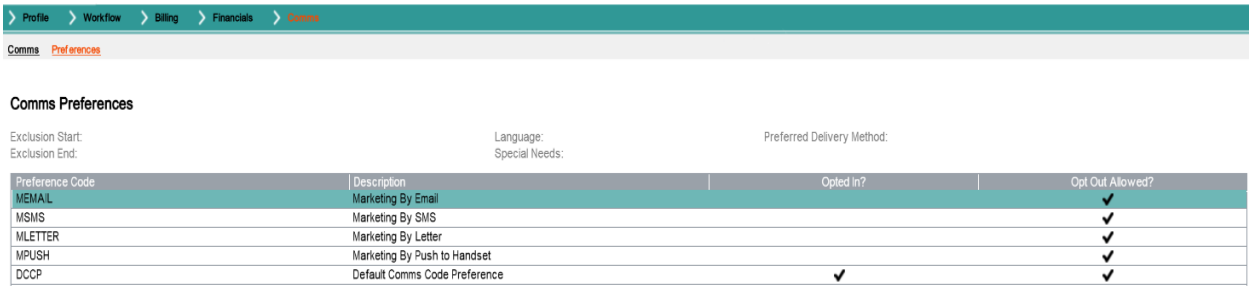
The stored document is then displayed to the user. The following is an example of an email communication:



7. 3 View and Edit Communication Preferences

7. 3. 1 View Communication Preferences

In AgentView, the communications preferences for a subscription or an account can be viewed in the **Comms** > **Preferences** lower panel in the relevant **Summary** screen.



The panel displays the customer preferences as follows:

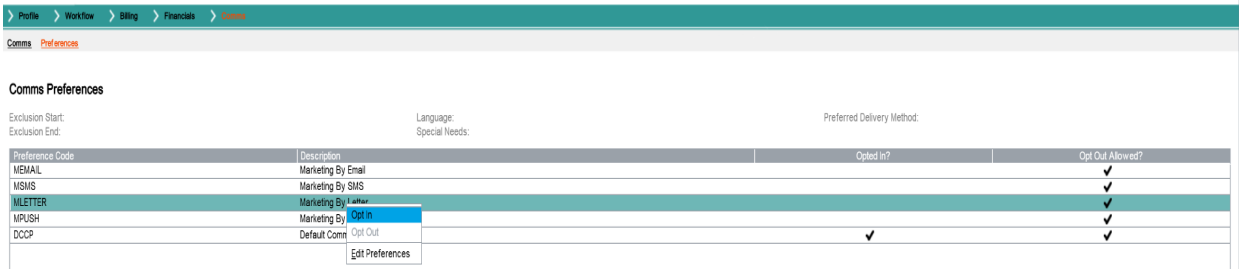
- **Exclusion Start** and **Exclusion End**: The start and end times if an exclusion period is set.
- **Language**: The customer's preferred language, if set, for example English or Spanish.
- **Special Needs**: Special needs preferences such as Braille.
- **Preferred Delivery Method**: If the customer has a preferred delivery method, for example SMS, EMAIL, PUSH etc.

The communications preferences are listed under the following column headings:

Field	Description
Preference Code	The alphanumeric code for the preference.
Description	A brief meaningful description name for the preference, for example, Marketing by Email.
Opted In	A checkmark in this column indicates that the customer has opted to receive the communication.
Opt Out Allowed?	A checkmark in the column indicates that the customer is allowed to opt out of receiving the communication.

7. 3. 2 Opt In to Communications

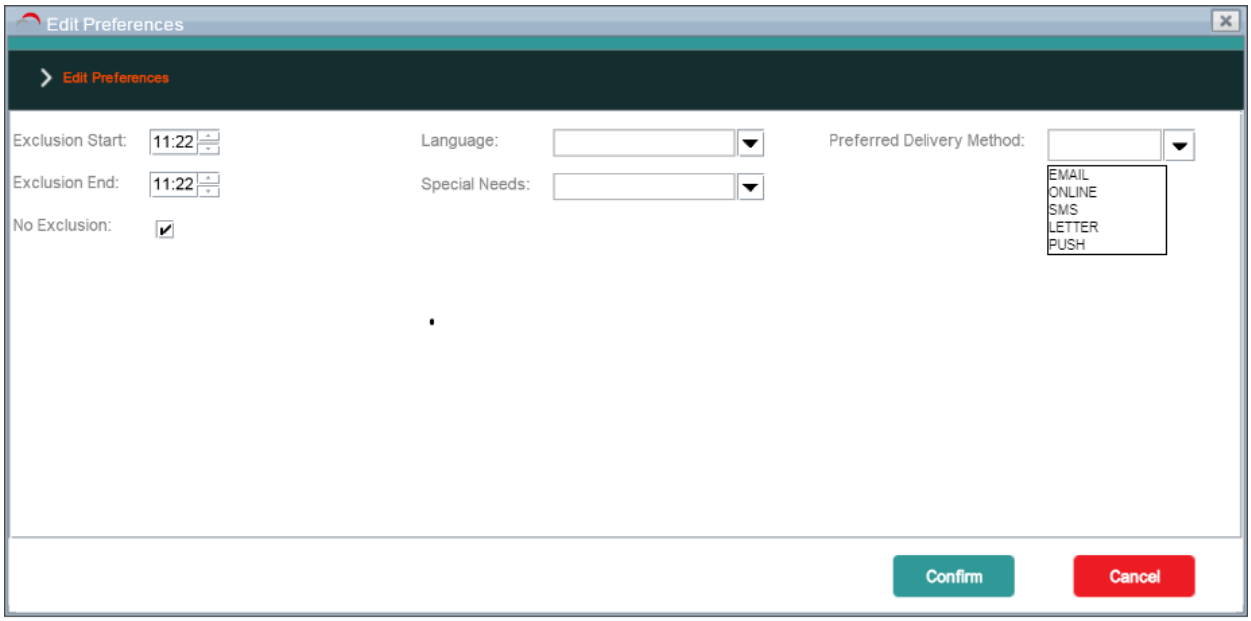
If a customer wants to opt in or out of a communication, the agent can set this up in the **Comms Preferences** pane by right-clicking a preference, choosing the appropriate **Opt In** or **Opt Out** option and confirming their action:



7. 3. 3 Edit Communication Preferences

Customer communication preferences can be edited in the **Edit Preferences** pop-up, which is access via the **Edit Preferences** right-click option in the **Comms Preferences**

panel.

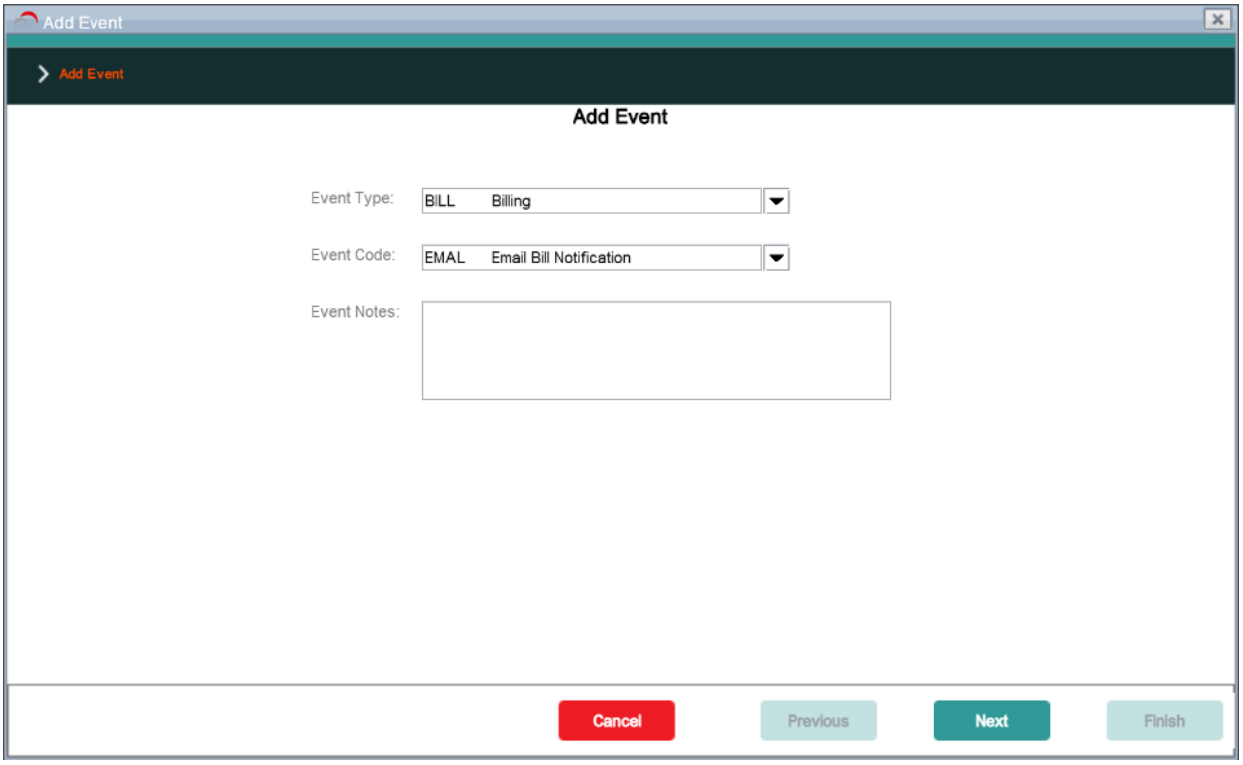


The pop-up contains the following options:

Field	Setting Description
Exclusion Start spin-box	If an exclusion is configured for the account/subscription, the time the exclusion starts.
Exclusion End spin-box	If an exclusion is configured for the account/subscription, the time the exclusion ends.
No Exclusion checkbox	A checkmark in this column indicates that no exclusion period is set.
Language drop-down	The preferred language for communications, for example, English or Spanish.
Special Needs drop-down	Any special needs accommodations required for communications, for example, Braille or Large Print.
Preferred Delivery Method	The preferred delivery method for communications, for example, letter, email, online, SMS, push notification or external system.

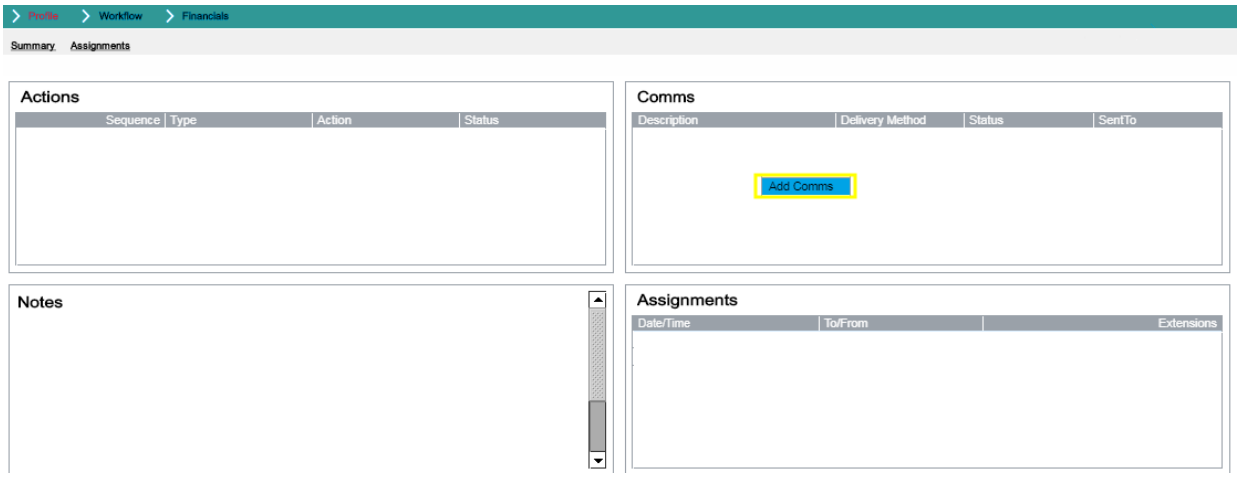
7. 4 Add a One-Off Communication

An agent can raise a communication in AgentView by raising a workflow event that triggers a communication, such as the one in the following screenshot, which sends an email notification that a customer's bill is ready:

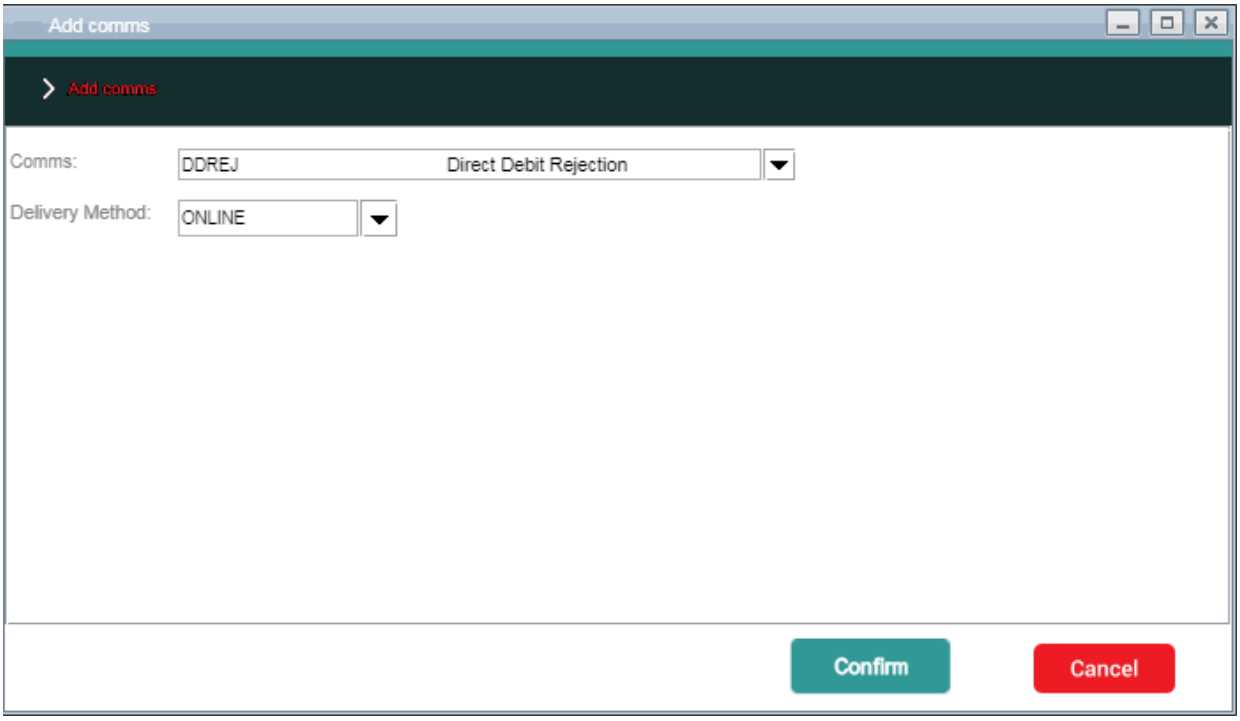


However, a one-off communication can also be added in the Comms lower panel without raising a workflow or can be added from within an unresolved workflow in AgentView, via the following:

- The **Add Comms** shortcut menu option in the **Comms > Comms** lower panel in the relevant **Subscription**, **Agreement** or **Account Summary** screen.
- The **Add Comms** shortcut option in the **Comms** pane in the **Event Summary** screen for that workflow event:



The launches the **Add Comms** pop-up, where the agent can choose a communication and a delivery method for the one-off communication:



See the online help for the AgentView application for more information on sending communications in AgentView.

7. 5 View Reasons for Comms Not Sent and Error Details

7. 5. 1 View Reasons Not Sent

The Reasons Not Sent functionality allows you to view the reasons why a comms was not sent. You can access this functionality for a comms with a request status of Not Sent.

1. Access the **Comms Not Sent Reasons** popup as follows:

   Either

   - Access the **Comms** lower panel for a Subscription , Account or Agreement.

   Or

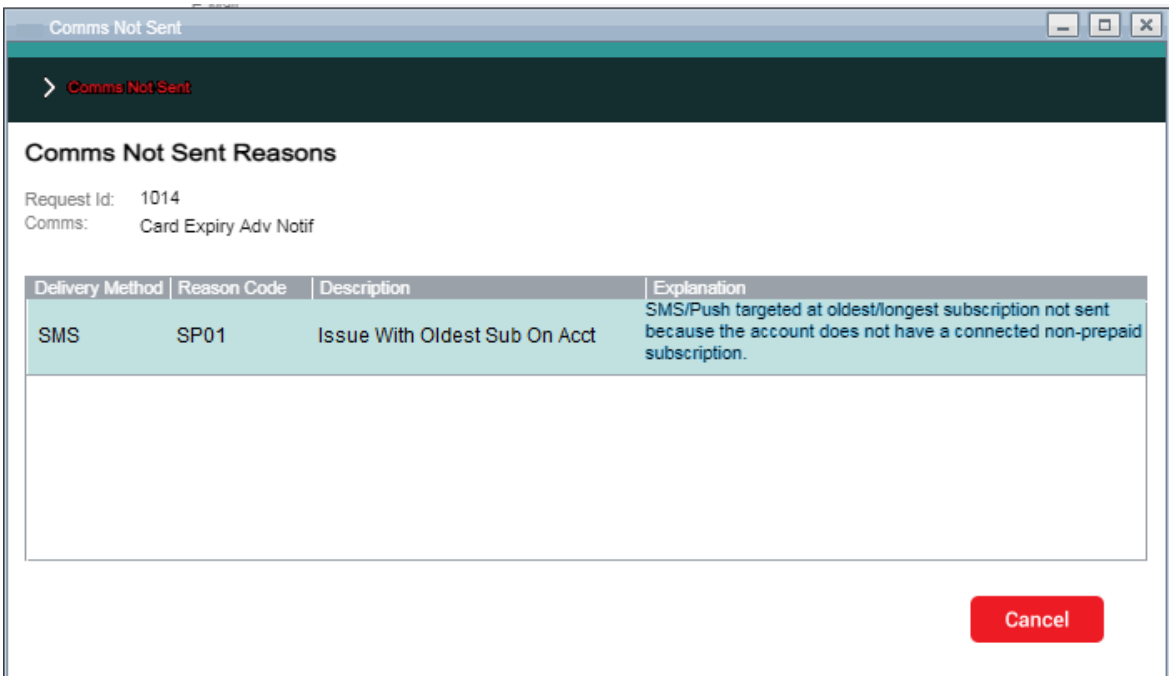
   - Open the **Event Summary Screen** for the workflow event associated with the comms.

     Locate the **Comms** pane in the lower panel.
2. In the list of comms, right-click the comms with the request status of **Not Sent** and select **Reasons Not Sent** in the shortcut menu.
3. The **Reasons Not Sent** popup appears. It displays an explanation why the comms was not sent.

   Explanation of pop-up fields

Field	Description
Request ID	The ID of the request that raised the comms.
Comms Code	The brief descriptive name of the comms, for example, Subscriber creation, Bill notification.
Reason Code	The alphanumeric code for the reason that the comms was not sent.
Description	A brief text description of the reason the comms was not sent.
Explanation	A more detailed explanation of the reason why the comms was not sent.

:



7. 5. 2 View Comms Error Details

The **Error Details** functionality allows you to view the reasons for a comms error. You can access this functionality for a comms with a request status of Error.

1. Access the **Comms Error Details** popup as follows:

   Either

   - Access the **Comms** lower panel for a Subscription , Account or Agreement.

   Or

   - Open the **Event Summary Screen** for the workflow event associated with the comms.

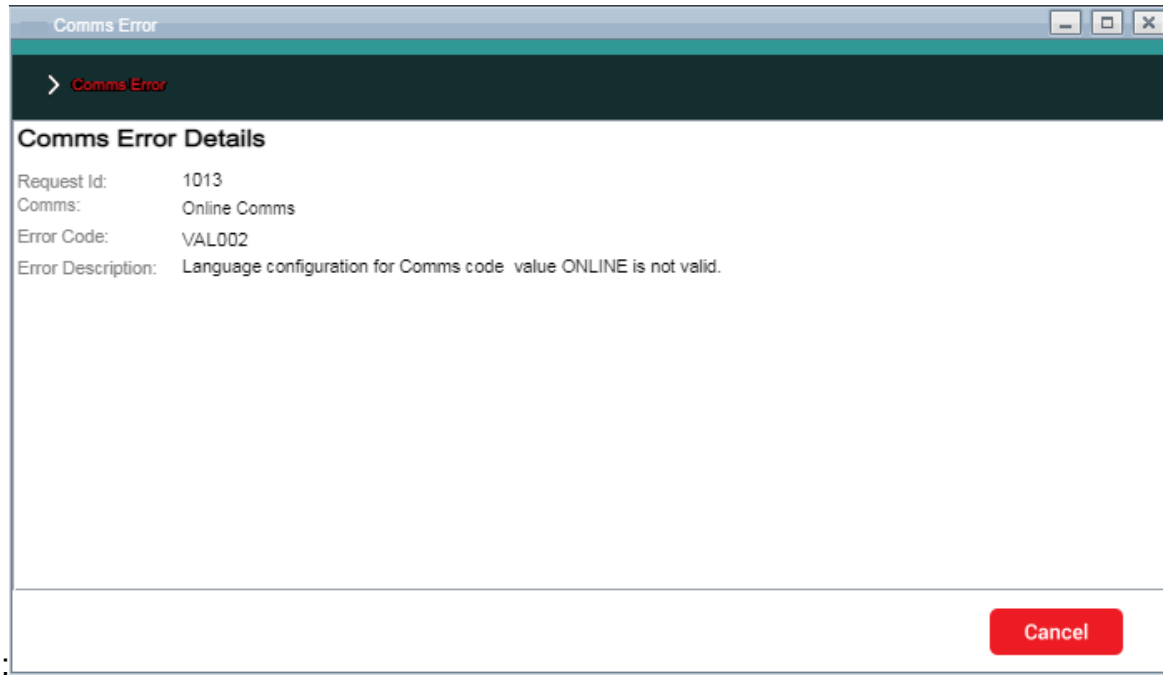
     Locate the **Comms** pane in the lower panel.

2. In the list of comms, right-click the comms with the request status of **Error** and select **Error Details** in the shortcut menu.
3. The **Error Details** pop-up appears. It displays the explanation why the comms was not sent.

Explanation of pop-up fields

Field	Description
Request ID	The ID of the request that raised the comms.
Comms	The brief descriptive name of the comms, for example, Subscriber creation, Bill notification.
Error Code	The alphanumeric code for the comms error.
Error Description	A brief text description of the comms error.

:



8.0 Appendix

8. 1 Appendix A: CMP Default Fields and Beans

The following table lists and describes the available default fields in CMP.

Default Field	Description
ACCOUNTAMOUNTDUE	This returns the balance on the account that is currently due.
ACCOUNTAMOUNTINARREARS	This returns the amount on the account that is in arrears.
ACCOUNTAMOUNTINQUERY	This returns the value of the balance on the account which is in query.
ACCOUNTATTRIBUTE1	This returns the value of account attribute number 1.
ACCOUNTBALANCE	This returns the full balance on the account.
ACCOUNTNUMBER	This returns the account number of the account or of the subscriptions account.
ACCOUNTSERIALNUMBER1	This returns the value of account serial number 1.
ACCOUNTTYPE	This returns the account type of the account.
ACTVTNCODE	The QR code for the eSIM provisioned.
ADDRESSLINE1	This returns address line 1.
ADDRESSLINE2	This returns address line 2.
ADDRESSLINE3	This returns address line 3.
ADDRESSLINE4	This returns address line 4.
ADDRESSLINE5	This returns address line 5.
AGREEMENTCONTRACTEXPIRE	The date on which an agreement's contracts are due to expire.
CARDLAST4DIGITS	This returns the last four digits of the credit card.
CARDEXPIRE	The date on which the card is due to expire.
CARDEXPIRYDATE	This returns the credit card expiry date.
CHANGINGSUBSERIALNUMBER1	This returns the value of subscription serial number 1.
CONTRACTEXPIRYDATE	The date when the contract is due to expire.
CORPORATETOTALAMOUNTDUE	This returns the total amount due at the Corporate level.
CORPORATETOTALBALANCE	This returns the total balance at the Corporate level.
CURRENTPRICEPLAN	This returns the current price plan.
CURRENTPRICEPLANCHARGE	The recurring charge for the current price plan.
DEARSIRMADAM	Static text to appear on letters: *Dear Sir/Madam*.
ENDEDPACKAGECLASSIFICATION	The classification of the package that has been ended.
ENDEDPACKAGEDESCRIPTION	The description of the package that has been ended.
ENDEDPACKAGEPRICE	The purchase price of the package that has been ended.
ENDEDPACKAGEPURCHASETIMESTAMP	The date and time that the package was originally purchased.
EVENTNUMBER	This returns the workflow event number.
FORENAME	This returns the forename.
FUTUREPRICEPLAN	This returns the future price plan.
FUTUREPRICEPLANCHARGE	This returns the recurring charge of the future price plan.
GROUPTOTALAMOUNTDUE	Group Total amount due.

Default Field	Description
GROUPTOTALBALANCE	This returns the total amount due at the Group level.
LATESTINVOICEAMOUNT	The latest amount that has been invoiced.
LATESTINVOICEDUEDATE	The payment due date of the latest invoice.
LATESTPRICEPLAN	This returns the latest price plan.
LATESTPRICEPLANCHARGE	This returns the recurring charge of the latest price plan.
MANUALPAYMENTDUEDATE	The date the manual payment is due.
MOBILELONGESTSUBONACCOUNT	This returns the mobile number of the longest non prepay sub where smsAllowed is True on the network type.
NETWORKSUBCODE1	This returns the value of subscription network sub code 1.
NEXTINVOICEDATE	This returns the date the next invoice is due.
NEXTPAYMENTDATE	This returns the date the next payment was made.
NEXTPAYMENTAMOUNT	This returns the amount due of the next payment.
NONMANSERIALNUMBER1	This returns the value of non-managed serial number 1.
NONMANSERIALNUMBER2	This returns the value of non-managed serial number 2.
OCSABSOLUTETHRESHOLD	This returns the absolute value of the threshold that the notification relates to, for example, 200(MB).
OCSBALANCEID	This returns the identifier of the specific allowance balance that the notification relates to.
OCSCYCLEENDDATE	This returns the end date of the allowance allocation that the notification relates to.
OCSCYCLELENGTH	This returns the length of the refresh interval for the allowance that the notification relates to, for example, 1 Month.
OCSENTITLEMENTNAME	This returns the name of the OCS Entitlement that the notification relates to.
OCSENTITLEMENTSIZE	This returns the overall inclusive allowance amount that the notification relates to.
OCSOFFERNAME	This returns the name of the OCS Offer (allowance name) that the notification relates to.
OCSREMAININGQUOTA	This returns the remaining allowance amount that the notification relates to.
OCSSUBSCRIPTIONID	This returns the identifier of the OCS subscription that the notification relates to.
OCSUSAGEAMOUNT	This returns the amount of allowance consumed that the notification relates to.
OCSVOLUMETHRESHOLDVALUE	This returns the amount of allowance consumed that the notification relates to.
ONEOFFBOLTONPACKAGEEXPIRY	The date on which the one off bolt-on package is due to expire.
ONETIMEPASSWORD	This returns the one time password (OTP).
PACKAGEDESCRIPTION	The description of the one off bolt-on package which is due to expire.
PACKAGEEXPIRYDATE	The date when the one off bolt-on package is due to expire.
PAYCARDLAST4DIGITS	This returns the last four digits of the credit card that made a payment.
PAYMENTACCNUM	This returns the CMP account number associated with the payment.
PAYMENTAMOUNT	This returns the amount of the payment.
PAYMENTDATETIME	This returns the date and time a payment was made.
POSTCODE	This returns the postcode of the address.
POSTCONNECTION	The date on which to send a notification to new subscriptions after connecting.

Default Field	Description
PREPAIDACTIVITYFINAL	The date on which to send a notification to a prepaid subscription alerting them of their disconnection after failing to purchase a bolt-on or top-up package.
PREPAIDACTIVITYWARNING	The date on which to send a communication to a prepaid subscriptions requesting to purchase a bolt-on or top-up package to avoid disconnection.
PREVIOUSPRICEPLAN	This returns the previous price plan.
PREVIOUSPRICEPLANCHARGE	This returns the recurring charge of the previous price plan.
PURCHASEPACKAGECLASS	Type of package purchased.
PURCHASEPACKAGECODE	The CMP code for the purchased package.
PURCHASEPACKAGEDESC	The description for the purchased package.
PURCHASEPACKAGEPRICE	The price of the purchased package.
PURCHASEPACKAGETIME	The timestamp for when the package was purchased.
RECPREPAYMENTPRODUCT	This returns the recurring prepayment product purchase description.
RECPREPAYMENTDATE	This returns the recurring prepayment due date.
RECPREPAYMENTLAST4DIGITS	This returns the last four digits of the recurring prepayment purchase.
RECPREPAYMENTAMOUNT	This returns the recurring prepayment amount.
RECURRINGPREPAYMENTBALANCE	This returns the due date of the prepayment of a recurring bolt-on package paid from the prepaid balance.
RECURRINGPREPAYMENTCARD	This returns the amount and due date of the prepayment of a recurring bolt-on package paid by card.
RESOLUTIONREQUIREDBYDATE	This returns the Resolution Required By Date of a workflow event.
RESOLVEDDATE	This returns the Resolved Date for a workflow event.
SUBALLOWANCEAMOUNT	The initial allowance amount allocated for the period on the OCS.
SUBALLOWANCEAMOUNTREMAINING	The amount of the allowance remaining on the OCS.
SUBALLOWANCEAMOUNTUSED	The amount of the allowance that has been consumed on the OCS.
SUBALLOWANCEDESCRIPTION	The description of the allowance associated with the OCS notification.
SUBALLOWANCETYPE	The type of the allowance associated with the OCS notification.
SUBCONTRACTENDDATE	This returns the date that the contract on the subscription is due to end.
SUBCONTRACTSTARTDATE	This returns the date that the current subscription contract commenced.
SUBPASSWORD	This field may be used to store a PIN depending on CMP implementation but it will not store a password of any sort.
SUBSCONNECTEDDATE	This returns the date that billing commenced.
SUBSCRIPTIONATTRIBUTE1	This returns the value of subscription attribute 001.
SUBSCRIPTIONNUMBER	This returns the subscription number of the subscription.
SUBSCRIPTIONCONTRACTEXPIRE	This returns the date on which a subscription's contracts are set to expire.
SUBSDISCONNECTEDDATE	This returns the date that billing stopped.
SUBSERIALNUMBER1	This returns the value of subscription serial number 1.
SUBSERIALNUMBER2	This returns the value of subscription serial number 2.

Default Field	Description
SUBSERIALNUMBER3	This returns the value of subscription serial number 3.
SUBSERIALNUMBER4	This returns the value of subscription serial number 4.
SUBSERIALNUMBER5	This returns the value of subscription serial number 5.
SUBSTERMINATIONDATE	This returns the date that the termination invoice was produced.
SUBSUSERNAME	This returns the username of the subscription.
SUBUSAGECAPAMOUNT	The initial spend cap amount defined for the period on the OCS.
SUBUSAGECAPAMOUNTREMAINING	The amount of the spend cap that is remaining on the OCS.
SUBUSAGECAPAMOUNTUSED	The amount of the spend cap that has been consumed on the OCS.
SUBUSAGECAPDESCRIPTION	The description of the spend cap associated with the OCS notification.
SURNAME	This returns the surname.
THRESHOLDAMOUNT	The Threshold amount.
THRESHOLDDESCRIPTION	The Threshold Description.
THRESHOLDPRORATA	The Threshold pro-rata text that is sent with the comm.
TITLE	This returns the title.
TODAYSDATE	This returns the current date.
TOTALSALESLEDGERADJUSTMENTS	This returns the Total Sales Ledger Adjustments.
YOURBILL	Static text *Your Bill*.

These are fields that are available to letter and email velocity templates by default and which do not need to be configured for a particular communication.

VelocityVariableName	Description
ADDRESSLINE1	Address Line 1
ADDRESSLINE2	Address Line 2
ADDRESSLINE3	Address Line 3
ADDRESSLINE4	Address Line 4
ADDRESSLINE5	Address Line 5
POSTCODE	Postcode
FIRSTNAME	First name
SHORTDELIVERYADDRESSNAME	If the address is a company address this is the Company Name If the address is a personal address this is the Title + FirstName + Middle Name + Surname

The values in default fields are retrieved by Java beans. The parameters to beans affect how they retrieve data. Beans are generally created over views or tables. This means that with appropriate configuration, any information from tables over which a bean has been created is accessible. In addition, it is also possible to enhance the bean to perform various calculations via methods to perform more complex logic. Fields are denoted by passing the field's attribute to the bean while more complex logic that invokes specific methods on the bean are defined by passing a method attribute to the bean as a parameter.

Beans have been created over the following tables making all fields on these tables accessible via communications:

Bean Name	Table Name
ACCOUNT	Account (view) - Account Extension + Account Master
ACCOUNTBALANCEBEAN	AccountBalanceView - access to the ledger and more
ACCOUNTPROPERTIES	AccountProperties (view) - access to account attributes and serial numbers
CORPORATEBALANCEBEAN	CorporateBalanceView
GROUPBALANCEBEAN	GroupBalanceView
CREDITDEBITCARDBEAN	CCDCCardDetails
PAYMENTBEAN	Payment, CardPayment, AddressPayment and BankPayment tables
PROPOSITIONBEAN	Mainly over the SubscriptionTariff View and various addition methods to calculate future tariffs etc.
SUBSCRIPTIONBEAN	Subscription (view)
SUBSCRIPTIONPROPERTIES	Managed, Non Managed Serial Numbers, Network Sub Codes and Subscription Attributes
WORKFLOW	Diary Event view (Diary Event and DiaryEventExtension)

The following table lists the default field categories and their corresponding bean IDs and bean parameters. This is how default fields have been set up in Business Configuration.

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
ACCOUNTAMOUNTINQUERY	FINANCIAL	field=amountInQuery	ACCOUNTBALANCE	This returns the value of the balance on the account which is in query.
ACCOUNTBALANCE	FINANCIAL	field=accountBalance	ACCOUNTBALANCE	This returns the full balance on the account.
ACCOUNTAMOUNTDUE	FINANCIAL	field=amountDue	ACCOUNTBALANCE	This returns the balance on the account that is currently due.
ACCOUNTAMOUNTINARREARS	FINANCIAL	field=amountInArrears	ACCOUNTBALANCE	This returns the amount on the account that is in arrears.
ACCOUNTTYPE	ACCOUNT	field=accountType	ACCOUNTBEAN	This returns the account type of the account.
ACTVTNCODE	ESIM	field=activationCode	RESOURCEATTRIBUTEBEAN	The QR code for the eSIM provisioned.
ADDRESSLINE1	DELIVERYADDRESS	field=address1	DELIVERYADDRESSBEAN	This returns address line 1.
ADDRESSLINE2	DELIVERYADDRESS	field=address2	DELIVERYADDRESSBEAN	This returns address line 2.

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
ADDRESSLINE3	DELIVERYADDRESS	field=address3	DELIVERYADDRESSBEAN	This returns address line 3.
ADDRESSLINE4	DELIVERYADDRESS	field=address4	DELIVERYADDRESSBEAN	This returns address line 4.
ADDRESSLINE5	DELIVERYADDRESS	field=address5	DELIVERYADDRESSBEAN	This returns address line 5.
CARDLAST4DIGITS	ADVANCE	field=textAttribute1	WORKFLOWEVENTATTRIBUTESBEAN	The last 4 digits of the payment card that is due to expire.
CONTRACTEXPIRYDATE	FINANCIAL	field=dateAttribute1	WORKFLOWEVENTATTRIBUTESBEAN	The date when the contract is due to expire.
CORPORATETOTALBALANCE	FINANCIAL	field=TotalBalance	CORPORATEBALANCE	The TotalBalance is the SUM of the Posting Account Balance and the Corporate Accounts Balance.
CORPORATETOTALAMOUNTDUE	FINANCIAL	field=TotalAmountDue	CORPORATEBALANCE	The TotalAmountDue is the SUM of the Corporate Posting Account Amount Due and the Corporate Accounts

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
				Amount Due.
CURRENTPRICEPLANCHARGE	SUBSCRIPTION	method=getCurrentPricePlanRecurringCharge	PROPOSITIONBEAN	The recurring charge for the current price plan.
ENDEDPACKAGEDESCRIPTION	FINANCIAL	field=packageDescription	ENDEDPACKAGEBEAN	The description of the package that has been ended.
ENDEDPACKAGECLASSIFICATION	FINANCIAL	field=packageClassification	ENDEDPACKAGEBEAN	The classification of the package that has been ended.
ENDEDPACKAGEPRICE	FINANCIAL	field=price	ENDEDPACKAGEBEAN	The purchase price of the package that has been ended.
ENDEDPACKAGEPURCHASETIMESTAMP	FINANCIAL	field=purchaseTimestamp	ENDEDPACKAGEBEAN	The date and time that the package was originally purchased.
EVENTNUMBER	WORKFLOW	field=event	WORKFLOWBEAN	This returns the Event Number.
FORENAME	DELIVERYADDRESS	field=foreName	DELIVERYADDRESSBEAN	This returns the forename.
FUTUREPRICEPLANCHARGE	SUBSCRIPTION	method=getFuturePricePlanRecurringCharge	PROPOSITIONBEAN	This returns the recurring charge of the

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
				future price plan.
GROUPTOTALBALANCE	FINANCIAL	field=TotalBalance	GROUPBALANCE	The TotalBalance is the SUM of the Corporate Posting Account Balance and the Group Accounts Balance.
GROUPTOTALAMOUNTDUE	FINANCIAL	field=TotalAmountDue	GROUPBALANCE	The TotalAmountDue is the SUM of the Group Posting Account Amount Due and the Group Accounts Amount Due.
LATESTPRICEPLANCHARGE	SUBSCRIPTION	method=getLatestPricePlanRecurringCharge	PROPOSITIONBEAN	This returns the recurring charge of the latest price plan.
NEXTINVOICEDATE	ACCOUNT	field=nextInvoiceDate	ACCOUNTBEAN	This returns the next invoice date.
NEXTPAYMENTAMOUNT	ADVANCE	field=monetaryAttribute1	WORKFLOWEVENTATTRIBUTESBEAN	The amount of the next payment that is due.

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
NEXTPAYMENTDATE	ADVANCE	field=dateAttribute1	WORKFLOWEVENTATTRIBUTES-BEAN	The date when the next payment is due.
ONETIMEPASSWORD	WORKFLOWBEAN	field=oneOffPassword	WORKFLOW	This returns the one time password.
POSTCODE	DELIVERYADDRESS	field=postcode	DELIVERYADDRESSBEAN	This returns the postcode of the address
RECPREPAYMENTPRODUCT	ADVANCE	field=textAttribute1	WORKFLOWEVENTATTRIBUTES-BEAN	The description of the purchased product.
SURNAME	DELIVERYADDRESS	field=surname	DELIVERYADDRESSBEAN	This returns the surname.
TITLE	DELIVERYADDRESS	field=title	DELIVERYADDRESSBEAN	This returns the title.
TODAYSDATE	STATIC	Todays date	STATICTEXTBEAN	Static text to appear on letters.
DEARSIRMADAM	SUNDRY	Dear Sir/Madam Note: No need for any prefix such as field= or method= here.	STATICTEXT	This returns the text *Dear Sir/Madam*.
ACCOUNTNUMBER	ACCOUNT	method=accountNumber	ACCOUNT	This returns the account number of the account or of the subscriptions account.
SUBSCRIPTIONNUMBER	SUBSCRIPTION	field=subscriptionNumber	SUBSCRIPTION	This returns

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
				the sub-scription num-ber of the subscription.
SUBSCONNECTEDDATE	SUBSCRIPTION	field=connectedDate	SUBSCRIPTION	This returns the date that billing com-menced.
SUBSDISCONNECTEDDATE	SUBSCRIPTION	field=disconnectedDate	SUBSCRIPTION	This returns the date that billing stopped.
SUBSTERMINATIONDATE	SUBSCRIPTION	field=subsTerminationDate	SUBSCRIPTION	This returns the date that the termination invoice was produced.
SUBPASSWORD	SUBSCRIPTION	field=password	SUBSCRIPTION	This field may be used to store a PIN depending on CMP imple-mentation but it will not store a password of any sort.
SUBCONTRACTSTARTDATE	SUBSCRIPTION	field=contractStartDate	SUBSCRIPTION	This returns the date that the current sub-scription con-

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
				tract commenced.
SUBCONTRACTENDDATE	SUBSCRIPTION	method=contractEndDate	SUBSCRIPTION	The date that the subscriptions contract is due to finish.
SUBSUSERNAME	SUBSCRIPTION	field=subscriptionUserName	SUBSCRIPTION	This returns the user name of the subscription.
MOBILELONGESTSUBONACCOUNT	ACCOUNT	method=mobileLongestNonPrepaySub	ACCOUNT	This returns the mobile number of the longest non prepay sub where smsAllowed is true on the network type.
EMAILLONGESTSUBONCMP	ACCOUNT	field=emailLongestNonPrepaySub	ACCOUNT	This is the email of the longest non prepay sub on CMP.
RECURCARDLAST4DIGITS	ACCOUNT	field=ccDcCardNumber	CREDITDEBITCARDBEAN	This returns the last 4 digits of the credit card that is used for recurring payments.

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
CARDEXPIRYDATE	ACCOUNT	field=ccDcExpiryDateYYMM	CREDITDEBITCARDBEAN	This returns the expiry date of the credit or debit card.
BANKACCOUNTNUM	FINANCIAL	field=bankIdentifier, isRecurring=true	DIRECTDEBITMASTERBEAN	This returns the bank account number associated with the direct debit that is active on the account.
BANKACCOUNTIDENTIFIER	FINANCIAL	field=bankAccountIdentifier, isRecurring=true	DIRECTDEBITMASTERBEAN	This returns the bank sort code associated with the direct debit that is active on the account.
NAMEOFPAYER	FINANCIAL	field=NameOfPayer, isRecurring=true	DIRECTDEBITMASTERBEAN	This returns the name of the payer associated with the direct debit that is active on the account.
SUBSERIALNUMBER1	SUBSCRIPTION	type=SSN, networkStatus=CURRENT, Id=1, field=value	SUBSCRIPTIONPROPERTIES	This returns the value of subscription

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
				serial number 1.
CHANGINGSUBSERIALNUMBER1	SUBSCRIPTION	type=SSN, networkStatus=CHANGING, Id=1, field=value	SUBSCRIPTIONPROPERTIES	This returns the valud of the changing subscirption serial number 1.
SUBSERIALNUMBER2	SUBSCRIPTION	type=SSN, networkStatus=CURRENT, Id=2, field=value	SUBSCRIPTIONPROPERTIES	This returns the value of subscription serial number 2.
SUBSERIALNUMBER3	SUBSCRIPTION	type=SSN, networkStatus=CURRENT, Id=3, field=value	SUBSCRIPTIONPROPERTIES	This returns the value of subscription serial number 3.
SUBSERIALNUMBER4	SUBSCRIPTION	type=SSN, networkStatus=CURRENT, Id=4, field=value	SUBSCRIPTIONPROPERTIES	This returns the value of subscription serial number 4.
SUBSERIALNUMBER5	SUBSCRIPTION	type=SSN, networkStatus=CURRENT, Id=5, field=value	SUBSCRIPTIONPROPERTIES	This returns the value of subscription serial number 5.
NONMANSERIALNUMBER1	SUBSCRIPTION	type=SSN, networkStatus=CURRENT, Id=6, field=value	SUBSCRIPTIONPROPERTIES	This returns

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
				the value of non managed serial number 1.
NONMANSERIALNUMBER2	SUBSCRIPTION	type=SSN, networkStatus=CURRENT, Id=7, field=value	SUBSCRIPTIONPROPERTIES	This returns the value of non managed serial number 2.
NETWORKSUBCODE1	SUBSCRIPTION	type=SNSC, networkStatus=CURRENT, Id=1, field=value	SUBSCRIPTIONPROPERTIES	This returns the value of Subscription network sub code 1.
SUBSCRIPTIONATTRIBUTE1	SUBSCRIPTION	type=SA, Id=001, field=value	SUBSCRIPTIONPROPERTIES	This returns the value of subscription attribute 001.
ACCOUNTSERIALNUMBER1	ACCOUNT	type=ASN,Id=1, networkStatus=CURRENT	ACCOUNTPROPERTIES	This returns the value of Account serial number 1.
ACCOUNTATTRIBUTE1	ACCOUNT	type=ASN,**Id=001**	ACCOUNTPROPERTIES	This returns the value of Account serial number 1.
FUTUREPRICEPLAN	PROPOSITION	method=FuturePricePlan	PROPOSITION	This returns a future price plan only if

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
				there is a pending plan/tariff change.
PREVIOUSPRICEPLAN	PROPOSITION	method=PreviousPricePlan	PROPOSITION	This returns a previous price plan if one exists.
CURRENTPRICEPLAN	PROPOSITION	method=CurrentPricePlan	PROPOSITION	Returns the current price plan.
LATESTPRICEPLAN	PROPOSITION	method=LatestPricePlan	PROPOSITION	This is the most recent record, this may be current or future.
RECURRINGPAYMENTREJECTAMT	FINANCIAL	field=MonetaryAttribute1	DIARYEVENTATTRIBUTEBEAN	
RECURRINGPAUMENTREJECTDAT-E	FINANCIAL	field=DateAttribute1	DIARYEVENTATTRIBUTEBEAN	
PAYMENTACCNUM	FINANCIAL	field=AccountNumber	PAYMENT	This is the CMP account number asso-ciated with the payment.
PAYMENTDATETIME	FINANCIAL	field=paymentDateTime	PAYMENT	This is the date and time a payment was made.

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
PAYMENTAMOUNT	FINANCIAL	field=PaymentAmount	PAYMENT	This is the amount of the payment.
PAYCARDLAST4DIGITS	FINANCIAL	field=CardLast4Digits	PAYMENT	This is the last 4 digits of the credit card that made a payment.
CASHTHRESHOLDVALUE	External System	triggerNotificationAttributeName = Cash.Threshold.Value, dataType=Money,field=triggerNotificationAttributeValue, (where Cash.Threshold.Value is the name of the attribute coming from the external system) dataType=Money - this controls the formatting of the data (e.g. puts €$£ in front of the amount)	EXTERNALTRIGGERATTRIBUTES	Actual Consumption of Cash which trigger the notification.
OCSABSOLUTETHRESHOLD	OCS	triggerNotificationAttributeName=absoluteThreshold	EXTERNALTRIGGERATTRIBUTES	The absolute value of the threshold that the notification relates to e.g. 200 (MB).
OCSBALANCEID	OCS	triggerNotificationAttributeName=balanceId	EXTERNALTRIGGERATTRIBUTES	The identifier of the specific allowance balance that the notification relates to.
OCSCYCLEENDDATE	OCS	triggerNotificationAttributeName=cycleEndDate	EXTERNALTRIGGERATTRIBUTES	The end date of the allow-

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
				ance allocation that the notification relates to.
OCSCYCLELENGTH	OCS	triggerNotificationAttributeName=cycleLength	EXTERNALTRIGGERATTRIBUTES	The length of the refresh interval for the allowance that the notification relates to e.g. 1 Month.
OCSENTITLEMENTNAME	OCS	triggerNotificationAttributeName=entitlementName	EXTERNALTRIGGERATTRIBUTES	Name of the OCS Entitlement that the notification relates to.
OCSREMAININGQUOTA	OCS	triggerNotificationAttributeName=remainingQuota	EXTERNALTRIGGERATTRIBUTES	The remaining allowance amount that the notification relates to.
OCSSUBSCRIPTIONID	OCS	triggerNotificationAttributeName=subscriptionId	EXTERNALTRIGGERATTRIBUTES	The identifier of the OCS subscription that the notification relates to.
OCSUSAGEAMOUNT	OCS	triggerNotificationAttributeName=usageAmount	EXTERNALTRIGGERATTRIBUTES	The amount of allowance consumed that the notification relates to.
OCSVOLUMETHRESHOLDVALUE	OCS	trigger-	EXTERNALTRIGGERATTRIBUTES	The percentage value

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
		Noti-ficationAttributeName=volumeThresholdValue		of the threshold that the notification relates to.
OFFERNAME	OCS	triggernotificationAttributeName=Offer.Name, dataType-e=Text,field=triggerNotificationAttributeValue, (where Offer.Name is the name of the attribute coming from the external system)	EXTERNALTRIGGERATTRIBUTE-S	Name of the Offer that the notification relates to.
PACKAGEDESCRIPTION	FINANCIAL	field=textAttribute1	WORKFLOWEVENTATTRIBUTES-BEAN	The descrip-tion of the one-off bolt-on package which is due to expire.
PACKAGEEXPIRYDATE	FINANCIAL	field=dateAttribute1	WORKFLOWEVENTATTRIBUTES-BEAN	The date when the one-off bolt-on is due to expire.
PREVIOUSPRICEPLANCHARGE	SUBSCRIPTION	method=getPreviousPricePlanRecurringCharge	PROPOSITIONBEAN	This returns the recurring charge of the previous price plan.
PURCHASEPACKAGECLASS	FINANCIAL	field=packageClassification	PURCHASEBEAN	The type of the package pur-chased by the subscriber or agreement administrator.
PURCHASEPACKAGECODE	FINANCIAL	field=packageCode	PURCHASEBEAN	The package code that has

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
				been purchased by the subscriber or agreement administrator.
PURCHASEPACKAGEDESC	FINANCIAL	field=packageDescription	PURCHASEBEAN	The description of the package purchased by the subscriber or agreement administrator.
PURCHASEPACKAGEPRICE	FINANCIAL	field=price	PURCHASEBEAN	The price of the package purchased by the subscriber or agreement administrator.
PURCHASEPACKAGETIME	FINANCIAL	field=purchaseTimestamp	PURCHASEBEAN	The date and time when the package was purchased by the subscriber or agreement administrator.
RECPREPAYMENTAMOUNT	ADVANCE	field=monetaryAttribute1	WORKFLOWEVENTATTRIBUTES-BEAN	The amount of the recurring prepayment.
RECPREPAYMENTDATE	ADVANCE	field=dateAttribute1	WORKFLOWEVENTATTRIBUTES-BEAN	The date that the recurring prepayment is due.
RECPREPAYMENTLAST4DIGITS	ADVANCE	field=textAttribute2	WORKFLOWEVENTATTRIBUTES-BEAN	The last 4 digits of the

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
				card used for the recurring prepayment.
RESOLUTIONREQUIREDBYDATE	FINANCIAL	field=resolutionReqByDate	WORKFLOWBEAN	This returns the Resolution Required by Date.
RESOLVEDDATE	WORKFLOW	field=resolvedDate	WORKFLOWBEAN	Resolved Date.
SUBALLOWANCEAMOUNT	OCS	triggerNotificationAttributeName=entitlementSize	OCSNOTIFICATIONATTRIBUTES	The initial allowance amount allocated for the period on the OCS.
SUBALLOWANCEDESCRIPTION	OCS	triggerNotificationAttributeName=subscriptionId	OCSNOTIFICATIONATTRIBUTES	The description of the allowance associated with the OCS notification.
SUBALLOWANCEAMOUNTREMAINING	OCS	triggerNotificationAttributeName=remainingQuota	OCSNOTIFICATIONATTRIBUTES	The amount of the allowance remaining on the OCS.
SUBALLOWANCEAMOUNTUSED	OCS	triggerNotificationAttributeName=usageAmount	OCSNOTIFICATIONATTRIBUTES	The amount of the allowance that has been consumed on the OCS.
SUBALLOWANCETYPE	OCS	triggerNotificationAttributeName=subscriptionId	OCSNOTIFICATIONATTRIBUTES	The type of the allowance associated with the OCS

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
				notification.
SUBUSAGECAPAMOUNT	OCS	triggerNotificationAttributeName=entitlementSize	OCSNOTIFICATIONATTRIBUTES	The initial spend cap amount defined for the period on the OCS.
SUBUSAGECAPAMOUNTREMAININ-G	OCS	trig-gerNotificationAttributeName=remainingQuota	OCSNOTIFICATIONATTRIBUTES	The amount of the spend cap that is remain-ing on the OCS.
SUBUSAGECAPAMOUNTUSED	OCS	triggerNotificationAttributeName=usageAmount	OCSNOTIFICATIONATTRIBUTES	The amount of the spend cap that has been consumed on the OCS.
SUBUSAGECAPDESCRIPTION	OCS	triggerNotificationAttributeName=subscriptionId	OCSNOTIFICATIONATTRIBUTES	The descrip-tion of the spend cap associated with the OCS notification.
TOTALSALESLEDGERADJUSTMEN-TS	FINANCIAL	field=totalSLAdjustments	WORKFLOWBEAN	The total Sales Ledger Adjust-ments.
THRESHOLDAMOUNT	FINANCIAL	field=thresholdAmount	THRESHOLDBEAN	The Threshold amount.
THRESHOLDDESCRIPTION	FINANCIAL	field=thresholdDescription	THRESHOLDBEAN	The Threshold Description.
THRESHOLDPRORATA	FINANCIAL	field=thresholdProRata	THRESHOLDBEAN	The Threshold pro-rata text that is sent

FieldCode	Field Category Code	Parameter (case sensitive)	BeanID	Explanation
				with the comm.
YOURBILL	STATIC	Your bill	STATICTEXTBEAN	Static text Your bill.
LATESTINVOICEDUEDATE	ADVANCE NOTIFICATION	field=dateAttribute1	WORKFLOWEVENTATTRIBUTES-BEAN	The expected payment pull date.

8. 2 Appendix B: OCS Notification Default Fields

The following table lists the default fields that use the OCSNOTIFICATIONATTRIBUTES bean:

Default Field	Description
SUBALLOWANCEAMOUNT	This returns The initial allowance amount allocated for the period on the OCS.
SUBALLOWANCEAMOUNTREMAINING	This returns the amount of the allowance that is remaining on the OCS.
SUBALLOWANCEAMOUNTUSED	This returns the amount of the allowance that has been consumed on the OCS.
SUBALLOWANCEDESCRIPTION	This returns the description of the allowance associated with the OCS notification.
SUBALLOWANCETYPE	This returns the type of the allowance associated with the OCS notification.
SUBUSAGECAPAMOUNT	This returns the initial usage cap amount defined for the period on the OCS.
SUBUSAGECAPAMOUNTREMAINING	This returns the amount of the usage cap that remains to be consumed on the OCS.
SUBUSAGECAPAMOUNTUSED	This returns the amount of the usage cap that has been consumed on the OCS.
SUBUSAGECAPDESCRIPTION	This returns the description of the usage cap associated with the OCS notification.