# Kofax Customer Communications Manager

5.0

ITP/MDK Repository API

KOFAX
from Lexmark

# Contents

Introduction ..... 3
Copyrights and trademarks ..... 3
Using the CCM Repository API ..... 4
XML specification ..... 5
Basics ..... 5
Version history ..... 6
<project> and <library> ..... 6
<folder> ..... 8
<data-folder> ..... 9
<style-folder> ..... 9
<content-wizard-folder> ..... 9
<textblock-folder> ..... 10
<nonitp-folder> ..... 10
<data-backbone> ..... 11
<did> ..... 12
<document> ..... 12
<style-document> ..... 13
<content-wizard> ..... 13
<textblock> ..... 14
Error Codes ..... 15
Known issues ..... 15

# Introduction

This API allows loading some types of objects into CCM Repository, and configuring projects, from an XML file. It will connect to the CCM Repository Server to perform the loading action.

The CCM Repository API is distributed in three files:

- repapi.exe - the actual API program
- repapi.xsd - a specification of the XML accepted by the API
- this document - documentation on the API.

> **Note**
> This API is currently a customer-specific beta version.

## Copyrights and trademarks

# Using the CCM Repository API

The CCM Repository API is implemented in the program repapi.exe. It is invoked by the command:

```
repapi /load /xml=<xml file> [/keeplocked]
       [/user=<account name> /password=<password>]
       [/server.host=<server name> /server.port=<server port>]
       [/logfile=<log file name>] [/resources=<resources directory>]
```

It is recommended to install the repapi.exe program in the CCM Repository client installation directory. You then don't need to provide many of these options.

The parameters have the following meaning:

- load: indicates that objects are to be loaded from the file specified. This is a required parameter.
- xml: indicates the XML file from which to load the objects. The format of this file is discussed in the next section.
- keeplocked: if specified, objects loaded will not be unlocked, and be kept [in development] by the user specified for the API run.
- user: the API will log on as the user specified in this parameter. It can be omitted, see below.
- password: password for the user account. It can be omited, see below.
- server.host: the host name of the computer running the CCM Repository server. By default it is localhost. This setting is normally provided by the client itprep.ini file, and does not have to be provided if repapi.exe is installed in the CCM Repository client installation directory.
- server.port: the port configured for the CCM Repository server. By default it is 2586. This setting is normally provided by the itprep.ini file, and does not have to be provided if repapi.exe is installed in the CCM Repository client installation directory.
- logfile: the log file to be written. By default this is repapi.log, but the standard setting in a CCM Repository client itprep.ini file is %temp%\itprep.log.
- resources: the directory containing resource files; i.e. the 'resources' subdirectory in the client installation directory. This option does not need to be provided if the repapi.exe is installed in the client installation directory.

The user account is used for authorization checking. If the /user and /password flag are both not specified, the API will attempt to log in using the Windows user ID as the CCM Repository user account. See the section Unified Logon in the Administrative Manual.

The CCM Repository API is a client program and must have access to the CCM Repository Server. It should have an itprep.ini file with the server address available in the current directory. Alternatively, pass a configuration file with /cfg=<config file>, or pass the command line parameters /server.host=... and /server.port=....

The API only adds new objects. If the XML specifies an existing object, an error message is given.

# XML specification

The file repapi.xsd contains an XSD for the XML format accepted by the API. Below you will find some explanation and semantic constraints.

## Basics

The XML file must be UTF-8 encoded.

It must contain a single top-level node <itpmdkrep version="1.3">. Previous versions were 1.0, 1.1 and 1.2.

```
<?xml version="1.0" encoding="UTF-8" ?>
<itpmdkrep version="1.3">
  ...
</itpmdkrep>
```

This top-level node may contain any number of project nodes, that in turn may contain nested object nodes. The configuration subnodes in these project nodes may refer to other objects. Any reference should refer to an object that already exists in the CCM Repository, or that occurs in the XML file before the reference.

All object nodes have an identifying "name" attribute. The name attribute specifies the name of the object to load. By nesting object nodes (document nodes in folder nodes, for example) you can specify in which location the object is to be loaded. Note that the predefined folders (like the Style Documents folder, the DIDs folder, the Data Backbones folder) need not and cannot be specified in the XML - these are implied.

For example:

```
<project name="My First Project">
  <folder name="A folder in the project">
    <document name="model document">
      <description>This document resides in the folder "A folder in the
project"</description>
    </document>
  </folder>
  <style-document name="a style document">
    <description>
        This style document will be created in the predefined Style Documents
folder
    </description>
  </style-document>
  <data-folder name="a data folder">
    <description>
        This folder will be created nested in the predefined Data Backbones
folder
    </description>
  </data-folder>
</project>
```

Apart from nested objects in folders or projects, sub nodes of an object node specify configuration,

description or content. If these are omitted, new objects will get default values.

Content of documents, DIDs and Data Backbones are not included in the XML, but should be stored in separate files and referred to in the <content> nodes.

## Version history

The following versions of the CCM Repository API XML are supported:

```
<itpmdkrep version="1.0">
```

The first version of the CCM Repository API XML.

```
<itpmdkrep version="1.1">
```

Introduced in CCM Repository 4.1.3, version 1.1 of the CCM Repository API XML will assume that project definitions are for new-style projects (having exactly one data backbone per project), unless the `legacy="true"` attribute is specified.

```
<itpmdkrep version="1.2">
```

Introduced in CCM Repository 4.2.1, version 1.2 of the CCM Repository API XML introduces support for `<nonitp-folder>` nodes.

```
<itpmdkrep version="1.3">
```

Introduced in CCM Repository 4.2.2, version 1.3 of the CCM Repository API XML adds support for the import of Content Wizards.

```
<itpmdkrep version="1.4">
```

Introduced in CCM Repository 4.2.3, version 1.4 of the CCM Repository API XML adds support for the import of Text Blocks.

## <project> and <library>

The <project> node defines a project, the <library> node defines a library project. Their content is identical.

Attributes:

- name: required. The name should adhere to the CCM Repository naming rules.
- legacy: optional. Indicates legacy-type projects. Allowed values are "true" and "false".

Sub nodes:

- description: optional. The description of the object. It should be less than 2000 characters long.
- config: optional. The configuration of the project.

With version 4.1.3 of the CCM Repository, new projects will have only a single data backbone, which resides directly under the project. This change is reflected in version 1.1 of the CCM Repository API XML. If you want to specify old-style projects in this version of the XML, add the attribute legacy="true" to the project or library node. If it is omitted or set to "false", the CCM Repository API will reject project definitions containing multiple data backbones.

The <config> subnode collects configuration of the project, and contains the following nodes:

- itp-lang: the ITP language. Valid values are ENG, NLD and DEU, the default value is ENG.
- cmp: the Create Model Package version, in the V.R.M format, e.g. 3.5.20. By default, the most recent installed version is used.
- online-uri: the CCM ComposerUI Server address for testing models from within the CCM Repository.
- doctype: the type of documents in this project. Valid values are msword-bin for binary format Microsoft Word documents, msword-oox for Microsoft Office Open XML documents, and openoffice.org for OpenOffice.org documents.
- style-document: the Style Document configured on the project. It should refer to a Style Document that is already present in the CCM Repository, or that occurs elsewhere in the XML file. The style document name should include the extension. The [current] revision of the Style Document is selected.
- includes: the list of include folders and libraries, as specified below.

The following two configurations are only available on legacy projects (with the legacy="true" attribute):

- did: the DID configured on the project. It should refer to a DID that is already present in the CCM Repository, or that occurs elsewhere in the XML file. The [current] revision of the DID is selected.
- data-backbone: the Data Backbone configured on the project. It should refer to a Data Backbone that is already present in the CCM Repository, or that occurs elsewhere in the XML file. The [current] revision of the Data Backbone is selected.

The <includes> subnode of the <config> node lists the include folders and libraries of a project. It has a single attribute:

- mode: determines how the specified include folders are merged with the previously loaded ones (as the <includes> node may occur multiple times). Valid values are
    - replace: the default. The existing list of include folders and libraries is replaced by the listed ones.
    - insert: the listed include folders and libraries are inserted before the existing ones.
    - append: the listed include folders and libraries are appended after the existing ones.

It has zero or more <include> sub nodes. The value of each node should be the name of a library project, or the full path to a folder (in the form project:folder\folder\includefolder).

Some restrictions:

- Folders and libraries should not be added multiple times.

- Folders added to the include list should be folders in the project being configured.

- Libraries added to the include list should be libraries, not projects.

> **Note**
> The API currently does not check for these restrictions. It may do so in a newer version.

There may be multiple <includes> nodes in a <config> node, allowing you to add include folders in different places. The nodes are processed in the order in which they occur in the XML file.

Example:

```
<project name="My Project">
  <description>This is an example project.</description>
  <config>
    <itp-lang>ENG</itp-lang>
    <cmp>3.5.20</cmp>

<online-uri>http://webserver/itp/app/sample2/modelbegin.aspx</online-uri
>
    <doctype>msword-bin</doctype>
    <style-document>My Project:Style sheet.doc</style-document>
    <includes mode="replace">
      <include>My Project:Includes</include>
      <include>My Project:Includes\More Includes</include>
      <include>My Library</include>
    </includes>
  </config>
</project>
<library name="My Library">
  <config>
    <includes>
      <include>My Library:Includes</include>
    </includes>
  </config>
</library>
```

The project configuration may refer to objects (style documents, include folders, etc.) that are defined elsewhere in the XML file, even if they occur after the <config> node.

Creating a project with a <project> or <library> node automatically creates all standard predefined folders, but does not add standard style documents.

## <folder>

The <folder> node defines a (document) folder.

Attributes:

- name: required. The name should adhere to the CCM Repository naming rules.

Sub nodes:

- description: optional. The description of the object. It should be less than 2000 characters long.

Example:

```
<folder name="Includes"/>
<folder name="More Includes">
  <description>More include documents</description>
</folder>
```

## <data-folder>

> **Note**
> This type of node is only available in legacy projects (with the legacy="true" attribute).

The <data-folder> node defines a folder containing Data Backbone objects.

Attributes:

- name: required. The name should adhere to the CCM Repository naming rules.

Sub nodes:

- description: optional. The description of the object. It should be less than 2000 characters long.

Example:

```
<data-folder name="Data">
  <description>Subfolder of the Data Backbones predefined
folder</description>
</data-folder>
```

> **Note**
> The predefined Data Backbones folder cannot be created by a <data-folder> node, and it is never referred to in a path.
> Data-folder nodes make no sense for project definitions without the legacy="true" attribute, as new-style projects have no data backbone folders.

## <style-folder>

The <style-folder> node defines a folder containing Style Documents.

Attributes:

- name: required. The name should adhere to the CCM Repository naming rules.

Sub nodes:

- description: optional. The description of the object. It should be less than 2000 characters long.

Example:

```
<style-folder name="More Styles">
  <description>Subfolder of the Style Documents predefined
folder</description>
</style-folder>
```

> **Note**
> The predefined Style Documents folder cannot be created by a <style-folder> node, and it is never referred to in a path.

## <content-wizard-folder>

The <content-wizard-folder> node defines a folder containing Content Wizards.

Attributes:

- name: required. The name should adhere to the CCM Repository naming rules.
- data-backbone: optional. The name of the Data Backbone that will be configured on the Content Wizard folder.

Sub nodes:

- description: optional. The description of the object. It should be less than 2000 characters long.

Example:

```
<content-wizard-folder name="Interactive Content Wizards">
  <description>Subfolder of the Content Wizards predefined
folder</description>
</content-wizard-folder>
```

**Note**
The predefined Content Wizards folder cannot be created by a <content-wizard-folder> node, and it is never referred to in a path.
If the `legacy="true"` attribute is present on the project or library node the data-backbone attribute is used to configure a Data Backbone on the folder. This attribute is ignored if `legacy="false"`.

## <textblock-folder>

The <textblock-folder> node defines a folder containing Text Blocks.

Attributes:

- name: required. The name should adhere to the CCM Repository naming rules.

Sub nodes:

- description: optional. The description of the object. It should be less than 2000 characters long.

Example:

```
<textblock-folder name="Signatures">
  <description>Subfolder of the Text Blocks predefined folder</description>
</textblock-folder>
```

**Note**
The predefined Text Blocks folder cannot be created by a <textblock-folder> node, and it is never referred to in a path.

## <nonitp-folder>

The <nonitp-folder> node defines a folder for non <itp> objects.

Attributes:

- name: required. The name should adhere to the CCM Repository naming rules.

Sub nodes:

- description: optional. The description of the folder. It should be less than 2000 characters long.

Example:

```
<nonitp-folder name="Non ITP Objects"/>
<nonitp-folder name="More Non ITP Objects">
  <description>To store non itp objects such as XSDs</description>
</nonitp-folder>
```

## <data-backbone>

The <data-backbone> node defines a Data Backbone.

Attributes:

- ext: optional. The file extension of the source part of the Data Backbone, e.g. "doc" for binary Microsoft Word documents, "docx" for Microsoft Office Open XML documents, "txt" for plain text documents. The extension may also be part of the name (name="data.doc"). If no extension is specified in the name or ext attribute, it is undefined what the actual value of the extension will be.

In legacy project definitions (with the legacy="true" attribute), the following attribute is mandatory; for other projects it is forbidden:

- name: The name should adhere to the CCM Repository naming rules.

Sub nodes:

- description: optional. The description of the object. It should be less than 2000 characters long.
- content-definition: required. The name of the file containing the definition source content of the Data Backbone. This value must be of the form "file://*filename*". If the filename does not specify an absolute path, the file is sought relative to the directory containing the XML file.
- content-retrieval: optional. The name of the file containing the retrieval source code of the Data Backbone. This value must be of the form "file://*filename*". If the filename does not specify an absolute path, the file is sought relative to the directory containing the XML file.
- config: optional. The configuration of the Data Backbone.

The <config> subnode collects configuration of the Data Backbone, and contains the following single node:

- did: the DID configured on the Data Backbone. It should refer to a DID that is already present in the CCM Repository, or that occurs elsewhere in the XML file. The [current] revision of the DID is selected.

Example:

```
<data-backbone ext="txt">
  <description>Company data</description>
  <content-definition>file://data07a.dat</content-definition>
  <content-retrieval>file://data07b.dat</content-retrieval>
  <config>
    <did>My Project:DBDATA</did>
  </config>
```

```
</data-backbone>
```

**Note**
A Data Backbone is loaded as source text, and then compiled.
If the legacy="true" attribute is not present on the project or library node, it must have at most one data-backbone node. If there is none, the project will get an automatically generated empty Data backbone.

## <did>

The <did> node defines a DID.

Attributes:

- name: required. The name should adhere to the CCM Repository naming rules.
- ext: optional. The file extension of the source part of the DID, e.g. "doc" for binary Microsoft Word documents, "docx" for Microsoft Office Open XML documents, "txt" for plain text documents. The extension may also be part of the name (name="did.doc").

**Note**
The extension for DIDs is added for future use. As the API currently loads compiled DID objects, it is not used.

Sub nodes:

- description: optional. The description of the object. It should be less than 2000 characters long.
- content: required. The name of the file containing the (compiled) content of the DID. This value must be of the form "file://*filename*". If the filename does not specify an absolute path, the file is sought relative to the directory containing the XML file.

Example:

```
<did name="Comp Data" ext="">
  <description>Company data</description>
  <content>file://did07.dat</content>
</did>
```

## <document>

The <document> node defines a document.

Attributes:

- name: required. The name should adhere to the CCM Repository naming rules.
- ext: optional. The file extension of the source part of the document, e.g. "doc" for binary Microsoft Word documents, "docx" for Microsoft Office Open XML documents, "txt" for plain text documents. If the ext attribute is omitted, the extension must be part of the name (name="document.doc"). If no extension is specified in the name or ext attribute, it is undefined what the actual value of the extension will be.

Sub nodes:

- description: optional. The description of the object. It should be less than 2000 characters long.
- content: required. The name of the file containing the content of the document. This value must

be of the form "file://*filename*". If the filename does not specify an absolute path, the file is sought relative to the directory containing the XML file.

Example:

```
<document name="My Model" ext="docx">
  <description>Master model</description>
  <content>file://doc07.dat</content>
</document>
<document name="functions_inc.docx">
  <description>Include document defining standard functions.</description>
  <content>file://doc17.dat</content>
</document>
```

## <style-document>

The <style-document> node defines a style document.

Attributes:

- name: required. The name should adhere to the CCM Repository naming rules.
- ext: optional. The file extension of the source part of the document, e.g. "doc" for binary Microsoft Word documents, "docx" for Microsoft Office Open XML documents, "txt" for plain text documents. If the ext attribute is omitted, the extension must be part of the name (name="document.doc"). If no extension is specified in the name or ext attribute, it is undefined what the actual value of the extension will be.

Sub nodes:

- description: optional. The description of the object. It should be less than 2000 characters long.
- content: required. The name of the file containing the content of the document. This value must be of the form "file://*filename*". If the filename does not specify an absolute path, the file is sought relative to the directory containing the XML file.

Example:

```
<style-document name="Standard Styles" ext="docx">
  <description/>
  <content>file://doc02.dat</content>
</style-document>
<style-document name="More Styles.doc">
  <description>Additional styles.</description>
  <content>file://c:\styles\doc03.dat</content>
</style-document>
```

## <content-wizard>

The <content-wizard> node defines a Content Wizard.

Attributes:

- name: required. The name should adhere to the CCM Repository naming rules.

Sub nodes:

- description: optional. The description of the object. It should be less than 2000 characters long.
- content: required. The name of the file containing the definition of the Content Wizard. This

value must be of the form "file://*filename*". If the filename does not specify an absolute path, the file is sought relative to the directory containing the XML file.
The definition must be a valid XML file that conforms to the definition of the Content Wizard XSD.

Example:

```
<content-wizard name="Interactive Letter">
  <description/>
  <content>file://wzrd02.xml</content>
</content-wizard>
<content-wizard name="Invoice">
  <description>Invoice Template.</description>
  <content>file://c:\wizards\wzrd03.xml</content>
</content-wizard>
```

## <textblock>

The <textblock node defines a Text Block.

Attributes:

- name: required. The name should adhere to the CCM Repository naming rules.
- rich: required. "true" for a Rich Text Block, "false" for a classic Text Block
- ext: optional. The file extension of the source part of the Text Block, e.g. "doc" for binary Microsoft Word documents, "docx" for Microsoft Office Open XML documents, "txt" for plain text documents. The extension is only relevant for Rich Text Blocks.

Sub nodes:

- description: optional. The description of the object. It should be less than 2000 characters long.
- content: required. The name of the file containing the definition of the Text Block. This value must be of the form "file://*filename*". If the filename does not specify an absolute path, the file is sought relative to the directory containing the XML file.

Example:

```
<textblock name="Signature" rich="true" ext="docx">
  <description/>
  <content>file://blk02.xml</content>
</textblock>
<textblock name="Disclaimer" rich="false">
  <description>Disclaimer Template.</description>
  <content>file://c:\tbs\blk03.xml</content>
</textblock>
```

# Error Codes

The API returns the following error codes:

0 - Success

7 - Command line parameters are incorrect

10 - Other error

A log file is written that logs some steps in the process, and any error message. By default, the name of this log file is repapi.log. It may be changed on the command line with the /logfile=... parameter or with the logfile=... setting in the [Configuration] section of the itprep.ini file. The file is written in the same directory as where the repapi.exe program resides, unless the logfile setting includes a path.

> **Note**
> The default value of the logfile setting in a CCM Repository client itprep.ini file is %temp%\itprep.log. This will also be the file the API writes as log file if this itprep.ini file is used for the API.

XML parsing errors will generate an error message with a small fragment of the XML, where the (approximate) location of the error is indicated with a pipe symbol '|'.

# Known issues

Due to limitations to XSD, the repapi.xsd file contains an XSD specification that is more restrictive than what is actually accepted by the API. In particular:

- the `<includes>` node may occur multiple times in a `<config>` node, but this is rejected by the XSD;
- the XSD poses some limitations on the order of subnodes which are not needed by the API. The XSD requires the `<description>` node (if present) to be first, followed by either the `<config>` or `<content>` node, depending on the type of object. After that, the other subnodes may follow.

Furthermore, the XSD does allow some things that are valid for project definitions with the `legacy="true"` attribute, but will cause an error for other projects:

- More than one `<data-backbone>` node in a project
- `<data-folder>` nodes
- The `name` attribute on Data Backbones definitions
- Configuring DIDs or Data Backbones on projects
- The `data-backbone` attribute on a `<content-wizard-folder>` node.

**Other known issues:**

`repapi.exe` fails with the following error:

```
Error: Could not open file ...\Resources\dbb_empty.itp (The system cannot
find the path specified. ), errorcode: 3
```

This error will occur if a project is created that does not contain a `<data-backbone>` node *and* `repapi.exe` is not installed in the CCM Repository client directory. Fix either condition to resolve this error.