Kofax Customer Communications Manager

5.0

Manual

KOFAX
from Lexmark

Contents

Introduction

This document describes the main entry point to Kofax Customer Communications Manager. This interface runs on a single port and provides:

1. A number of web services (SOAP).
2. Access to the CCM Designer via http.
3. Access to an example CCM ComposerUI for HTML5 web application via http.

Documentation for the package and the separate components, can be found on the server in the following directory: "<provided rootpath>\CCM\Programs\5.0\Documentation".

Creating CCM Designer Users

Before you can log on to CCM Designer, you will need to create one or more users.

Creating users is a task that has to done through CCM Designer for Windows. This component must be installed manually. After deployment, the installer can be found on the server in the following directory: "<provided rootpath>\CCM\Work\5.0\Instance_01\designer\Client". The readme file contains the values that will be asked by the installer. Once the Windows client is installed, users can be created by performing the following steps:

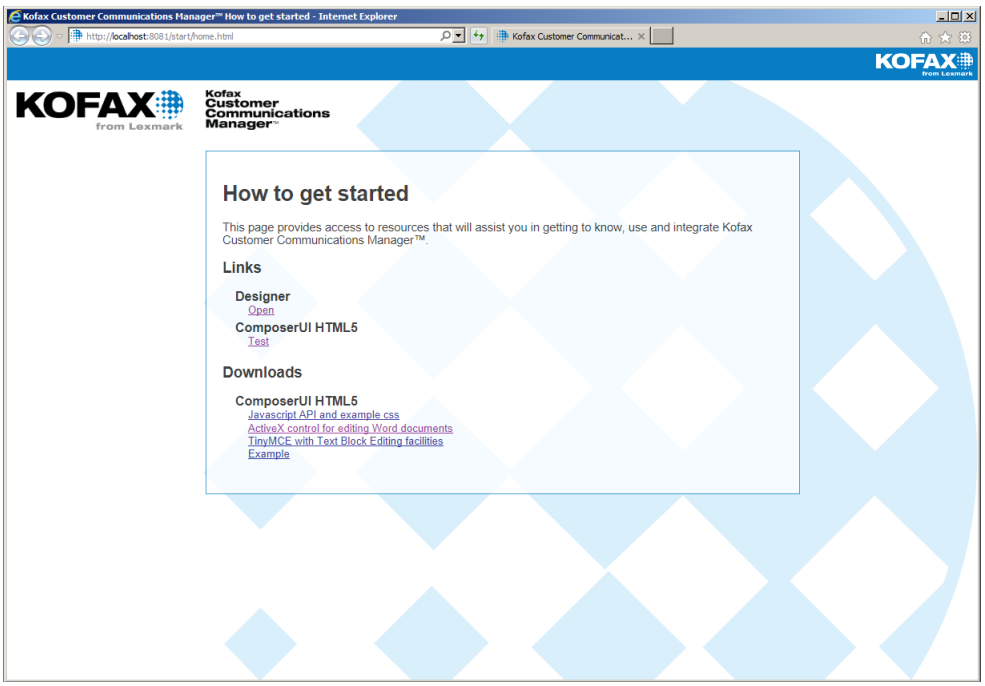
1. Login is "itpadmin" (default password: "www.aia-itp.com", must be changed on first login)
2. Right click "Users"
3. Select "New User"
4. Fill in the details
5. Click "Ok"
6. In case the user has to be able to import projects
    a. Select "Users"
    b. Right click on the newly created user
    c. Select "Configuration"
    d. In the tab "Authorization" enable the checkbox "Allow login as admin"

Authorization for the users can be set in CCM Designer. People with the permission (e.g. the person who imported the project) to set authorization can authorize other users. To be able to view content in the example project, authorization is required that allows the user to view the project.

Access to the Designer

After the creation of users, it is possible to logon to CCM Designer. In order to do so, open the following url in a brower: http://<ccm server>:8081/start/home.html.

This will provide you with the screen below:



By clicking on the link "Open" for the Designer, a logon screen will show up where the credentials can be entered that you have provided earlier for the users that you have defined.

Minimal requirements and settings

To be able to use CCM Designer on a system the following requirements must be met:

General

Supported browsers:

- Microsoft Internet Explorer 11
- Microsoft Edge
- Recent versions of Google Chrome
- Recent versions of Firefox

JavaScript needs to be enabled and pop-up blockers need to be disabled.

To work with Master Templates, Libraries, Includes and Quick Documents

Supported versions of Microsoft Word:

- Microsoft Word 2010
- Microsoft Word 2013

- Microsoft Word 2016

Supported browsers:

- Microsoft Internet Explorer 11

Permission to install ActiveX controls is required.

To work with Content Wizards and Forms

Supported browsers:

- Microsoft Internet Explorer 11

.Net 3.5 and .Net 4 need to be installed and the Microsoft Internet Explorer security setting 'XAML browser applications' needs to be enabled.

The CCM application programming interface

SOAP Web Services

There are several groups of web services, called contract types. In addition, there exist multiple versions of certain contract types.

Currently, Kofax Customer Communications Manager supports the follow contract types.

- CCMInteractive, version V1
- CCMInteractive, version V2
- CCMDistribution, version V1

These are all exposes as SOAP web services. Their respective WDSL's can be obtained via the following URLs.

```
http://<ccm
server>:8081/ccm/wsdl?contracttypename=CCMInteractive&contracttypeversio
n=V1
http://<ccm
server>:8081/ccm/wsdl?contracttypename=CCMInteractive&contracttypeversio
n=V2
http://<ccm
server>:8081/ccm/wsdl?contracttypename=CCMDistribution&contracttypeversi
on=V1
```

The version of each contract type basically tells which calls are available in it. Each contract type version provides in itself a consistent and complete set of calls without any "legacy".

The calls within a contract type also have a version associated to them. This version will change whenever the interface or behavior of a particular call changes. This version is essentially global for that call and unrelated to the version of the contract type(s) that the call is contained in.

The API reference describes exactly which call versions are contained within which contract type. The sections below will largely ignore these versions and describe the functionality of Kofax Customer Communications Manager on a more abstract level.

> **Note**
> The WSDL contains explicit anchors (^ and $) in the added to the pattern definitions. These anchors cannot be found in the documentation (Type descriptions (page 65)).

CCMInteractive:

The CCMInteractive contract type deals with the definition, management, and composition of documents and document packs. The calls that it contains are organized in a number of categories, each identified by a prefix in their name:

- Designer: These calls provide access to the CCM Designer.
- Compose: These calls create a document or a document pack, possibly via CCM ComposerUI for HTML5. The V1 version creates single documents. The V2 version creates document packs

(which contain one or more documents).
- DocumentPack: These calls process existing document packs. They allow reviews and modification of document packs for example, as well as document format conversions.
- Admin: These calls provide administrative functionality.
- SystemCheck: These calls provide functionality for checking the installation.

There are two versions of this contract type:

- V1: This contract type offers functionality that focuses on single documents. The compose calls deliver a single document. The contract offers no support for processing these documents further.
- V2: This contract type offers functionality that focuses on document packs. The compose calls deliver a complete document pack, and the contract offers functionality for manipulating and reviewing these these.

CCMDistribution

The CCMDistribution contract type deals with the distribution of composed document packs. It contains the following functionality:

- DistributeDocumentPack: This takes a document pack as an argument and hands it off to the CCM Core DistributeDocumentPack exit point.
- SignDocumentPack: This call takes a document pack as an argument and hands off to Kofax SignDoc.

The following chapters will provide an overview of the functionality that Kofax Customer Communications Manager offers for each of these contract types. A detailed description of the available calls can be found in API reference sections.

General Request Format

All web service requests have at least the following parameters. The top four parameters identify the contract type that you are using, and are the same for all calls within that contract type. The jobid parameter is an id that you provide and that can be used to relate the execution of Kofax Customer Communications Manager jobs to the execution of your own business application. This value consists of ASCII alphanumeric characters and must not be empty. It can be any value of your choice, but we advise to provide a meaningful id, as this will be helpful when diagnosing problems.

- partner: CCM
- customer: Local
- contracttypename: CCMInteractive or CCMDistribution. This identifies the contract type.
- contracttypeversion: V1 or V2. This identifies the version of the contract type.
- jobid: A non-empty ASCII alphanumeric value of your choice

The returned result of each web service request always contains a requestinfo substructure that contains these parameters as well.

Designer

The designer requests give access to the CCM Designer. These requests are placed in the CCMInteractive contract type. There are three types of requests: GUI requests, explore requests, and administrative requests that are specific to the CCM Designer.

GUI Requests

DesignerStartSession

This call creates a session context in which one end user can access the CCM Designer environment. It returns a url to which the end user can be guided to edit templates, text blocks etc.

The return values are:

- an identifier for the session
- a url that is relative to the url of Kofax Customer Communications Manager (i.e. http://<ccm server>:8081/ccm/). This url will lead to the login page of CCM Designer.

Explore Requests

The explore requests allow a business application to obtain information about objects stored in the CCM Repository so that the business application can start an appropriate composition request. Please see the included reference for details.

- DesignerGetDataBackBoneDefinition: Returns XML data that contains the definition of a Data Backbone.
- DesignerGetDocumentPackTemplate: Returns XML data that contains the definition of a Document Pack Template.
- DesignerGetDocumentTemplate: Returns XML data that contains the definition of a Document Template.
- DesignerGetLetterbook: Returns XML data that contains the definition of a letterbook.
- DesignerListLetterbooks: Returns XML data that contains a list of all letterbooks in a project.
- DesignerListDocumentTemplates: Returns XML data that contains a list of all document templates in a project.
- DesignerListProjects: Returns XML data that contains a list of all all available projects.
- DesignerListTemplates: Returns XML data that contains a list of all available templates in a projext (both document templates and document pack templates).

Designer Administration Requests

DesignerAddUser

This request gets two arguments:

- user: The name of the user that will be added. This name may contain spaces.
- role: The role of the user. You will find a list of possible roles below. The user will get this role on all existing projects.

User Roles

The following roles are allowed.

- author
- project administrator

- content viewer
- publisher
- publishing author
- publishing reviewer
- viewer

Please see the documentation of the CCM Designer for more information about these roles.

Composer

The composer requests allows someone to run templates and create documents or document packs. These requests are placed in the CCMInteractive contract type. The exact functionality depends on the version of the contract type:

- V1: The compose calls deliver single documents. There are separate calls for different output formats running interactive templates and non-interactive (on demand) templates.
- V2: The compose calls deliver document packs. The are separate calls for running interactive templates and non-interactive (on demand) templates.

We will focus on both versions separately below, starting with the V2 interface.

Document Pack Composition

Compose

The on demand compose requests allow non-interactive templates to be run. The non-interactive ComposeDocumentPack call composes a document pack with one or more documents. The produced document packs will contain a docx document for each Document Template in the pack.

The ComposeDocumentPack call takes the following parameters in addition to the standard ones:

- project: CCM Designer project that contains the template that you want to use.
- templatetype: the type of the template that you want to use. This can either be the value "documenttemplate" or "documentpacktemplate".
- templatename: The name of the template that you want to use. This can either be the name of a document template or of a document pack template.
- status: the status of the template that you want to use.
- databackbonexml: The data backbone XML to use as input. This must be the contents of a data backbone XML file, base-64 encoded.

The result of these calls contains:

- a session identifier
- a base-64 encoded result document pack.

If an error occurs, a soap error will be returned.

Compose Interactive

Executing interactive requests requires the implementation of a CCM ComposerUI for HTML5 web application. Please see the CCM ComposerUI for HTML5 documentation for details. The interactive web service calls described below allow you to start a CCM ComposerUI for HTML5 composition run, and they allow you to obtain the result that such a run produces.

A default CCM installation contains an example CCM ComposerUI for HTML5 web application. This application can be reached via the following URL, where server and port are the server and port that CCM Interactive has been deployed on.

```
http://<ccm server>:8081/start/home.html
```

This example will provide access to the login page of the CCM Designer, and it will allow you to run an example interactive Template. This example uses the calls that are described in the following sections.

Start

The ComposeDocumentPackInteractiveStart call starts a new interactive run that (eventually) produces a document pack. It has the following additional arguments:

- project: CCM Designer project that contains the template that you want to use.
- templatetype: the type of the template that will be used. This can either be the value "documenttemplate", "documentpacktemplate", or "letterbook".
- templatename: The name of the template or letterbook that you want to use. The template can either be a document template or a document pack template.

- status: the status of the template that you want to use.
- databackbonexml: The data backbone XML to use as input. This must be the contents of a data backbone XML file, base-64 encoded.
- allowsuspend: whether the user is allowed to suspend (save) the interactive run, and resume it later.

The result of these calls contains:

- an identifier for the run
- url: a url that is relative to the endpoint of Kofax Customer Communications Manager (i.e. http://<ccm server>:8081/ccm/). This url refers to JSON data that serves as input to your interactive CCM ComposerUI for HTML5 web application.

See the included reference for more details on the parameters for this call.

Suspend and resume

If the allowsuspend flag was passed on the earlier start call, users will be allowed to suspend (save) their sessions and resume (restore) it later. The CCMInteractive V2 contract type contains 2 calls that support this:

- ComposeInteractiveGetSuspendedSession: gets a session identifier as a parameter and returns a (base-64 encoded) representation of the session state.
- ComposeInteractiveResumeSuspendedSession: gets the (base-64 encoded) session state that was returned by ComposeInteractiveGetSuspendedSession as a parameter and returns a new session identifier and a url that allows the CCM ComposerUI for HTML5 web application to proceed where it left off earlier.

See the included reference for more details on the parameters for these calls.

Result

The ComposeDocumentPackGet call obtains the result of an interactive run. It gets a session identifier as an argument and delivers the document pack that was produced in that session. If an error occurs, a soap error will be returned.

Document Composition

Compose

The on demand requests allow non-interactive templates to be run. The CCMInteractive V1 contract type has two non-interactive compose calls, one for generating a docx document and one for generating a PDF document:

- ComposeDocxV1: creates a docx document.
- ComposePdfV1: creates a PDF document.

These calls take the following parameters next to the standard ones:

- project: CCM Designer project that contains the document template that you want to use.
- documenttemplate: The document template that you want to use.
- status: the status of the document template that you want to use.
- databackbonexml: The data backbone XML for the document template. This must be the contents of a data backbone XML file, base-64 encoded.

Apart from these, ComposePdfV1 has the following additional parameter:

- securedocument (PDF only): specifies that the PDF document will be secured, the content cannot be copied, printed or changed afterwards.

The result of these calls contains:

- document: a base-64 encoded result document. This is either a PDF or a docx, depending on the call.
- databackbonexml: a base-64 encoded data backbone XML

If an error occurs, a soap error will be returned.

Compose Interactive

Executing interactive requests requires the implementation of a CCM ComposerUI for HTML5 web application. Please see the CCM ComposerUI for HTML5 documentation for details. The interactive web service calls described below allow you to start a CCM ComposerUI for HTML5 composition run, and they allow you to obtain the result that such a run produces.

A default CCM installation contains an example CCM ComposerUI for HTML5 web application. This application can be reached via the following URL, where server and port are the server and port that CCM Interactive has been deployed on.

```
http://<ccm server>:8081/start/home.html
```

This example will provide access to the login page of the CCM Designer, and it will allow you to run an example interactive Template. This example uses the calls that are described in the following sections.

InteractiveStart

There are two calls that start a new interactive run. One that produces a docx result, and one that produces a PDF document:

- ComposeDocxInteractiveStart
- ComposePdfInteractiveStart

Both calls work in a similar way. They have the following additional arguments:

- project: The Designer project that contains the document template or letterbook to use.
- documenttemplate: If specified, the document template that you want to start an interactive run for. Either documenttemplate or letterbook must be specified, but not both.
- letterbook: If specified, the letterbook that you want to start an interactive run for. Either documenttemplate or letterbook must be specified, but not both.
- status: the status of the document template or letterbook that you want to start an interactive run for.
- previewformat: If specified, the format of the document previews that you want to have during the interactive run. Currently only PDF is supported.
- databackbonexml: The data backbone XML for the interactive run. This must be the contents of a data backbone XML file, base-64 encoded.

The result of these calls contains:

- itpcloudid: an identifier for the run
- url: a url that is relative to the endpoint of Kofax Customer Communications Manager (i.e. http://<ccm server>:8081/ccm/). This url refers to JSON data that serves as input to your interactive CCM ComposerUI for HTML5 web application.

See the included reference for more details on the parameters for these calls.

InteractiveGet

There are two calls that obtain the result of an interactive run. One for docx result documents, and one for PDF result documents:

- ComposeDocxInteractiveGet
- ComposePdfInteractiveGet

These calls have the following parameters next to the standard ones:

- itpcloudid: specifies the identifier of the run that has been started earlier.
- securedocument (PDF only): specifies that the PDF document will be secured, the content cannot be copied, printed or changed afterwards.

The result of these calls contains:

- document: a base-64 encoded result document. This is either a PDF or a docx, depending on the call.
- databackbonexml: a base-64 encoded data backbone XML

If an error occurs, a soap error will be returned.

InteractiveFinish

The ComposeInteractiveFinish call signals that the interactive composition run has ended. This will remove any remaining server side data related to the composition run. It has only a single parameter and no output:

- itpcloudid: specifies the identifier of the run that has been started earlier.

Document Pack Processing

Both the CCMInteractive and CCMDistribution contract types offers functionality for processing

document packs that have been generated earlier.

Reviewing

The DocumentPackInteractiveReview call in the CCMInteractive V2 contract type allows one to take an existing document pack and have a user review the documents that it contains.

The call gets the document pack as its argument. It delivers the same result as for the interactive compose calls. This can be used by a CCM ComposerUI for HTML5 web application to show an interactive reviewing tool to the user. If a user finds any errors during this review, the DocumentPackInteractiveModifyFromReview call can be used to make changes.

Modification

The DocumentPackInteractiveModify and DocumentPackInteractiveModifyFromReview calls allows a user to interactively correct any interactive answers that were provided earlier for a particular document pack. These calls restart the interactive process that was used to create the document pack, only now with all answers of the initial composition already filled out in the interactive forms.

The DocumentPackInteractiveModify call gets the document pack as an argument. The DocumentPackInteractiveModifyFromReview call operates on the document pack that was reviewed via DocumentPackInteractiveReview session. The latter operates on the session identifier of the review session and thus avoids the need to pass the complete document pack again as an argument.

The result of these calls is the same as for the interactive compose calls. This allows the CCM ComposerUI for HTML5 web application to show an interactive session to the user that allows her to alter the earlier answers.

Conversion

The CCMInteractive contract type contains a DocumentPackConvert call that allows one to convert documents in the pack. The call gets a document pack and an output format as an argument. It will convert all docx documents that are present in the document pack to the specified output format.

The output format may be either:

- "pdf"
- "pdf/a-1b"

The resulting documents will be placed next to the existing document in the document pack, i.e. de document pack will be augmented with the files in the requested format. The convert call will return this augmented document pack, as well as a session identifier.

Distribution

The DocumentPackDistribute call will distribute a previously created document package. The actual Distribution functionality will be defined in the CCM Core DistributeDocumentPack exit point.

Distribution

The DocumentPackDistributeWorkplace call distributes a previously created document package to a project or folder in Perceptive Workplace.

This call must be passed a number of parameters:

- The url to the Perceptive Workplace server, in the form of http://<server>:<port>[/<service-context>], e.g. https://pw.psft.co/fns-service/.
- Credentials in the form of a user and password. This will give access to the workplace of this user. Sharing of uploaded documents can be arranged from within Perceptive Workplace.
- A path to a space or folder where the documents should be stored. If this path does not exist yet in Perceptive Workplace it will be created by the call.
  This path should be constructed as follows: <path seperator><path seperator><projectname> followed by 0 or more <path seperator><foldername> combinations.
  Example: //MyProject/MyFolder/MySubfolder.
- An output format. This can be either 'native', 'pdf' or 'pdf/a-1b'. This parameter has two functions. On the one hand it determines which document formats will be uploaded. On the other hand, it will try to convert documents to this format if needed.

The documents within the perceptive workplace destination will either be named as follows:

- If a slot reference has been defined for the document template, the name will be: <n>-<slot reference>.<extension>, where 'n' is the location of the document in the generated document pack, i.e. 'n' specifies the order.
- If no slot reference has been defined, the name will be: <n>-<document template name>.<extension>

For example:

```
1-Welcome Letter Agricultural.pdf
2-Policy.pdf
3-Terms and Conditions 2016.pdf
```

Please see the reference section of the CCMDistribution contract type for details.

Signing

The SignDocumentPack request allows one to hand off composed document packs to Kofax SignDoc. This request can be found in the CCMDistribution contract type. It will create a Kofax SignDoc signing package from the MS Word docx files and the PDF documents in the document pack. Kofax SignDoc will subsequently take care that the documents in this signing package get signed.

Document Pack Signing

Currently, Kofax Customer Communications Manager only supports signing MS Word docx files and PDF files. The document pack must contain at least one MS Word docx file that contains a signature line. Kofax SignDoc uses these lines to determine who needs to sign this particular document (the signer). If the document pack contains no documents with signature lines Kofax SignDoc cannot proceed, as no signer has been specified, and an error will be raised.

Signing will proceed for all MS Word docx files and all PDF files in the document pack. Note however that if a document has both a docx version and a PDF version in the same document pack slot, only the docx version will end up in the signing package. Some documents may contain multiple signature lines and others may contain none. For all signers in a document pack, Signdoc will make sure that:

- Docx files that have a signature line for that signer get signed.
- Docx files that have no signature line for that signer get marked as viewed.
- PDF files that have no docx equivalent get marked as viewed.

Please see the the SignDoc documentation for details on this signing process and how signature lines are related to signers.

A document pack may contain documents in other formats than MS Word docx files and PDF files. This may be the case for static document for example. If that is the case SignDocumentPack will raise an error.

Please note that composed documents are *always* included at least as MS Word documents in a document pack.

Authentication

Kofax Customer Communications Manager requires a proper SignDoc API key with sufficient rights to be passed on the SignDocumentPack call.

SignDoc uses 'API keys' to authenticate callers. This key is generated in the Kofax SignDoc UI when a user account is created. To find the API key generated by Kofax SignDoc the user has to navigate to the preference page of his individual account. There he can find a separate section for API keys. Please see the Kofax SignDoc documentation for details.

Signing Package Identification

Kofax SignDoc identifies signing packs through an ID. This id is currently mandatory. If a caller does not specify an id, and error will be raised. The ID can contain alphanumerical characters, minus signs, and underscores (see the SignDoc documentation for details). If the key already exists, the SignDocumentPack call will raise an error. The SignDocumentPack call will not overwrite existing signing packages.

The passed id will be returned by the SignDocumentPack call. Although the returned ID is currently the same as the one that is passed as input, we advise applications to use the returned ID for future reference to this document pack, as future release may return a modified ID.

Resource handling

Various calls involve some server-side session state in Kofax Customer Communications Manager. The resources that are associated with this will be reclaimed automatically, but this may take some time.

The FinishSession call signals that any server side resources that are associated with a particular session id can be reclaimed. It has the session identifier as input and no output.

Administration

The Administration requests are placed in the CCMInteractive contract type.

AdminGetLogs

This call will provide access to a set of logs that have been produced between a certain start date and end date. It gets the following parameters, next to the standard parameters.

- fromdate: a date in YYYY-MM-DD format that specifies the start date (including the date itself) of the logs.
- todate: a date in YYYY-MM-DD format that specifies the end date (including the date itself) of the logs.

This call returns the following:

- url: a url that is relative to the url of Kofax Customer Communications Manager (i.e. http://<ccm server>:8081/ccm/).

System Check

The System Check functionality allows you to verify that a deployed installation is up and running. These requests are placed in the CCMInteractive contract type. There are three calls that essentially check the base document composition functionality without having to define appropriate content in the CCM Designer first:

SystemCheck

This SystemCheck call checks the non-interactive document composition functionality. It will produce a test document (PDF). This call takes the following parameter, next to the standard ones.

- text: a text that ends up in the produced test document.

If successful, this call returns:

- response: the number of milliseconds that it took the check to complete.
- pdf: a base-64 encoded result PDF document.

SystemCheckInteractive

The following two calls check the interactive composition functionality. In order to use these calls you will need to set up a CCM ComposerUI for HTML5 web application first. The use of these calls is analogous to the use of the ComposeInteractive calls.

SystemCheckInteractiveStart

This call is equivalent to the Compose InteractiveStart calls. It will initiate an interactive run that produces a PDF test document. It takes the following parameter next to the standard ones:

- text: a text that is included in the produced test document (after the interactive run has finished).

The results of this call are:

- a session identifier for the run
- a url that is relative to the url of Kofax Customer Communications Manager (i.e. http://<ccm server>:8081/ccm/). This url refers to JSON data that serves as input to your interactive CCM ComposerUI for HTML5 web application.

After the interactive runs has completed, the SystemCheckInteractiveGet call can be used to retrieve the produced PDF document.

SystemCheckInteractiveGet

This call can be used to obtain the result of a run that was initiated earlier via SystemCheckInteractiveStartV1. It takes the following parameter next to the standard ones:

- the session identifier of the run that has been started earlier.

The result of this call is:

- document: a base-64 encoded result PDF document.

Examples

Example request

Below an example for a SystemCheckV1 request can be found:

```
POST endpoint HTTP/1.1
Host: hostname
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "http://www.aiasoftware.com/cloud/v1/system/check/v1"

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header/>
   <soapenv:Body>
      <v1:SystemCheckV1Request>
         <v1:partner>CCM</v1:partner>
         <v1:customer>Local</v1:customer>
         <v1:contracttypename>CCMInteractive</v1:contracttypename>
         <v1:contracttypeversion>V1</v1:contracttypeversion>
         <v1:jobid>CheckSystem</v1:jobid>
         <v1:text>Check the system</v1:text>
      </v1:SystemCheckV1Request>
   </soapenv:Body>
</soapenv:Envelope>
```

Example response

```
Below an example for a SystemCheckV1 response can be found:
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <tns:SystemCheckV1Response
xmlns:tns="http://www.aiasoftware.com/cloud/v1">
         <tns:requestinfo>
            <tns:partner>CCM</tns:partner>
            <tns:customer>Local</tns:customer>
            <tns:contracttypename>CCMInteractive</tns:contracttypename>
            <tns:contracttypeversion>V1</tns:contracttypeversion>
            <tns:jobid>CheckSystem</tns:jobid>
         </tns:requestinfo>
         <tns:response>885</tns:response>
         <tns:document>VGhlIHJlc3VsdCBkb2N1bWVudC4=</tns:document>
      </tns:SystemCheckV1Response>
   </soap:Body>
</soap:Envelope>
```

CCMInteractive V1 API Reference

The CCMInteractive V1 contract type of Kofax Customer Communications Manager contains the following web services:

- AdminGetLogsV1
- DesignerStartSessionV1
- DesignerAddFieldsV1
- DesignerAddUserV1
- DesignerListDocumentTemplatesV1
- DesignerGetLetterbookV1
- DesignerListLetterbooksV1
- DesignerListProjectsV1
- ComposeDocxInteractiveStartV1
- ComposePdfInteractiveStartV1
- ComposeDocxInteractiveGetV1
- ComposePdfInteractiveGetV1
- ComposeInteractiveFinishV1
- ComposeDocxV1
- ComposePdfV1
- SystemCheckV1
- SystemCheckInteractiveStartV1
- SystemCheckInteractiveGetV1

AdminGetLogsV1

The AdminGetLogsV1 call retrieves a zip file with the CCM logs that have been produced between a certain start date and end date.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
fromdate	message	A date in YYYY-MM-DD format that specifies the start date of the logs.
todate	message	A date in YYYY-MM-DD format that specifies the end date of the logs.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
url	message	A relative url that gives access to a zipped set of logs. These logs are organised in a folder structure that reflects the internal components of CCM Interactive.

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

ComposeDocxV1

The ComposeDocxV1 call composes a docx document from a document template.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
project	repositoryobjectname	The Designer project that contains the document template that you want to use.
documenttemplate	repositoryobjectname	The document template that you want to use.
status	message	Optional: The status of the document template that you want to use. The available statuses are "Current", "Accepted" or "Published". If omitted "Published" will be used.
databackbonexml	base64Binary	The data backbone XML for the document template. This must be the contents of a data backbone XML file, base-64 encoded.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
document	base64Binary	A base-64 encoded docx document.
databackbonexml	base64Binary	A base-64 encoded data backbone XML

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

ComposeDocxInteractiveStartV1

The ComposeDocxInteractiveStartV1 call starts an interactive run that composes a docx document from a document template or a letterbook.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
project	repositoryobjectname	The Designer project that contains the document template, or the letterbook that you want to start an interactive run for.
documenttemplate	repositoryobjectname	Optional: If specified, the document template that you want to start an interactive run for. Either a documenttemplate or letterbook must be specified, but not both.
lettterbook	repositoryobjectname	Optional: If specified, the letterbook that you want to start an interactive run for. Either a documenttemplate or letterbook must be specified, but not both.
status	message	Optional: The status of the document template that you want to use. The available statuses are "Current", "Accepted" or "Published". If omitted "Published" will be used.
previewformat	message	If specified, the format of the document previews that you want to have during the interactive run. This can be either PDF or html. Please note that html conversions proceed via the MakeHTMLDocument.dss core scripting exit point. This exit point makes uses of the msxsl transformation tool, which may not work out of the box. Please make sure that this tool is installed on the system and that it is in the binary search path.
databackbonexml	base64Binary	The data backbone XML for the document template. This must be the contents of a data backbone XML file, base-64 encoded.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**

Field	Type	Description
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
itpcloudid	sessionId	An identifier for the interactive run.
url	url	A url that is relative to the url of Kofax Customer Communications Manager (i.e. http://<ccm server>:8081/ccm/). This url refers to JSON data that serves as input to your interactive ComposerUI web application.

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

ComposeDocxInteractiveGetV1

The ComposeDocxInteractiveGetV1 call gets the results for an interactive run that was started earlier via ComposeDocxInteractiveStartV1.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
itpcloudid	sessionId	Specifies the identifier of the run that has been started earlier.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
document	sessionId	A base-64 encoded docx document.
databackbonexml	url	A base-64 encoded data backbone XML.

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

ComposePdfV1

The ComposePdfV1 call composes a PDF document from a document template.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
project	repositoryobjectname	The Designer project that contains the document template that you want to use.
documenttemplate	repositoryobjectname	The document template that you want to use.
status	message	Optional: The status of the document template that you want to use. The available statuses are "Current", "Accepted" or "Published". If omitted "Published" will be used.
securedocument	boolean	Optional: Indicates whether a secured PDF document must be produced. If omitted, the default value is false.
databackbonexml	base64Binary	The data backbone XML for the document template. This must be the contents of a data backbone XML file, base-64 encoded.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
document	base64Binary	A base-64 encoded PDF document.
databackbonexml	base64Binary	A base-64 encoded data backbone XML

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

ComposePdfInteractiveStartV1

The ComposePdfInteractiveStartV1 call starts an interactive run that composes a PDF document from a document template or a letterbook.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
project	repositoryobjectname	The Designer project that contains the document template, or the letterbook that you want to start an interactive run for.
documenttemplate	repositoryobjectname	Optional: If specified, the document template that you want to start an interactive run for. Either a documenttemplate or letterbook must be specified, but not both.
lettterbook	repositoryobjectname	Optional: If specified, the letterbook that you want to start an interactive run for. Either a documenttemplate or letterbook must be specified, but not both.
status	message	Optional: The status of the document template that you want to use. The available statuses are "Current", "Accepted" or "Published". If omitted "Published" will be used.
previewformat	message	Optional: If specified, the format of the document previews that you want to have during the interactive run. If omitted, no preview will be shown. Allowed values: "pdf".
databackbonexml	base64Binary	The data backbone XML for the document template. This must be the contents of a data backbone XML file, base-64 encoded.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
itpcloudid	sessionId	An identifier for the interactive run.
url	url	A url that is relative to the url of Kofax Customer Communications Manager (i.e. http://<ccm server>:8081/ccm/). This url refers to JSON data that serves as input to your interactive ComposerUI web application.

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

ComposePdfInteractiveGetV1

The ComposePdfInteractiveGetV1 call gets the results for an interactive run that was started earlier via ComposePdfInteractiveStartV1.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
itpcloudid	sessionId	Specifies the identifier of the run that has been started earlier.
securedocument	boolean	Optional: Indicates whether a secured PDF document must be produced. If omitted, the default value false will be used.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
document	sessionId	A base-64 encoded PDF document.
databackbonexml	url	A base-64 encoded data backbone XML.

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

ComposeInteractiveFinishV1

The ComposeInteractiveFinishV1 call indicates that a previously started interactive run has finished. This allows CCM Interactive to free system resources early.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
itpcloudid	sessionId	Specifies the identifier of the run that has been started earlier.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

DesignerAddUserV1

The DesignerAddUserV1 call allows you to create new CCM Designer users.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
user	message	The name of the user that will be added. This name may contain spaces.
role	message	The role of the user.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

DesignerAddFieldsV1

The DesignerAddFielsV1 call allows you to add fields to an existing fieldset. This works for fieldsets in the root of the fieldsets folder.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
project	repositoryobjectname	The name of the project to which the fields should be added.
fieldset	repositoryobjectname	The name of the fieldset to which the fields should be added. This fieldset has to be in the root of the fieldsets folder.
fieldlist	message	A comma separated list of fields that have to be added to the specified fieldset.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

DesignerStartSessionV1

The DesignerStartSessionV1 call starts a designer session. It will return a relative url that provides access to CCM Designer.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
user	message	Reserved. This parameter is currently being ignored.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
itpcloudid	sessionId	An identifier for the interactive run.
url	url	A url that is relative to the url of Kofax Customer Communications Manager (i.e. http://<ccm server>:8081/ccm/). This url will lead to the login page of CCM Designer.

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

DesignerListProjectsV1

The DesignerListProjects returns an XML structure that describes all available projects..

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
projects	base64Binary	A base-64 encoded XML structure that lists the available projects.

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

DesignerListDocumentTemplatesV1

The DesignerListDocumentTemplatesV1 call returns an XML structure that describes all available document templates in a particular project.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
project	repositoryobjectname	The name of the project of which the document templates must be listed.
status	message	Optional: The status of the listed document templates that are requested. The available statuses are "Current", "Accepted" or "Published". If omitted "Published" will be used.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
documenttemplates	base64Binary	A base-64 encoded XML structure that lists the available document templates in the project.

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

DesignerListLetterbooksV1

The DesignerListLetterbooksV1 call returns an XML structure that describes all available letter books in a particular project.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
project	repositoryobjectname	The name of the project of which the letter books must be listed.
status	message	Optional: The status of the listed letterbooks that are requested. The available statuses are "Current", "Accepted" or "Published". If omitted "Published" will be used.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
letterbooks	base64Binary	A base-64 encoded XML structure that lists the available letter books in the project.

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

DesignerGetLetterbookV1

e DesignerGetLetterbookV1 call returns an XML structure that describes a particular letter book.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
project	repositoryobjectname	The name of a project.
letterbook	repositoryobjectname	The name of a letterbook.
status	message	Optional: The status of the letterbook that you want to use. The available statuses are "Current", "Accepted" or "Published". If omitted "Published" will be used.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
letterbook	base64Binary	A base-64 encoded XML structure that describes the specified letter book.

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

SystemCheckV1

Checks whether the base non-interactive document composition works. This call will produce a PDF test document.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
text	message	A text that ends up in the produced test document.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
response	message	The number of milliseconds that it took to run the test.
document	base64Binary	A base-64 encoded PDF document.

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

SystemCheckInteractiveStartV1

This call will start an interactive test run. Its use is analogous to ComposePdfInteractiveStartV1, so it needs to be integrated in a ComposerUI application.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
text	message	A text that ends up in the produced test document.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
itpcloudid	sessionId	An identifier for the interactive run.
url	url	A url that is relative to the url of Kofax Customer Communications Manager (i.e. http://<ccm server>:8081/ccm/). This url refers to JSON data that serves as input to your interactive ComposerUI web application.

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

SystemCheckInteractiveGetV1

This call obtains the PDF test document that has been produced by the interactive run that was initiated via SystemCheckInteractiveStartV1.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
itpcloudid	sessionId	Specifies the identifier of the run that has been started earlier.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
document	sessionId	A base-64 encoded PDF test document.

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

CCM Interactive V2 API Reference

The CCMInteractive V2 contract type of Kofax Customer Communications Manager contains the following web services:

- AdminGetLogsV2
- ComposeDocumentPackV1
- ComposeDocumentPackGetV1
- ComposeDocumentPackInteractiveStartV1
- ComposeInteractiveGetSuspendedSessionV1
- ComposeInteractiveResumeSuspendedSessionV1
- DesignerAddFieldsV1
- DesignerGetDataBackBoneDefinitionV1
- DesignerGetDocumentPackTemplateV1
- DesignerGetDocumentTemplateV1
- DesignerGetLetterbookV2
- DesignerListLetterbooksV2
- DesignerListProjectsV2
- DesignerListTemplatesV1
- DesignerStartSessionV2
- DocumentPackConvertV1
- DocumentPackInteractiveModifyV1
- DocumentPackInteractiveModifyFromReviewV1
- DocumentPackInteractiveReviewV1
- FinishSessionV1
- SystemCheckV1
- SystemCheckInteractiveGetV2
- SystemCheckInteractiveStartV2

AdminGetLogsV2

This call will provide access to a set of logs that have been produced between a certain start date and end date.

It gets the following parameters, next to the standard parameters:

Field	Type	Description
fromdate	message	A date in YYYY-MM-DD format that specifies the start date (including the date itself) of the logs.
todate	message	A date in YYYY-MM-DD format that specifies the end date (including the date itself) of the logs.

The response will contain the following fields:

Field	Type	Description
url	url	A URL that is relative to the url of Kofax Customer Communications Manager (i.e. http://<ccm server>:8081/ccm/).

ComposeDocumentPackV1

ComposeDocumentPackV1 is used to create a document pack from Document Templates and Document Pack Templates which are not interactive. The document pack is returned in the response, and can also be retrieved by performing a ComposeDocumentPackGetV1.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
project	repositoryobjectname	Name of the CCM Repository project.
templatetype	name	Either ‘documentpacktemplate’ or ‘documenttemplate’.
templatename	repositoryobjectname	Name of the template object to compose the Document Pack from.
databackbonexml	base64Binary	Base64 encoded representation of the data backbone XML.
status	name	Optional. Status of the selected template object. Either 'published', 'accepted' or 'current'. Defaults to 'published'.

The response will contain the following fields:

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier.
documentpack	base64Binary	Base64 encoded binary representation of the resulting Document Pack data structure.

ComposeDocumentPackGetV1

The ComposeDocumentPackGetV1 usually is the follow-up web-service after finishing the interactive part of running a template. It can be used to retrieve the result of all composition calls that create a document pack.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier. This is the CCM session identifier which was returned at the start of the composition call of which you would like to retrieve the resulting document pack.

The response will contain the following fields:

Field	Type	Description
documentpack	base64Binary	Base64 encoded binary representation of the document pack.

ComposeDocumentPackInteractiveStartV1

ComposeDocumentPackInteractiveStartV1 is used to create a Document Pack based on either an interactive Document Template or an interactive Document Pack Template. The call will result in a session id and (partial) url which can be used to start the interaction in ComposerUI. After the interaction has been completed the resulting document pack can be retrieved using ComposeDocumentPackGetV1.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
project	repositoryobjectname	Name of the CCM Repository project.
templatetype	name	Either ‘documentpacktemplate’ or ‘documenttemplate’.
templatename	repositoryobjectname	Name of the template object to compose the Document Pack from.

Field	Type	Description
databackbonexml	base64Binary	Base64 encoded representation of the data backbone XML.
allowsuspend	boolean	Optional. Enables option to suspend interactive composition when set to true. Defaults to False.
status	name	Optional. Status of the selected template object. Either 'published', 'accepted' or 'current'. Defaults to 'published'.

The response will contain the following fields:

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier.
url	url	Relative URL for starting the interactive process.

ComposeInteractiveGetSuspendedSessionV1

ComposeInteractiveGetSuspendedSessionV1 can be used to retrieve a suspended session from the <ccm_server>. This can only be done for interactive sessions that have been initiated with their allow suspend parameter set to true. This functionality can be used to pause longer interactive processes, for example if you need input from others, or if you want to continue on another day.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier. This is the CCM session identifier which was returned at the start of the interactive process that has been suspended.

The response will contain the following fields:

Field	Type	Description
sessionfile	base64Binary	A base64 encoded binary representation of a CCM session archive data structure. This file can be used in ComposeInteractiveResumeSuspendedSessionV1 to resume the suspended interaction.

ComposeInteractiveResumeSuspendedSessionV1

ComposeInteractiveResumeSuspendedSessionV1 can be used to resume a suspended session retrieved by ComposeInteractiveGetSuspendedSessionV1. The returned values can be used to resume the interactive process at the point the user left, with all values still set to the value he set them to.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
sessionfile	base64Binary	A base64 encoded binary representation of a CCM session archive data structure. This is the suspended session retrieved by ComposeInteractiveGetSuspendedSessionV1 and contains all information to continue the interactive process where the user left of.

The response will contain the following fields:

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier. This is the CCM session identifier of the suspended session.
url	url	The URL from which the interactive session can be resumed.

DesignerAddFieldsV1

The DesignerAddFielsV1 call allows you to add fields to an existing fieldset. This works for fieldsets in the root of the fieldsets folder.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
project	repositoryobjectname	The name of the project to which the fields should be added.
fieldset	repositoryobjectname	The name of the fieldset to which the fields should be added. This fieldset has to be in the root of the fieldsets folder.
fieldlist	message	A comma separated list of fields that have to be added to the specified fieldset.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

DesignerGetDataBackBoneDefinitionV1

DesignerGetDataBackBoneDefinitionV1 can be used to retrieve the data backbone definition. The result will contain a Base-64 Encoded zip file containing three XSDs describing the data backbone. The data backbone definition is useful when building and filling a testdata.xml to use in the composition process.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
project	repositoryobjectname	The name of the project from which to retrieve the data backbone.
status	name	Optional. The status of the data backbone that you want to retrieve. The available statuses are "current", "accepted" or "published". If omitted "published" will be used.

The response will contain the following fields:

Field	Type	Description
databackbonedefinition	base64Binary	A Base-64 Encoded zip file containing three XSDs describing the data backbone.

DesignerGetDocumentPackTemplateV1

DesignerGetDocumentPackTemplateV1 returns a document pack template as defined in the CCM Designer as Base-64 encoded XML. This XML can be used to check which documents will be produced for this document pack and which documents are optional for example.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
project	repositoryobjectname	The name of a project from which to retrieve the document pack template.
documentpacktemplate	repositoryobjectname	The name of the requested document pack template.
status	name	Optional: The status of the document pack template that you want to retrieve. The available statuses are 'current', 'accepted' or 'published'. If omitted 'published' will be used.

The response will contain the following fields:

Field	Type	Description
documentpacktemplate	base64Binary	A Base-64 Encoded XML describing the requested document pack template. The format of this XML can be found in the section Format Definitions.

DesignerGetDocumentTemplateV1

DesignerGetDocumentTemplateV1 returns a document template as defined in the CCM Designer

as Base-64 encoded XML. This XML can be used to check which parameters this document template uses or on which repository object it is based.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
project	repositoryobjectname	The name of the project from which to retrieve the document template.
documenttemplate	repositoryobjectname	The name of the requested document template.
status	name	Optional. The status of the document pack template that you want to retrieve. The available statuses are 'current', 'accepted' or 'published'. If omitted 'published' will be used.

The response will contain the following fields:

Field	Type	Description
documenttemplate	base64Binary	A Base-64 Encoded XML describing the requested document pack template. The format of this XML can be found in the section Format Definitions.

DesignerGetLetterbookV2

The DesignerGetLetterbookV2 call returns an XML structure that describes a particular letterbook. A letterbook is a collection of document templates, which usually are related in a way. It can be used by business applications to provide end-users with a list of document templates to choose from.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
project	repositoryobjectname	The name of the project from which to retrieve the letterbook.
letterbook	repositoryobjectname	The name of the letterbook that is to be retrieved.
status	name	Optional. The status of the letterbook that you want to retrieve. The available statuses are 'current', 'accepted' or 'published'. If omitted 'published' will be used.

The response will contain the following fields:

Field	Type	Description
letterbook	base64Binary	A base-64 encoded XML structure that describes the specified letter book. The format of this XML can be found in the section Format Definitions.

DesignerListLetterbooksV2

The DesignerListLetterbooksV2 call returns an XML structure that describes all available letter books in a particular project. Letterbooks usually represent a category or collection of document templates which have some cohesion. The list of letterbooks can for example be used by the integrating application to help in selecting a document template.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
project	repositoryobjectname	The name of the project of which the letter books must be listed.
status	name	Optional: Only letter books that have the requested status will be listed. The available statuses are 'current', 'accepted' or 'published'. If omitted 'published' will be used.

The response will contain the following fields:

Field	Type	Description
letterbooks	base64Binary	A base-64 encoded XML structure that lists the available letter books in the project. The format of the returned XML can be found in the Format Definitions section.

DesignerListProjectsV2

DesignerListProjectsV2 returns a base64 Encoded XML structure that describes all available projects.

This call only requires the default parameters.

The response will contain the following fields:

Field	Type	Description
projects	base64Binary	A base-64 encoded XML structure that lists the available projects. The format of the returned XML can be found in the format definition section.

DesignerListTemplatesV1

DesignerListTemplatesV1 is used to retrieve an Base 64 encoded XML representing a list of templates from the CCM Designer. This list includes both document templates and document pack templates which are differentiable by the type attribute. These templates can be used in other calls, for example: ComposeDocumentPackV1 to create a document pack based on the template, or getDocumentTemplate to get more information about a certain template.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
project	repositoryobjectname	The name of the project of which the templates must be listed.
status	name	Optional. Only templates that have the requested status will be listed. The available statuses are 'current', 'accepted' or 'published'. If omitted 'published' will be used.
ondemandonly	boolean	Optional. Filter applied to template list. From CCM 5.0 and up the list will contain only templates that are not interactive when set to true.

The response will contain the following fields:

Field	Type	Description
templatelist	base64Binary	A base-64 encoded XML structure that lists the available templates in the project. The format of the returned XML can be found in the Format Definitions section

DesignerStartSessionV2

DesignerStartSessionV2 is used to acces the CCM Designer. It wil return a relative URL to the login page of the CCM Designer. Within the CCM Designer users can manage the content in the repository.

This interface requires no additional parameters.

The response will contain the following fields:

Field	Type	Description
ccmsessionid	sessionId	CCM Contract Manager session identifier.
url	url	A URL that is relative to the URL of Kofax Customer Communications Manager (i.e. http://<ccm server>:8081/ccm/). This URL will load CCM Designer.

DocumentPackConvertV1

DocumentPackConvertV1 can be used to convert the documents in a document pack to another file format, for example pdf. The new file format will be added to the document pack. Existing documents will not be removed.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
documentpack	base64Binary	Base64 encoded binary representation of a Document Pack data structure. This document pack contains the documents that will be converted.
outputformat	message	The output format that one requires. Current only "pdf" and "pdf/a-1b" are supported.

The response will contain the following fields:

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier. The identifier of the session that was used for the conversion.
documentpack	base64Binary	Base64 encoded binary representation of a Document Pack data structure. This document pack contains the converted documents and updated references to these; otherwise it is identical to the pack that was originally sent on the request. Please note that only docx documents will be converted.

DocumentPackInteractiveModifyV1

DocumentPackInteractiveModifyV1 can be used when the document has to be altered after its original composition has been finished. For example, if the document pack has been reviewed by a second person and some feedback has been provided to the first person who needs to make some small modifications.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
documentpack	base64Binary	Base64 encoded binary representation of a Document Pack data structure. This document pack contains the documents that can be modified.

The response will contain the following fields:

Field	Type	Description

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier.
url	url	Relative URL for starting the interactive process.

DocumentPackInteractiveModifyFromReviewV1

DocumentPackInteractiveModifyFromReviewV1 is used after a Document Pack Review has been started, typically when a user found a problem during the review and wants to correct it. This call restarts the interactive process that was used to create the document pack, only now with all changes made during the initial composition present.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
ccmsessionid	sessionId	CCM Contract Manager session identifier. This is the session identifier returned by the DocumentPackInteractiveReviewV1 that was used to initiate the review session.

The response will contain the following fields:

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier.
url	url	Relative URL for starting the interactive process

DocumentPackInteractiveReviewV1

DocumentPackInteractiveReviewV1 can be used to inspect an interactively composed document pack. It opens an interactive reviewing tool that allows the user to review all produced documents. If a user finds any errors during this review, DocumentPackInteractiveModifyFromReviewV1 can be used to make changes.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
documentpack	base64Binary	Base64 encoded binary representation of a Document Pack data structure. This document pack contains the documents that can be reviewed.

The response will contain the following fields:

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier.
url	url	Relative URL for starting the interactive process.

FinishSessionV1

FinishSessionV1 can be used to explicitly notify CCM all interaction with the session has been finished.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier. The identifier of the session that should be terminated.

This call only returns the default parameters.

SystemCheckV1

Checks whether the base non-interactive document composition works. This call will produce a PDF test document.

A request takes the following fields. See Type descriptions for more information on the types.

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes
text	message	A text that ends up in the produced test document.

For logging and traceability, the response to the request will contain the following request fields nested in a **requestinfo** node:

Field	Type	Description
partner	name	**CCM**
customer	name	**Local**
contracttypename	name	**CCMInteractive**

Field	Type	Description
contracttypeversion	version	**V1**
jobid	jobid	Custom identifier for diagnostic purposes

In addition the response will contain the following fields:

Field	Type	Description
response	message	The number of milliseconds that it took to run the test.
document	base64Binary	A base-64 encoded PDF document.

In case of an error, the response will contain a SOAP Fault structure. See Error handling for more details on error handling.

SystemCheckInteractiveGetV2

SystemCheckInteractiveGetV2 obtains the PDF test document that has been produced by the interactive run that was initiated via SystemCheckInteractiveStartV2.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
ccmsessiondid	sessionid	The CCM Contract Manager session identifier that was return by the SystemCheckInteractiveStartV2 used to initiate the test run.

The response will contain the following fields:

Field	Type	Description
document	base64Binary	A base-64 encoded PDF test document.

SystemCheckInteractiveStartV2

SystemCheckInteractiveStartV2 will start an interactive test run. Like any interactive call it needs to be integrated in to a CCM ComposerUI application. It will produce a simple test document to verify the ccm installation is up and running and can produce documents as intended. This document can be retrieved using SystemCheckInteractiveGetV2 using the returned ccmsessionid.

To start the call the following parameters have to be provided in addition to the default parameters, unless stated otherwise:

Field	Type	Description
text	message	Text to be added to the test.

The response will contain the following fields:

Field	Type	Description
ccmsessiondid	sessionid	The CCM Contract Manager session identifier to be used by SystemCheckInteractiveGetV2 to retrieve the resulting document.
url	url	Relative URL for starting the interactive process.

CCMDistribution V1 API Reference

The CCMDistribution V1 contract type of Kofax Customer Communications Manager contains the following web services:

- DocumentPackDistributeV1
- DocumentPackDistributeWorkplaceV1
- DocumentPackSignV1

DocumentPackSignV1

DocumentPackSignV1 submits a previously produced document pack to a Kofax SignDoc installation for signing and further handling. This call should be used when the documents in a generated Document Pack have to be digitally signed.

To start the call the following parameters have to be provided in addition to the default parameters:

Field	Type	Description
documentpack	base64Binary	Base64 encoded binary representation of a Document Pack data structure. This is the document pack that will be submitted for signing and it should contain at least one MS Word document with signature lines that can be processed by Kofax SignDoc.
signdocurl	url	URL of the SignDoc Rest API to which the Document Pack will be submitted. This URL should be formatted like <signdocServerAddress>/<SignDocApplication>/rest/ i.e.: "http://signdocserver.com/cirrus/rest".
signdocapikey	message	The API key used for authenticating calls to the SignDoc API. This key can be retrieved from the SignDoc installation.
signdocpackageid	message	The SignDoc id (the name) of the signing package. This should be a new (unique) id consisting of alphanumerical characters, underscores and minus signs (See the SignDoc documentation for details). If an existing id or empty id is passed, an error will be raised.

The response will contain the following fields:

Field	Type	Description
signdocpackageid	message	The SignDoc id of the created signing package. This is currently the same as the signdocpackageid that is passed as an input argument. We advise application builders to use the returned signdocpackageid for future reference, as future releases may return a modified id.

DocumentPackDistributeWorkPlaceV1

DocumentPackDistributeWorkplaceV1 submits a previously produced document pack to a Perceptive Workplace server.

To start the call the following parameters have to be provided in addition to the default parameters:

Field	Type	Description
documentpack	base64Binary	Base64 encoded binary representation of a Document Pack data structure. This is the document pack that will be submitted for distribution to workplace.
user	message	The user name of the account that will be used to upload documents.
password	message	The password of the account that will be used to upload documents.
workplaceurl	url	URL of the workplace server to which the Document Pack will be submitted. This URL should be formatted like http://<server>:<port>[/<service-context>] i.e.: "https://pw.psft.co/fns-service/".
location	message	A full 'path' to the space or folder in which the documents will be placed. This path must start with 2 separator characters. This character determines where new subfolders will be generated along this path. In principle this can be any character that does not occur in any of the object names (space or folder), but typically this will be a '/'.

Field	Type	Description
outputformat	message	Optional. The requested output format for the documents that will be placed in workplace. Currently only 'native', 'pdf' and 'pdf/a-1b' are supported. The meaning of these are as follows: • native: The original document format will be uploaded to workplace. For quick documents and document templates that are based on a master template this means: docx. For static documents this means: the document as it is stored in CCM Repository. Import document are uploaded 'as is'. • pdf: If a PDF version of a document is already present, that will be uploaded. Otherwise, if a docx version is present, it will be converted to PDF and the result will be uploaded. The remaining documents are uploaded 'as is'. • pdf/a-1b: If a PDF/A-1b version of a document is already present, that will be uploaded. Otherwise, if a docx version is present, it will be converted to PDF/A-1b and the result will be uploaded. The remaining documents are uploaded 'as is'. This includes 'ordinary' PDF documents that have no docx variant. The default value of this parameter is 'native'.

The response will contain the following fields:

Field	Type	Description
workplaceid	message	The workplace id of the space or folder to which the documents have been uploaded.

DocumentPackDistributeV1

DocmentPackDistributeV1 is used to distribute a previously created document package by CCM. The actual Distribution functionality will be defined in the CCM Core DistributeDocumentPack exit point.

To start the call the following parameters have to be provided in addition to the default parameters:

Field	Type	Description
documentpack	base64Binary	Base64 encoded binary representation of a Document Pack data structure. This is the document pack that will be distributed.

Field	Type	Description
channel	name	The Distribution channel to distribute to. Possible values are: 'print', 'email', 'portal' and 'archive'.
organizationalmetadata	base64Binary	Base64 encoded binary representation of organizational metadata. This data will be passed to the exit point "as is".

Except for the default requestinfo structure this call does not return anything.

API Type descriptions

The following table describes the field types of the WebServices requests and responses. This applies to all contract types.

Type	Format description	Example
base64Binary	Standard XSD type 'base64Binary'. A file encoded as an ASCII string according to the Base64 encoding specification.	"QSBCYXNlNjQgZW5jb2RlZCB0ZXh0IGZpbGU="
boolean	Standard XSD type 'boolean'. Matches regular expression: `true|false|1|0`	"true"
jobid	Text matching regular expression: `[A-Za-z0-9]+`	"Ref12345"
message	Text matching regular expression: `.*`	"Unknown session ID."
name	Text matching regular expression: `[A-Za-z0-9]+`	"AiaSoftware"
repositoryobjectname	The name of an object in the repository, consisting of text matching regular expression: [^:?/<>*"\\()]+	"My Document Template"
sessionid	Text matching regular expression: `[A-Za-z0-9\-]+`	"3ff62c53-a2c6-4417-b1c5-379a0a8db0bf"
url	Standard XSD type 'anyURI'. A relative or absolute URL according standard URL specifications.	"/itpmdk/login.jsf"

Format Definitions

DocumentPackManifestXML

The Document Pack Manifest XML can be found at the root of each document pack and describes the contents of the document pack. Most notably it specifies for the slots where documents can be found within the pack.

Databackbone

The databackbone node at top level indicate either the result of the Data Preparation Template, or the input databackbone when no Data Preparation Template is present in a Document Pack Template.
In case of a Document Template the databackbone mentioned here will be equal to the result of the Document Template as preparation will take place there.
Databackbone nodes for Document Templates indicate the resulting databackbone after the template was run for this item.

Item

Each item node refers to a document within the Document Pack.
For every Document Template, Import or Upload that is included in the Document Template Pack there must be a matching item node. The result node indicates the result documents available for this item.

Closed Loop Identifier

This element is only present when KTA has passed a Closed Loop Identifier on the request and/or the Document Template has modified this.

The XML will be conform the following XSD:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://www.kofax.com/ccm/documentPack/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="documentPack">
    <xs:complexType>
      <xs:sequence>
            <xs:element name="databackbone" type="xs:string">
                  <xs:annotation>
                        <xs:documentation>
                              This element contains the path to the
databackbone within the ZIP file. This path is relative to the root of the
ZIP file.

                              The databackbone mentioned here is either
the result of the Data Preparation Template or the input databackbone when
no Data Preparation Template is present in a DocumentPackTemplate.
                                          In case of a DocumentTemplate the
databackbone mentioned here will be equal to the result of the
DocumentTemplate as Preparation will take place there.
```

```
                        </xs:documentation>
                    </xs:annotation>
                </xs:element>
            <xs:element maxOccurs="unbounded" name="item">
              <xs:annotation>
                <xs:documentation>
                  Each item node refers to a document within the Document Pack.
                  For every Document Template, Import or Upload that is included
in the Document Template Pack there must be a matching item node.
                </xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="description" type="xs:string">
                    <xs:annotation>
                      <xs:documentation>
                        Human-readable description of the element in the
Document Pack.
                      </xs:documentation>
                    </xs:annotation>
                  </xs:element>
                  <xs:element name="closedLoopIdentifier" type="xs:string"
minOccurs="0">
                    <xs:annotation>
                      <xs:documentation>
                        Closed Loop Identifier.

                        This element is only present when KTA has passed a Closed
Loop Identifier on the request and/or the Document Template has modified this.
                      </xs:documentation>
                    </xs:annotation>
                  </xs:element>
                  <xs:choice>
                    <xs:annotation>
                      <xs:documentation>
                        This choice indicates the type of element that produced
the document and its name within the Document Template Pack.
                      </xs:documentation>
                    </xs:annotation>
                    <xs:element name="uploadDocument">
                      <xs:annotation>
                        <xs:documentation>
                          Document uploaded by the user.
                        </xs:documentation>
                      </xs:annotation>
                    </xs:element>
                    <xs:element name="importDocument">
                      <xs:annotation>
                        <xs:documentation>
                          Document provided by the calling application.
                        </xs:documentation>
                      </xs:annotation>
                    </xs:element>
                    <xs:element name="documentTemplate">
                      <xs:annotation>
                        <xs:documentation>
                          Document produced by a CCM Document Template.
                        </xs:documentation>
                      </xs:annotation>
                      <xs:complexType>
```

```
                    <xs:sequence>
                      <xs:element name="description" type="xs:string">
                        <xs:annotation>
                          <xs:documentation>
                            Description of ...
                          </xs:documentation>
                        </xs:annotation>
                      </xs:element>
                    </xs:sequence>
                    <xs:attribute name="name" type="xs:string"
use="required">
                      <xs:annotation>
                        <xs:documentation>The name of the Document Template
selected in this slot. (currently always matches the slot
name)</xs:documentation>
                      </xs:annotation>
                    </xs:attribute>

                  </xs:complexType>
                </xs:element>
              </xs:choice>
            <xs:element name="databackbone" type="xs:string">
                          <xs:annotation>
                                <xs:documentation>
                                      This element contains the path
to the databackbone within the ZIP file. This path is relative to the root
of the ZIP file.

                                      The databackbone mentioned here
is the resulting databackbone after the template run for this item.
                                </xs:documentation>
                          </xs:annotation>
                       </xs:element>
                   <xs:element name="result" >
                <xs:annotation>
                  <xs:documentation>
                    The result node describes the document(s) associated
with the source.
                    If there are multiple document nodes within this
element, these nodes describe different representations of the same document.
                    (for example:
                      - The original DOCX version produced by the Document
Template.
                      - A PDF version generated from the DOCX version.
                      - A PDF/A version generated from the DOCX version.
                      - A structured XML version of the document produced
by the Document Template.
                     )
                  </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0"
name="document">
                      <xs:annotation>
                        <xs:documentation>
                          Each document node describes an instance of the
result document. These should all contain the same content, but represented
differently.
```

```
                          This element contains the path to the document
within the ZIP file. This path is relative to the root of the ZIP file.
                      </xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                      <xs:simpleContent>
                        <xs:extension base="xs:string">
                          <xs:attribute name="format" type="xs:string"
use="optional">
                            <xs:annotation>
                              <xs:documentation>
                                Generic description of the file format
(DOCX, PDF, PDF/A, etc).
                              </xs:documentation>
                            </xs:annotation>
                          </xs:attribute>
                          <xs:attribute name="mimetype"
type="xs:string" use="optional">
                            <xs:annotation>
                              <xs:documentation>
                                MIME type describing the file format.
                              </xs:documentation>
                            </xs:annotation>
                          </xs:attribute>
                          <xs:attribute name="secure" type="xs:boolean"
use="optional" default="false">
                            <xs:annotation>
                              <xs:documentation>
                                This flag is only applicable to all PDF
(and it's subttypes) files. It describes whether or not the PDF has been
secured.
                              </xs:documentation>
                            </xs:annotation>
                          </xs:attribute>
                          <xs:attribute name="verbatim"
type="xs:boolean" use="optional" default="false">
                            <xs:annotation>
                              <xs:documentation>
                                If true, this document has been included
as it was provided by the end-user or calling application, without determining
or verifying any information regarding format or mimetype.
                                                        This flag will
(currently) only be present for Import or Upload Documents.
                                                        When this flag is set
to true the format and mimetype attributes will not be set.
                              </xs:documentation>
                            </xs:annotation>
                          </xs:attribute>
                        </xs:extension>
                      </xs:simpleContent>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attribute name="name" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>
```

```
                                The name of the corresponding slot in the
Document Pack Template.


                </xs:documentation>
              </xs:annotation>
            </xs:attribute>
            <xs:attribute name="coverLetter" type="xs:boolean"
use="optional" default="false">
              <xs:annotation>
                <xs:documentation>
                  Indicates if this document is the Cover Letter of the
Document Pack.

                  There MUST be exactly ONE item which has this flag set to
true.

                  Default: false.
                </xs:documentation>
              </xs:annotation>
            </xs:attribute>
            <xs:attribute name="optional" type="xs:boolean" use="optional"
default="false">
              <xs:annotation>
                <xs:documentation>
                  Indicates that this document is produced by an optional
component in the Document Template Pack. This choice is explicitly performed
by the user or the controlling application.

                  Default: false.
                </xs:documentation>
              </xs:annotation>
            </xs:attribute>
            <xs:attribute name="conditional" type="xs:boolean"
use="optional" default="false">
              <xs:annotation>
                <xs:documentation>
                  Indicates that this document is produced by a conditional
component in the Document Template Pack. If the condition was evaluated to
a false condition no result node will be present.

                  Default: false.
                </xs:documentation>
              </xs:annotation>
            </xs:attribute>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="version" type="xs:unsignedByte" use="required"
fixed="1" >
        <xs:annotation>
          <xs:documentation>
            Indicates the version of the Manifest.xml file.

            This MUST be "1" for this version of the specification.
        </xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
```

```
</xs:schema>
```

DocumentPackTemplateXML

A Document Pack Template XML is returned by DesignerGetDocumentPackTemplateV1. It describes the structure of a document pack template by listing which slots the pack consists of and how their content is to be created or added. The slot types are documenttemplate, uploadDocument, importDocument and selection. It also contains information about whether a slot is optional or conditional, and marks the first template explicitly as cover letter.

The XML will be conform the following XSD:

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://www.kofax.com/ccm/documentPackTemplate/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.kofax.com/ccm/documentPackTemplate/1.0">
    <xs:complexType name="documentTemplateType">
        <xs:sequence>
            <xs:element name="description" type="xs:string" />
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required" />
    </xs:complexType>

    <xs:element name="documentPackTemplate">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="item">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="description" type="xs:string" />
              <xs:choice>
                <xs:element name="uploadDocument">
                  <xs:complexType>
                    <xs:attribute name="name" type="xs:string"
use="required" />
                  </xs:complexType>
                </xs:element>
                <xs:element name="importDocument">
                  <xs:complexType>
                    <xs:attribute name="name" type="xs:string"
use="required" />
                  </xs:complexType>
                </xs:element>
                <xs:element name="documentTemplate" type
="documentTemplateType" />
                <xs:element name="selection">
                  <xs:complexType>
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded"
name="documentTemplate" type="documentTemplateType"/>
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
              </xs:choice>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="optional" />
```

```
            <xs:attribute name="coverLetter" type="xs:boolean"
use="optional" />
            <xs:attribute name="optional" type="xs:boolean" use="optional"
/>
            <xs:attribute name="conditional" type="xs:boolean"
use="optional" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
        <xs:attribute name="version" type="xs:unsignedByte" use="required"
/>
        <xs:attribute name="name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

DocumentTemplateXML

A Document Template XML is returned by DesignerGetDocumentTemplateV1.
The Document Template XML describes a specific document template as defined in the designer. It contains a name and description, information about which master template it is based on, and which parameters it takes.

The XML will be conform the following XSD:

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://www.kofax.com/ccm/getDocumentTemplate"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.kofax.com/ccm/getDocumentTemplate">
    <xs:simpleType name="parameterType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Boolean"/>
            <xs:enumeration value="ContentWizard"/>
            <xs:enumeration value="Date"/>
            <xs:enumeration value="Form"/>
            <xs:enumeration value="Number"/>
            <xs:enumeration value="Text"/>
            <xs:enumeration value="TextBlock"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="documentTemplateParameterType">
          <xs:simpleContent>
                <xs:extension base="xs:string">
                      <xs:attribute name="name" type="xs:string"
use="required" />
                      <xs:attribute name="type" type="parameterType"
use="required" />
                </xs:extension>
          </xs:simpleContent>
   </xs:complexType>

    <xs:complexType name="documentTemplateParametersType">
          <xs:sequence>
                <xs:element minOccurs="0" maxOccurs="unbounded"
name="parameter" type="documentTemplateParameterType"/>
          </xs:sequence>
   </xs:complexType>
```

```
    <xs:complexType name="masterTemplateType">
          <xs:attribute name="name" type="xs:string" />
   </xs:complexType>

    <xs:element name="documentTemplate" >
          <xs:complexType>
                <xs:sequence>
                      <xs:element name="description" type="xs:string" />
                      <xs:element minOccurs="0" maxOccurs="1"
name="masterTemplate" type="masterTemplateType" />
                      <xs:element minOccurs="0" maxOccurs="1"
name="parameters" type="documentTemplateParametersType" />
                </xs:sequence>
                <xs:attribute name="version" type="xs:unsignedByte"
use="required" />
            <xs:attribute name="name" type="xs:string" use="required"/>
        </xs:complexType>
   </xs:element>
</xs:schema>
```

LetterBookXML

A Letter Book XML describes a letter book and its contents, which is a list of document templates and folders.

The XML will be conform the following XSD:

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://www.kofax.com/ccm/getLetterBook"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.kofax.com/ccm/getLetterBook">
    <xs:simpleType name ="templateType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="documentTemplate"/>
            <xs:enumeration value="documentPackTemplate"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="letterbookTemplateType">
        <xs:sequence>
            <xs:element name="description" type="xs:string" />
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required" />
        <xs:attribute name="type" type="templateType" use="required" />
    </xs:complexType>

    <xs:complexType name="letterbookFolderType">
        <xs:sequence>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element name="template" type="letterbookTemplateType"
/>
                <xs:element name="folder" type="letterbookFolderType" />
            </xs:choice>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required" />
    </xs:complexType>

    <xs:element name="letterbook">
        <xs:complexType>
```

```
            <xs:sequence>
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element name="template"
type="letterbookTemplateType" />
                    <xs:element name="folder" type="letterbookFolderType"
/>
                </xs:choice>
            </xs:sequence>
            <xs:attribute name="version" type="xs:unsignedByte"
use="required" />
            <xs:attribute name="name" type="xs:string" use="required" />
        </xs:complexType>
    </xs:element>
</xs:schema>
```

ListLetterbooksXML

The List Letterbooks XML will contain a list of letter books in a project including descriptions.

The XML will be conform the following XSD:

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://www.kofax.com/ccm/listLetterBooks"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="letterbooks">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="letterbook">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="description" type="xs:string" />
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="version" type="xs:unsignedByte" use="required"
/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

ListProjectsXML

The List Projects XML will contain a list of all projects in the repository.

The XML returned will conform to the following XSD:

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://www.kofax.com/ccm/listProjects"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="projects">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="project">
          <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required" />
```

```
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="version" type="xs:unsignedByte" use="required"
/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

ListTemplatesXML

The List Templates XML will contain a list of templates, both the document templates and the document pack templates, in a project.

The XML will conform to the following XSD:

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://www.kofax.com/ccm/listTemplates/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.kofax.com/ccm/listTemplates/1.0">
    <xs:simpleType name ="templateType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="documentTemplate"/>
            <xs:enumeration value="documentPackTemplate"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:element name="templates">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="template">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="description" type="xs:string" />
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required" />
            <xs:attribute name="type" type="templateType" use="required"
/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="version" type="xs:unsignedByte" use="required"
/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

API Error handling

In case of an error, the response message will contain a standard SOAP Fault structure. This structure will contain some error details in a structure called **AiaFaultV1**. This structure will contain the following fields. This applies to all contract types.

Field	Type	Description
errorcode	errorcode	A return code indicating the type of error.

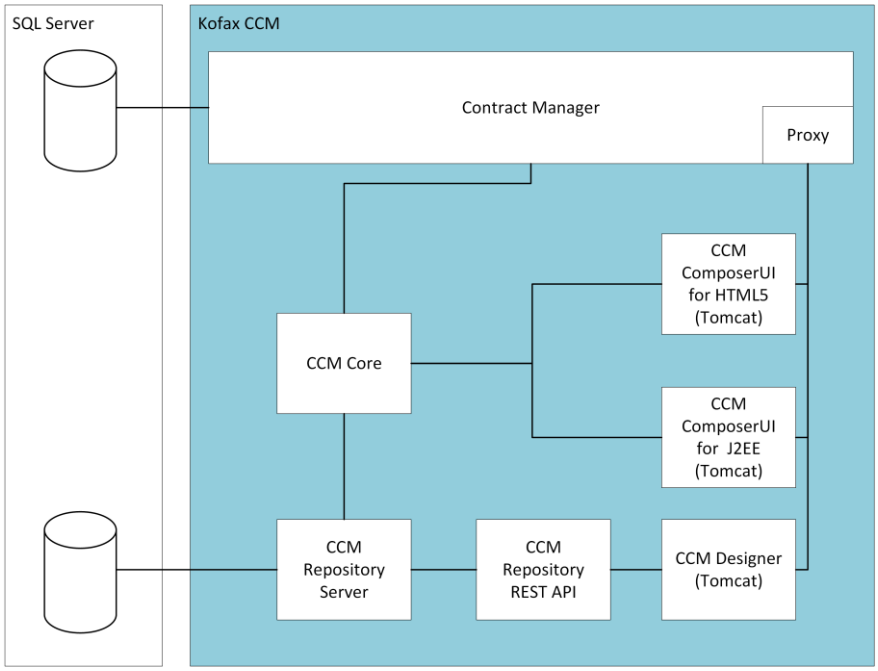
Field	Type	Description
message	message	A message describing the error.

Technical description

In this chapter there is a description of the packaged setup. It provides an overview of all the components that are available.

Component overview

This section shows you how all Kofax CCM components are configured to work together.



The Contract Manager is the single entry point to the Kofax Customer Communications Manager environment. The Contract Manager will communicate with CCM Core directly over TCP/IP. Although CCM ComposerUI for J2EE is deployed, it will not be used by default. It will be CCM ComposerUI for HTML5 instead.

After installation, the server contains two instances of CCM Core. One instance is called "core_01" and the other instance is called "core_Manager". The manager instance is for internal administrative use only. It is not available for content production purposes.

Contract Manager

The Contract Manager is the interface to Kofax Customer Communications Manager for all communications (run time, design time, system management). It is the single point of access for the Kofax Customer Communications Manager instance. Currently the Contract Manager has two roles:

1. Expose services (SOAP)
2. Proxy http traffic

The Contract Manager has its own database in which all the interface definitions will be stored.

Server administration

Open CCM Core Administrator

The CCM Core Administrator is installed on the server. It can be accessed via the start menu in the taskbar. In the list of programs there is a folder called "Kofax CCM". This folder contains the shortcut to start the CCM Core Administrator.

Add CCM Core services

In order to add CCM Core services in a simple way, a tool called 'AddCoreServices.ps1' is available in the CCM Management Folder (`CCM\Programs\5.0\Management`).

```
.\AddCoreServices.ps1 C:\MyServices
```

The argument that is passed to the tool should point to a folder that has the following structure:

```
<Service1>\<Service1>.dss
           <Service1>.ini
<Service2>\<Service2>.dss
           <Service2>.ini
...
```

Substitute `<Service1>` and `<Service2>` with your desired service names. The dss file should contain the service's Process Language definition. The ini file should contain the service's configuration as found in the dp.ini on the system where you developed the services.

Note
This feature should be used to add CCM Core services to a system that will be used for production. You are strongly encouraged to develop the services on a development system, using CCM Core Administrator.

When upgrading to a newer version, custom services should be added to that version again.

Load a CCM Repository project

To load an existing CCM Repository project into an installation, the repimport.exe tool, that can be found in `CCM\Programs\5.0\ITPMDKRepositoryServer` can be used. This tool accepts the following parameters

/file=<file>	The path to the file to be imported.
/merge, /replace or /new	The action to perform with the project. /merge will merge the project with an existing project with the same name, should such a project exist. /replace will replace an existing project with the same name, should such a project exist. /new will abort the import if a project with the same name already exists.

/roles=<roles>	*Optional. Semicolon separated list of roles to assign to the project.*
/account=<account>	*Optional. The user account to assign the roles to. If omitted, the roles are assigned to all accounts.*
/user=<user>	The user account that performs the import
/password=<password>	The password for the user that performs the import
/cfg=<file>	The location of an itprep.ini configuration file that contains server information. For a typical installation, this would be \CCM\Work\5.0\Instance_01\designer\Config\itprep.ini
/clearCmp	*Optional. Upon import, the Create Model Package setting of a project will be cleared.*

Creating an export

To create a project export, it is advised to create a project on the content development environment. Publish all relevant objects in the project and run the repexport.exe tool that can be found in `CCM\Programs\5.0\ITPMDKRepositoryServer`. This tool accepts the following parameters:

/file=<file>	The path to the file to be exported to.
/transfer or /archive	Choose one to designate what kind of export is made. For the purpose described in this chapter, /transfer should be used.
/user=<user>	The user account that performs the export
/password=<password>	The password for the user that performs the export
/cfg=<file>	The location of an itprep.ini configuration file that contains server information. For a typical installation, this would be \CCM\Work\5.0\Instance_01\designer\Config\itprep.ini