Kofax Communications Manager

Batch & Output Management Getting Started Guide

Version: 5.3.0

Date: 2019-05-28

KOFAX

Table of Contents

Preface **5**
- Related documentation 5
- Getting help with Kofax products 6

Chapter 1: Introduction **8**

Chapter 2: Using KCM B&OM **9**
- Lifecycle of objects 9

Chapter 3: Define slot and correspondence types **11**
- Define a slot type 11
- Define a correspondence type 12

Chapter 4: Define contact data and organizational metadata **13**
- Define contact configuration 13
- Define organizational metadata configuration 14

Chapter 5: Define channels **15**
- Define a channel 15
  - Define a channel for archiving 15

Chapter 6: Create application, correspondence, and communication rules **18**
- About application rules 18
  - Configure application rules 18
- About correspondence rules 24
  - Configure correspondence rules 24
- About communication rules 28
  - Configure communication rules 29

Chapter 7: Create requests **37**
- Correspondence request 37
  - Components: correspondence 37
- Application request 39
  - Components: application event 39
- Import request 41
  - Components: import correspondence 41

Chapter 8: About standard processes **43**
- Information transfer between standard processes 44
- Configure global settings for standard processes 45
- Standard processes 46
  - Electronic output 46

Printed output....47

CCM_CleanUpAll standard process....50

Change the standard processes....50

Chapter 9: Configure standard processes....51

Configure processing components in KCM Studio....51

Registration....51

Application....52

Communication....52

ElectronicDelivery....53

PrintDelivery....55

XpsConversion....55

Bundling....55

Stacking....55

Streaming....56

Conversion....56

Distribution....57

GarbageCollection and Timer....57

Chapter 10: Define overlays....59

Appendix A: Glossary....60

Appendix B: Views columns in KCM Studio....62

Preface

This guide explains how to start working with the Batch & Output Management component of Kofax Communications Manager (KCM).

Related documentation

The documentation set for Kofax Communications Manager is available here:[1]

https://docshield.kofax.com/Portal/Products/CCM/530-1h4cs6680a/CCM.htm

In addition to this guide, the documentation set includes the following items:

Kofax Communications Manager Release Notes

Contains late-breaking details and other information that is not available in your other Kofax Communications Manager documentation.

Kofax Communications Manager Installation Guide
Describes the installation procedure.

Kofax Communications Manager Batch & Output Management Developer's Guide
Describes the Batch & Output Management scripting language used in B&OM related scripts.

Kofax Communications Manager Getting Started Guide
Describes how to use Contract Manager to manage instances of Kofax Communications Manager.

Help for Kofax Communications Manager Designer
Contains general information and instructions on using Kofax Communications Manager Designer, which is an authoring tool and content management system for Kofax Communications Manager.

Kofax Communications Manager Repository Administrator's Guide
Describes administrative and management tasks in Kofax Communications Manager Repository and Kofax Communications Manager Designer for Windows.

Kofax Communications Manager Repository User's Guide
Includes user instructions for Kofax Communications Manager Repository and Kofax Communications Manager Designer for Windows.

[1] You must be connected to the Internet to access the full documentation set online. For access without an Internet connection, see "Offline documentation" in the Installation Guide.

Kofax Communications Manager Repository Developer's Guide
Describes various features and APIs to integrate with Kofax Communications Manager Repository and Kofax Communications Manager Designer for Windows.

Kofax Communications Manager Template Scripting Language Developer's Guide
Describes the KCM Template Script used in Master Templates.

Kofax Communications Manager Core Developer's Guide
Provides a general overview and integration information for Kofax Communications Manager Core.

Kofax Communications Manager Core Scripting Language Developer's Guide
Describes the KCM Core Script.

Kofax Communications Manager API Guide
Describes Contract Manager, which is the main entry point to Kofax Communications Manager.

Kofax Communications Manager ComposerUI for HTML5 JavaScript API Web Developer's Guide
Describes integration of ComposerUI for HTML5 into an application, using its JavaScript API.

Kofax Communications Manager DID Developer's Guide
Provides information on the Database Interface Definitions (referred to as DIDs), which is an alternative method retrieve data from a database and send it to Kofax Communications Manager.

Kofax Communications Manager ComposerUI for ASP.NET and J2EE Customization Guide
Describes the customization options for KCM ComposerUI for ASP.NET and J2EE.

Kofax Communications Manager ComposerUI for ASP.NET Developer's Guide
Describes the structure and configuration of KCM ComposerUI for ASP.NET.

Kofax Communications Manager ComposerUI for J2EE Developer's Guide
Describes JSP pages and lists custom tugs defined by KCM ComposerUI for J2EE.

Getting help with Kofax products

The Kofax Knowledge Base repository contains articles that are updated on a regular basis to keep you informed about Kofax products. We encourage you to use the Knowledge Base to obtain answers to your product questions.

To access the Kofax Knowledge Base, go to the Kofax website and select **Support** on the home page.

Note The Kofax Knowledge Base is optimized for use with Google Chrome, Mozilla Firefox or Microsoft Edge.

The Kofax Knowledge Base provides:

- Powerful search capabilities to help you quickly locate the information you need.

  Type your search terms or phrase into the **Search** box, and then click the search icon.
- Product information, configuration details and documentation, including release news.

  Scroll through the Kofax Knowledge Base home page to locate a product family. Then click a product family name to view a list of related articles. Please note that some product families require a valid Kofax Portal login to view related articles.
- Access to the Kofax Customer Portal (for eligible customers).

  Click the **Customer Support** link at the top of the page, and then click **Log in to the Customer Portal**.
- Access to the Kofax Partner Portal (for eligible partners).

  Click the **Partner Support** link at the top of the page, and then click **Log in to the Partner Portal**.
- Access to Kofax support commitments, lifecycle policies, electronic fulfillment details, and self-service tools.

  Scroll to the **General Support** section, click **Support Details**, and then select the appropriate tab.

Chapter 1

Introduction

KCM Batch & Output Management (also known as KCM B&OM) is an independent extension to KCM Interactive, responsible for the Document Pack composition, modification, and distribution to multiple recipients.

Tip To make reading this guide easier, refer to the Glossary that explains the basic KCM B&OM terminology used in this document.

The core functionality of KCM B&OM gives you the ability to:

- Continuously process large volumes of requests
- Continuously (and asynchronously) produce distribution output
- Distribute Document Packs to multiple recipients
- Distribute Document Packs over multiple channels
- Modify Document Packs based on recipient and/or channel
- Combine output for the same recipient in the same envelope
- Convert output to a device-specific format

To access KCM B&OM, start KCM Studio.

When you start KCM Studio for the first time, click each of the following buttons:

- : Opens the upper and lower Navigator panes
- : Opens the Toolbox
- : Opens the Object Inspector
- : Opens the Output pane

For convenience, you can filter the objects listed in the lower Navigator pane to display only the processes:

1. On the toolbar, click Show filter list .
   The Filter Settings dialog box appears.
2. Clear all of the boxes so only **Process** remains selected and click **OK**.

Chapter 2

Using KCM B&OM

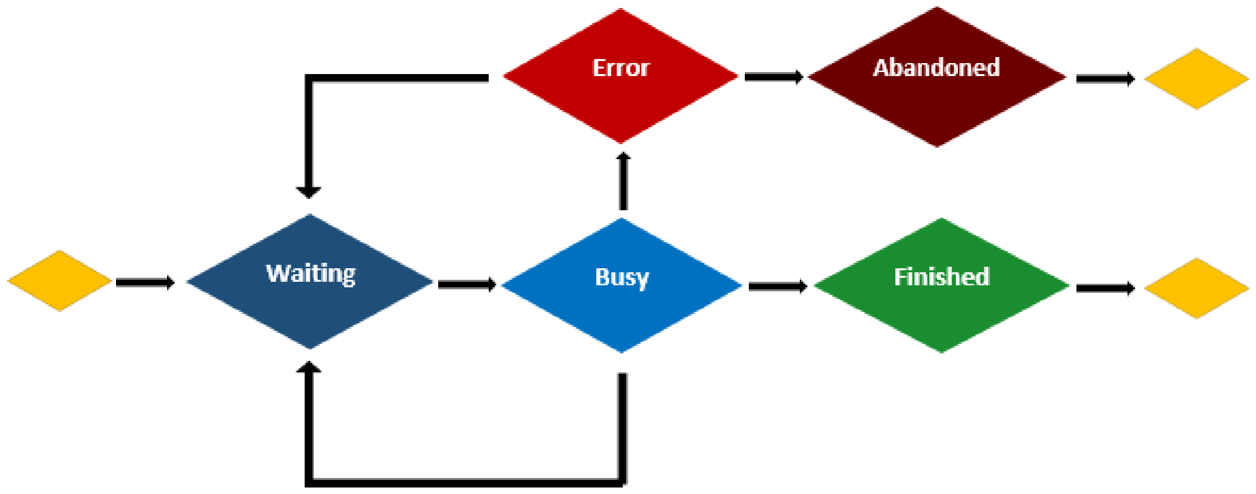
Using KCM B&OM for the first time involves the following steps:

1. Install KCM Interactive.
   See the *Kofax Communications Manager Installation Guide*.
2. Create a Document Pack Template in KCM Designer.
   See the KCM Designer online Help.
3. Define slot and correspondence types to associate with the Document Pack Template.
   See Define slot and correspondence types.
4. Configure contact data and organizational metadata.
   See Define contact data and organizational metadata.
5. Create and apply application, correspondence, or communication rules.
   See Create application, correspondence, and communication rules.
6. Create a request XML and store it in the watch folder.
   See Create requests.
7. Configure the standard processes.
   See About standard processes and Configure standard processes.
8. Start the standard processes step by step from KCM Studio or using Windows Service Host.
9. Retrieve the produced print file from the output folder.

When you continue to use KCM B&OM, you may add new content by creating additional Document Pack Templates, correspondence types, and slot types. Additionally, you may also define channels to create different distribution methods for the output, or to implement archiving (see Define channels).

Lifecycle of objects

Every object processed by KCM B&OM, such as a correspondence, print and electronic communications, process, envelope, or stack, is part of the object lifecycle. It moves from one status to another in the following way.



- When an object is created, it is assigned the **Waiting** status.
- Standard processes pick up an object in the **Waiting** status and immediately move it to the **Busy** status. When the processing is finished, the object is moved to one of the following statuses:
  - **Waiting** if it requires further processing.
  - **Finished** if it does not require further processing.
  - **Error** if an error is encountered during the processing.
- If an object has the **Error** status, you can either resubmit it, which moves it to the **Waiting** status, or abandon it, which moves it to the **Abandoned** status.
- If an object is kept in the **Busy** status for a long time, you can manually move it to the **Error** status.

  Important Only use the "Move to error" function when you are certain that the object is stuck in the Busy state. Moving an unfinished object to the Error status may result in a duplicated output if the object is resubmitted later.

- When objects are either **Finished** or **Abandoned**, you can perform a cleanup with a dedicated standard process.

To performs the resubmit, abandon, or move to error actions on an object, follow these steps:

1. In KCM Studio, on the **Administration** tab, click the button and select a view for the object.
   For a description of the views columns, see Views columns in KCM Studio.
2. In the view, right-click the object and then click the necessary action.

Chapter 3

Define slot and correspondence types

To use a Document Pack Template in B&OM, you need to define slot and correspondence types in the B&OM Repository to represent the Document Pack Template.

- The main purpose of correspondence types is to associate the slots of the Document Pack Template with slot types in B&OM.
- The main purpose of a slot type is to determine the paper type, and to define a category of slots required by the communication rules to customize the processing of these slots.

Note You must define the slot types for the documents of a correspondence type (Document Pack Template) prior to creating the correspondence type itself.

Define a slot type

1. On the toolbar, click **Select System** to select the system to store the correspondence type.
2. In the **Navigator** pane, on the **Folder View** tab, select the folder to store the correspondence type.
3. In the lower Navigator pane, click **New** > **Slot Type**.
4. Type the name for the slot type.
   A new tab appears.
5. In the **Form** section, select the paper type for this slot type. This form is used for printing the pages of the documents that have this slot type.
6. Optionally, you can provide custom properties for the slot type to be used by the correspondence and communication rules. To do so, select the **Communication configuration** tab and then enter or paste an XML file in the editor.
   The XML file must have a root node with name `PluginConfig`.
   Example
   ```
   <PluginConfig>
     <CustomField>CustomValue</CustomField>
     <OptionalField/>
   </PluginConfig>
   ```
   When such an XML file is provided in the "Communication configuration" editor, the following information is inserted in the slots of that slot type, both in the correspondence and communication inputs (see the CcmBomCorrespondenceInput and CcmBomCommunicationInput XSDs, respectively):
   ```
   <ccm:Slot Name="2">
    <ccm:SlotType Name="SlotTypeOther">
     <ccm:Customs>
      <st:CustomField>CustomValue</st:CustomField>
   ```

```
    <st:OptionalField/>
  </ccm:Customs>
 </ccm:SlotType>
</ccm:Slot>
```

The top-level node is automatically renamed to `Customs`, and the namespaces are automatically set to the standard `ccm` (top-level node) and `st` (inner nodes) namespaces.

7. Save the changes.

Define a correspondence type

1. On the toolbar, click **Select System** to select the system to store the correspondence type.
2. In the **Navigator** pane, on the **Folder View** tab, select the folder to store the correspondence type.
3. In the lower Navigator pane, click **New** > **Correspondence Type**.
4. Type the name for the correspondence type.
   This name does not need to correspond to the Document Pack Template name.
   A new tab appears.
5. In the **Document pack template** section, fill in the **name** and **project** fields.
   These values must correspond exactly to the name and project of the Document Pack Template as defined in KCM Designer. The name and project are case-sensitive.
6. Click **Add** to add a document for each slot of the Document Pack Template.
7. For each entry, provide the following information:
   a. The slot reference. Case-sensitive. Type the reference that corresponds exactly to the slot identifier as set in the KCM Designer.
   b. The slot type. From the drop-down list, select a slot type object.
   c. *Optional*. Overlay. From the drop-down list, select an overlay object.

   The order of the added documents does *not* need to correspond to the order of the slots in KCM Designer.
8. Save the changes.

Chapter 4

Define contact data and organizational metadata

Before submitting requests to B&OM, you need to define the information to provide for senders and recipients as well as specify the content for the organizational metadata:

- The contact configuration defines the fields to fill in for a sender and each recipient.
  - If a request XML has too many contact fields, KCM B&OM rejects it.
  - If a request XML has too few contact fields, KCM B&OM adds the missing fields with empty values.
- The organizational metadata configuration defines the keys that must be present in the organizational metadata. Optionally, you can configure default values for certain keys.
  - If a request XML has too many organizational metadata keys, KCM B&OM rejects it.
  - If a request XML has too few organizational metadata keys, KCM B&OM also rejects it, except for organizational metadata keys having default values.

By creating these configurations, you can customize the information required in request XML files.

Note In a request XML file, the values for the contact fields and organizational metadata keys must represent plain text.

Define contact configuration

1. On the toolbar, click **Select System** to choose the system where to store the contact configuration.
2. In the **Navigator** pane, on the **Folder View** tab, select the folder where to store the contact configuration.
3. In the lower **Navigator** pane, click **New** > **CCM XML Configuration**.
4. Type the name for the contact configuration.
   A new tab appears.
5. In the editor, enter or paste the content of an XML file that conforms to the CcmBomContactConfig XSD.
6. In the **Object Inspector**, set the **ConfigurationType** property to **ContactConfiguration**.
7. Save the changes.
8. Now you need to activate the new contact configuration.
   To do so, on the **Administration** tab, click **System Administration**, click the system object, and then set the **Contact configuration** property to the object you created.

Define organizational metadata configuration

1. On the toolbar, click **Select System** to choose the system where to store the organizational metadata configuration.
2. In the **Navigator** pane, on the **Folder View** tab, select the folder where to store the organizational metadata configuration.
3. In the lower **Navigator** pane, click **New** > **CCM XML Configuration**.
4. Type the name for the organizational metadata configuration.
   A new tab appears.
5. In the editor, enter or paste the content of an XML file that conforms to the CcmBomOrganisationalMetadataConfig XSD.
6. In the **Object Inspector**, set the **ConfigurationType** property to **OrganisationalMetadata**.
7. Save the changes.
8. Now you need to activate the new organizational metadata configuration.
   To do so, on the **Administration** tab, click **System Administration**, click the system object, and then set the **Organisational metadata configuration** property to the object you created.

Chapter 5

Define channels

With KCM B&OM, you can create different channels for document distribution and configure which output should be distributed over which channel with the communications rules.

Define a channel

1. On the toolbar, click **Select System** to choose the system where to store the channel.
2. In the **Navigator** pane, on the **Folder View** tab, select the folder where to store the channel.
3. In the lower **Navigator** pane, click **New** > **Channel**.
4. Type the name for the channel.
   A new tab appears.
5. In the **Output behavior** section, select **Separate** for electronic output or **Combined** for print output.
   - Communications for combined channels are distributed by these processes: CCM_PrintDelivery, CCM_XpsConversion, CCM_Bundling, CCM_Stacking, CCM_Streaming, and CCM_Conversion. The documents of the communication processed in this sequence are combined into one result.
   - Communications for separate channels are distributed directly by the CCM_ElectronicDelivery process. The exit point script in this process receives the documents of the communication separately.
6. For combined channels, to enable CCM_Bundling to compose additional cover letters for envelopes distributed over this channel, in the **Cover letter** section, check **Enable creation of cover letter** and then fill in the **Name** and **Project** of the Document Pack Template for this cover letter. The name and project are case-sensitive.

   Note Additionally, you need to configure the composition of cover letters in the bundling component with the `AddChannelCoverLetter` function. For more information, see the section "Bundling component functions" in the *Batch & Output Management Scripting Developer's Guide*.

7. Save the changes.

Define a channel for archiving

Archiving in KCM B&OM is carried out by means of channels. In order to implement archiving, perform the following steps:

1. Create one or multiple dedicated channels for archiving (for instructions, see the previous section).

2. In the **Output behavior** section, make the following selection:
   - If you need an electronic archive channel, select **Separate**. In this case, all documents in the communication will be archived separately.
   - If you need a print archive channel, select **Combined**. In this case, the entire communication will be archived.
3. Create correspondence rules to determine which copies of the communication need to be sent for archiving.

   In the correspondence rules, you need to add a `CreateCommunication` clause for every communication that needs to be archived, with its own `Label`.

   For more information on creating the rules, see About correspondence rules.
4. Create communication rules for the archived communications.

   For more information on creating the rules, see About communication rules.

   Example 1 (no archiving)

```
<CorrespondenceRule Reference="Customer">
  <MatchAllCorrespondences/>
  <CreateCommunication Label="Customer"/>
</CorrespondenceRule>

<CommunicationRule Reference="RecipientCustomer"/>
  <MatchCommunication WhenXPath="@Label = 'Customer'"/>
  <SelectRecipientType><Text Value="Customer"/></SelectRecipientType>
</CommunicationRule>

<CommunicationRule Reference="ContentCustomer"/>
  <MatchCommunication WhenXPath="@Label = 'Customer'"/>
  <SelectSlots WhenXPath="1=1"/>
</CommunicationRule>

<CommunicationRule Reference="ChannelCustomer"/>
  <MatchCommunication WhenXPath="@Label = 'Customer'"/>
  <SetChannel><Text Value="Print"/></SetChannel>
</CommunicationRule>
```

   Example 2 (archiving added)

```
<CorrespondenceRule Reference="Customer">
  <MatchAllCorrespondences/>
  <CreateCommunication Label="Customer"/>
  <CreateCommunication Label="CustomerArchive"/>
</CorrespondenceRule>

<CommunicationRule Reference="RecipientCustomer"/>
  <MatchCommunication WhenXPath="@Label = 'Customer' or @Label =
 'CustomerArchive'"/>
  <SelectRecipientType><Text Value="Customer"/></SelectRecipientType>
</CommunicationRule>

<CommunicationRule Reference="ContentCustomer"/>
  <MatchCommunication WhenXPath="@Label = 'Customer' or @Label =
 'CustomerArchive'"/>
  <SelectSlots WhenXPath="1=1"/>
</CommunicationRule>

<CommunicationRule Reference="ChannelCustomer"/>
  <MatchCommunication WhenXPath="@Label = 'Customer'"/>
  <SetChannel><Text Value="Print"/></SetChannel>
</CommunicationRule>
```

```
<CommunicationRule Reference="ContentArchive"/>
  <MatchCommunication WhenXPath="contains(@Label,'Archive')"/>
  <DeselectSlots WhenXPath="@Name = 'Terms and Conditions'"/>
</CommunicationRule>

<CommunicationRule Reference="ChannelArchive"/>
  <MatchCommunication WhenXPath="contains(@Label,'Archive')"/>
  <SetChannel><Text Value="Archive"/></SetChannel>
</CommunicationRule>
```

In the second example, the following modifications are made for the archived communications:

- The `CustomerArchive` communication is created next to the communication `Customer`.
- `SelectRecipientType` is set to select the same recipient for `CustomerArchive` as for `Customer`.
- `SelectSlots` is set to select the same recipient for `CustomerArchive` as for `Customer`.
- `DeselectSlots` is set to remove `Terms and Conditions` from the archived communications.
- `SetChannel` is set for the archived communications.

5. Using the configuration functions, configure the bundling, stacking, streaming, and conversion components to determine the properties of the archived communications.
6. Depending on the archive type, proceed with the following steps:
   - If the archive channel is an electronic channel, you need to modify the Electronic Delivery exit point on KCM Core. The exit point script resides here: `[drive]\KCM\Work\<version>\Instance_<number>\core\Scripts\User Library`.
   - If the archive channel is a print channel, you need to create a new, separate CCM_Conversion process to ensure that the print files for archiving get into a separate folder.

     For example, create CCM_ConversionArchiving.

     To do so, you need to create a new process in KCM Studio and then copy the content of the standard CCM_Conversion process script, modifying it for archiving:
     - For the new CCM_Conversion process, you need to manually create and configure a dedicated watch folder to process the files for archiving, and a separate output folder.
     - In the new script, you need to set the `SelectByChannelName` function to the name of the new archive channel.
7. Save the changes.

Chapter 6

Create application, correspondence, and communication rules

By creating application, correspondence, and communication rules, you define the communications that will be distributed for a particular correspondence and provide the necessary information about those communications.

About application rules

Application rules determine which correspondences, if any, are created for a particular application event and provide important details. Application rules determine the Document Pack Template to compose for an application event. Optionally, they can modify the Data Backbone, the optional documents, and any other data, such as sender, recipients, and organizational metadata for the created correspondences.

For example, application rules can be used in the following situations:

- If a business application sends out many events, and only some of these events should result in documents being sent to a customer. In this case, the application rules can determine which events require correspondence to be created.
- If recipient data is necessary for the composition, but the business application is not able to modify the Data Backbone accordingly. In this case, the application rules can copy recipient data from the application event into the Data Backbone.

Note Application rules are only used for processing application requests. They are optional for processing correspondence or import requests.

Configure application rules

To configure application rules, execute the following steps:

1. In KCM Studio, create an object of type **CCM XML Configuration**.
   Right-click in the lower Navigator pane and then click **New** > **CCM XML Configuration**.
2. In the **Object Inspector** pane, set the **ConfigurationType** property to **ApplicationStep**.
3. Edit the XML object in the central pane.
4. Define the application rules by means of XML elements as described in the next two sections.
5. Save the changes.

6. Activate the application rules.

   To do so, on the **Administration** tab, click **System Administration**, and then set the **ApplicationStep** configuration property to the object you created.

   Note Only one rule set can be active at a time.

Apply application rules

The application rules are applied by the CCM_Registration standard process as the first step of the application request processing. For correspondence and import requests, the application rules are not put into use.

The application rules are used one by one on the application events for the request. If a rule matches, it can either create a correspondence or explicitly ignore the event, preventing further rules from being applied. Multiple rules can match on a single event, so you can create multiple correspondences for the same event.

The application rules are applied to an XML serialization of the application events that are being processed by the CCM_Registration standard process. This XML serialization is defined by the `ApplicationEvent` complex type in the CcmBomRequest XSD. You can find the B&OM XSD files in: `<deploy root>\KCM\Documentation\5.3.0\Resources\Schemas\Output Management`.

Example

```
<ccm:ApplicationEvent Reference="AEV_4894" Brand="BR22"
                                          Application="MyApp1" Process="P1"
 Event="EV_4894_19"
                                          xmlns:ccm="http://www.kofax.com/ccm"
                                          xmlns:dbb="http://www.kofax.com/ccm/dbb"
                                          xmlns:ct="http://www.kofax.com/ccm/ct"
                                          xmlns:data="http://www.kofax.com/ccm/data">
    <ccm:Recipients>
      <ccm:Recipient Type="Customer">
        <ct:Name>M.L. Surname</ct:Name>
        <ct:Street>Sunny Avenue 12</ct:Street>
        <ct:Zipcode/>
        <ct:Country>HolidayResolt</ct:Country>
      </ccm:Recipient>
      <ccm:Recipient Type="Intermediary">
        <ct:Name>H.T. Work</ct:Name>
        <ct:Street>Busy Lane 52</ct:Street>
        <ct:Zipcode>ABCD 78</ct:Zipcode>
        <ct:Country>OfficeLand</ct:Country>
      </ccm:Recipient>
    </ccm:Recipients>
    <ccm:OrganisationalMetadata>
        <data:Key1>value</data:Key1>
    </ccm:OrganisationalMetadata>
    <ccm:DataBackbone>
      <dbb:Data_Backbone>
         <dbb:Part>17</dbb:Part>
         <dbb:System>S45</dbb:System>
      </dbb:Data_Backbone>
    </ccm:DataBackbone>
    <ccm:OptionalDocuments>
        <ccm:OptionalDocument>Terms&amp;Conditions</ccm:OptionalDocument>
    </ccm:OptionalDocuments>
```

```
  </ccm:ApplicationEvent>
```

The following information of the application event is available to the application rules:

- Properties of the application event: reference, application, process, event, and brand, if any
- Contact data of the sender, if any
- Type and contact data of the allowed recipients
- Organisational metadata
- Data Backbone
- Optional documents
- Import documents, if any (not shown in the preceding example)

Syntax for application rules

Each application rule consists of a "match condition" and either a "create correspondence" clause with a number of "rule actions," or an empty "ignore event" clause.

- The "match condition" must be `MatchEvent`.
- The "rule actions" can either be `SetDocumentPackTemplate`, `ModifyOrganisationalMetadata`, `ModifyDatabackbone`, `AddRecipient`, `ModifyRecipientData`, `AddSender`, `ModifySenderData`, or `AddDocument`.

When a rule is applied on an application event, it leads to one of the following results:

1. When the match condition is not satisfied, the rule skips the event.
2. When the match condition is satisfied, and the rule has a "create correspondence" clause, the rule actions are applied to a copy of the event, and the result is then transformed to an output correspondence. The original event is not changed.
3. When the match condition is satisfied, and the rule has an "ignore event" clause, the application event is ignored, and no further rules are applied.

The following situations result in an error:

- When no application rules in the input match on an application event. However, to have application events with no output, you can use rules that explicitly match on the event but then ignore it.
- When an "ignore event" is carried out on an application event for which a previous rule already created output.

The rules may contain XPath expressions that are evaluated on the input application event. XML context is selected based on the application correspondence `(root)node` as defined in the CcmBomApplicationCorrespondence XSD. The expressions are evaluated according to the XPath 1.0 standard and must use the following fixed name spaces to navigate through the application event.

Namespace	Prefix	Nodes
http://www.kofax.com	`ccm`	All nodes, except the next two nodes
http://www.kofax.com/ccm/data	`data`	Keys of `OrganisationalMetadata`
http://www.kofax.com/ccm/contact	`ct`	Contact data of `Senders` and `Recipients`

For example, to navigate to the `Building` value in the organizational metadata of the application event, use `ccm:OrganisationalMetadata/data:Building`.

The following elements are part of the syntax:

- **MatchEvent** (match condition)

  ```
  <MatchEvent WhenXPath="…"/>
  ```

  The `MatchEvent` condition evaluates the XPath expression on the input application event. If the result is true, the condition succeeds. Otherwise, the condition fails.

  Attribute:

  `WhenXPath`. XPath expression to be evaluated on the input application event as a whole.

  The `MatchEvent` condition throws an error when the result of the XPath expression is not a boolean value.

- **SetDocumentPackTemplate** (rule action)

  ```
  <SetDocumentPackTemplate>
  <Project>…</Project>
  <Name>…</Name>
  </SetDocumentPackTemplate>
  ```

  The `SetDocumentPackTemplate` action evaluates the expressions contained in the `Project` and `Name` subelements to strings. Then, it sets the Document Pack Template of the application event to the computed project and name. According to the CcmBomApplicationRules XSD, the `SetDocumentPackTemplate` action is called exactly once for each correspondence created, as the first rule action.

  The rule action has no attributes, but must instead have two XML subelements that contain expressions. The expressions must compute the project and name of the Document Pack Template. `SetDocumentPackTemplate` throws an error when the expressions cannot be evaluated to strings, or when no correspondence type is defined in theB&OM Repository for the computed Document Pack Template.

- **ModifyOrganisationalMetadata** (rule action)

  ```
  <ModifyOrganisationalMetadata
  Key="…">…</ModifyOrganisationalMetadata>
  ```

  The `ModifyOrganisationalMetadata` action evaluates the subexpression to a string, and then sets the given key in the organizational metadata of the application event to the computed value.

  Attribute:

  `Key`. String. The key to set in the organizational metadata of the application event.

  The action also has an XML subelement of type expression. This expression must compute the value to set in the organizational metadata.

  `ModifyOrganisationalMetadata` throws an error when the expression cannot be evaluated to a string, or when the given key does not appear in the organizational metadata.

- **ModifyDataBackbone** (rule action)

  ```
  <ModifyDataBackbone
  ```

```
Location="…">…</ModifyDataBackbone>
```

The `ModifyDataBackbone` action evaluates the subexpression to a string, and then sets the given location in the Data Backbone to the computed value.

Attribute:

`Location`. XPath expression. The location to set in the Data Backbone of the application event. The XPath expression must evaluate to a node in the Data Backbone, and is evaluated on the `ccm:DataBackbone` subnode of the application event.

The action also has an XML subelement of type expression. This expression must compute the value to set in the Data Backbone.

`ModifyDataBackbone` throws an error when the subexpression cannot be evaluated to a string, or when the location cannot be evaluated to a subnode of the Data Backbone.

- **AddRecipient** (rule action)

  ```
  <AddRecipient Type="…"/>
  ```

  The `AddRecipient` action adds a recipient with the given type to the application event. The added recipient has empty values for all configured contact data. You can modify these values by calling `ModifyRecipientData`.

  Attribute:

  `Type`. String. The type of the added recipient. The application event may not already have another recipient with the same type.

  `AddRecipient` throws an error when the given recipient type already exists in the application event.

- **ModifyRecipientData** (rule action)

  ```
  <ModifyRecipientData Type="…"
  Field="…">…</ModifyRecipientData>
  ```

  `ModifyRecipientData` evaluates the subexpression to a string, and then adds or modifies the given field in the contact data of the recipient of the given type to the computed value.

  Attributes:

  `Type`. String. The recipient type to modify. A recipient of this type must exist in the application event.

  `Field`. String. The field in the contact data of the recipient to add or modify. The field must be defined in the contact configuration.

  `ModifyRecipientData` throws an error when the application event does not contain a recipient of the given type, or when the contact configuration does not contain the given field, or when the subexpression cannot be evaluated to a string.

- **AddSender** (rule action)

  ```
  <AddSender/>
  ```

  The `AddSender` action adds a sender to the application event. The added sender has empty values for all configured contact data. You can modify these values by calling `ModifySenderData`.

  `AddSender` throws an error when the application event already has a sender.

- **ModifySenderData** (rule action)

  ```
  <ModifySenderData
  ```

```
Field="…">…</ModifyRecipientData>
```

`ModifySenderData` evaluates the subexpression to a string, and then adds or modifies the given field in the contact data of the sender to the computed value.

Attribute:

`Field`. String. The field in the contact data of the sender to add or modify. The field must be defined in the contact configuration.

`ModifySenderData` throws an error when the application event does not contain a sender, or when the contact configuration does not contain the given field, or when the subexpression cannot be evaluated to a string.

- **AddDocument** (rule action)

  ```
  <AddDocument Name="…" WhenXPath="…"/>
  ```

  The `AddDocument` action evaluates the given XPath condition to a boolean, and when the result is true, adds the optional document with the given name to the application event.

  Attribute:

  `Name`. String. The slot identifier of the optional document to add. When composing the Document Pack Template for the created correspondence later, this optional slot is no longer removed from the result, if it existed.

  Attribute:

  `WhenXPath`. XPath expression to be evaluated on the input application event.

  `AddDocument` throws an error when the XPath expression cannot be evaluated to a boolean.

Expressions

Expressions are used in rules to denote either a static value that does not depend on the input, or a dynamic value that is computed out of the input. A dynamic value is based on concatenations of XPath expressions and constant values.

An XML node of type expression can be one of the following:

1. `<Text Value="…"/>` Denotes a constant value.
2. `<Eval XPath="…"/>` Denotes a dynamic value computed by an XPath expression.
3. `<Concat>…</Concat>` Denotes a concatenation, and must contain two or more subnodes of type expression.

Examples of expressions:

1.
   ```
   <Text Value="A"/>
   ```
2.
   ```
   <Eval XPath="ccm:OrganisationalMetadata/data:Field"/>
   ```
3.
   ```
   <Concat>
       <Eval XPath="ccm:OrganisationalMetadata/data:Field1"/>
       <Text Value=";"/>
       <Eval XPath="ccm:OrganisationalMetadata/data:Field2"/>
   </Concat>
   ```

Example

```
<ApplicationRuleSet xmlns="http://www.kofax.com/ccm">
  <ApplicationRule Reference="A1">
    <MatchEvent WhenXPath="(@Application = 'Legacy')"/>
```

```
    <IgnoreEvent/>
  </ApplicationRule>

  <ApplicationRule Reference="A2">
    <MatchEvent WhenXPath="(@Event = 'E12')"/>
    <CreateCorrespondence>
      <SetDocumentPackTemplate>
         <Project><Text Value="All"/></Project>
         <Name><Text Value="Prolongation Letter"/></Name>
      </SetDocumentPackTemplate>
      <ModifyDataBackbone Location="dbb:Data_Backbone/dbb:Main/dbb:Sender">
        <Eval XPath="ccm:Sender/ct:Name"/>
      </ModifyDataBackbone>
      <AddDocument Name="Terms and Conditions" WhenXPath="count(ccm:Recipients/
ccm:Recipient[@Type='Customer']) > 0"/>
    </CreateCorrespondence>
  </ApplicationRule>

  <ApplicationRule Reference="A3">
    <MatchEvent WhenXPath="(@Event = 'E77')"/>
    <CreateCorrespondence>
      <SetDocumentPackTemplate>
         <Project><Text Value="All"/></Project>
         <Name><Text Value="Bill"/></Name>
      </SetDocumentPackTemplate>
      <AddRecipient Type="Bank"/>
      <ModifyRecipientData Type="Bank" Field="Address"><Text Value="BankAddress"/></
ModifyRecipientData>
    </CreateCorrespondence>
  </ApplicationRule>
</ApplicationRuleSet>
```

About correspondence rules

Correspondence rules determine which kinds of communications should be distributed for a particular correspondence.

For example, correspondence rules can be used in the following situations:

- If the original communication for a correspondence is sent to a customer, and the customer has an intermediary. In this case, a copy of the communication can also be sent to the intermediary.
- If the original communication for a correspondence is sent to a customer through a print channel, and the customer has a portal address. In this case, a copy of the communication can also be placed on the portal.

Configure correspondence rules

To configure correspondence rules, execute the following steps:

1. In KCM Studio, create an object of type **CCM XML Configuration**.
   Right-click in the lower Navigator pane and then click **New** > **CCM XML Configuration**.
2. In the **Object Inspector** pane, set the **ConfigurationType** property to **CorrespondenceConfiguration**.
3. Edit the XML object in the central pane. Define the correspondence rules by means of XML elements as described in the next two sections.

4. Save the changes.
5. Activate the correspondence rules.
   To do so, on the **Administration** tab, click **System Administration**, and then set the **Correspondence rule configuration** property to the object you created.

   Note Only one rule set can be active at a time.

Apply correspondence rules

The correspondence rules are applied by the CCM_Communication standard process, after the Document Pack of a correspondence is composed. The rules are applied one by one on the input correspondence, in the order defined in the rule set.

Whenever a correspondence rule creates a new communication, the application of the correspondence rules is interrupted to apply all communication rules to the new communication. Once the communication rules are applied, a summary of the final version of the communication is attached to the input correspondence, and the application of correspondence rules resumes.

The correspondence rules are applied to an XML serialization of the input correspondence that is being processed by the CCM_Communication standard process. The XML serialization of the input correspondence is defined by the CcmBomCorrespondenceInput XSD. You can find the B&OM XSD files here: `<deploy root>\KCM\Documentation\5.3.0\Resources\Schemas\Output Management`

Example

```
<ccm:Correspondence xmlns:ccm=”http://www.kofax.com/ccm”
                    xmlns:data="http://www.kofax.com/ccm/data"
                    xmlns:ct=”http://www.kofax.com/ccm/contact”
                    xmlns:st="http://www.kofax.com/ccm/slottype">
  <ccm:Sender>
    <ct:Id>PQR</ct:Id>
    <ct:Name>John</ct:Name>
  </ccm:Sender>
  <ccm:Recipients>
    <ccm:Recipient Type="Customer">
      <ct:Id>XYZ</ct:Id>
      <ct:Name>Bob</ct:Name>
    </ccm:Recipient>
    <ccm:Recipient Type="Intermediairy">
      <ct:Id>ABC</ct:Id>
      <ct:Name>Alice</ct:Name>
    </ccm:Recipient>
  </ccm:Recipients>
  <ccm:OrganisationalMetadata>
    <data:Key1>value1</data:Key1>
    <data:Key2>value2</data:Key2>
  </ccm:OrganisationalMetadata>
  <ccm:Slots>
    <ccm:Slot Name="Cover">
      <ccm:SlotType Name="MySlotType">
        <ccm:Customs>
          <st:Field/>
        </ccm:Customs>
      </ccm:SlotType>
    </ccm:Slot>
    <ccm:Slot Name="Document"
                          Overlay="MyOverlay">
      <ccm:SlotType Name="MySlotType">
        <ccm:Customs>
          <st:Test>Test</st:Test>
        </ccm:Customs>
      </ccm:SlotType>
    </ccm:Slot>
  </ccm:Slots>
  <ccm:Communications>
    <ccm:Communication Label="Original" RecipientType="Customer" Channel="Email"/>
    <ccm:Communication Label="Copy” RecipientType="Intermediairy" Channel="Print"/>
  </ccm:Communications>
</ccm:Correspondence>
```

The following information for the correspondence is available to the correspondence rules:

- Contact data of the sender, if any.
- Type and contact data of the allowed recipients.
- Organisational metadata.
- Composed slots. The following information is available:
  - Slot type
  - Slot identifier
  - Overlay to be applied, if any
- Summary of the communications created for this correspondence at this point. For each created communication, the following information is available:
  - Communication label
  - Recipient type
  - Channel

Note Objects defined in the B&OM Repository are always referenced by their object names. In the input correspondence, this affects the slot type, the overlay of a slot, and the channel of a created communication.

Syntax for correspondence rules

Each correspondence rule consists of a "match condition" and one or more "rule actions." If the "match condition" is satisfied, the "rule actions" are applied, one by one. If not, the rules have no effect.

- The "match condition" can either be `MatchCorrespondence` or `MatchAllCorrespondences`.
- The "rule actions" can either be `CreateCommunication`, `RemoveCommunication`, or `RaiseError`.

The rules may contain XPath expressions that are evaluated on the input correspondence. XML context is selected based on the correspondence `(root)node` as defined in the CcmBomCorrespondenceInput XSD. The expressions are evaluated according to the XPath 1.0 standard and must use the following fixed name spaces to navigate through the correspondence.

Namespace	Prefix	Nodes
http://www.kofax.com/ccm	`ccm`	All nodes, except the next three nodes
http://www.kofax.com/ccm/data	`data`	Keys of `OrganisationalMetadata`

Namespace	Prefix	Nodes
http://www.kofax.com/ccm/contact	`ct`	Contact data of `Senders` and `Recipients`
http://www.kofax.com/ccm/slottype	`st`	Customization of `SlotType`

For example, to navigate to the `Name` field of the sender of the input correspondence, use `ccm:Sender/ct:Name`.

The following elements are part of the syntax:

- **MatchCorrespondence** (match condition)

  ```
  <MatchCorrespondence WhenXPath="…"/>
  ```

  The `MatchCorrespondence` condition evaluates the XPath expression on the input correspondence. If the result is true, the condition succeeds. Otherwise, the condition fails.

  Attribute:

  `WhenXPath`. XPath expression to be evaluated on the input correspondence as a whole.

  The `MatchCorrespondence` condition throws an error when the result of the XPath expression is not a boolean value.

- **MatchAllCorrespondences** (match condition)

  ```
  <MatchAllCorrespondences/>
  ```

  The `MatchAllCorrespondences` condition is a condition that always succeeds. You can use it to specify correspondence rules that must be applied unconditionally.

  The condition has no attributes.

- **CreateCommunication** (rule action)

  ```
   <CreateCommunication Label="…"WhenEmpty="…"/>
  ```

  The `CreateCommunication` action initiates the creation of a new communication. The communication is started as an empty copy of the input correspondence, which does not yet have any content, recipient type, or channel. These properties are filled in by communication rules.

  Attributes:

  - `Label`. String. Expresses the communication purpose. It is assigned to the empty correspondence, and is available for use in the communication rules. `Label` must be unique within all the communications generated for this correspondence.
  - `WhenEmpty`. *Optional*. Values are either `RaiseError` or `AutoRemove`. Default is `RaiseError`. Determines the behavior when the final communication (after the application of communication rules) does not contain any of the slots from the correspondence. When set to `RaiseError`, an empty communication is treated as an error. When set to `AutoRemove`, an empty communication is removed silently, without triggering an error.

  The `CreateCommunication` action throws an error when `Label` is not unique, when an empty communication is created and `WhenEmpty` is not set to `AutoRemove`, or when the created communication is not empty, but no recipient type or channel is selected.

- **RemoveCommunication** (rule action)

  ```
  <RemoveCommunicationLabel="…"/>
  ```

  The `RemoveCommunication` action removes the communication with the given label.

  Attribute:

  `Label`. String. The label to assign.

  The `RemoveCommunication` action throws an error when no communication with the given label exists.

- **RaiseError** (rule action)

  ```
  <RaiseError Message="…"/>
  ```

  The `RaiseError` action forcibly puts the input correspondence in error with the given error message.

  Attribute:

  `Message` String. The error message.

Example

```
<CorrespondenceRuleSet xmlns="http://www.kofax.com/ccm">

  <CorrespondenceRule Reference="C1">
    <MatchCorrespondence WhenXPath="boolean(ccm:Recipients/
ccm:Recipient[@Type='Customer'])"/>
    <CreateCommunication Label="Original" WhenEmpty="AutoRemove"/>
  </CorrespondenceRule>

  <CorrespondenceRule Reference="C2">
    <MatchCorrespondence WhenXPath=
    "ccm:Communiations/ccm:Correspondence[@Label='Original']/@Channel = 'Print'"/>
    <CreateCommunication Label="CopyForPortal"/>
  </CorrespondenceRule>

  <CorrespondenceRule Reference="C3">
    <MatchCorrespondence WhenXPath=
"ccm:Correspondences/ccm:Correspondence[@Label='CopyForPortal']/@Channel =
 'MyChannel'"/>
    <RemoveCommunication Label="CopyForPortal"/>
  </CorrespondenceRule>

  <CorrespondenceRule Reference="C4">
    <MatchAllCorrespondences/>
    <RaiseError Message="Fatal"/>
  </CorrespondenceRule>

</CorrespondenceRuleSet>
```

About communication rules

Communication rules provide details on the communications created by the correspondence rules.

For example, communication rules are used in the following situations:

- For the original communication of a correspondence, rules can determine whether to send the communication directly to the customer or indirectly through an intermediary.

- For the original communication of a correspondence when sent to the customer directly, rules can determine whether to send the communication through email or physical mail.
- For a copy sent to the intermediary, rules can determine which documents in the original communication to include, and which to leave out.

Configure communication rules

To configure communication rules, execute the following steps:

1. In KCM Studio, create an object of type **CCM XML Configuration**.
   Right-click in the lower Navigator pane and then click **New** > **CCM XML Configuration**.
2. In the **Object Inspector** pane, set the **ConfigurationType** property to **CommunicationStep**.
3. Edit the XML object in the central pane.
4. Define the communication rules by means of XML elements, as described in the next two sections, and divide the communication rules into rule sets.
5. Create an alias table.
   Right-click in the lower Navigator pane and then click **New** > **Alias table**.
6. Edit the alias table. Create entries for each of the rule sets.
   Right-click the alias table and click **Create new Key**. The keys must be subsequent natural numbers, starting from 1.
7. Activate the communication rules.
   To do so, on the **Administration** tab, click **System Administration**, and then set the **Communication step configuration** property to the object you created.

Apply communication rules

The communication rules are applied by the CCM_Communication standard process on each communication that it creates. The communications to create for a correspondence are determined by the correspondence rules, which are also applied by CCM_Communication.

The communication rules are applied one by one, first in the order in which the rule sets are referenced from the alias table, and then within each rule set.

The communication rules are applied to an XML serialization of a communication under construction, which is an intermediary form between a correspondence and a communication. A communication under construction begins as a copy of a correspondence, and is gradually extended with information about the communications. After the communication rules are applied, the correspondence specific components are removed, leaving a normal communication at the end.

For the communication rules, a communication under construction is simply called a communication. The XML serialization is defined by the CcmBomCommunicationInput XSD. You can find the B&OM XSD files in: `<deploy root>\KCM\Documentation\5.3.0\Resources\Schemas\Output Management`.

Example

```
<ccm:Communication Label=”Original” RecipientType="Customer" Channel="Print"
                   xmlns:ccm=”http://www.kofax.com/ccm”
                   xmlns:data="http://www.kofax.com/ccm/data"
                   xmlns:ct=”http://www.kofax.com/ccm/contact”
```

```
                  xmlns:st="http://www.kofax.com/ccm/slottype">
  <ccm:Sender>
    <ct:Id>PQR</ct:Id>
    <ct:Name>John</ct:Name>
  </ccm:Sender>
  <ccm:Recipients>
    <ccm:Recipient Type="Customer">
      <ct:Id>XYZ</ct:Id>
      <ct:Name>Bob</ct:Name>
    </ccm:Recipient>
    <ccm:Recipient Type="Intermediairy">
      <ct:Id>ABC</ct:Id>
      <ct:Name>Alice</ct:Name>
    </ccm:Recipient>
  </ccm:Recipients>
  <ccm:OrganisationalMetadata>
    <data:Key1>value1</data:Key1>
    <data:Key2>value2</data:Key2>
  </ccm:OrganisationalMetadata>
  <ccm:Slots>
    <ccm:Slot Name="Cover"
              Selected=”true” Disposable=”true”>
      <ccm:SlotType Name="MySlotType">
        <ccm:Customs>
          <st:Field/>
        </ccm:Customs>
      </ccm:SlotType>
    </ccm:Slot>
    <ccm:Slot Name="Document"
              Overlay="MyOverlay"
              Selected=”false” Disposable=”false”>
      <ccm:SlotType Name="MySlotType">
        <ccm:Customs>
          <st:Test>Test</st:Test>
        </ccm:Customs>
      </ccm:SlotType>
    </ccm:Slot>
  </ccm:Slots>
  <ccm:AllowedChannels>
    <ccm:All>Print:Email</ccm:All>
    <ccm:First>Print</ccm:First>
  </ccm:AllowedChannels>
</ccm:Communication>
```

The following information of the communication under construction is available to the communication rules:

- Contact data of the sender, if any.
- Selected recipient type. Initially not filled in.
- Type and contact data of the allowed recipients. Removed at the end, with the exception of the contact data of the selected recipient type.
- Organisational metadata.

- Composed slots. Of each slot, the following information is available:
  - Slot type.
  - Slot identifier.
  - Overlay to be applied, if any.
  - `Selected` flag. Modifiable, boolean. Initially set to `false`. At the end, all slots with `Selected` set to `false` are removed, and the `Selected` flag itself is removed.
  - `Disposable` flag. Non-modifiable. Initially set to `false`. You can set it to `true` for cover letters that are added to the communication.
  - List of allowed channels. Modifiable. Initially, this list is empty and is removed at the end. Access to the channels is provided to the list as a whole, and to the first element on the list.
- Selected channel. Initially not filled in.

Objects defined in the B&OM Repository are always referenced by their names. In the input communication, this concerns the channels, slot type, and overlay of a slot.

Syntax for communication rules

Each communication rule consists of a "match condition" and one or more "rule actions." If the "match condition" is satisfied, the "rule actions" are applied, one by one. If not, the rules have no effect.

- The "match condition" can either be `MatchCommunication` or `MatchAllCommunications`.
- The "rule actions" can either be `SelectSlots`, `DeselectSlots`, `AddCoverLetter`, `SelectRecipientType`, `SetChannel`, `InitializeAllowedChannels`, `FilterAllowedChannels`, `RemoveAllowedChannels`, `SortAllowedChannels`, or `ModifyOrganisationalMetadata`.

The rules may contain XPath expressions that are evaluated on the input correspondence. XML context is selected based on the communication `(root)node` as defined in the CcmBomCommunicationInput XSD. The expressions are evaluated according to the XPath 1.0 standard and must use the following fixed name spaces to navigate through the correspondence.

Namespace	Prefix	Nodes
http://www.kofax.com/ccm	`ccm`	All nodes, except the next three nodes
http://www.kofax.com/ccm/data	`data`	Keys of `OrganisationalMetadata`
http://www.kofax.com/ccm/contact	`ct`	Contact data of `Senders` and `Recipients`
http://www.kofax.com/ccm/slottype	`st`	Customization of `SlotType`

For example, to navigate to the `Name` field of the sender of the input correspondence, use `ccm:Sender/ct:Name`.

The following elements are part of the syntax:

- **MatchCommunication** (match condition)

  ```
  <MatchCommunicationWhenXPath="…"/>
  ```

  The `MatchCommunication` condition evaluates the XPath expression on the input communication. If the result is true, the condition succeeds. Otherwise, the condition fails.

  Attribute:

  `WhenXPath`. XPath expression to be evaluated on the input communication as a whole.

  The `MatchCommunication` condition throws an error when the result of the XPath expression is not a boolean value.

- **MatchAllCommunications** (match condition)

  ```
  <MatchAllCommunications/>
  ```

  The `MatchAllCommunications` condition always succeeds. You can use it to express communication rules that must be applied unconditionally.

  The condition has no attributes.

- **SelectSlots** (rule action)

  ```
  <SelectSlots WhenXPath="…"SetOverlay="…"/>
  ```

  The `SelectSlots` action sets the selected flag to true for the slots for which the given XPath expression evaluates to true. When specified, the given overlay is also applied to these slots, replacing any existing overlay. `SelectSlots` does not consider slots that are already selected.

  Attributes:

  - `WhenXPath`. XPath expression to be evaluated on `ccm:Slot` nodes.
  - `SetOverlay`. *Optional*. Object name of an overlay in the B&OM Repository.

  `SelectSlots` throws an error when the result of the XPath expression is not a boolean value, or when the overlay parameter does not refer to an overlay object name in the B&OM Repository.

- **DeselectSlots** (rule action)

  ```
  <DeselectSlots WhenXPath="…"/>
  ```

  The `DeselectSlots` action sets the selected flag to false for the slots for which the given XPath expression evaluates to true. `DeselectSlots` does not consider slots that are already not selected.

  Attributes:

  `WhenXPath` XPath expression to be evaluated on `ccm:Slot` nodes.

  The `DeselectSlots` action raises an error when the result of the XPath expression is not a boolean value.

- **AddCoverLetter** (rule action)

  ```
  <AddCoverLetter Project="…" Template="…" Disposable="…"/>
  ```

  The `AddCoverLetter` action adds a cover letter to the communication, by composing a Document Pack Template and selecting the cover letter slot from it. The cover letter is inserted as the first slot of the communication and gets the parameter value for its `Disposable` flag.

  Note The cover letter is only added to the communication at the end of the communication creation, so it has access to the final content of the communication, but it is not affected by the other rule actions.

  You can only call `AddCoverLetter` once for each communication.

  Attributes:

  - `Project` String. The name of the project in KCM where the Document Pack Template of the cover letter is located. A correspondence type must exist for this Document Pack Template (see Define a correspondence type).
  - `Template` String. The name of the Document Pack Template with this cover letter. The Document Pack Template is given a Data Backbone that will be used to compose the cover letter. This Data Backbone is defined by the CcmBomCoverLetterBackbone XSD.
  - `Disposable` Boolean flag. Indicates whether the cover letter must be disposed in case another cover letter is added later by the bundling process.

  `AddCoverLetter` throws an error in the following cases:

  - When no correspondence type is defined for the given Document Pack Template.
  - When composition of the Document Pack Template fails.
  - When the composed Document Pack contains other slots than the cover letter.
  - When another communication cover letter was added earlier.
  - When the slot identifier of the added cover letter is already used by one of the selected slots of the communication. This check is delayed until all slot selections are finished.

- **SelectRecipientType** (rule action)

  ```
  <SelectRecipientType>…</SelectRecipientType>
  ```

  The `SelectRecipientType` action evaluates the subelement expression to a string, and verifies that it is the type of one of the allowed recipients of the communication. Then, it sets this type as the selected recipient type of the communication.

  `SelectRecipient` must be called exactly once for each communication: if it is not called at all, the created communication is rejected by the `CreateCommunication` correspondence rule: if it is called more than once, it fails the second time.

  The condition has no attributes, but must instead have an XML subelement of type expression. This expression must compute the recipient type to set.

  `SelectRecipient` throws an error when the expression cannot be evaluated to a valid recipient type, or when the recipient type is set already.

- **SetChannel** (rule action)

  ```
  <SetChannel>…</SetChannel>
  ```

  The `SetChannel` action evaluates the subelement expression to a string, and verifies that it is an object name that refers to a channel object in the B&OM Repository. Then, it sets this channel as the selected channel of the communication.

  `SetChannel` must be called exactly once for each communication: if it is not called at all, the created communication is rejected by the `CreateCommunication` correspondence rule; if it is called more than once, it fails the second time.

  The action has no attributes, but must instead have an XML subelement of type expression. This expression must compute the object name of the channel to set.

  `SetChannel` throws an error when the expression cannot be evaluated to a valid channel object name, or when the channel is set already.

- **InitializeAllowedChannels** (rule action)

  ```
  <InitializeAllowedChannels>…</InitializeAllowedChannels>
  ```

  The `InitializeAllowedChannels` action evaluates the subelement expression to a string, and verifies that it is a colon-separated list of object names of channel objects in the B&OM Repository. Then, it sets the allowed channels of the communication to this list of object names. You can only call `InitializeAllowedChannels` once for each communication.

  The action has no attributes, but must instead have an XML subelement of type expression. This expression must compute a colon-separated list of channel object names.

  `InitializeAllowedChannels` throws an error when the expression cannot be evaluated to a valid colon-separated list of channel object names, or when the allowed channels are already initialized.

- **FilterAllowedChannels** (rule action)

  ```
  <FilterAllowedChannels>…</FilterAllowedChannels>
  ```

  The `FilterAllowedChannels` action evaluates the subelement expression to a string, and verifies that it is a colon-separated list of object names of channel objects in the B&OM Repository. Then, it filters the allowed channels from the communication, only keeping the channels that appear in the computed list.

  The action has no attributes, but must instead have an XML subelement of type expression. This expression must compute a colon-separated list of channel object names.

  `FilterAllowedChannels` throws an error when the expression cannot be evaluated to a valid colon-separated list of channel object names, or when the allowed channels are not yet initialized.

- **RemoveAllowedChannels** (rule action)

  ```
  <RemoveAllowedChannels>…</RemoveAllowedChannels>
  ```

  The `RemoveAllowedChannels` action evaluates the subelement expression to a string, and verifies that it is a colon-separated list of object names of channel objects in the B&OM Repository. Then, it removes the channels on the computed list from the allowed channels of the communication.

  The action has no attributes, but must instead have an XML subelement of type expression. This expression must compute a colon-separated list of channel object names.

  `RemoveAllowedChannels` throws an error when the expression cannot be evaluated to a valid colon-separated list of channel object names, or when the allowed channels are not yet initialized.

- **SortAllowedChannels** (rule action)

  ```
  <SortAllowedChannels>…</SortAllowedChannels>
  ```

  The `SortAllowedChannels` action evaluates the subelement expression to a string, and verifies that it is a valid colon-separated list of object names of channel objects in the B&OM Repository. Then, it sorts the allowed channels of the communication, using the computed list as the preferred order. Channels that do not appear in the computed list are moved at the end of the allowed channels list. For example, if the current allowed channels are `C:X:A`, and they are sorted by the list `A:B:C:D`, then the end result would be `A:C:X`.

  The action has no attributes, but must instead have an XML subelement of type expression. This expression must compute a colon-separated list of channel object names.

  `SortAllowedChannels` raises an error when the expression cannot be evaluated to a valid colon-separated list of channel object names, or when the allowed channels are not yet initialized.

- **ModifyOrganisationalMetadata** (rule action)

  ```
   <ModifyOrganisationalMetadataKey="…">…</ModifyOrganisationalMetadata>
  ```

  `ModifyOrganisationalMetadata` evaluates the subelement expression to a string, and sets the given key in the organizational metadata of the communication to the computed value.

  Attribute:

  `Key`. String. The key to set in the organizational metadata of the communication.

  The action also has an XML subelement of type expression. This expression must compute the value to set in the organizational metadata.

  `ModifyOrganisationalMetadata` throws an error when the expression cannot be evaluated to a string, or when the given key does not appear in the organizational metadata configuration.

Expressions

Expressions are used in rules to denote either a static value that does not depend on the input, or a dynamic value that is computed out of the input. A dynamic value is based on concatenations of XPath expressions and constant values.

An XML node of type expression can be one of the following:

1. `<Text Value="…"/>` Denotes a constant value.
2. `<Eval XPath="…"/>` Denotes a dynamic value computed by an XPath expression.
3. `<Concat>…</Concat>` Denotes a concatenation, and must contain two or more subnodes of type expression.

Examples of expressions:

1.
   ```
   <Text Value="A"/>
   ```

2.
   ```
   <Eval XPath="ccm:OrganisationalMetadata/data:Field"/>
   ```

3.
   ```
   <Concat>
   <Eval XPath="ccm:OrganisationalMetadata/data:Field1"/>
   <Text Value=";"/>
   <Eval XPath="ccm:OrganisationalMetadata/data:Field2"/>
   </Concat>
   ```

Example

```
<CommunicationRuleSet xmlns="http://www.kofax.com/ccm">
```

```
  <CommunicationRule Reference="E1">
    <MatchCommunication WhenXPath="(@Label = 'Original') or (@Label =
 'CopyForPortal')"/>
    <SelectSlots WhenXPath="ccm:SlotType/ccm:Customs/st:RecipientType = 'Customer'"/>
    <DeselectSlots WhenXPath="ccm:SlotType/ccm:Customs/st:SupplyOnce = 'true'"/>
  </CommunicationRule>

  <CommunicationRule Reference="E2">
    <MatchCommunication WhenXPath="(@Label = 'Original') and
 (ccm:OrganisationalMetadata/data:ReturnToDepartment = 'true')"/>
    <SelectRecipientType><Text Value="Department"/></SelectRecipientType>
    <SetChannel><Text Value="Print"/></SetChannel>
    <AddCoverLetter Project="X" Template="CoverLetterDepartment" Disposable="false"/>
  </CommunicationRule>

  <CommunicationRule Reference="E3">
    <MatchCommunication WhenXPath="@Label = 'CustomerOriginal'"/>
    <SelectRecipientType><Text Value="Customer"/></SelectRecipientType>
    <AddCoverLetter Project="X" Template="CoverLetterCustomer" Disposable="false"/>
  </CommunicationRule>

  <CommunicationRule Reference="E4">
    <MatchAllCommunications/>
    <InitializeAllowedChannels><Text Value="Print:Emal"/></InitializeAllowedChannels>
  </CommunicationRule>

  <CommunicationRule Reference="E5">
    <MatchCommunication WhenXPath="exists(ccm:Slots/ccm:Slot/ccm:SlotType[@Name='OLA'])
 and (ccm:OrganisationalMetadata/data:OLA_Must_Print = 'true')"/>
    <FilterAllowedChannels><Eval XPath="metadata2"/></FilterAllowedChannels>
    <RemoveAllowedChannels><Eval XPath="metadata"/></RemoveAllowedChannels>
    <SortAllowedChannels><Text Value="Print:Email""/></SortAllowedChannels>
    <SetChannel><Eval XPath="ccm:AllowedChannels/ccm:First"/></SetChannel>
  </CommunicationRule>

  <CommunicationRule Reference="E6">
    <MatchCommunication WhenXPath="none"/>
    <ModifyOrganisationalMetadata Key="A"><Text Value="A"/></
ModifyOrganisationalMetadata>
    <ModifyOrganisationalMetadata Key="B"><Eval XPath="B"/></
ModifyOrganisationalMetadata>
  </CommunicationRule>

</CommunicationRuleSet>
```

Chapter 7

Create requests

This chapter provides information on the format for B&OM requests, which are submitted to KCM by means of XML files. The XML files conform to the XSD files delivered in the schemas directory:

```
<deploy root>\KCM\Documentation\5.3.0\Resources\Schemas\Output Management
```

Each request file contains a single request, which can be one of the following types:

- Correspondence request. Consists of one or more correspondences, which are already in the correct form for document composition. This is the most common type of request.
- Application request. Consists of one or more application events, which will be transformed into correspondences by the application rules.
- Import request. Consists of a combination of a single correspondence and an already composed Document Pack. The correspondence bypasses composition, using the provided document pack instead.

To send the request, place it in the watch folder of the Registration component.

Correspondence request

A correspondence request is a request that already contains all input data for document composition in the right form. It does not need to pass through the application step.

The request consists of the following components:

- Request reference. This is an external identification of the request. A meaningful and unique reference allows the request to be tracked easily throughout its processing in Batch & Output Management. Processing does not fail if the external reference is not unique.
- A correspondence. This is a conceptual set of documents to be composed. The correspondence is distributed to one or more recipients over one or more channels. For more information, see the next section.

Components: correspondence

A correspondence is a conceptual set of documents to be composed and distributed to one or more recipients over one or more channels. The documents are determined by a Document Pack composition, and the correspondence contains the necessary input for this composition.

A correspondence consists of the following attributes:

- A correspondence `Reference`. This is an external identification.

- *Optional*. `Brand`. The brand identifies the style to be used in the HTML version of the HTML cover letter. This attribute is only used if the cover letter is not an HTML template and needs to be converted to HTML with a style sheet.
- *Optional*. `Sender`. The sender describes the contact data of the sender of the correspondence. The active contact configuration object defines which fields are allowed in contact data.

  Each field may have an XML element with its name and simple text content. Nested nodes are not allowed.

  Example

```
<ccm:Sender>
  <Name>Jan</Name>
  <City>MyCity</City>
  …
</ccm:Sender>
```

- One or more `Recipients`. The recipients describe the contact data of the possible recipients of the correspondence. Each recipient is named with a type, and can be referenced and selected by this type. The active contact configuration object defines which fields are allowed in contact data. Each `Recipient` has a `Type` attribute, which may have subnodes for each allowed field with their names and simple text content. Nested nodes are not allowed below that point.

  Example

```
<ccm:Recipients>
  <ccm:Recipient Type="Klant">
    <Name>Jan</Name>
    …
  </ccm:Recipient>
  <ccm:Recipient Type="Intermediair>
    <Name>Jack</Name>
    <City>MyCity</City>
    …
  </ccm:Recipient>
  …
</ccm:Recipients>
```

- Zero or more `OrganisationalMetadata` values. An organizational metadata value is a key/value pair accessible from the KCM components that process the correspondence, to implement custom behavior. The active organizational metadata configuration object defines which keys are allowed. It is required to have keys without a defined default value.

  Each key may have an XML element with its name and simple text content. Nested nodes are not allowed.

  Example

```
<ccm:Recipients>
  <ccm:Recipient Type="Klant">
    <Name>Jan</Name>
    …
  </ccm:Recipient>
  <ccm:Recipient Type="Intermediair>
    <Name>Jack</Name>
    <City>MyCity</City>
    …
  </ccm:Recipient>
  …
</ccm:Recipients>
```

- `DocumentPackTemplate`. The Document Pack Template refers to a definition in KCM Repository by a combination of a project and a name. Additionally, a correspondence type linked to the Document Pack Template must exist in the B&OM Repository.

- `DataBackbone`. The Data Backbone describes the raw input data for the composition of the Document Pack Template. It must conform to the format expected by KCM Core.
- Zero, one, or more `OptionalDocuments`. The optional documents describe which optional slots of the Document Pack Template to keep. The others are removed.
- Zero, one, or more `ImportDocuments`. Import documents are base64 encoded files inserted as they are in the import slots of a composed Document Pack Template. They are placed under the `ImportDocuments` node of a correspondence, and the Import Documents XML must conform to its XSD. The XSD resides here:

  `<deploy root>\KCM\Documentation\5.3.0\Resources\Schemas\Output Management`

  Example

  ```
  <ccm:ImportDocuments>
    <im:importDocuments version=”1” xmlns:im=
                   ”http://www.kofax.com/ccm/importDocuments/1.0”>
      <im:importDocument name=”1” mimetype=”application/pdf”> …
            </im:importDocument>
    </im:importDocuments>
  </ccm:ImportDocuments>
  ```

The `'...'` element stands for base64 content. The recommended location for the namespace attribute for import documents is on the second level of the `ImportDocuments` node.

Application request

An application request contains the input data for composition that needs to be derived by the application rules for the provided business data.

The request consists of the following components:

- Request reference. This is an external identification of the request. A meaningful and unique reference allows the request to be tracked easily throughout its processing in Batch & Output Management. Processing does not fail if the external reference is not unique.
- One or more application events. An application event is a partial correspondence of which the actual correspondences are generated by the application rules. For more information, see the next section.

Components: application event

An application event is a partial correspondence of which the actual correspondences are generated by the application rules. In the application event, you cannot specify a Document Pack Template. It is set by the application rules instead.

An application event also contains `Application`, `Process`, and `Event` attributes that provide additional business data of the submitting application that can be used to compute the Document Pack Template.

Tip To create an application event out of a correspondence, or vice versa, simply copy and modify (remove and/or add) the existing data.

An application event consists of the following attributes:

- A correspondence `Reference`. This is an external identification.

- *Optional*. `Brand`. The brand identifies the style to be used in the HTML version of the HTML cover letter. This attribute is only used if the cover letter is not an HTML template and needs to be converted to HTML with a style sheet.
- `Application`.
- `Process`.
- `Event`.

  Example

  ```
   <ApplicationEvent Reference="Order"
  Brand="MyBrand" Application="A" Process="P" Event="E">
  ```
- *Optional*. `Sender`. The sender describes the contact data of the sender of the correspondence. The active contact configuration object defines which fields are allowed in contact data.

  Each field may have an XML element with its name and simple text content. Nested nodes are not allowed.

  Example

  ```
  <ccm:Sender>
    <Name>Jan</Name>
    <City>MyCity</City>
    …
  </ccm:Sender>
  ```
- One or more `Recipients`. The recipients describe the contact data of the possible recipients of the correspondence. Each recipient is named with a type, and can be referenced and selected by this type. The active contact configuration object defines which fields are allowed in contact data. Each `Recipient` has a `Type` attribute, which may have subnodes for each allowed field with their names and simple text content. Nested nodes are not allowed below that point.

  Example

  ```
  <ccm:Recipients>
    <ccm:Recipient Type=”Klant”>
      <Name>Jan</Name>
      …
    </ccm:Recipient>
    <ccm:Recipient Type=”Intermediair>
      <Name>Jack</Name>
      <City>MyCity</City>
      …
    </ccm:Recipient>
    …
  </ccm:Recipients>
  ```
- Zero or more `OrganisationalMetadata` values. An organizational metadata value is a key/value pair accessible from the KCM components that process the correspondence, to implement custom behavior. The active organizational metadata configuration object defines which keys are allowed. It is required to have keys without a defined default value.

  Each key may have an XML element with its name and simple text content. Nested nodes are not allowed.

  Example

  ```
  <ccm:Recipients>
    <ccm:Recipient Type=”Klant”>
      <Name>Jan</Name>
      …
    </ccm:Recipient>
    <ccm:Recipient Type=”Intermediair>
      <Name>Jack</Name>
      <City>MyCity</City>
  ```

```
    …
  </ccm:Recipient>
  …
</ccm:Recipients>
```

- `DataBackbone`. Data Backbone describes the raw input data for the composition of the Document Pack Template. It must conform to the format expected by KCM Core.
- Zero, one, or more `OptionalDocuments`. Optional documents describe which optional slots of the Document Pack Template to keep. The others are removed.
- Zero, one, or more `ImportDocuments`. Import documents are base64 encoded files inserted as they are in the import slots of a composed Document Pack Template. They are placed under the `importDocuments` node of a correspondence, and the Import Documents XML must conform to its XSD. The XSD resides here:

  `<deploy root>\KCM\Documentation\5.3.0\Resources\Schemas\Output Management`

  Example

  ```
  <ccm:ImportDocuments>
    <im:importDocuments version="1" xmlns:im=
                    "http://www.kofax.com/ccm/importdocuments/1.0">
      <im:importDocument name="1" mimetype="application/pdf"> …
            </im:importDocument>
    </im:importDocuments>
  </ccm:ImportDocuments>
  ```

Import request

An import request is a request with a single correspondence that has already been composed externally.

The request consists of the following components:

- Request reference. An external identification of the request. A meaningful and unique reference allows the request to be tracked easily throughout its processing in Batch & Output Management. Processing does not fail if the external reference is not unique.
- One import correspondence. A correspondence for which all input data for composition has been removed. The removed data are the Data Backbone, Optional Documents, and any Import Documents file. All other data remains, which allows the import correspondence to be treated the same as a normal correspondence for the purpose of communication and delivery. For more information, see the next section.
- One Document Pack (external .zip file). The Document Pack is the saved result of an earlier external document composition. When an import request is registered, it looks for a .zip file with the same file name as the request XML.

  Note If an active Registration component is scanning the requests folder, the Document Pack .zip file should be inserted first.

Components: import correspondence

An import correspondence consists of the following attributes:

- A correspondence `Reference`. This is an external identification.

- *Optional*. `Brand`. The brand identifies the style to be used in the HTML version of the HTML cover letter. This attribute is only used if the cover letter is not an HTML template and needs to be converted to HTML with a style sheet.
- *Optional*. `Sender`. The sender describes the contact data of the sender of the correspondence. The active contact configuration object defines which fields are allowed in contact data.

  Each field may have an XML element with its name and simple text content. Nested nodes are not allowed.

  Example

```
<ccm:Sender>
  <Name>Jan</Name>
  <City>MyCity</City>
  …
</ccm:Sender>
```

- One or more `Recipients`. The recipients describe the contact data of the possible recipients of the correspondence. Each recipient is named with a type, and can be referenced and selected by this type. The active contact configuration object defines which fields are allowed in contact data. Each `Recipient` has a `Type` attribute, which may have subnodes for each allowed field with names and simple text content. Nested nodes are not allowed below that point.

  Example

```
<ccm:Recipients>
  <ccm:Recipient Type="Klant">
    <Name>Jan</Name>
    …
  </ccm:Recipient>
  <ccm:Recipient Type="Intermediair>
    <Name>Jack</Name>
    <City>MyCity</City>
    …
  </ccm:Recipient>
  …
</ccm:Recipients>
```

- Zero or more `OrganisationalMetadata` values. An organizational metadata value is a key/value pair accessible from the KCM components that process the correspondence, to implement custom behavior. The active organizational metadata configuration object defines which keys are allowed. It is required to have keys without a defined default value.

  Each key may have an XML element with its name and simple text content. Nested nodes are not allowed.

  Example

```
<ccm:Recipients>
  <ccm:Recipient Type="Klant">
    <Name>Jan</Name>
    …
  </ccm:Recipient>
  <ccm:Recipient Type="Intermediair>
    <Name>Jack</Name>
    <City>MyCity</City>
    …
  </ccm:Recipient>
  …
</ccm:Recipients>
```

- `DocumentPackTemplate`. The Document Pack Template refers to a definition in KCM Repository by a combination of a project and a name. Additionally, a correspondence type linked to the Document Pack Template must exist in B&OM Repository.

Chapter 8

About standard processes

The KCM B&OM functionality is implemented in a set of standard processes that are installed with the product. Running the set of standard processes enables you to transform input request files into distribution output.

The following tables list and describe the standard processes in the order you need to launch them.

Table 1: General steps

Step in the processing sequence	Purpose	Process	Description	Link to the section
Step 1	This process registers submitted requests	CCM_Registration	This process registers submitted requests.	Step 1: CCM_Registration
Step 2	Parse the requests	CCM_Application	This process decomposes the requests into correspondences.	Step 2: CCM_Application
Step 3	Determine the output	CCM_Communication	This process determines what output is delivered electronically and what output is printed.	Step 3: CCM_Communication

Refer to Table 2 for the next steps to perform when producing electronic output.

Refer to Table 3 for the next steps to perform when producing printed output.

Table 2: Steps for electronic output

Step in the processing sequence	Purpose	Process	Description	Link to the section
Step 4	Deliver electronic output	CCM_ElectronicDelivery	This process sends electronic communications to the Electronic Delivery exit point on KCM Core.	Step 4: CCM_ElectronicDelivery

Table 3: Steps for printed output

Step in the processing sequence	Purpose	Process	Description	Link to the section
Step 4	Deliver print output	CCM_PrintDelivery	Transforms print communications into a different representation needed for the next steps.	Step 4: CCM_PrintDelivery
Step 5		CCM_XpsConversion	This process converts the documents for print communications to the XPS format.	Step 5: CCM_XpsConversion
Step 6		CCM_Bundling	This process combines print communications into envelopes.	Step 6: CCM_Bundling
Step 7		CCM_Stacking	This process combines envelopes into print stacks.	Step 7: CCM_Stacking
Step 8		CCM_Streaming	This process determines page modifications for print stacks.	Step 8: CCM_Streaming
Step 9		CCM_Conversion	This process creates print files for print stacks (one print file per stack).	Step 9: CCM_Conversion
Step 10		CCM_Distribution	This process distributes print files.	Step 10: CCM_Distribution
Not applied	Perform the cleanup	CCM_CleanUpAll	This process cleans up database entries and files stored on the file system.	CCM_CleanUpAll standard process

The following applies to all standard processes:

- They can be run within Windows Service Hosts or directly from KCM Studio.
- They need to be configured before they can be used.
- They can be modified if custom behavior is required.

For more information on each process, see Standard processes.

Information transfer between standard processes

The standard processes transfer information to one another by means of the run-time database and the internal storage folder.

Note To run the standard processes remotely, allow the remote machine access to both the database and the storage folder.

These tables demonstrate how the information is exchanged among the processes.

Table 1: general steps

Step in the processing sequence	Standard process	Reads	Writes
Step 1	CCM_Registration	request files [1]	request records [2]
Step 2	CCM_Application	request records [2]	correspondence records [2]
Step 3	CCM_Communication	correspondence records [2]	communication records [2]

Table 2: electronic output

Step in the processing sequence	Standard process	Reads	Writes
Step 4	CCM_ElectronicDelivery [5]	communication records [2]	communication records and electronic output [1]

Table 3: printed output

Step in the processing sequence	Standard process	Reads	Writes
Step 4	CCM_PrintDelivery	communication records [2]	process and job records [2,3]
Step 5	CCM_XpsConversion [4]	process records [2]	process records [2]
Step 6	CCM_Bundling	process records [2]	envelope records [2]
Step 7	CCM_Stacking	envelope records [2]	stack records [2]
Step 8	CCM_Streaming [4]	stack records [2]	stack records [2]
Step 9	CCM_Conversion [4]	stack records [2]	stack records [2]
Step 10	CCM_Distribution [4]	stack records [2]	stack records and print output [1]

Note the following:

[1] Stored in public input and output folders, to be maintained by the customer.

[2] These records are stored in the run-time database and can be monitored in KCM Studio.

[3] A process record is an alternate representation of a print communication.

[4] These standard processes transform an input record, instead of creating a new output record.

Configure global settings for standard processes

The following settings are configured on the system and are mandatory to run the standard processes.

1. Select the **Administration** tab and then click **System Administration**.

2. In the central pane, click the B&OM system object.
3. In the **CCM Configuration** section on the right, review the following system settings:
   - Default output folder.
   - Default requests folder.

     Note To start the B&OM processes, ensure that you have full access rights on this folder.

   - Runtime DB Alias. Specifies the run-time database to connect to.
   - Storage folder. This setting is automatically set during the installation of B&OM, but you can modify it. It specifies the root directory path of the B&OM storage folder. This folder is used to store the internal files, such as the results of the different processing steps.
4. Click the Save button in the upper left corner to save the changes.
   These changes are applied to all objects in the system.

To configure the time when the cleanup of the database entries and files is performed on your system, follow this step:

1. In the **Garbage Collection** section on the right, set the **Clean Up Delay** setting to the number of minutes that the system must wait until performing the cleanup of unneeded and finished stacks and requests.
2. Save the changes.

Also, at this point, you can set the conversion method for the DOCX to XPS conversion that happens during the processing. By default, the Rendition engine is used. You can configure the system to perform this type of conversion using Microsoft Word:

1. In the **General** section, set the **Convert Docx using** property to **MSWord**.
2. Save the changes.
   For information on the conversion methods, see the *Kofax Communications Manager Core Scripting Developer's Guide*.

Standard processes

Electronic output

Step 1: CCM_Registration

The CCM_Registration standard process registers the submitted requests by creating database records to be processed by CCM_Application.

The process monitors a watch folder for request files and registers them in the database.

Step 2: CCM_Application

The CCM_Application standard process decomposes request records created by CCM_Registration into correspondence records to be processed by CCM_Communication. This process performs the following functions:

- Parses the stored request files.
- Decomposes requests into correspondences, which are later processed individually.
- Applies the application rules to fill in missing data in application requests, transforming the contained application events into correspondences.

Step 3: CCM_Communication

The CCM_Communication standard process implements the processing of correspondences created by CCM_Application into two communication types:

- Print communications to be processed by CCM_PrintDelivery
- Electronic communications to be processed by CCM_ElectronicDelivery

The process performs the following functions:

- Composes the documents of a correspondence on KCM Core.
- For each correspondence, the process applies communication rules to determine:
    1. Which recipients receive the output
    2. Which channel is used to send the output to each recipient
    3. Which documents are part of the output for each recipient/channel
- For each document of each print communications, the process selects which representation is used. For example, the Document Pack Template can contain both a DOCX and a PDF representation of a single document; CCM_Communication takes one of these representations for distribution, using PDF > DOCX > XPS > TIFF as the preferred order. The other representation is discarded.

Step 4: CCM_ElectronicDelivery

The CCM_ElectronicDelivery standard process sends electronic communications created by CCM_Communication to the Electronic Delivery exit point on KCM Core. This exit point handles the actual distribution of electronic communications.

This process ensures that the Electronic Delivery exit point is provided with input in the correct format, but does not perform additional processing. The actual electronic delivery is implemented in the exit point.

Printed output

Step 1: CCM_Registration

The CCM_Registration standard process registers the submitted requests by creating database records to be processed by CCM_Application.

The process monitors a watch folder for request files and registers them in the database.

Step 2: CCM_Application

The CCM_Application standard process decomposes request records created by CCM_Registration into correspondence records to be processed by CCM_Communication. This process performs the following functions:

- Parses the stored request files.
- Decomposes requests into correspondences, which are later processed individually.
- Applies the application rules to fill in missing data in application requests, transforming the contained application events into correspondences.

Step 3: CCM_Communication

The CCM_Communication standard process implements the processing of correspondences created by CCM_Application into two communication types:

- Print communications to be processed by CCM_PrintDelivery
- Electronic communications to be processed by CCM_ElectronicDelivery

The process performs the following functions:

- Composes the documents of a correspondence on KCM Core.
- For each correspondence, the process applies communication rules to determine:
    1. Which recipients receive the output
    2. Which channel is used to send the output to each recipient
    3. Which documents are part of the output for each recipient/channel
- For each document of each print communications, the process selects which representation is used. For example, the Document Pack Template can contain both a DOCX and a PDF representation of a single document; CCM_Communication takes one of these representations for distribution, using PDF > DOCX > XPS > TIFF as the preferred order. The other representation is discarded.

Step 4: CCM_PrintDelivery

The CCM_PrintDelivery standard process implements the processing of print communications created by CCM_Communication into process records to be processed by CCM_XpsConversion.

This process also converts all DOC documents of the communication to XPS. When MS Word is configured on the system as the conversion engine, also all DOCX are transformed to XPS by CCM_PrintDelivery (for information on the conversion methods, see Configure global settings for standard processes).

Step 5: CCM_XpsConversion

The CCM_XpsConversion standard process implements the transformation of process records created by CCM_PrintDelivery into modified process records to be processed by CCM_Bundling. These process records represent print communications.

The process performs the following functionality:

- Creates an XPS file for each document of a print communication by converting the print representation chosen by CCM_Communication into the XPS format.

Step 6: CCM_Bundling

The CCM_Bundling standard process implements the bundling of process records as transformed by CCM_XpsConversion into envelopes to be processed by CCM_Stacking. The bundled process records represent print communications.

The process performs the following functionality:

- Combines process records for the same channel and recipient into the same envelope
- Puts process records that cannot be combined in their own envelopes
- Calculates size and weight of each envelope
- Determines postage class of each envelope
- Sets the printer for each envelope
- *Optional*. Composes a new cover letter for envelopes

Step 7: CCM_Stacking

The CCM_Stacking standard process implements the batching of envelopes as left behind by CCM_Bundling into stacks to be processed by CCM_Streaming.

This process allows envelopes to be printed in configurable batches.

Step 8: CCM_Streaming

The CCM_Streaming standard process implements the transformation of stacks created by CCM_Stacking into modified stacks to be processed by CCM_Conversion.

The process performs the following functions:

- Inserts separator pages in stacks
- Adds bar codes, OMR codes, or other page modifications to stacks
  For more information, see the section "Include additional pages for stack, envelope, or document" in the *KCM Batch & Output Management Scripting Developer's Guide*.

Step 9: CCM_Conversion

The CCM_Conversion standard process implements the processing of stacks as transformed by CCM_Streaming into print files to be distributed by CCM_Distribution.

The process performs the following functions:

- Creates print files
- Combines all documents of all envelopes in a stack into a single file
- Applies the page modifications that were determined by CCM_Streaming
- Converts documents from XPS into a configurable output format

Step 10: CCM_Distribution

The CCM_Distribution standard process implements the distribution of print files as converted from stacks by CCM_Conversion. These print files constitute the output of B&OM.

The process performs the following functions:

- Creates a print stack description file that describes the output
- Distributes print files and description files according to configuration

CCM_CleanUpAll standard process

The CCM_CleanUpAll standard process cleans up database entries and files stored on the file system. By default, once the process is started, the cleanup is performed at 1 AM on a daily basis (see GarbageCollection and Timer).

The CCM_CleanUpAll process identifies the finished requests and stacks for which the cleanup delay is already passed. Then it removes the identified requests and their associated correspondences and communications, as well as the identified stacks and their associated envelopes, processes, and jobs. Additionally, the process immediately removes all abandoned requests, processes, envelopes and stacks, as well as their associated items. Abandoned correspondences and communications are not cleaned up directly and only removed when their requests are cleaned up.

If a stack, request or process cannot be deleted, the process does not stop. It retries the cleanup during the next cycle.

Change the standard processes

You can change the processing objects, scripts and conditions to configure the standard processes or to customize their behavior. These objects are not affected by a KCM B&OM upgrade. For more information, see the next chapter.

Also, you can create duplicates of standard processes if it is necessary to run multiple instances with different behavior. When multiple different instances are not required, you should directly change the original standard process.

Chapter 9

Configure standard processes

The core functionality used by the standard processes is defined by their processing components.

To configure any standard process, you need to configure the properties of its processing component. Some processing components have only one property, Runtime DB Alias, which is already configured globally on the system (see Configure global settings for standard processes).

Additionally, configuring the standard processes involves writing correspondence, communication, and possibly application rules. For more information, see Create application, correspondence, and communication rules.

Also, you can use the configuration functions to change the process behavior. For more information, see the section "Processing component functions" in the *Kofax Communications Manager Batch & Output Management Scripting Language Developer's Guide*.

Configure processing components in KCM Studio

This section lists the processing components that constitute the standard processes and provides information on how to configure their available properties in KCM Studio.

Registration

To configure the CCM_Registration standard process, you need to adjust the properties of the Registration component:

1. In the **Navigator** upper pane, select the folder. In the **Navigator** lower pane, find and double-click **CCM_Registration**.
   The process appears in the central pane.
2. Click **Registration** in the lower central pane.
3. In the **Object Inspector** pane on the right, click the ellipsis button to configure the following Registration properties, if applicable.
   - RequestDirectory. Select the watch folder from which requests are read.
   - ErrorDirectory. Select the folder to which erroneous requests are moved.
4. Click the Save button in the upper left corner to save the changes.
   The changes are saved in the process itself and only affect the Registration component of CCM_Registration.

Application

To configure the CCM_Application standard process, you need to adjust the properties of the Application component:

1. In the **Navigator** upper pane, select the folder. In the **Navigator** lower pane, find and double-click **CCM_Application**.

   The process appears in the central pane.
2. Click **Application** in the lower central pane.
3. In the **Object Inspector** pane on the right, click the ellipsis button to configure the following Application property, if applicable.
   - TraceDirectory. Select the folder to which trace files are written for the application rules. When specified, Trace Mode is enabled, and trace XML files with information on the performed steps are written. When left empty, no trace files are created.

     When Trace Mode is enabled, the Application component creates a subfolder in the folder specified in the TraceDirectory property for each application request that it processes. In this subfolder, the component creates trace XML files for all steps performed by the application rules. The files are written in the order they were processed and provide an information thread from the application events in the input request to the created output correspondences.

     Therefore, both the input and output are written as XML files to the trace subfolder for each of the following items:
     - Application event
     - Rule that matches the application event
     - Individual action of the rule
4. Click the Save button in the upper left corner to save the changes. The changes are saved in the process itself and only affect the Application component of CCM_Application.

Communication

To configure the CCM_Communication standard process, you need to adjust the properties of the Communication component.

1. In the **Navigator** upper pane, select the folder. In the **Navigator** lower pane, find and double-click **CCM_Communication**.

   The process appears in the central pane.
2. Click **Communication** in the lower central pane.
3. In the **Object Inspector** pane on the right, click the ellipsis button to configure the following Communication property, if applicable.
   - TraceDirectory. Select the folder to which trace files are written for the correspondence and communication rules. When specified, Trace Mode is enabled, and trace XML files with information on the performed steps are written (see below for more details). When left empty, no trace files are created.

4. Click the Save button in the upper left corner to save the changes.

   The changes are saved in the process itself and only affect the Communication component of CCM_Communication.

When Trace Mode is enabled, the Communication component creates a subfolder in the folder specified in the TraceDirectory property for each correspondence that it processes. In this subfolder, the component creates trace XML files for all steps performed by both the correspondence and communication rules. The files are written in the order they were processed and provide an information thread from the correspondences in the input to the created output communications.

Therefore, both the input and output are written as XML files to the trace subfolder for each of the following items:

- Correspondence
- Correspondence rule that matches it
- Individual action of that rule
- Communication created by CreateCommunication
- Communication rule that matches it
- Individual action of that rule

ElectronicDelivery

To configure the CCM_ElectronicDelivery standard process, you need to customize the ElectronicDelivery.dss script in the KCM Core instance configured on the system.

The script resides here: `[drive]\KCM\Work\<version>\Instance_<number>\core\Scripts\User Library`. To edit the script, use the built-in script editor. To compile the changed script and restart the Document Processors, use KCM Core Administrator.

The ElectronicDelivery.dss script runs in a session where the Document Pack, which contains the documents for the electronic communication, is opened.

The script requires the following parameters:

- `Channel`. *String*. Channel name.
- `Metadata`. *String*. Full path to the Metadata XML.

The script handles the complete distribution of the electronic communication. If the script throws an error, the electronic communication is put into an error state.

Note Batch & Output Management may produce a modified Document Pack that may not comply with the definition of the Document Pack Template used as the basis for the Document Pack. This introduces the following limitations:

- You cannot submit such a Document Pack for processing to KCM.
- Some script operations are not supported for such a Document Pack:
  1. `CopyDocumentPack`
  2. `SaveDocumentPack`
- The Document parameter passed to the `IterateDocumentPack` command running on a modified Document Pack may contain the value `"*none"`, which indicates that this slot was explicitly made empty for the given channel.

The following operations and functions can be used on a modified Document Pack that may be produced by B&OM:

- `IterateDocumentPack` command
- `get_document_from_pack`, `get_slots_from_pack`, and `get_channels_from_pack` functions

For information on these commands and functions, see the *Kofax Communications ManagerCore Scripting Language Developer's Guide*.

Here is an example of the ElectronicDelivery.dss script.

```
/*
 * B&OM Electronic Delivery exit point.
 *
 * This script will be called whenever a channel is configured for electronic
 * delivery.
 *
 * The content of the Document Pack is already activated and can be manipulated
 * using the standard Document Pack APIs.
 */

Parameter Text Channel;  /* Name of the selected channel */
Parameter Text Metadata; /* Full path to the Metadata XML */

/* Document Pack manifest file */
Const Text ManifestXML = "manifest.xml" [ _document_pack, ];

IterateDocumentPack
 Script (ListSlots)
 Channel ("")
 Context ("");
```

```
Example script (ListSlots):

Parameter Text Slot;
Parameter Text Document;
Parameter Text Status;
Parameter Text Context;

If Document <> "*none" Then
  Progress Message ("-------------------> " + Slot);
Fi;
```

PrintDelivery

You do not need to configure the PrintDelivery component of the CCM_PrintDelivery standard process, as it is configured globally on the system.

XpsConversion

You do not need to configure the XpsConversion component of the CCM_XpsConversion standard process as it is configured globally on the system.

When you convert PDF files to XPS, place the DocBridge license file PdfToXpsCpsdk.lic to the B&OM installation folder: `<deploy root>\KCM\Programs\<version>\B&OM`.

Bundling

By default, the CCM_Bundling standard process runs once a day at 2 AM. To override this default interval, follow these steps:

1. In the **Navigator** upper pane, select the folder. In the **Navigator** lower pane, find and double-click **CCM_Bundling**.
   The process appears in the central pane.
2. Double-click the **Timer** object in the central pane.
3. In the dialog that appears, adjust the timer settings and then click **OK**.
4. Click the Save button in the upper left corner to save the changes.
   The changes are saved in the process itself and only affect the Bundling component of CCM_Bundling.

Also, you can use the Bundling component configuration functions to change the process behavior (see the section "Bundling component functions" in the *Batch & Output ManagementScripting Developer's Guide*).

Stacking

By default, the CCM_Stacking standard process runs once a day at 3 AM. To override this default interval, follow theses steps:

1. In the **Navigator** upper pane, select the folder. In the **Navigator** lower pane, find and double-click **CCM_Stacking**.
   The process appears in the central pane.
2. Double-click the **Timer** object in the central pane.
3. In the dialog that appears, adjust the timer settings and then click **OK**.
4. Click the Save button in the upper left corner to save the changes.
   The changes are saved in the process itself and only affect the Stacking component of CCM_Stacking.

Also, you can use the Stacking component configuration functions to change the process behavior. See the section "Stacking component functions" in the *Batch & Output Management Scripting Developer's Guide*.

Streaming

You do not need to configure the Streaming component of the CCM_Streaming standard process, as it is configured globally on the system. But you can use the Streaming component configuration functions to change the process behavior. See the section "Streaming component functions" in the *Batch & Output Management Scripting Developer's Guide*.

Conversion

To configure the CCM_Conversion standard process, you need to adjust the properties of the Conversion component.

Also, you can change the Conversion component configuration functions to change the process behavior. See the section "Conversion component functions" in the *Kofax Communications Manager Batch & Output Management Scripting Developer's Guide*.

Conversion relies on the DocBridge library to carry out file handling.

1. In the **Navigator** upper pane, select the folder. In the **Navigator** lower pane, find and double-click **CCM_Conversion**.
   The process appears in the central pane.
2. Click **Conversion** in the lower central pane.
3. In the **Object Inspector** pane on the right, configure the following Conversion properties, if applicable.
   - ConvertOptions. *Optional*. A comma-separated list of custom conversion options, which are passed to DocBridge.
   - LicenseFileName. Full path to the DocBridge license file.
   - OutputFormat. Format in which the output print file is written, such as AFP, PCL, PDF, POS, PXL, and TIFF. Only output formats for which an output filter is configured in the DocBridge license file are supported.
   - ProfilePath. Folder where DocBridge looks for profiles.
   - TraceFileName. Name of the file where DocBridge writes log messages.
     The actual trace file name is composed of the name you specify in this property, plus the process name, component name, and Windows process identifier.
     Example
     If you specify the file name as `C:\KCM\Work\5.3.0\Output Management\Logs\Compart_trace.txt`, it becomes: `C:\KCM\Work\5.3.0\Output Management\Logs\Compart_trace_CCM_Conversion_Conversion_8796.txt`.
     When this property is left empty, no trace file is written.
   - TraceLevel. Log level for DocBridge.

4. Click the Save button in the upper left corner to save the changes.

   The changes are saved in the process itself and only affect the Conversion component of CCM_Conversion.

Distribution

To configure the CCM_Distribution standard process, you need to adjust the properties of the Distribution component.

Also, you can use the Distribution component configuration function to change the process behavior (see the section "DistributeStack function" in the *Kofax Communications Manager Batch & Output Management Scripting Language Developer's Guide*).

By default, distribution copies the print file generated during conversion and the XML description file created during distribution from the storage folder to the output folder. If the files with the same names already exist in the output folder, the script shows an error message, and if an error is encountered while saving the files, the cleanup of the already distributed files is performed so you can rerun the distribution.

By default, the folders reside here:

- Storage: `<deploy root>\KCM\Work\5.3.0\Output Management\Storage`
- Output: `<deploy root>\KCM\Work\5.3.0\Output Management\Output`

To change the locations, see Configure global settings for standard processes.

To override the default behavior, follow these steps:

1. In the **Navigator** upper pane, select the folder. In the **Navigator** lower pane, find and double-click **CCM_Distribution**.

   The process appears in the central pane.
2. Click **Distribution** in the lower central pane.
3. In the **Object Inspector** pane on the right, configure the following Distribution properties, if applicable.
   - OnStackDistribute. The exit point script to use when distributing the stack.

     By default, the property is set to a sample pre-configured exit point called DistributeStack that scripts the default behavior of distribution. This sample exit point is shipped with the installation. For more information, see the section "Configure the distribution functionality" in the *Kofax Communications Manager Batch & Output Management Scripting Language Developer's Guide*.
   - OutputPath. The path in which the print files are saved.

     If you omit this property, the print files are saved to the default output folder.
4. Click the Save button in the upper left corner to save the changes.

   The changes are saved in the process itself and only affect the Distribution component of CCM_Distribution.

GarbageCollection and Timer

The CCM_CleanUpAll standard process functionality is defined by two parts: a timer, and the CCM_DoCleanUp script. The cleanup timeout is configured globally on the system (see Configure global settings for standard processes).

The timer is responsible for the schedule of the process. By default, once CCM_CleanUpAll is started, the cleanup is performed once a day at 1 AM. To override this default, follow these steps:

1. In the **Navigator** upper pane, select the folder. In the **Navigator** lower pane, find and double-click **CCM_CleanUpAll**.
   The process appears in the central pane.
2. Double-click the **Timer** object in the central pane.
3. In the dialog that appears, adjust the timer settings and then click **OK**.
4. Click the Save button in the upper left corner to save the changes.

Chapter 10

Define overlays

An overlay is a watermark or standard logos or paper marks that you can add to your documents. An overlay can be either placed in front of the other document content, or behind it.

In the printed output, overlays are processed by DocBridge, so you can use any file in a format supported by your DocBridge license. For overlay files in paged formats, such as PDF, PS, or XPS, only the first page is applied, and the other pages are ignored.

Note When defining overlays for printed output, we strongly recommend that you avoid using transparency, which may result in an interleaving or low-resolution output with substantially degraded quality.

In the electronic output, you provide information about the configured overlays to the Electronic Delivery exit point in KCM Core. The exit point then reads and applies the overlays by using the Core Scripting Language commands, such as MergePDF. See the *Kofax Communications ManagerCore Scripting Language Developer's Guide*.

To define an overlay in KCM Studio:

1. In the **Navigator** pane, on the **Folder View** tab, select the folder to store the overlay.
2. In the lower **Navigator** pane, click **New** > **Overlay**.
3. Enter the name for the overlay.

   A new tab appears.
4. In the **Overlay mode** section, make a selection:
   - Select **Above Page** to place the overlay in front of the document content.
   - Select **Below Page** to place the overlay behind the document content.
5. In the **Overlay file path** section, enter the path to the overlay file.
   - If the path is relative, it is resolved relative to the KCM B&OM Storage directory, which resides in: `<deploy root>\KCM\Work\<Version>\Output Management\Storage`. We recommend that you create a subdirectory in the Storage directory to store overlays.
   - If the path is absolute, it is used as is.

     Note If the files are moved or deleted, you need to update the absolute paths manually.
6. Save the changes.

Appendix A

Glossary

This glossary explains the key terms used in this guide in alphabetical order.

- **Channel**

  The way used to distribute correspondences to the recipient. Channels are user-defined. For example, print, portal, and email.
- **Communication**

  A Document Pack [2] customized for *one* particular recipient and *one* particular channel. KCM B&OM can deliver communications in two different ways: as printed output and as electronic output.
- **Correspondence**

  A Document Pack that will be sent to one main recipient and optionally secondary recipients over one or more channels. This Document Pack can be customized for each recipient and channel. The Document Pack Template and the list of possible recipients are specified in the correspondence request.
- **Cover letter**

  The first document in an entity. There are three types of cover letters, which can be used jointly or interchangeably, depending on the desired output:
  - Cover letter as defined for the Document Pack Template in KCM Designer.
  - Cover letter customized for a particular recipient, as optionally defined per communication during the communication phase.
  - Cover letter customized for the set of communications contained in each envelope, as optionally defined per envelope during the bundling phase.
- **Envelope**

  A set of communications for the same recipient.
- **Request**

  The XML file submitted to B&OM that contains input data used to compose documents and produce output for a number of correspondences. There are three types of requests: correspondence, application, and import.
- **Recipient**

  Possible receiver of the correspondence. There are two types of recipients:
  - Main recipient. The main receiver of the correspondence. Information on this recipient is obtained from the Data Backbone for the Document Pack Template.
  - Actual recipients. All possible recipients of the correspondence as specified in the correspondence request. The list of actual correspondence recipients contains the original main recipient and/or different, secondary recipients.

[2] A Document Pack is a KCM term for a set of related documents created from a Document Pack Template.

- **Rules**

  The XML files that determine which communications should be distributed for a particular correspondence, supply information about these communications, and also determine which Document Pack Template to compose for a correspondence. There are three types of rules: correspondence, communication, and application rules.
- **Organizational metadata**

  The XML element that supports inclusion of custom business metadata in the Document Pack.

Appendix B

Views columns in KCM Studio

To get detailed information on the various KCM B&OM objects, in KCM Studio, on the **Administration** tab, click the button and a select a view. The following tables list and describe the columns that appear in each view.

Table 1: Request Summary

Column	Description
Request ID	Identifier of the request.
Reference	Reference of the request as set in the request XML.
Progress	Progress of KCM B&OM in creating and distributing the output of this request, such as: Registered, In progress, Completed.
Correspondences	The number of correspondences that are part of this request and divided into: • number of correspondences successfully passed through CCM_Communication • number of abandoned correspondences • total number of correspondences Initially empty. Set by CCM_Application and updated by CCM_Communication for each processed correspondence.
Communications	The number of communications that are part of this request and divided into: • number of distributed communications (successfully passed through CCM_ElectronicDelivery or are part of a stack that successfully passed through CCM_Conversion) • number of abandoned communications • total number of communications Initially empty. Updated by CCM_Communication for each created communication and by both CCM_ElectronicDelivery and CCM_Conversion for each distributed communication.
Registration time	In UTC, time when the request record was registered (the same as the Creation time in the Request view).
Start time	In UTC, time when the processing for this request is started (the same as the Start time in the Request view). Initially empty.

Column	Description
Completion time	In UTC, time when all output of the request is either distributed by KCM B&OM or abandoned. Initially empty.

Table 2: Request

Column	Description
Request ID	Identifier of the request.
Reference	Reference of the request as set in the request XML.
Status	Current status of the request after being processed by CCM_Application: Waiting, Busy, Finished, Error, or Abandoned.
Creation time	In UTC, time when the request record was created.
Start time	In UTC, time when the processing for this request is started. Initially empty. Set by the CCM_Application standard process.
Finish time	In UTC, time when the processing by CCM_Application for this request is finished. Initially empty. Set by the CCM_Application standard process.
Error time	In UTC, time when an error occurred. Initially empty.
Error message	Error message text. Initially empty.
Abandon time	In UTC, time when the request is abandoned. Initially empty.

Table 3: Correspondence

Column	Description
Correspondence ID	Identifier of the correspondence.
Status	Current status of the correspondence, such as: Waiting, Busy, Finished, Error, or Abandoned.
Correspondence reference	Correspondence reference as set in the request XML.
Request ID	Identifier of the correspondence request.
Request reference	Request reference as set in the request XML.
Project	KCM Designer project that contains the Document Pack used for this correspondence.
Document pack template	Document Pack used for this correspondence.
Brand	Brand as set in the request XML. Empty when no brand was entered.
Sender	Sender as set in the request XML. Can be changed with the application rules. Empty when no sender is entered.
Recipients	List of recipients for the correspondence.
Organisational metadata	Organisational metadata of the correspondence.

Column	Description
Creation time	In UTC, time when the correspondence record was created.
Start time	Time when the communication for this correspondence is started. Initially empty. Set by the CCM_Communication standard process.
Finish time	Time when the correspondence is finished. Initially empty. Set by the CCM_Communication standard process.
Error time	Initially empty. In UTC, time when the error occurred.
Error message	Initially empty. Error message text.
Abandon time	Initially empty. In UTC, time when the correspondence is abandoned.
Data Backbone	Data Backbone used as input for this correspondence. Can be changed with the application rules.
Optional document	Optional document that is included in this correspondence as set in the request XML. Can be extended with the application rules.

Table 4: Communication

Column	Description
Communication ID	Identifier of the communication.
Communication type	Type of communication, such as: Print or Electronic.
Status	Current status of the communication, such as: Waiting, Busy, Finished, Error, or Abandoned.
Request Id	Identifier of the communication request.
Correspondence ID	Identifier of the correspondence for this communication.
Label	Label as set by the correspondence rules to identify the communication in the correspondence.
Channel	Channel used to deliver this communication.
Brand	Brand as set in the request XML. Empty when no brand is entered.
Recipient type	Recipient type to which the communication is delivered, as set in the communication rules.
Sender	Sender as set in the request XML. Empty when no sender is entered. Can be changed with the application rules.
Recipient	Recipient of the communication as selected by the communication rules.
Organisational metadata	Organisational metadata of the communication.
Creation time	In UTC, time when the communication record is created.

Column	Description
Start time	Time when delivery is started. Initially empty. Set by the CCM_PrintDelivery or CCM_ElectronicDelivery standard process.
Finish time	Time when delivery is finished. Initially empty. Set by the CCM_PrintDelivery or CCM_ElectronicDelivery standard process.
Error time	Initially empty. In UTC, time when the error occurred.
Error message	Initially empty. Error message text.
Abandon time	Initially empty. In UTC, time when the correspondence is abandoned.

Table 5: Open Jobs

Column	Description
ID	Unique identifier of the job.
Envelope ID	Identifier of the envelope where this job is incorporated. Initially not filled in. Set by the CCM_Bundling standard process.
State	Current state of the job, such as: • Created. Initial state after creation. • Rendition created. Converted to XPS. • Error creating rendition. Error while converting to XPS. • Disposed. Communication cover letter that is disposed. (This is a cover letter with Disposable set to "true" that is later replaced by a new cover letter during bundling.)
Seq. number	Before CCM_Bundling, this is the sequence number of the job in the process. After CCM_Bundling, this is the sequence number of the job in the envelope. The sequence numbers start at 1.
Created at	Time on the server when the job record is created.
Printername	Printer for which print distribution is created. Initially not filled in. Set by the CCM_Bundling standard process.
Form	Form used when print distribution is created for this job.
Annex	Reserved for future use.
Document	Identifier of the slot from which this job originated.
Document type	Format of the input document of the job, such as: DOCX, PDF, TIFF, or XPS.
Filename	Full path to the input document of the job.

Column	Description
Rendition	Full path to the XPS file of the job, which is generated from the input document. Initially not filled in. Set by the CCM_XpsConversion standard process.
Pagecount	Number of pages of the job. Initially not filled in. Set by the CCM_XpsConversion standard process.
Papercount	Number of papers needed for printing the job. Takes duplex printing into account. Initially not filled in. Set by the CCM_Bundling standard process.
Simplex/Duplex	Mode of file printing, such as: • Simplex • Duplex/horizontal • Duplex/vertical
Document collection - document-nr.	Sequence number of the job in the process. The sequence number starts at 1.
Process ID	Identifier of the process linked to this job.
Request ID	Identifier of the request from which this job originated.
Correspondence ID	Identifier of the correspondence from which this job originated.
Channel name	Channel used to distribute the job.
Slot type name	Slot type from which this job originated.
Cover letter flag	Type of the cover letter in this job, such as: • 0. Not a cover letter • 1. Cover letter that cannot be disposed ("Disposable" set to "false") • 2. Cover letter that may optionally be disposed ("Disposable" set to "true")
Overlay file name	Overlay to optionally apply to the job.
Overlay option	Method to apply the overlay, such as: • 0. above page • 1. below page • 2. no overlay

Table 6: Processes

Column	Description
ID	Global unique identifier of the process.
Status	Current status of the process, such as: Waiting, Busy, Finished, Error, or Abandoned.

Column	Description
Process state	Current state of the process, such as: • Imported. Initial state after creation. • Rendition created. XPS variant is created for all process jobs. • Locked. Blocked for processing. • Delete. Marked for deletion.
Created at	In UTC, time when the process record was created.
Title	Label of the communication for which the process is created.
Error message	Error message. Set when the process is in error.
Communication Id	Communication for which the process is created.
Envelope Id	Envelope where the process is incorporated. Initially not filled in. Set by the CCM_Bundling standard process.
Seq. number	Sequence number of the process in the envelope where it is incorporated. Initially not filled in. Set by CCM_Bundling standard process.

Table 7: Envelope

Column	Description
ID	Unique identifier of the envelope.
Status	Current status of the envelope, such as: Waiting, Finished, Error, or Abandoned.
Stack ID	Identifier of the stack containing this envelope. Initially empty. Set by the CCM_Stacking standard process.
Seq. number	Sequence number of the envelope in the stack where it is incorporated. Initially empty. Set by the CCM_Stacking standard process.
Channel	Channel to distribute the envelope.
Created at	In UTC, time when the envelope record was created.
Error message	Error message. Set when the envelope is in error.
Postage	Postage class selected for the envelope. Set by the CCM_Bundling standard process.
Pagecount	Number of pages in the envelope.
Sheet number / Papercount	Number of papers needed for printing the envelope. Takes duplex printing and annexes into account.

Column	Description
Weight	Weight of the envelope, which includes the number of papers, and the weight of the envelope cover. The weight is shown in grams. Set by the CCM_Bundling standard process.
Name	Name of the envelope definition (repository object) that applies to this envelope record (runtime object). Selected from the postage class.
Organisational metadata	Organizational metadata of this envelope, in JSON.
Annex	Reserved for future use.

Table 8: Stack

Column	Description
ID	Unique identifier of the stack.
Status	Current status of the process, such as: Waiting, Busy, Finished, Error, or Abandoned.
Stacktype	Current type of the stack, such as: • Streaming (20). Created, ready for CCM_Streaming • Convert (30). Streamed, ready for CCM_Conversion • Distribute (70). Converted, ready for CCM_Distribution • Finished (99). Print file created
Channel	Channel to distribute the stack.
Printer	Printer to distribute the stack.
Created at	In UTC, time when the stack record was created.
Organisational metadata	Organisational metadata of the stack.
Distributed at	In UTC, time when the stack was distributed.
Error message	Error message, only set when the stack is in error.
Streamfile	Full path to the stream file created for the stack. Initially not filled in. Set by the CCM_Streaming standard process.
Papercount	Number of pages in the stack.
Pagecount	Number of papers needed to print the stack. Takes duplex printing into account.
Job count	Number of jobs contained in the stack.
Envelope count	Number of envelopes contained in the stack.