# Kofax Communications Manager

Repository User's Guide

Version: 5.4.0

Date: 2020-08-26

# Table of Contents

**Preface....... 5**

Related documentation....... 5

Getting help with Kofax products....... 6

**Chapter 1: Introduction....... 8**

Older versions of Repository....... 8

Roles and authorization....... 9

Predefined roles....... 9

**Chapter 2: Work with KCM Repository....... 11**

CM Repository as a file system....... 11

Projects and folders....... 12

Create projects....... 13

View and edit objects....... 14

Lifecycle management....... 14

Use revisions of objects....... 15

Restore deleted items....... 16

Label objects....... 17

Assign characteristics to objects....... 17

Create copies of objects....... 17

Dependencies between different objects....... 19

About Master Templates....... 21

Use auto-includes....... 22

Search paths....... 22

**Chapter 3: Develop templates and content....... 25**

Getting started....... 25

Work with data....... 26

Use Data Backbone....... 26

Use DIDs....... 28

Work with content....... 28

Building blocks....... 28

Master Templates and Master Template scripts....... 29

Text Blocks....... 30

Rich Text Blocks....... 30

Content Wizards....... 30

Includes....... 31

Forms........................................................................................................ 32
QForms...................................................................................................... 32
Field Sets...................................................................................................33
Text Block Lists..........................................................................................36
Data Backbone........................................................................................... 36
Work with document layout.............................................................................36
Style Document..........................................................................................36
Page Style document.................................................................................. 37
**Chapter 4: Configure contents........................................................................................... 39**
Projects............................................................................................................ 39
Dynamic objects.............................................................................................. 40
Style Documents..............................................................................................41
DIDs................................................................................................................. 41
Master Templates............................................................................................ 41
Run Master Templates on CM ComposerUI from CM Repository...................................... 42
Run Master Templates on CM Core from CM Repository.................................................. 42
Run Master Templates with dynamic objects.....................................................................42
Includes............................................................................................................44
**Chapter 5: Manage data..................................................................................................... 45**
Save and Load................................................................................................ 45
Import and Export............................................................................................46
Set the identifier for revisions.....................................................................46
Export........................................................................................................ 47
Import........................................................................................................ 48
**Appendix A: Error report settings.................................................................................. 49**
**Appendix B: Word processor settings............................................................................ 51**
**Appendix C: Other settings............................................................................................. 53**

# Preface

This guide provides general information on Kofax Communications Manager Repository (KCM Repository) and describes common tasks that you can perform using this KCM component.

## Related documentation

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

https://docshield.kofax.com/Portal/Products/KCM/5.4.0-cli2a1c07m/KCM.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 Technical Specifications***
Provides information on supported operating system and other system requirement for Kofax Communications Manager.

***Kofax Communications Manager Installation Guide***
Contains instructions on installing and configuring Kofax Communications Manager and its components.

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

***Kofax Communications Manager Batch & Output Management Getting Started Guide***
Describes how to start working with Batch & Output Management.

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

***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.

[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 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 Batch & Output Management Developer's Guide***
Describes the Batch & Output Management scripting language used in KCM Studio related scripts.

***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 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.

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

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

## 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 Repository is the authoring environment and document management system of KCM. In KCM Repository, templates and content blocks are developed and maintained.

KCM Repository has the following features:

- Central storage of documents and templates, supporting backup and security
- Project organization of documents and templates
- Version control
- Reporting facilities
- User activity tracking
- Multi-User Access control, protecting document integrity
- Dependency tracking
- Document editing facilities
- Development support for compiling and running Master Templates
- Storage and development of data interfaces (Data Backbone and DID)

## Older versions of Repository

This chapter describes the differences between the latest KCM Repository versions and older versions of this component (ITP version 4 and older).

KCM Repository projects have a new folder structure, and the behavior of the projects differs. When upgrading, all existing projects keep the old structure. These projects are referred to as Legacy projects and Legacy shared projects.

It is not possible to create a legacy project other than copying a legacy project that is already in use. The "New project" action creates a new style project.

**Shared projects**

Version 4 of KCM Repository renames Libraries to Shared projects. Legacy projects can use legacy Shared projects. New style projects cannot use Shared projects. For more information on Shared projects, see the ITP version 3.5 documentation.

**Data Backbone**

Data Backbone and Data Definitions have been introduced as a beta feature in ITP version 3.5.18. Starting from ITP version 4.2.0, the official release of Data Backbones is available.

A project has exactly one Data Backbone. That means that all the content in that project uses the same Data Backbone. All content in that project can be shared.

When updating your ITP installation, existing projects are marked as Legacy projects. That means that they still can use multiple Data Backbones in one projects. It is possible to add Data Backbones in such a Legacy project.

We advise that you upgrade your project to a new style project. This means that you have to create a new project for each Data Backbone and transfer this Data Backbone to the new project along with all of its content.

**Legacy projects**

Legacy projects had the option to mark folders as "does not contain ITP documents." This functionality is implemented in new-style projects with the introduction of a special Documentation folder. (Only visible in the web version of KCM Designer.)

Legacy projects allowed you to configure include folders and the order in which they were used. New-style projects have a single predefined Includes folder, simplifying its use.

Legacy projects allowed you to use Shared projects. New-style projects are intended to be self-contained and do not support Shared projects.

## Roles and authorization

To be able to see and work on specific objects in KCM Repository, a user is assigned a certain role within the project.

Every role requires knowledge of how to perform certain tasks and permission to do so.

When a user has no authorization on a folder or object, the object is hidden for the user. To learn more on administrative tasks that include tasks to create roles, add permissions, and assign roles to users, see the *Kofax Communications Manager Repository Administrator's Guide*.

### Predefined roles

KCM Repository comes with a set of predefined roles:

- **Author**. A user with the Author role can create, edit, publish, and delete most of the content objects in Designer for Windows, with a few exceptions. For example, an Author can create and edit Text Blocks but cannot perform these actions on Master Templates, Libraries, Includes, Data Backbone, brands, as well as on Style Sheets and Page Style documents for brands.
- **Content Reviewer**. A user with the Content Reviewer role has the ability to accept or reject changes to objects.
- **Publisher**. A user with the Publisher role has the ability to publish objects and remove the published status of an object.
- **Publishing Author**. A user with the Publishing Author role is granted permissions of both an Author and a Publisher.
- **Publishing Reviewer**. A user with the Publishing Reviewer role is granted permissions of both a Content Reviewer and a Publisher.

- **Viewer**. A user with the Viewer role can view most of the objects but cannot change them.
- **Project Administrator**. A user with the Project Administrator role is granted all permissions mentioned here. A Project Administrator has the ability to develop templates as well as create and modify Master Templates and Data Backbone. Also, Project Administrator manages projects, folders, reports, and other resources.

In general, the following rules apply to all permissions:

- Roles and corresponding permissions can only be granted to users through the assignment of roles
- No permission is granted by default

Chapter 2

# Work with KCM Repository

This chapter presents KCM Repository in its basic form and provides information on what is stored in KCM Repository and how it is involved in the development of templates.

## CM Repository as a file system

In general, KCM Repository is a file system. It maintains a database of documents and other objects organized in folders. Projects, folders, documents, and other objects are analogous to an ordinary file system. In addition, KCM Repository offers version control and dependency tracking functionality.

KCM Repository stores the following objects:

- **Documents**
  KCM Repository can store documents in different formats, such as TXT, DOC, and DOCX. KCM Repository has integrated support for opening and editing such files. For Master Templates, which are also ordinary text processing files, KCM Repository has additional support for compiling and running.
- **Master Templates**
  A Master Template is a program responsible for the document creation. During the Master Template composition process, Master Template scripts are processed.
- **Text Blocks**
  A Text Block is a meaningful piece of text with some variables and it can be reused for different purposes. You can use Text Blocks in a Master Template Document and in Content Wizards.
- **Forms**
  Forms are a means to get information, to store it in Fields in Data Backbone, and to use it to produce documents.
- **Text Block Lists**
  A Text Block List is a collection of Text Blocks. Text Block Lists are used in Forms and in Text Block selection groups of Content Wizards.
- **Content Wizards**
  Content Wizards describe the document structure. They hold sections and Text Blocks.
- **Field Sets**
  A Field Set is a collection of Fields used to contain data.
- **DIDs**
  The DID file describes how the data is accessed. It is used during the Master Template creation process.
- **Data Backbone**
  Data Backbone describes the structure of the data and the data fields that can be used in the documents. The Data Backbone is used in the content.
- **Style documents**
  Style documents describe the layout of a document.
- **Folders**
  Objects in KCM Repository are organized in folders. There are a few different types of folders.
- **Projects**
  Projects contain the most important settings for creating and running templates. Project groups can be used to group projects.

## Projects and folders

All objects in KCM Repository are organized in projects and folders. Projects can be grouped in project groups.

### Projects and project groups

The work that you do in KCM Repository is organized in projects. In the KCM Repository structure, projects are organized in folders, containing all documents and objects related to that project. A project also contains configuration information.

You can group projects into project groups. Project groups are the top level folders in KCM Repository structure. You can navigate to a project through the project group it belongs to. You can add and remove projects from a project group using the project Configuration function.

For more information on configuration, see Configure contents.

**Folders**

Folders are lower level objects that can hold any KCM Repository object, such as other folders, documents, Text Blocks, and so on. To distinguish between folders that contain these different objects, folders are named, depending on the objects that they hold. For example, "Text Blocks."

You can give names to objects that you create. The maximum length of a name is 254 characters. The system removes trailing dots (.) and leading spaces in the name. Field Sets are subject to the naming rules of the Template scripting language, so they start with a capital letter and they can contain letters, digits, and underscores. Newly created objects are always locked at first, which is indicated by a lock sign. If you are the one who created the object, your user name appears between brackets next to the name of the object. Other users can only view a locked object, but they cannot edit it.

## Create projects

A project is the starting point of all work in KCM Repository. If you are the first user of KCM Repository, it probably does not yet contain any projects. If so, at least one new project has to be created. This section is for users whose role is to create and configure projects.

A user must be assigned the roles of Project Administrator to be able to configure or create projects. For more information, see Roles and authorization.

To create a project, click the folder Projects and then click **File** > **New Project**. In the dialog that appears, select the document type and the scripting language of the new project. The document type determines if the project contains MS Word .doc or MS Word .docx documents. The scripting language determines if the keywords in templates are English, Dutch, or German. You cannot change these settings once the project has been created.

When a project is created, a number of the predefined folders are automatically added to the project. These folders are used to store the objects after which they are named.

After a project is created, the Configuration dialog for the project opens. You must specify a number of project settings before you can create and run Master Templates. To configure the project, see Configure contents.

**Tip** You can create an unconfigured project if you do not need to create and run Master Templates. This allows you to store documents in this project, without adjusting the project settings. Such projects are marked as [unconfigured].

## View and edit objects

You can open objects in KCM Repository in two ways -- for editing or only viewing. When you open an object, the corresponding editor is opened in view or edit mode. For the different types of objects, different editors are integrated in KCM Repository. For documents, Microsoft Word is used. For all other objects, dedicated editors are integrated.

Once you start editing an object, a new revision of that object is created and the object becomes locked by you.

## Lifecycle management

This section describes the lifecycle management that KCM Repository provides for content.

Some revisions have a special status. Those revisions have a revision icon and a status text.

Revisions are part of the lifecycle management. The lifecycle of an object starts when you create a new object, which automatically gets the status [in development] when you close and save it. When you unlock this object, it gets the status [draft]. If a revision is tested and meets the requirements, it is promoted to the status [accepted]. To make an object available for the production run, it has to be promoted to the status [published].

**[in development]**

At most one revision of an object has the status [in development]. An [in development] revision exists if a user has locked an object either for editing, for template creation, or if a DID has just been loaded. The [in development] revision is the working non-public copy of the user who has locked the object.

**[draft]**

There is always just one [draft] revision for an object. It is possible to have no [draft] revision when a new object has never been unlocked. In that case, there is a [in development] revision, but no [draft] revision. The [draft] revision can be regarded as the most recent public version of the object. When an object gets unlocked, the last saved working copy automatically loses its [in development] status and becomes the [draft] revision.

**[accepted]**

The marking [accepted] is only available for run-time objects, that is, for Master Templates and dynamic objects (see About Master Templates). At most a single revision of a run-time object may be marked as [accepted]. This may also be the [draft] revision so a Master Template script revision may be both [draft] and [accepted]. Failed objects and [in development] revisions cannot be made [accepted].

**[published]**

The status [published] reflects that a revision is used in production runs. The marking [published] is only available for run-time objects. Only a revision marked as [accepted] can be marked as [published]. After marking a revision as [published], its previous [accepted] marking is automatically removed.

## Use revisions of objects

KCM Repository keeps track of all versions of your objects, also known as revisions. Whenever you start editing a file, you do not edit the original file, but its copy. Once you unlock the file, the original file does not get overwritten, but your edited copy gets placed next to it, as a new revision.

If you want to see the revisions of an object, on the menu, click **View**, select **Show Details**, and then expand the tree view of that particular item.

Revisions are shown under the object they belong to.

A revision is identified by a revision number, and in the tree view they are ordered with the most recent revision on top. In the content pane, you can order revisions in other ways by clicking on the headers of the shown list. You can double-click a revision to view it.

### Draft revisions

There is always just one [draft] revision for an object. It is possible to have no [draft] revision when a new object has never been unlocked. In that case, there is a [in development] revision, but no [draft] revision. The [draft] revision can be regarded as the most recent public version of the object. When an object gets unlocked, the last saved working copy automatically loses its [in development] status and becomes the [draft] revision.

Normally, you edit and unlock the latest version of an object so that it becomes the new [draft] revision. However, there may be situations when you cannot use the latest version of an object. This may happen if the latest version has a bug or when the [draft] revision contains some prototype code that is not developed further.

In that case, you can make an earlier revision the [draft] revision. To do so, right-click this earlier revision and click **Make draft**. Then if you edit the object, the new development revision will be based on the earlier revision that is now marked as [draft].

### Revision numbering

When a file is revised, it is assigned a new version number. The number consists of a major and minor version number in the format <major>.<minor> with a period (.) as the separator. The first version of a file is 1.0. When the file is revised, the minor version number is incremented to 1.1, 1.2, and so on.

The major number is incremented when the entire project is revised. The major version number of individual files cannot be incremented.

Use the New Major action to create a new version of the project. The New Major action applies to all new versions. The current version of the project and each file is copied, labeled with the next major version number, such as 2.0, and made current.

Follow these steps to create a new major version:

1. Make sure there are no locked files in the project. If there are, unlock them.
2. Select the project you want to assign.
3. Click **Action** > **New Major**.

### Compare revisions

To compare revisions, follow these steps:

1. Select the revisions you want to compare.
2. Right-click the revision and select **Compare**.
3. Select the version you want to compare.

You can also compare the [in development] revision of the selected document with the [draft] revision to see the changes that you made. This action is only available for the user who locked the document.

1. Select a document or a Rich Text Block.
2. Click **View my changes**.
   The result is a document showing the differences underlined and in a contrasting font color.

#### Compare document revisions

KCM Repository offers the possibility to compare two revisions of a document.

A revision can be compared with another revision when it is opened in the ITP/Model Development Kit or it can be compared with another revision from within KCM Repository.

1. When a document is opened in the ITP/Model Development Kit, on the menu, click **Edit** and then click **Compare**.

#### Compare with local file

A document revision can be compared with a file on your local file system. You can use this functionality to compare a revision in KCM Repository with a revision on the local file system that has been altered to fix a problem.

1. Select the revision, click **Edit** on the menu and then click **Compare**.
   If a document is selected and not a specific revision, the [draft] revision is used to compare with.

## Restore deleted items

A deleted object can be restored.

If an object exists with the same name as the object you attempt to restore, the restore process fails.
If you restore a folder containing multiple objects with names that already exist, the restore process is successful, but a dialog listing all the objects that cause conflicts appears. You have to eliminate these name conflicts before you start promoting objects to other statuses.

1. On the menu, click **View** > **Show deleted items**.
2. Click the item to restore and on the menu click **File** > **Restore**.

Restored objects have the [in development] status or no status. You can promote restored objects to any status, but first check the dependency. To do so, right-click the object and click **Compute Dependencies**.

## Label objects

Use labels to identify revisions of a document, such as a milestone release like "major release," the type of component, such as "Form," or the type of source, such as "Master Template." You can then specify a label to obtain specific versions.

You can assign labels on several levels, labeling different objects:

- Project level, labeling the [draft] revision of every object in the project
- Folder level, labeling the [draft] revision of every object in the folder
- Object level, labeling the [draft] revision
- Revision level, labeling a specific revision of an object

When a label is applied to a project, all shared projects used in that project are also labeled.

### Assign a label to an object

A label can have a maximum length of 49 characters.

All information available in KCM Repository about the labels used in a project can be gathered in one document. To do so, select a project and choose **Label Listing** on the **Report** menu. This procedure runs a Master Template that will collect all information and present it in a Microsoft Word document.

**Note** Labels are not supported in the web version of KCM Designer. We recommend that you use Changesets instead of labels for managing changes to multiple objects.

1. In the tree view, click the object to assign a label to and on the menu click **File** > **Assign Label**.
2. Type a new label or select an existing label and click **OK**.
   You will receive a notification that the label is assigned.

## Assign characteristics to objects

You can use characteristics to identify and group objects in KCM Repository. Once characteristics are defined in the projects level Configuration window, they can be assigned to the lower level objects.

You can search objects using their characteristics with the search function available on the toolbar. For information on how to define and configure characteristics for an object, see Projects.

**Note** Characteristics are not supported in the web version of KCM Designer.

## Create copies of objects

You can use this functionality to copy objects to another location.

Consider the following when copying objects:

- Only the [draft] revision of objects with revisions is copied.
- Locks are not copied, except when copying a revision directly.

- The revision number of a copied revision is set to x.0, where x is the current major revision number of the project where the copied revision is pasted. For more information on revision numbering, see Revision numbering.
- The copied object gets the authorization of the folder it is copied to.
- An object can only be copied where it would also be allowed to create a new object of that type. This includes authorization.
- Copies of folders contain a copy of the complete content in these folders, with the exception of templates and objects that have no current revision.
- Usage information and Includes information is not valid after a copy. The copied objects are marked with [Dependency info outdated]. To refresh the information, right-click the object and click **Compute Dependencies**.
- Labels are only copied when an entire project is duplicated.

### Copy a revision of an object

1. To copy a single revision of an object, right-click the revision in a project and click **Copy**.
2. Select an appropriate folder and paste your copied revision into this folder.

KCM Repository creates a new base object to hold the revision. This object is locked by the current user, and the copied revision is the in-development revision.

**Note** Single revision copy is the only way to copy an [in development] revision and the only type of copy that keeps a lock on an object.

### Create a duplicate of a project

You can create a duplicate of a project to transfer it to another KCM Repository. A duplicate of a project is a complete, faithful copy. It includes all configuration settings, revisions, and objects in the project. Also, objects that are locked remain locked in the duplicate.

To make a duplicate of a project and its data, right-click the project and click **Duplicate**.

### Create a copy of a project

To prevent documents that are in production from breaking, KCM Designer explicitly forbids changes to the Data Backbone or a Library that would cause existing documents to fail. It is not possible to remove a Field Set from a Data Backbone after an [accepted] revision is created for the Data Backbone. It is not possible to remove a FORMAT function from a Library.

This restriction may cause problems when you develop documents for a new version of the business application, as the data in that application may also change.

In this case, create a copy of the project for the new version of the application. This copy only contains the revisions that were [draft] in the source project, and they will be [in development] in the new project. This allows you to change Libraries and the Data Backbone without putting at risk the documents that are in production.

To create such a copy, in the tree view, click the project and then click **File** > **Restart Project** on the menu.

The new project has the same name as the original, with [1] appended to it. You need to rename it to give it a meaningful name.

## Dependencies between different objects

Revisions, Master Templates and Includes, DIDs, and other objects are dependent in various ways on each other. To view these dependencies, on the menu, click **View** > **Show Details**. After that, the dependency information is shown below the objects involved.

KCM Repository adjusts the dependency information whenever needed. However, there are situations when the information is not present or incomplete. KCM Repository lets you know when the dependency information it holds for a project, folder, or object (revision) is out of date by displaying a clock in the icon and by putting [Dependency info outdated] next to the name. To refresh all dependency information, right-click the project or object and click **Compute Dependencies**.

### Document revision dependencies

Documents have different dependencies, depending on their role in KCM Repository. Only dependencies that are relevant for a document are shown. For example, "The "included by" dependency is only shown on an Include; the "stylesheet of" dependency is only shown on a Style Document.

#### Derived Revisions folder

Each document revision has a folder "Derived Revisions" that shows which revisions are directly based on it:

- When a Document Revisions folder is selected in the tree view, the information for all revisions of a document is shown in the content pane on the right.
- When the Derived Revisions folder of a specific revision is selected, the revisions based on that revision are shown in the content pane.

#### Includes folder

The Includes folder of Master Template document revisions contains all documents included in this revision. It does not refer to revisions because an Includes statement inside a Master Template script does not refer explicitly to a particular revision. Whenever this revision of the template gets compiled, it includes the revision of the indicated document that is [draft] or [in development] at that moment.

#### "Included by" folder

Each Include has an "included by" subfolder. It contains all documents that directly include this source. To view which revision of the document includes this source, click to expand the document in the tree view.

#### Derived Models folder

For each document revision, the Derived Models folder lists all Master Template revisions that have been created with it. To view the specific template revisions created with a document revision, click to expand the Master Template inside the Derived Models folder.

#### "Stylesheet of" folder

Each Style Sheet revision comes with a "Stylesheet of" folder that lists the projects and folders in which the Style Sheet revision is configured.

## Master Template revision dependencies

**"Created with" folder**

For each Master Template revision, the "Created with" folder lists all documents, the DID, and Data Backbone used to create it. Also, the version of KCM used to create the Master Template script is shown here. If the "Show details" option is checked, the document revisions are shown.

**"Uses" folder**

This folder lists the following information:

- Text Blocks inserted in the Master Template with the instruction `TEXTBLOCK NAME`.
- Forms inserted in the Master Template script with the instruction `FORM NAME`.
- Content Wizards inserted in the Master Template script with the instruction `WIZARD NAME`.
- Content Wizard folders inserted in the Master Template script with the instruction `WIZARD FOLDER`.
- Views (Text Block Lists) used in the Master Template.

## DID revision dependencies

**"derived models" folder**

For each DID revision, the "derived models" folder lists all Master Templates created with this DID revision. This folder is only shown when the "Show details" option is derived.

**"configured in" folder**

Each DID revision comes with a "configured in" folder that lists the projects and folders in which the DID revision is configured.

**"DID entries" folder**

This folder lists all the DID entries used by the Master Template.

## Text Block folder dependencies

**"used in" Text Block List revision**

This folder lists the Text Block Lists that are an alias of this folder.

## Form dependencies

**"used in" folder**

This folder lists the Master Template revisions that use this Form. It lists only those Master Templates that insert the Forms using the `FORM NAME` instruction.

## Form revision dependencies

**"uses" folder**

Lists the Fields Sets, Fields, and Text Block Lists that the Form revision uses.

### Field Set dependencies

**"used in" folder**

Lists the Form revisions, Text Block revisions, and Content Wizard revisions that use this Field Set.

### Text Block Lists dependencies

**"used in" folder**

Lists the Master Template revisions, Forms, and Content Wizard revisions that use this Text Block List.

## About Master Templates

Master Templates are compiled Master Template documents. Before you can work with Master Templates, you need to configure the project.

**Local configuration**

To run a template, you can use KCM Core or KCM ComposerUI. For information on how to use KCM Core to run templates, see the *Kofax Communications Manager Core Developer's Guide*.

**Creating a Master Template**

To compile any document in KCM Repository, right-click the document and click **Create Master Template**.

Once you created a Master Template, you automatically get a lock for it, and it becomes unavailable for editing by other users.

When you are ready to publish the Master Template, unlock it so others can start using it.

**Testing a Master Template**

Once a Master Template has been successfully created, you can test it. Right-click it and click **Test**. This action appears in grey if the Master Template has not yet been successfully created.

Testing means that you are running your working copy of the template. This allows you to test various iterations of your template without affecting other users' work. Other users can only run the original public version of the template, provided that it exists.

**Dynamic objects**

Dynamic objects are used to create content. They can be changed rapidly and easily by business users. The changed content is directly available for document creation, without needing time-consuming and complicated compilation or deployment steps, as they are retrieved by the template at run time.

Dynamic objects are objects that are retrieved at runtime when composing a document. Dynamic objects include Forms, Text Blocks, Text Block Lists, Data Backbone, Libraries, Master Templates, Content Wizards, Style Sheets, and Page Styles.

**Running a Master Template with dynamic objects**

You do not need to recreate a Master Template when a dynamic object is changed. When a Master Template is run, it collects the dynamic objects directly from KCM Repository. If an object has no [published] revision, it is not returned. Therefore, marking a dynamic object revision as [published] indicates that the revision is accepted to be used when the Master Template is run.

If the KCM Repository installation is upgraded from a version older than 3.5.10, by default, the compatibility option "Handle Accepted revisions as Published" is turned on. In this case, the status [accepted] takes over the role of the [published] status.

When a Master Template is run from within KCM Repository, it retrieves the [in development] version of the dynamic object if such a status is present and locked by the user who runs the Master Template. If not, it retrieves the [current] revision.

## Use auto-includes

You can configure projects, folders, and documents with an auto-include. An auto-include is an Include that is automatically included at the beginning or end in Master Templates.

Master Template scripts with auto-includes do not need an instruction `#BEGIN…END#`.

To use auto-includes, configure the project accordingly:

1. In the tree view, right-click a project, folder, or a document for which you want to configure Includes.
2. Right-click it and click **Configuration**.
3. Select the **Auto-includes** tab.

   Select **Use auto-includes** and select Includes to add and to end the Master Templates of this project, folder, or document.

   Select **Hide Auto-includes in Error Document** to make an error document shorter and more readable.

When a Master Template is created, the following Master Template script is composed.

```
#BEGIN
Contents of auto-include configured on the folder as 'Add to end'
Contents of auto-include configured on the project as 'Add to end'
END#
```

## Search paths

When a Master Template script is run, KCM Core retrieves Text Blocks, Forms, Text Block Lists, and Field Sets from KCM Repository. The system searches for these run-time objects based on their names. The following sections describe rules that apply to the search.

### Search on the project level

All objects reside in a project. KCM Repository determines the project in which it looks for an object in a specific way.

**Specification in insert statement**

If the object is referred in the Master Template with a name that includes the project, the object is searched for in that project. You can include the project in a reference by adding the project name and a period in front of the actual name:

```
TEXTBLOCK NAME "MyProject.my Text block"
FORM NAME "Questionnaires.customers\order form"
```

#### CM Core setting

For objects that do not directly refer to a project, the RepositoryProject setting of the appropriate environment in the KCM Core configuration is used. This setting is meant as an override.

#### Master Template location

If none of the above applies, KCM Repository looks for the objects in the same project that also holds the Master Templates. This is the normal situation.

### Search on the folder level

Except for Field Sets, objects with the same name may reside in different folders. A folder name can also be used when referring to such an object. But a folder specification never includes top-level folder for these objects, as it is implied.

For example, to insert a Text Block that resides in a certain Text Block folder such as "My Folder," you do not need to mention the top-level Text Block folder name such as "Text Blocks" in the `TEXTBLOCK NAME` statement: "My Folder" is sufficient.

#### Absolute path

When inserting an object, you can specify an absolute location placing a backslash in front of the name or path of the object. The backslash comes after a possible project specification.

```
TEXTBLOCK NAME "\my Text block"
TEXTBLOCK NAME "\folder\another Text block"
FORM NAME "Questionnaires.\customers\order form"
```

Objects specified in this way are searched only and in that exact location.

#### Relative path

If an object specification does not start with a backslash, KCM Repository looks for the object in a number of folders that were specified during project configuration. Together, these folders form a search path. By default, the search paths only contain the top-level folder for the object, such as Text Blocks, Forms, and so on.

Text Blocks, Forms, and Text Block Lists have their own search paths. To configure their paths, see Dynamic objects.

Field Sets do not need search paths, as the name of a Field Set is unique within a project.

KCM Repository looks for the object in each folder in the order that they are given in the search path. If the object reference also includes folder names, they are considered to be subfolders of the folder in the search path. Searching ends when an object is found with the proper name and an applicable revision. This object is returned and used.

When a folder is added to the search path, only that folder itself is searched. Subfolders are not included in the search. You can either add the subfolders to the search path or add the subfolder name to the insert statement.

**Example** Assume that "folder" is present in the search path, but its subfolder "subfolder" is not present. The Text Block "text block" resides in "subfolder." This Text Block will not be found if it is inserted with `TEXTBLOCK NAME "text block"`, as "subfolder" is not searched. However, if you insert the Text Block with `TEXTBLOCK NAME "subfolder\text block"`, it will be found.

**Applicable revisions**

KCM Repository not only looks for an object with the proper name, but also with an applicable revision. The first object on the search path with an applicable revision is returned. For Master Template run outside of KCM Repository, the object must have a [published] revision to be found.

For example, assume that you have a Text Block "Customer Details" both in the Text Block folder "folder1" and in Text Block folder "folder2," where "folder1" precedes "folder2" in the search path. A Master Template looking for the Text Block "Customer Details" gets the Text Block residing in "folder1," provided that this Text Block has a [published] revision. If not, searching will continue to "folder2," and the Text Block residing there will be retrieved, but only if it has a [published] revision.

Master Template runs started from within KCM Repository (test runs) only require a [draft] revision to exist for an object to be found, or an [in development] revision for objects locked by the user running the Master Template.

**Shared projects**

When looking for the object in the search path of the project does not succeed, KCM Repository continues looking for it in the Shared projects in the order they are configured in the project and uses the search path of the shared project to find the object.

## Refresh usage information

Changing the search paths invalidates usage information for the entire project.

If this information becomes outdated, revisions are marked with [Dependency info outdated] and a clock sign. This signifies that the information in the "uses" folder is not accurate anymore. To refresh the information, right-click the project that contains the revisions and click **Compute Dependencies**.

Chapter 3

# Develop templates and content

When you are new to KCM, you probably have a set of documents you want to convert to templates. This chapter describes how to start implementing these documents in KCM.

## Getting started

Before you can start implementing templates in KCM, you have to analyze the documents that you need to create in order to benefit fully from the features that KCM offers.

**Step 1**: Identify the data requirements

In this step, determine how the data should be offered in the documents. Starting point is that the data needs to be prepared for the documents so that it can be used in a simple way.

The application data used in documents differs in various documents. To create the connection to the data, you need to know which data is required. The goal is to define the Data Backbone, the interface between the business application data and the templates data.

For more information on creating Data Backbone, see Work with data.

**Step 2**: Identify shared content

KCM allows you to benefit from sharing parts of documents -- you can reuse content components that are common to a group of documents. Sharing is important because it increases the maintainability of documents. That means that you have to analyze your documents to determine which parts can be shared.

Some reusable content is reused within the documents that use the same Data Definition, some reusable content is used by all documents. The first content is just created as components within a KCM Repository project, the second content is created in a KCM Repository shared project.

For more information on how to share content, see Work with content.

**Step 3**: Apply corporate style

The previous steps concentrate on what documents share in content and data. This step concentrates on what they share in look and feel.

Decide on the corporate style that has to be applied to all of the documents. Examine the fonts that have to be used, the company logo and its position on the page, headers and footers, margins of the documents, and so on. Find characteristics that apply to every document. Find characteristics that apply to the subsets of documents you created in the first step. Decide which styles you need and the characteristics of the styles.

Microsoft Word styles play an important role in KCM documents. We advise that you use styles as much as possible to lay out documents. Consistent use of styles guarantees a consistent layout of the result documents. Furthermore, consistent use of styles speeds up the maintenance of the document and can prevent unexpected layout of the result document.

For more information on using and creating styles in Microsoft Word, see the Microsoft Word documentation.

For more information on how to work with document layout in KCM, see Work with document layout.

**Step 4**: Setting up the KCM Repository structure

Determine how you want to organize your content. Develop a structure of folders. Content that uses the same Data Backbone belongs together in a project. Start with ordering documents using the same Data Backbone in sets of documents that belong together, department-wise, function wise, or for another reason. The idea is to design a structure that is simple enough to be usable and complex enough to allow for future growth.

**Step 5**: Develop the building blocks

In this phase, you are going to produce the projects/folders and building blocks you identified in Steps 1, 2, and 3, and test them.

**Step 6**: Develop the style

In this phase, you are going to produce the Style Documents you need and test them.

**Step 7**: Develop templates and content

With the structure of projects, folders, Includes folders, Includes, Style projects, and Style Documents in place, you can start developing your templates and content or import existing documents and adapt them to the new structure.

## Work with data

When you have analyzed the data requirements of the documents, create a Data Definition the describes the structure of the data as it is organized in XML format. The schema used by the data is called the Data Backbone. You can export the Data Backbone as an XSD file. Once the Data Definition has been created, an XSD export can be made from the Data Backbone. This XSD file describes the data XML that the application delivers needs to comply to.

The following section describes how the Data Backbone is created and used in KCM Repository.

### Use Data Backbone

The Data Backbone of a project is always available in Master Templates, Text Blocks, Forms, and Content Wizards.

Every Content Wizard has the Data Backbone pane. This pane gives the Content Wizard Editor information on the structure of the data and Fields that are available on a certain level in the Data Backbone. In the Content Wizard, you can drag the Data Backbone elements to the Content Wizard pane and combine them with Text Blocks. If the Data Backbone element is repetitive, all Text Blocks on that location are repeated in the result document. The repetitions are dependent on the number of occurrences of the Data Backbone element.

The Text Blocks used in a Content Wizard that uses a Data Backbone may also include Fields to add variable data to the document. If such a Text Block is added to a repetitive part of the Content Wizard, KCM replaces these Fields with the correct data at the moment the result document is composed. For more information on working with Content Wizards and Data Backbones, see Content Wizards.

## Data Backbone components

The Data Backbone is stored in the project so that it is always present in the project. Creating a Data Backbone means creating the Data Definition and creating the Data Retrieval.

**Data Definition**

The Data Definition component of a Data Backbone describes the structure of the data and the Fields used in the structure. When adding a Field Set to the Data Backbone, verify that the Field Set has already been created.

You can view the structure of Data Backbone and the Fields that can be used in the Data Backbone Viewer when it has been compiled with the action Create Data Backbone. This view is also available in the Content Wizard Editor.

It is possible to create an XSD from the Data Backbone once a Data Definition has been added. This .xsd described the structure of the data XML as it is needed by the Master Templates.

**Data Retrieval**

The Data Retrieval component of the Data Backbone describes how values are added to the elements of the Data Backbone.

In this component, all elements in the Data Definition are assigned to a value. This can be a value from a Field in the DID, an answer to a question, or a fixed value. Adding a value to an element of a Data Definition is done with the already existing keyword `ASSIGN`.

There is a functionality to easily assign values from Fields in a repeating entry to an element in the Data Definition. For information on this functionality, see the section "FIELDSET" in the *Kofax Communications Manager Template Scripting Language Developer's Guide*.

## Edit and view Data Backbone

1. Select the Data Definition component you want to edit or view.
2. Right-click the component and click **Edit Data Definition** or **View Data Definition**.
3. If you change or create a data component, create a new version of the object. Right-click **Data Backbone** and click **Create Data Backbone**.
   A new version of the object is created.
4. Optionally, export an XSD file of the updated Data Backbone. In the Data Backbone Viewer, select **File** > **Export XSD**.

#### Outdated revisions of a Content Wizard

A Content Wizard or Content Wizard revision is outdated if it is based on a Data Backbone revision that is changed and does not exist anymore. In other words, the outdated [in development] Content Wizard revision is based on obsolete data and should be reviewed. Saving the Content Wizard brings it up to date again, reflecting the changes to the Data Backbone revision.

Deleting, renaming, changing, or moving the elements of the types Fieldset Array, Data Structure, or Data Structure Array in a Data Definition component of the Data Backbone makes the Content Wizards based on this Data Backbone outdated.

#### Data Backbone revision dependencies

For each Data Backbone revision, the Derived Models folder lists all templates created with this particular Data Backbone revision. This folder is only shown when the "Show details" option is checked.

### Use DIDs

A DID is a description of a database. Documents created with KCM use data from a data source. Usually, a data XML is sent to KCM to create documents. In cases where the data needs to be retrieved directly from the database by KCM, you can use DIDs.

The data that is retrieved from the database using DIDs is transformed to document data with a Data Backbone. Data Backbones are used to describe the data in such a way that they can conveniently be used in documents.

A project always has exactly one Data Backbone. That means that all content in this project share the same Data Backbone.

The DID object resides in the predefined project folder DIDs. Wizards exist to easily create a DID. When editing a DID object, the DID help is available.

**Note** To create DIDs in the KCM Repository, you need a DID development (SDK/MP) license.

For more information on DIDs, see the *Kofax Communications Manager DID Developer's Guide*.

## Work with content

KCM Repository is designed to develop and maintain Master Templates and building blocks used in these Master Templates. The following sections explains how to develop Master Templates. It also provides information on the building blocks that exist and how they can be used in the Master Templates.

### Building blocks

Building blocks are reusable fragments in Master Templates. You can use the following building blocks in a Master Template:

- Data Backbone

- Includes
- Text Blocks
- Rich Text Blocks
- Forms
- Content Wizards
- Style Documents

## Master Templates and Master Template scripts

A Master Template script is a Microsoft Word file containing the text, scripting language instructions, and layout that is used to create the Master Template.

Master Templates are used by KCM Core to produce the result document. A Master Template is an object that contains non-human readable content, which can only be interpreted by KCM Core.

A Master Template in itself cannot be edited. When a Master Template is opened to be edited, the associated Master Template script is opened.

- To create a Master Template from a Master Template script, in the tree view, right-click the Template folder and click **Create Master Template**.
- To edit a Master Template, right-click it and click **Edit**.
  The associated Master Template script is opened and locked.

### Master Templates and their revisions get outdated

A **Master Template** can be outdated for two reasons:

- If it contains an [in development] Master Template revision based on a Master Template script set, usually a Master Template script revision and one or more Includes revisions, of which one ore more do not exist anymore because they have been changed.
- If it is based on a Data Backbone revision that does not exist anymore because it has been changed.

A **Master Template revision** can be outdated for two reasons:

- If it is based on a Master Template script set, usually a Master Template script revision and one or more Includes revisions, of which one or more revisions do not exist anymore because they have been changed.
- If it is based on a Data Backbone revision that does not exist anymore because it has been changed.

**Note** Not all changes in a Data Backbone result in outdated Master Templates. Additions to the Data Definition component of a Data Backbone can be made without consequences.

Creating the Master Template revision makes it up-to-date again, reflecting the changes to the Data Backbone revision or the document revisions.

The outdated [in development] Master Template revision cannot be recreated exactly as it is in the current state.

If KCM encounters errors during Master Template creation, an error message is generated and KCM Repository is opened showing a copy of the Master Template script with error messages marked in red. A [in development, failed] revision is added to show that the revision is not a valid Master Template.

When an [in development, failed] revision is unlocked, it is changed into a proper revision but that revision is not set as the [draft] revision.

## Text Blocks

A Text Block contains plain text and variable Fields that insert unformatted text strings. Text Blocks are created and maintained in their own editor, the Text Block Editor. Layout possibilities are limited. When a Text Block is included in the document, its text is printed in the final document.

A Text Block can, like any building block, be used in more than one Master Template.

Text Blocks are used in Content Wizards but they can also be directly inserted in Master Templates using the instruction `TEXTBLOCK`. See the *Kofax Communications Manager Template Scripting Language Developer's Guide* for more information on this instruction.

Text Blocks that use variable Fields need to have Field Sets attached to their Text Block folder. A Field Set describes the available variable fields. Fields are defined in a Field Set. For information on Field Sets, see Field Sets. The Master Template developer assigns values to these variable Fields with the instruction `FIELDSET`. See the *Kofax Communications Manager Template Scripting Language Developer's Guide* for more information on this instruction.

Text Blocks are stored in the special Text Blocks folder. This folder is automatically created when you create a new project. You can create Text Blocks subfolders in this folder.

## Rich Text Blocks

A Rich Text Block offers the same functionality as a simple Text Block and all the layout possibilities of Microsoft Word. A Rich Text Block can, just like a simple Text Block, be included with the instruction `TEXTBLOCK`. See the *Kofax Communications Manager Template Scripting Language Developer's Guide* for more information on this instruction.

Rich Text Blocks which use variable Fields need to have Field Sets attached to their Text Block folder. For information on Field Sets, see Field Sets.

The use of headers and footers in Rich Text Blocks is not supported.

## Content Wizards

Content Wizards enable you to define large documents by assigning existing Text Blocks and Rich Text Blocks to subsections of the document. You need to define the Text Blocks and Rich Text Blocks first before selecting them in the Content Wizard.

It is possible to add user prompts to the Content Wizard. In that case, the end user needs to select subsections or Text Blocks to be part of the result document. In the Content Wizard Editor, these Text Blocks or Sections are either collected in a Selection Group or are defined as Optional. A Selection Group specifies whether the user can select multiple or only one of the Text Blocks or Sections. Text Blocks defined as Editable are presented to the end user in the Text Block Editor to allow changes to the Text Block.

QForms can be attached to various levels in the Content Wizard. For more information on the QForms, see QForms.

Content Wizards are stored in the folder Content Wizards. This folder is automatically created when you create a new project. You can create subfolders in this folder.

Content Wizards are inserted in a Master Template using the `WIZARD` instruction. The instruction `FOREACH WIZARD` is available to process the business user's selection from that Content Wizard. For more information, see the *Kofax Communications Manager Template Scripting Language Developer's Guide*.

### Requirements to use Content Wizards

To run Content Wizards, you need KCM ComposerUI.

### Content Wizard and their revisions get outdated

A **Content Wizard** can be outdated for two reasons

- If it has an [in development] revision based on a Data Backbone revision that has been changed and therefore does not exist anymore.
- If you delete, rename, or move the elements of the types Fieldset Array, Data Structure, or Data Structure Array in the Data Definition component of the Data Backbone that the Content Wizards are based on.

Additions to a Data Definition component do not result in outdated Content Wizards. Any changes to the Data Retrieval component of the Data Backbone also do not result in outdated Content Wizards.

When an outdated Content Wizard is unlocked, the outdated [in development] revision is changed into a proper revision.

A **Content Wizard revision** gets outdated for two reasons:

- If it is based on a Data Backbone revision that does not exist anymore, because it has been changed.
- Deleting, renaming, or moving elements of the types Fieldset Array, Data Structure, or Data Structure Array in the Data Definition component of a Data Backbone makes the Content Wizards based on this Data Backbone outdated.

Additions to a Data Definition component do not result in outdated Content Wizards. Any changes to the Data Retrieval component of the Data Backbone also do not result in outdated Content Wizards.

The outdated [in development] Content Wizard revision cannot be recreated exactly as it is in the current state. Save the Content Wizard to make it up-to-date again, reflecting the changes to the Data Backbone revisions.

When an outdated Content Wizard is unlocked, the outdated [in development] revision is changed into a proper revision.

## Includes

To reuse parts of a Master Template, you can use Includes.

Includes reside within Includes folders. Projects contain a predefined Includes folder. In Legacy Projects, you need to create and configure Include folders yourself.

Add Includes in another document using their name and extension and the `__INC` instruction. Documents inside subfolders of an Includes folder must be included with their folder as well.

**Example** `__INC(date functions.doc)` includes the Include with the name "date functions."

To prevent errors, you cannot rename Includes files once they are included.

If you want to include a document in a particular folder, check the order in which the Include paths are searched. The list in the project configuration is searched from top to bottom. If a document with the same name exists in one of the earlier searched paths, that document is found first and used.

You can also include documents in the Includes folders as an auto-include. To use auto-includes, configure the project accordingly. See Includes.

## Forms

A Form collects information from users during a Master Template run by means of a questions/answers form. It contains elements such as questions, instruction, and help texts. When a Form is included in the Master Template script, KCM ComposerUI prompts the user to answer the questions, and shows instructions and help texts.

Forms are created and maintained in the Form Editor.

Forms are stored in the Forms folder. This folder is automatically created when you create a new project.

Forms are included with the instruction `FORM NAME` or `FORM VAR`. For more information on this instruction, see the *Kofax Communications Manager Template Scripting Language Developer's Guide*.

The answers to questions are stored in Fields of Field Sets in the Data Backbone. In Legacy projects, it is possible to configure a Forms folder with a restriction on the Field Sets that can be used in Forms inside it.

A Text Block question allows a user to select one of more Text Blocks from the Text Block List to insert in the result document. In Legacy projects, it is possible to configure a Forms folder with a restriction on the Text Block Lists that can be used in Forms inside it.

It is possible to save an erroneous Form. The error message of the Form Editor is stored with the revision and shown in the description pane. The revision of an erroneous Form is tagged as failed. Such an erroneous Form revision cannot be unlocked.

## QForms

A QForm is a special Form. A QForm is meant to ask the user questions for Fields used in inserted Text Blocks, which do not have a value yet. The other questions are hidden. A QForm can therefore only be used when inserting a Text Block using the `TEXBLOCK` keyword or in a Content Wizard.

### Differences between Forms and QForms

It is easy to distinguish a QForm from a regular Form. When the QForm Editor is open, you can see the text "You are editing a QForm."

The distinction between a regular Form and a QForm is important because a QForm serves a different purpose. QForms are meant to enter values for Fields used in Text Blocks, which do not have a value yet when the Text Block is called in the Master Template script. For this reason the following features of the regular Form are not available or are different in the QForm:

- By default, validation of questions in a QForm is set to "A non-empty answer is required." This ensures that the user enters data for Fields used in the Text Block. If this is not desired, select "None can be selected" on the tab Validation of the question.
- You cannot insert Text Block Questions. When you use a Text Block Question, you generate new Text Blocks, which can in turn contain more empty Fields. Those should be checked again. To prevent a recursive circle from Text Block Questions to the QForm, you cannot select Text Block Questions.
- You cannot make a question read-only, because the user must always be able to fill in the Fields by providing an answer to the question on the QForm.
- The feature to show or hide groups of questions is not available. Questions are always shown when the assigned Field is empty, and is always hidden when they already have a value.
- As a consequence, you cannot enter a reference for a question. The function of references is to present or hide groups of questions in response to an answer to a previous question. In a QForm, questions are never hidden or shown conditionally, depending on a previous answer.
- You can only select a Field from a Field Set as answer for one question per Form. When a user wants to select a Field as value for an answer that is already in use for another question, the user receives a warning. It is possible to save an erroneous QForm. The error message of the QForm Editor is stored with the revision and shown in the description pane. The revision of an erroneous QForm is tagged as "failed," it cannot be made [accepted] and therefore also not promoted to [published].

## Field Sets

A Field Set defines a group of Fields associated with a Text Block or Forms Definition folder. They provide an interface from the Master Template script to Text Blocks, Forms, and Content Wizards.

In Legacy projects, you can connect Field Sets to a Text Block folder or to a Forms Definitions folder on the Field Sets tab of the project Configuration window.

The questions in the Forms can use the Fields from the Field Set to save the answer to those Fields. The user who creates or changes a Text Block can use these Fields in the Text Block. Content Wizards can use these Field Sets for conditions.

To use the Field Set in a Master Template with a Form, Text Block, or Content Wizard, first create the Field Set, then add the Field Set to the Data Backbone, and finally create the Data Backbone.

You can also use the Field Set interface in a Master Template by declaring a special `FIELDSET` variable with the name of the Field Set. See the *Kofax Communications Manager Template Scripting Language Developer's Guide* for more information on using the `FIELDSET` variable.

Also, you can use a special Field Set type, Status Field Set, in combination with Content Wizards. In a Content Wizard, you can optionally add Sections and Text Blocks as well as create editable Text Blocks. During composition of a document or a Document Pack, the user can interactively choose to include the Section or the Text Block and/or to change the content of the editable Text Block. Status Field Sets can be used to store information on the user's choice and to store the content of the changed editable Text Block.

You can indicate your intention to include and object or not in the Data Retrieval, by adding either Yes or No value to such a Field directly. The Status Field Sets are invisible in every other place where Field Sets could be used. Fields of the Status Field Set have no default value and format.

Field Sets are stored in the special Field Set folder. This folder is automatically created when you create a new project. You can create Field Set subfolders in this folder.

## Add a Field to a Field Set

1. Click the project and locate to the Field Sets folder in the tree view.
2. Right-click the Field Sets folder and click **New** > **New Field Set**.
   The new Field Set appears under the Field Sets folder. You can change the name of the Field Set.
3. Right-click the new Field Set and click **Edit**.
   The Field Set editor is opened.
4. Click **Add** to add a Field to the Field Set.
5. When the Field is added, click **Edit** to assign a default value, a description, and a format to the Field.
   The default value is used if the Field is used in a Text Block and no other value has been assigned to it yet. Default values of Fields are not shown for Fields used in Master Template scripts, Forms, or Content Wizards.
   The selected format determines how the content of the Field is presented (formatted) when that Field is used in a Text Block. The selected format is not applied to Fields used in Master Template scripts, Forms, or Content Wizards, but only to Fields used in Text Blocks.
   You can select the following formats. The result of the applied format is language dependent. This applies to the thousand separators and the date representation.

| Format | Description |
|---|---|
| none | No formatting is applied to the content of the Field. The content is presented as is. |
| date | Presents the content of the Field as a date. The day and the year are presented as numbers; the month is presented as the month's name. The Field must contain a numerical date value.<br>The result depends on the output language.<br>**Example** 20160621<br>**Results**<br>English US: June 21 2016<br>English UK: 21 June 2016<br>Dutch: 21 juni 2016 |
| amount | Rounds the numerical content of the Field to two decimals and adds thousand separators, if applicable. Use this format if the numerical content of the Field should be represented as money amounts. No currency signs or other additions are added. The Field must contain a numerical value.<br>**Example** 1234.56<br>**Result** 1,234.56 |

| | |
|---|---|
| numerals without format | Rounds the numerical content of the Field to zero decimal positions. No thousand separators are added. The Field must contain a numerical value.<br>**Example** 1234.56<br>**Result** 1235 |
| uppercase | Represents the text content of the Field in uppercase characters.<br>**Result** NEW YORK |
| lowercase | Represents the text content of the Field in lowercase characters.<br>**Result** new york |

**Note** The formats that expect the Field content to be a number (amount, numerals without format, and date) only provide output for the content up to the first non-numerical character. For example, if the Field contains the text "12.99 euro," the result after applying the amount format is "12.99" only.

The format you select here is linked to built-in functions of the KCM Template Script. To learn more on these functions, see the *Kofax Communications Manager Template Scripting Language Developer's Guide*. These linked functions cannot be overridden by user defined functions.

| **Format** | **Corresponding Template scripting language function** |
|---|---|
| date | date |
| amount | amount |
| numerals without format | numerals |
| uppercase | uppercase_of_characters |
| lowercase | lowercase_of_characters |

**Note** Selecting a format does not change the content of a Field. It only determines how the content of a Field is presented when that Field is used in a Text Block. This means that the selected format does not affect conditions used Forms or Content Wizards.

**Combine default value and format**

If both a default value and a format are specified on a Field and the default is used in the Text Block, then the Format is applied to the specified default. This means that the default value should be specified unformatted. In other words, the default value should be correct input for the applied format. For example, you should specify the default for a Field that represents a date and therefore has the Date format set in the ITP date format YYYYMMDD. For example, 20110621. When that default is then presented in a Text Block, it may be shown as June 21 2011.

6. Click **OK** and then click **Save and Close**.

## Add a Field Set to the Data Backbone

1. In the project structure, right-click **Data Backbone** and click **Edit Data Definition**.

2. Type the keyword `FIELDSET` and the name of the Field Set.
   **Example** `FIELDSET CompanyData`
3. Save the Data Definition object.
4. Right-click **Data Backbone** and click **Create Data Backbone** to create a new version of the object.

### Text Block Lists

A Text Block List is a selection of Text Blocks and Rich Text Blocks. Text Block Lists are used in KCM ComposerUI Forms in a Text Block Question. A Text Block Question allows a user to select one of more Text Blocks to insert in the result document. By default, all Text Block Lists are available to all Form folders.

Text Block Lists are stored in the Text Block Lists folder. This folder is automatically created when you create a new project.

### Data Backbone

Data Backbone provides the data to use in the content. For more information on Data Backbone, see Work with data.

## Work with document layout

In most cases, the documents that you produce with KCM need to have the same look and feel. For example, the company logo can be printed in the top right corner, the margins are set to certain values, a certain font is used for the text of a document, and so on. All these characteristics can be classified as the corporate style.

KCM offers support to define and maintain the layout definitions for the document separately from the content. Such a layout definition is called a Brand. A Brand contains Style Documents and Page Style documents.

A Style Document defines the styles that can be used in the documents. A Page Style document defines the margins and the section properties of the documents.

### Style Document

A Style Document contains Microsoft Word styles.

A Style Documents describes the styles used in the documents. When using Text Blocks, the Style Document should at least define the proper Text Block styles.

You can use a Style Document in two ways: you can configure it on a project and/or use to determine the layout of the result document at the moment the Master Template is run.

**A Style Document configured on a project**

When a Style Document is configured on a project, all documents that use styles defined in the configured Style Document use the style as it is defined in the Style Document. A Master Template in such a project also includes these style definitions.

For more information on how to configure Style Documents, see Style Documents.

The Style Documents folder for a newly created project already contains predefined Style Documents with the names "Text Block Style Document" and "Text Block Docx Style Document." The Style Document "Text Block Style Document" is initially used by Microsoft Word .doc projects; the Style Document "Text Block Docx Style Document" is used by Microsoft Word .docx projects. They define the default styles needed to lay out Text Blocks.

**Note** Although it is possible to configure a Style Document for a folder, we advise that you configure a Style Document on the project level only.

#### Use a Style Document when the Master Template is run

You can override style definitions when the Master Template is run. This is done using the multi-brand support: Style Documents are created for one or more brands and called when the Master Template is run.

The type of the Style Document should match the type of the other documents in the project. When no Style Documents are used, the styles as defined in the Master Template document are applied to the result document. You should use Style Documents because this guarantees consistent layout of your result documents. Style Documents also offer an easy way to manage styles across documents. Updating styles is faster and easier when using Style Documents.

#### Styles and protected documents

This only applies to Microsoft Word. If a project or folder is configured with a Style Document, KCM Repository attempts to apply the styles from it to a document within this project when it is opened for editing or viewing. This can only be done if the document has no section protection. If the document is protected, KCM Repository asks you whether to temporarily lift the protection in order to apply the styles. You may be prompted to provide a password.

## Page Style document

To set the margins and the section properties, use a Page Style Document. This Page Style Document is a normal word processor document in which the margins and section properties are set to the values that need to be set in the result document.

By default, the margins and the section settings defined in the Master Template are applied to the result document. You can use a Page Style document to obtain consistent layout of your result documents and to easily manage the layout across documents.

Page Style documents are defined in a Brand folder. Use Brands to manage the page style documents, even for organizations that only use one brand. For more information on creating style documents for specific brands, see the next section.

### Multi-brand support

1. To create a new Brand, click the project and on the menu, click **File** > **New** > **New Brand**. The Brand folder holds the Style Documents and Page Style Documents for this specific Brand.
   The new Brand folder appears under the Style Documents folder.

Style Documents and Page Style Documents can be applied by the Master Template, using the functions `pagestyle` and `stylesheet`. For information on these functions, see the *Kofax Communications Manager Template Scripting Language Developer's Guide*.

Chapter 4

# Configure contents

This chapter describes common configuration tasks that you can perform from the Configuration window in KCM Repository.

You can configure every object in KCM Repository. To do so, right-click any object in KCM Repository and then click **Configuration**. Which tabs you see depends on the selected object and on authorization. For more information on authorization, see Roles and authorization.

The Configuration window has its own Help, which describes in detail the settings on the various Configuration tabs.

## Projects

You can view and change general information on a project, create a group of projects, and specify an export alias for the project. Setting an export alias allows you to change the project name during export.

1. In the tree view, right-click the project and click **Configuration**.
2. Select the **General** tab.
   In the **Name** field, you can rename the object.
   In the **Description** field, you can add a description for this object.
   In the **Project Group** field, type the name for the new project group and then click **Apply**.
   In the **Export alias** field, type the project name that will be used during export.
3. In the **Involved in imports and exports** section, you can see in which imports and exports this object has been involved, if any.
   When an object is imported from another KCM Repository, this KCM Repository is listed as source KCM Repository.
   The source KCM Repository is listed with the company name of the license and the postfix as set by the administrator.
   Export and Import actions are listed by date and time. Double-clicking a import or export action to open a dialog that shows all objects involved in that export or import action. You can view and edit documents from this list.

   **Note** Opening this dialog may take long time.

4. To apply the changes, click **Apply** and then click **OK**.

Also, you can define characteristics on in the project level and then assign those characteristics to different folders/objects belonging in the project.

1. In the tree view, right-click the project and click **Configuration**.
2. Select the **Define Characteristics** tab.
3. Click **New** to add a characteristic and then click **Add**.
   You can also click **New Group** to define a set of characteristics.
4. Click **Apply**.
5. Select the **Characteristics** tab and check the characteristic that you want to apply to this project.
   You can also apply the characteristic to a folder/object belonging in this project on the Characteristics tab of the Configuration dialog for this folder/object.

## Dynamic objects

You can set an expiration date for a dynamic object so when it is used for document composition on or after that date, KCM Core reports a run-time error.

You can configure search paths for Text Blocks, Text Block List, and Forms. A search path lists a number of folders in which an object is searched for to run a Master Template.

To configure an expiration date:

1. In the tree view, right-click the dynamic object and click **Configuration**.
2. Select the **General** tab.
3. Check **Expires on** and in the drop-down calendar select the date on which the object will expire.
4. To apply the changes, click **Apply** and then click **OK**.

To configure a search path:

1. Select the **Runtime** tab.
2. In each section, add or remove a search path for the object.

   **Note** Changing the search paths on this tab invalidates the usage information in the project and you should recompute it. To do so, right click the project and click **Compute Dependencies**.

3. To apply the changes, click **Apply** and then click **OK**.

   **Important** Changing the search paths takes effect immediately for published objects. It affects the behavior of the production environment.

## Style Documents

A Style Document contains style elements that you can use to format result documents. You can link a Style Document to a project or a document folder.

1. In the tree view, select a project or a document folder to assign a Style Document to.
2. Right-click it and then click **Configuration**.
3. Select the **Style document** tab.
   In the **Style document** section, click **Browse** and select a Style Document to be used for this project/document folder.
   Click **Select** to select a version of the Style Document.
4. To clear all check boxes, click **Clear**. To apply the changes, click **Apply** and then click **OK**.

## DIDs

If you are using projects created before KCM version 4.4, you can configure which DID is used in the Master Templates in a certain project or folder on the Data tab of the Configuration window.

## Master Templates

Master Template is a program responsible for the document creation. In KCM Repository, you can create and run Master Templates to test them and select a platform to perform the test.

You can run Master Templates for testing on KCM Core and KCM ComposerUI:

1. In the tree view, select a project with Master Templates to test.
2. Right-click it and then click **Configuration**.
3. Select the **Test Model** tab.
4. To run the Master Templates on KCM ComposerUI, select **KCM ComposerUI**.
   In the **KCM ComposerUI URL** field, specify the URI to use for starting the template. **Example** http://server/itp/app/sample/modelbegin.aspx
5. To run the Master Templates on KCM Core, select **KCM Core**.
   In the settings sections, specify the host and port where the KCM Core installation resides. You can also provide an environment, keys, and extras for the run. Additionally, you can provide the location of a XML file with test data which is then used to fill the Data Backbone. The file must be accessible on that location by KCM Core, and its contents should conform to the XSD of the Data Backbone.
6. To apply the changes, click **Apply** and then click **OK**.

Only selecting KCM ComposerUI for both ASP.NET and J2EE or KCM Core as run platform does not suffice to enable this functionality. For more information on what is needed to test Master Templates using these platforms, see the next sections.

## Run Master Templates on CM ComposerUI from CM Repository

You can run Master Templates on KCM ComposerUI for both ASP.NET and J2EE version 3.1.14 or later from KCM Repository.

If a Master Template is run using KCM ComposerUI for both ASP.NET and J2EE, KCM ComposerUI decides which KCM Core environment should be used. If no environment is passed by KCM ComposerUI, the KCM Core default environment is used. You can configure an alternate environment for a project by passing a KCM Core environment on the KCM ComposerUI URI.

In the following example, the environment `live` is used to run Master Templates.

```
http://itponlineserver/itp/app/sample/modelbegin.aspx?env=live
```

## Run Master Templates on CM Core from CM Repository

You can run Master Templates on KCM Core version 3.2.19 or higher from KCM Repository.

KCM Core before version 4.4 needs to be configured as follows:

1. In KCM Core Administrator, click the **Services** node and select the **Constants** tab. Verify that the constant AllowRepositoryModelRun is set to Y. If it is not set to Y, KCM Core denies running Master Templates from KCM Repository.
2. On the **Services** tab, click **Compile scripts** to recompile the scripts, and then click **Save & Apply**.

## Run Master Templates with dynamic objects

When a Master Template is run, the [published] revision of the Master Template and all related objects are retrieved from KCM Repository. If an object is used that has no [published] revision, retrieving fails.

By default, a Master Template or a dynamic object with the status [published] is used for the production run. If the KCM Repository installation is upgraded from a version older than 3.5.10, the option "Handle Accepted revisions as Published" is checked. In this case, the status [accepted] takes over the role of the status [published].

By default, this compatibility option is turned off for new installations. You cannot select this option when the status [published] is already introduced on objects.

### Retrieve building blocks

The following applies when retrieving Forms, QForms, Text Blocks, Text Blocks Lists, and Content Wizards:

If a project is passed:

- If a building block is inserted using a fully qualified path, the building block is retrieved from this particular project.

If no project is passed:

- If a Master Template uses Text Blocks, Text Block Lists, or Forms, KCM Repository looks for them in a number of folders. For information on how to configure the search paths, see Dynamic objects.

## Retrieve Style Sheets and Page Styles from a brand

As of KCM Core version 4, Style Sheets and Page Style Documents are managed by KCM Repository, and KCM Core can retrieve the documents as dynamic objects. Use of this feature also requires the KCM Repository version 4 or later.

If the setting LabelDocuments is not configured on the General tab of the Environments node in KCM Core Administrator, KCM Core always tries to retrieve the Style Sheet and Page Style documents from KCM Repository. The Style Sheet is retrieved from the `stylesheet` or `pagestyle` folder under the specified brand.

**Example**

```
# BEGIN

#@(pagestyle ("Insurance For Life"; "Invoice"))#
#@(stylesheet ("Insurance For Life"; "Invoice"))#

#
... content ...
#
END #
```

This example uses the invoice `stylesheet` and `pagestyle` stored under the insurance brand `For Life`.

If the setting LabelDocuments is configured, these styles are always retrieved from the file system.

Style Sheets must use the following naming convention: *<base directory>\<brand>\Style-<stl>.<ext>*

Page Styles must use the following naming convention: *<base directory>\<brand>\Page-<stl>.<ext>*

where:

- *<base directory>* is the folder configured with the setting LabelDocuments on the General tab of the Environments node in KCM Core Administrator.
- *<brand>* is the name specified by the brand parameter.
- *<stl>* is the name specified by the stylesheet parameter.
- *<ext>* is the extension of the document. This is either DOC or DOCX, depending on the type of the Master Template document.

All settings from these documents are copied into the result document, overriding the styles and settings that might already be defined in the document.

The preceding example uses the following two documents to apply the styles (assuming a DOC Master Template document):

- *<base directory>\Insurance For Life\Style-Invoice.doc*
- *<base directory>\Insurance For Life\Page-Invoice.doc*

## Retrieve a Master Template and its Data Backbone

Master Templates and Data Backbone are always retrieved from KCM Repository.

In order to instruct KCM Core to retrieve a Master Template and its Data Backbone, the name of the Master Template should be specified as a rep:/ URI or a Letter Book URI. For more information, see the *Kofax Communications Manager Core Developer's Guide*.

## Includes

Includes contain functions and procedures and can be reused in more than one Master Template. You can add Includes at the beginning or at the end of a Master Template. For more information, see Use auto-includes.

Chapter 5

# Manage data

There are three ways to import and export data from KCM Repository:

1. "Save to" and "Load from" functions can be used to save files to the file system as documents and to load documents from the file system. The main use of this functionality is to load a new KCM Repository with an existing collection of templates. See Save and Load.
2. The Import/Export functionality gives you the ability to export objects from KCM Repository to be archived or to imported again in another KCM Repository installation. Exported objects are stored in a file that can be read by the import process. See Import and Export.

## Save and Load

You can save documents, and text files, and DIDs from KCM Repository to your computer and load documents and text files from your computer to KCM Repository.

To save documents:

1. In the tree view, click the object or folder to save and then on the menu click **File** > **Import/Export** > **Save to**.

   The "Save to" dialog is opened.
2. Select the destination for the file/folder.
3. Click **OK**.

To load documents:

1. In the tree view, click the folder to load the files to and then on the menu click **File** > **Import/Export** > **Load from**.

   The "Load from" dialog is opened.
2. Check **Include subdirectories** to load the subfolders of the select folder and rebuild the folder structure in KCM Repository.

   In the **Directory** field, select the folder to load the files from.

   In the **Mask** drop-down list, select a mask. The mask determines which files are shown during selection. To select several masks, separate them by a semicolon, comma, or space.

   In the **Files** section, select the files to load. If you select a folder, all files in that folder are loaded. If **Include subfolders** is checked, all subfolders and their content are loaded as well, and the folder structure is rebuilt in KCM Repository. Empty subfolders or subfolders containing no files that comply with the mask are not loaded. If no file or folder is selected in the **Files** section, all files in the **Files** section are loaded, including all files in subfolders.

3. If a file to load already exists in the target KCM Repository folder:

   Select **Prompt** to skip or create a new revision of the file.

   Select **Skip** to skip the file.

   Select **Create new Revision** to create a new revision of the file in KCM Repository.

   Select **Abort** to stop the loading process once a duplicate file is encountered. In that case, no documents are imported.

   Select **Rename** to have the ability to rename the duplicate file that is loaded.

   **Note** The duplicate check is performed on folder level.

4. In the **Target folder** section, select the target folder in KCM Repository. You cannot load to the root of a project.

   You can right-click the project and select to create a new project or a new folder. If you create a new folder, it is created in the project that is selected in the **Target folder** section. If you create a new project, the configuration window for the new project is opened allowing you to configure the new project.

   **Note** The loaded files are copied and placed inside KCM Repository. After the load, there is no relation between the loaded files and the original files. Altering one has no effect on the other.

5. Click **OK**.

## Import and Export

You can export selected objects as KCM Repository objects and import objects in a different KCM Repository. You can perform an export action on every object or object revision in KCM Repository. Whatever object or object revision selected, the export action assembles a set of objects for export related to the selected object. For example, if a document is selected for export, the project, the folder that contains the document, the Style Document, DID file, and other components are exported.

Exporting an object does not remove that object from KCM Repository.

You need to have permission to use this functionality.

### Set the identifier for revisions

When development takes place on a project that was imported, it can be relevant to see whether a revision originated from the exporting Repository or was edited locally. This can be achieved by assigning an identification mark or word as a postfix to revision numbers wherever a revision is listed. This postfix is added to each new revision, indicating in which Repository it was created.

The postfix must be set by the administrator. Typically, it is a code with a limited number of characters such as "dev" for a development KCM Repository, or "prod" for a production KCM Repository. We advise that you use a short code because the postfix is also used in the tree view, where space is limited, to show the origin of imported objects.

1. To set the identifier, on the menu, click **Action** > **Installation properties**.
2. Enter the revision number postfix.
3. Optionally, you may provide a Repository Role and Repository Role signaling color. They are used in the status bar of KCM Designer for Windows. When you have multiple active Repository installations, this may help you quickly identify in which one you are working.
4. Click **OK**.

## Export

The export procedure depends on how you will use the exported data. You have the following options:

- Transfer the data to another system.
- Save a copy for archival use to submit data to the helpdesk for debugging.

### Transfer to another system

Transfer to another system option exports the current, accepted, and published revisions and everything they need (DID, Includes) without exporting the KCM Repository specific data (authorization). Labels, Characteristics, and older revisions can be exported as additional options.

This export option makes it possible to move objects from one KCM Repository to other KCM Repository without unnecessary information.

The user who imports the export must be aware that the roles and permissions are not exported, so they have to be reassigned.

1. To perform this export type, in the tree view, click the object to export and on the menu click **File** > **Import/Export** > **Export**.
   The Export dialog is opened.
2. Select the option.
3. Select the location to export the object.
4. If you are using labels and the object to export has several revisions, check **Select exported objects by label** and select the label from the drop-down list. For more on labels, see Label objects.
5. Check **Include history (older revisions)** to export all revisions of the object. This option is not available if an object revision is selected.
6. Check **Include label and characteristics** to export labels and characteristics of the object and its revisions.

### Archival and helpdesk

**Archival or helpdesk** option exports everything, including authorization. To export a project under a different name, use the "Export alias" setting. For more information, see Projects.

The "Archival or helpdesk" export makes a copy of the selected objects. It can be useful to archive some projects from KCM Repository before deleting them or to send a troublesome template to the helpdesk. This export type maintains authorization information.

**Note** If the export is used to send a troublesome section of KCM Repository to the helpdesk, do not change anything and export the section as is. Also, for security reasons, the global authorization of users (authorization granted to a user directly, not for a particular project) is not exported.

This export also exports the [in development] revision of objects. When KCM Repository data is archived, make sure to store a copy of the current version of the KCM Repository installers.

1. To perform this export type, select the object to export and on the menu click **File** > **Import/Export** > **Export**.

   The Export dialog is opened.
2. Select the option.
3. Select the location to export the object.

   **Note** By default, when this export type is selected, all revisions of the object, its labels, and characteristics are always exported.
4. Click **OK**.

## Import

The Import wizard lets you select an exported file to load. By default, this is a file with an .mdk extension. It shows which projects are being imported, and whether they are overwriting existing projects in your KCM Repository, or whether new projects are added.

**Note** Prior to importing, back up your database.

Consider the following when importing project exports:

- When importing a project from an older KCM Repository, a number of predefined folders are added to it, such as DIDs, Style Documents, Text Blocks, Forms, Field Sets, and Views. This is similar to the creation of a new project.
- When importing an "Archival or helpdesk" export, the import fails if the project with the same name already exists. To complete the import, rename the existing project.
- When importing a regular export, the import fails if the project with the same name already exists and contains Changesets created and controlled by an external workflow. To complete the import, delete such Changesets from the existing project. For more information on these Changeset types, see the KCM Designer online Help.

You can view the export/import history of an object. See Projects.

1. To import an object, in the tree view, click the object to import and on the menu and then click **File** > **Import/Export** > **Import**.

   The Import Wizard is opened.
2. Select the file to import and click **Next** > **Next** > **Finish**.

Appendix A

# Error report settings

This and the next one sections describe the settings that can be set with Other Settings option on the Project Configuration tab. With this option you can configure the Create Master Template process.

**Note** These settings are rarely used, and therefore are not often added to the Other Settings option.

The error report settings influence the appearance of the error document and adjust its look and feel. The following settings can be used:

| Setting | Default | Description |
|---|---|---|
| ITPERRORCOLORS | 6,6 | With this setting you can change the color of errors and include markers in a Microsoft Word error document. The value must be specified as x,y where x is the color for errors and y is the color for include statements. Possible values are:<br>**1.** Black<br>**2.** Blue<br>**3.** Cyan<br>**4.** Green<br>**5.** Magenta<br>**6.** Red<br>**7.** Yellow<br>**8.** White<br>**9.** Dark Blue<br>**10.** Dark Cyan<br>**11.** Dark Magenta<br>**12.** Dark Red<br>**13.** Dark Yellow<br>**14.** Dark Grey<br>**15.** Light Grey<br>**Note** This option only has effect on error documents for Microsoft Word, HTML and AmiPro Master Template documents. |
| ITPMAXERRORS | 15 | Number of errors to report before terminating compilation. |
| ITPSTRICTLANGUAGE | Y | Enable the PAR4299 warning that will be reported if KCM finds illegal characters in the Master Template document. This warning will appear if the Master Template contains statements like ASSIGN x := @(y). |

Appendix B

# Word processor settings

The settings below affect the result document. Note that all of these settings are supported only for Microsoft Word DOC documents, except ITPINCLUDELEVEL which is supported for all formats except Microsoft Word DOCX.

| Setting | Default | Description |
|---|---|---|
| ITPCOMPATIBLEFIELD | N | Enable this setting if you don't want to handle KCM instructions within Microsoft Word fields. |
| ITPCOMPATIBLEFORM | N | Enable this setting if you don't want to handle KCM instructions within Microsoft Word Form Fields. |
| ITPEXPANDVISFIELD | Y | Disable this setting to disable the handling of KCM instructions within the visible part of Microsoft Word fields. |
| ITPCOMPATIBLEBOOKMARKS | N | Enable this setting to deactivate the range support of bookmarks. |
| ITPINCLUDELEVEL | 16 | Maximum number of nested documents allowed when using the built-in __INC(…) support. |
| ITPWFWSUBDOCS | N | Enable to have the built-in __INC(…) support include subdocuments into a master document. |
| ITPCOMPRESSLISTS | Y | When set, this setting will remove the left overhead if multiple included documents contain the same named bulleted/numbered styles.<br>It strips list overrides from includes which do not use their overrides or where the override has been removed due to style re-mapping.<br>**Note** This might not work if each include uses at least one list override. |

| Setting | Default | Description |
|---|---|---|
| ITPLISTSUPPORT | Y | This setting includes the bullet/ numbering gallery from the Master Template document into the Master Template for later use in the result document.<br><br>**Note** Master Templates created with this option enabled cannot be executed with pre 2.1.14 KCM versions. |

Appendix C

# Other settings

| Setting | Default | Description |
|---|---|---|
| ITPDIDCP | System code page of the AS/400 | Code page in which the KCM/SDK DID files are stored. |
| ITPSUPWPFILE | | If this setting is set, the default settings KCM uses for the Microsoft Word processor support are overwritten by the settings in this supwp.ini configuration file. |
| ITPALLOWMISSINGINC | N | Controls whether or not a missing `__INC` document causes an error when translating a Master Template. If this setting is set to Y, KCM removes the `__INC()` statement and continue translating. If this setting is set to N, KCM will report an error and fail the translation. This behavior is in compliance with the behavior of Office Vision. |

The next settings affect the amount of memory KCM Microsoft Word processor Support uses and the way it handles temporary files. The Microsoft Word support produces a large number of temporary files for administrative purposes. With the default settings only a few of these files should ever exceed the default setting ITPOUTPUTCACHE. Increasing either ITPOUTPUTCACHE and/or ITPMAXTMPOPEN can improve performance while producing large result documents.

| Setting | Default | Description |
|---|---|---|
| ITPMAXINPUTCACHE | 1024 | Sets the maximum amount of memory (in Kb) that KCM Microsoft Word processor support uses while reading a file.<br>If files exceed this size, KCM will read only as much of the file as fits in this size and update from disk if it needs other parts of the file.<br>**Note** This setting affects the amount of memory that KCM Microsoft Word processor Support uses during translation and execution of a model. |

| Setting | Default | Description |
|---|---|---|
| ITPOUTPUTCACHE | 256 | Sets the maximum amount of memory in Kb that KCM Microsoft Word processor support can use while writing a file. If the file exceeds this size, KCM will flush parts of the file to disk. **Note** This setting affects the amount of memory that KCM Microsoft Word processor Support uses during execution of a model. |
| ITPMAXTMPOPEN | 1 | **Note** Sets the number of temporary files that is kept open simultaneously. If the size of an output file exceeds the size indicated with ITPOUTPUTCACHE, this file will be flushed to disk. |

ITPMAXINPUTCACHE primarily affects performance during Master Template translation and post-include.

ITPOUTPUTCACHE and ITPMAXTMPOPEN affect performance during Master Template execution and post-include.

**Note** The effect of assigning more memory with the ITPOUTPUTCACHE setting should be primarily noticeable while producing large result documents on a system that has sufficient hardware resources available.