Kofax Customer Communications Manager

API Guide

Version: 5.2

Date: 2018-11-19

Table of Contents

Preface

This guide describes the Contract Manager interface, which serves as the main entry point to Kofax Customer Communications Manager. The interface runs on a single port and provides:

- A number of SOAP web services
- Access to CCM Designer through HTTP or HTTPS

Related documentation

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

https://docshield.kofax.com/Portal/Products/CCM/520-nz7r6s9geq/CCM.htm

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Getting help for Kofax products

Kofax regularly updates the Kofax Support site with the latest information about Kofax products.

To access some resources, you must have a valid Support Agreement with an authorized Kofax Reseller/Partner or with Kofax directly.

Use the tools that Kofax provides for researching and identifying issues. For example, use the Kofax Support site to search for answers about messages, keywords, and product issues. To access the Kofax Support page, go to www.kofax.com/support.

The Kofax Support page provides:

- Product information and release news
  Click a product family, select a product, and select a version number.
- Downloadable product documentation
  Click a product family, select a product, and click **Documentation**.
- Access to product knowledge bases

Click **Knowledge Base**.

- Access to the Kofax Customer Portal (for eligible customers)

  Click **Account Management** and log in.

  To optimize your use of the portal, go to the Kofax Customer Portal login page and click the link to open the *Guide to the Kofax Support Portal*. This guide describes how to access the support site, what to do before contacting the support team, how to open a new case or view an open case, and what information to collect before opening a case.
- Access to support tools

  Click **Tools** and select the tool to use.
- Information about the support commitment for Kofax products

  Click **Support Details** and select **Kofax Support Commitment**.

Use these tools to find answers to questions that you have, to learn about new functionality, and to research possible solutions to current issues.

Chapter 1

Getting started

The chapter explains how to perform the primary tasks required to get started working with CCM.

CCM package

This section provides an overview of all components that are available in the CCM package.

By default, the Kofax Customer Communication Manager installation package deploys a single instance of the CCM software consisting of CCM Core, CCM Repository, CCM Designer, CCM ComposerUI for HTML5, CCM ComposerUI for J2EE, and the Contract Manager. The installation creates one instance ready for use, but you can append additional CCM instances if required. The Contract Manager is the interface to Kofax Customer Communications Manager for all communications (run time, design time, and system management). Currently, the Contract Manager has two roles:

- Expose services (SOAP)
- Proxy HTTP(S) traffic

The Contract Manager has its own database in which all the interface definitions are stored. It communicates with CCM Core directly over TCP/IP. Although CCM ComposerUI for J2EE is deployed, CCM ComposerUI for HTML5 is used by default.

Install CCM Designer for Windows

To perform the primary tasks, you need to manually install the CCM Designer for Windows client. The installer ITP MDK Repository Client setup.exe resides in: `<deploy root>\CCM\Work\5.2\Instance_01\designer\Client`.

The installer prompts you to enter the installation name that will determine the directory name where the client will be residing. It also prompts you to enter the information on the CCM Repository server and host that you can retrieve from the readme.txt file located in the same folder as the installer.

By default, the client is deployed in: `C:\Program Files (x86)` for a 64-bit OS.

Configure Tomcat settings

By default, the Tomcat connector used for CCM Designer is set to accept objects of 25 MB at most. You can configure this limit in the server.xml file that resides in: `<tomcat root>\instance-WebDesigner-5.2\conf`. Use the setting MaxPostSize to change the limit.

Create CCM Designer Users

To log on to CCM Designer, you need to create one or more users.

During installation, a default user is created. The default user is "Administrator". The default password is "0fec6d3d54b8df23067a7f7e79db8b0f".

To create CCM Designer users, you can use SOAP calls. For more information, see DesignerAddUserV1, DesignerAddUserV2, and DesignerAddRoleV1.

Also, you can create users with CCM Designer for Windows. Use the following steps to log on as administrator to CCM Designer.

1. Start CCM Designer for Windows.
2. Log in as ITP Admin.
   a. In the **Name** box, type: itpadmin or itp admin.
   b. In the **Password** box, type: www.aia-itp.com
3. The first time you log on, you are prompted to change your password.
   - Also, you can log on using a user account that has the Allow login as Admin right, which allows to create new users. For more information on this right, see the next section. To log on with such a user account, log on to CCM Designer for Windows as this user, click **File** > **Log in as administrator** and enter the name and password for this user again.
4. In the tree view, right-click **Users** and then click **New User**.
   The **New User** window appears.
5. Enter a name, a full name, and a password, and then click **Add User**.
   A password must comply with the password policy requirements. For more information, see Requirements for password.

Allow login as Admin right

1. To allow the user to log in with the same rights as ITP Admin, in the tree view, click **Users**.
2. Right-click a user name in the right pane and then click **Configuration**.
3. Select the **Authorization** tab and select **Allow login as Admin**. When selected, the user can assign roles and create projects.
4. Click **OK**.

Requirements for password

The following password requirements are enforced in a standard configuration:

- Minimum password length is 8 characters
- If length is fewer than 20 characters, password should contain at least three of the following character types: lowercase letters, uppercase letters, digits, and other (symbols, punctuation, and so on)
- For languages without lowercase/uppercase distinction, all three of the following character types must be present: letters, digits, and other (symbols, punctuation, and so on)
- Password cannot be repeated for the same user during a 12-month period
- If applicable, password should not appear on an organization's list of forbidden passwords

For information on how to configure the password policy, see the section "Configure the password policy" in the *Kofax Customer Communications Manager Repository Administrator's Guide*.

Create a new project

1. Start CCM Designer for Windows.
2. To create new projects, you need the corresponding permissions. Contact your Administrator or see Allow login as Admin right.
3. Click **File** > **New** > **New Project**.
   You are prompted to select a document type and a model (**Master Template**) language.
4. Adjust the settings to fit the requirements of your project and click **Create**.
   The New Project - Configuration window appears.
5. On the **General** tab, enter a name for the project.
6. Click **Apply** and then click **OK**.
   The new project appears in CCM Designer.

   Note You may need to wait some time for the new project to appear. If the project does not appear, click **Refresh** to refresh the screen.

Edit Data Backbone

Data Backbone determines what data should be passed in the Data Backbone XML required by some of the SOAP calls described in this guide. If you make no changes to the Data Backbone, the Data Backbone XML should look this way:

```
<?xml version="1.0" encoding="utf-8"?><Test></Test>
```

Retrieve the XSD for the Data Backbone XML with the call DesignerGetDatabackboneDefintion (see DesignerGetDataBackBoneDefinitionV1).

You can only edit Data Backbone using CCM Designer for Windows. For more information, see the section "Work with data" in the *Kofax Customer Communications Manager Repository User's Guide*.

Control calling applications and their access to Contract Manager interfaces

If the ContractManager!UseAuthentication deployment parameter is set to True during installation, the authentication and authorization mechanism is enabled for the Contract Manager. It only grants applications access to authorized SOAP calls after they are properly authenticated.

This mechanism uses HTTP basic authentication. This means that you should only use authentication and authorization in combination with an encrypted connection. CCM expects preemptive authentication.

When a calling application executes SOAP calls on the Contract Manager, it must supply an application name and the key. You can manage calling applications using the ManageCM tool.

For detailed information on how to do so, see the *Kofax Customer Communications Manager Getting Started Guide*.

Chapter 2

CCM API overview

This chapter provides an overview of the functionality that Kofax Customer Communications Manager offers for the available contract types. For a detailed description of the contract types, see CCM API description.

SOAP web services

There are several groups of web services called contract types, some of which have multiple versions.

Currently, Kofax Customer Communications Manager supports the following contract types:

1. CCMInteractive, versions V1, V2, and V3
2. CCMDistribution, versions V1 and V2
3. CCMAdministration, version V1
4. CCMContentApprovalWorkflow, version V1

Also, CCMCapabilities is available as a service.

To learn how to use the available calls and the CCMCapabilities service, see the chapter CCM API description.

The contract types are exposed as SOAP web services. You can obtain their respective WDSLs through the following URLs:

1. CCMInteractive

```
http(s)://<ccm server>:<port>/ccm/wsdl?
contracttypename=CCMInteractive&contracttypeversion=V1
```

```
http(s)://<ccm server>:<port>/ccm/wsdl?
contracttypename=CCMInteractive&contracttypeversion=V2
```

```
http(s)://<ccm server>:<port>/ccm/wsdl?
contracttypename=CCMInteractive&contracttypeversion=V3
```

2. CCMDistribution

```
http(s)://<ccm server>:<port>/ccm/wsdl?
contracttypename=CCMDistribution&contracttypeversion=V1
```

```
http(s)://<ccm server>:<port>/ccm/wsdl?
contracttypename=CCMDistribution&contracttypeversion=V2
```

3. CCMAdministration

```
http(s)://<ccm server>:<port>/ccm/wsdl?
contracttypename=CCMAdministration&contracttypeversion=V1
```

```
http(s)://<ccm server>:<port>/ccm/wsdl?
contracttypename=CCMAdministration&contracttypeversion=V2
```

4. CCMContentApprovalWorkflow

```
http(s):<ccm server>:<port>/ccm/wsdl?
contracttypename=CCMContentApprovalWorkflow&contracttypeversion=V1
```

5. CCMCapabilities

```
http(s)://<ccm server>:<port>/ccm/Capabilities
```

Each contract type version provides a consistent and complete set of calls. The calls within a contract type also have a version associated to them. This version is changed whenever the interface or behavior of a particular call is changed. This version is essentially global for that call and unrelated to the version of the contract types that the call is contained in.

The WSDL contains explicit anchors (^ and $) added to the pattern definitions.

CCMInteractive contract type

The CCMInteractive contract type handles the definition, management, and composition of documents and document packs. It contains calls that 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 through CCM ComposerUI for HTML5. The V1 version creates single documents. The V2 version creates document packs that contain one or more documents.
- DocumentPack: These calls process existing document packs. For example, they give you the ability to review and modify document packs and document format conversions.
- Admin: These calls provide administrative functionality.
- SystemCheck: These calls provide functionality for checking the installation.

There are three versions of this contract type:

1. 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.
2. 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 them.
3. V3: This contract type allows import documents to be passed when composing a document pack, next to the functionality offered by V2.

For the list of available calls, see the next sections. For detailed information on these calls, see Available calls.

CCMInteractive V1

The CCMInteractive V1 contract type contains the following calls:

- AdminGetLogsV1
- ComposeDocxInteractiveStartV1

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

CCMInteractive V2

The CCMInteractive V2 contract type contains the following calls:

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

- SystemCheckInteractiveGetV2
- SystemCheckInteractiveStartV2

CCMInteractive V3

The CCMInteractive V3 contract type contains the following calls:

- AdminGetLogsV2
- ComposeDocumentPackGetV1
- ComposeDocumentPackInteractiveStartV2
- ComposeDocumentPackV2
- ComposeInteractiveGetSuspendedSessionV1
- ComposeInteractiveResumeSuspendedSessionV1
- DesignerAddFieldsV1
- DesignerGetDataBackBoneDefinitionV1
- DesignerGetDocumentPackTemplateV1
- DesignerGetDocumentTemplateV1
- DesignerGetLetterbookV2
- DesignerListLetterbooksV2
- DesignerListProjectsV2
- DesignerListTemplatesV1
- DesignerStartSessionV2
- DocumentPackConvertV1
- DocumentPackInteractiveModifyFromReviewV1
- DocumentPackInteractiveModifyV1
- DocumentPackInteractiveReviewV1
- FinishSessionV1
- SystemCheckInteractiveGetV2
- SystemCheckInteractiveStartV2
- SystemCheckV1

CCMDistribution contract type

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

- DistributeDocumentPack: This call hands off a document pack to the CCM Core DistributeDocumentPack exit point. For more information, see DistributeDocumentPack.dss.
- DocumentPackSign: This call hands off a document hands to Kofax SignDoc.
- DocumentPackDistributeWorkPlace: This call is no longer supported.
- DocumentPackDistributeOutputManagement: This call hands off a document pack to CCM Batch & Output Management for further distribution.

For the list of available calls, see the following section. For detailed information on these calls, see Available calls.

CCMDistribution V1

The CCMDistribution V1 contract type contains the following calls:

- DocumentPackDistributeV1
- DocumentPackDistributeWorkplaceV1 (no longer supported)
- DocumentPackSignV1

CCMDistribution V2

The CCMDistribution V2 contract type contains the following calls:

- DocumentPackDistributeV1
- DocumentPackDistributeOutputManagementV1
- DocumentPackSignV1

CCMAdministration contract type

The CCMAdministration contract type deals with common administrative tasks, such as adding a new user, assigning a role to the user for a particular project, and starting a CCM Designer session.

For the list of available calls, see the next sections. For detailed information on these calls, see Available calls.

CCMAdministration V1

The CCMAdministration V1 contract type contains the following calls:

- DesignerAddUserV2
- DesignerAddRoleV1
- DesignerStartSessionV3

CCMContentApprovalWorkflow contract type

CCM differentiates between two Changeset types:

1. Changesets created and managed internally by CCM Designer (for more information, see *Help for CCM Designer*).
2. Changesets created and controlled by an external workflow (see CCMContentApprovalWorkflow V1).

CCMContentApprovalWorkflow handles the second type of Changeset. With this contract type, you can assign a new state to a Changeset, retrieve information on this Changeset, and get a list of all Changesets in a project.

The following rules apply to the Changesets:

- Changesets created in CCM Designer are always and only controlled by CCM Designer.

- Changesets created with the DesignerCreateWorkV1call are always controlled externally by CCMContentApprovalWorkflow.

CCMContentApprovalWorkflow V1

The CCMContentApprovalWorkflow V1 contract type contains the following calls:

- DesignerCreateWorkV1
- DesignerRejectWorkV1
- DesignerApproveWorkV1
- DesignerPublishWorkV1
- DesignerGetWorkStateV1
- DesignerGetWorkV1
- DesignerListAllWorkV1

In the preceding call names, the term "work" is used to denote a Changeset.

CCMCapabilities service

The CCMCapabilities service provides information about the contract type capabilities of a contract, which is defined by a partner and a customer. It is exposed as a SOAP web service and can be used by integrating applications to guarantee upward compatibility.

The CCMCapabilities service contains the following call:

- CCMCapabilitiesVerifyContractTypeAvailableV1

For detailed information on this call, see Available calls.

Designer requests

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

GUI requests

DesignerStartSession call creates a session context in which one end user can access the CCM Designer environment. It returns a URL that the end user can use to edit templates, Text Blocks, and so on. See DesignerStartSessionV1, DesignerStartSessionV2, and DesignerStartSessionV3.

Explore requests

The explore requests allow the business application to obtain information about objects stored in CCM Repository so that the business application can start an appropriate composition request.

- DesignerGetDataBackBoneDefinition: Returns XML data that contains the definition of a Data Backbone. See DesignerGetDataBackBoneDefinitionV1.

- DesignerGetDocumentPackTemplate: Returns XML data that contains the definition of a Document Pack Template. See DesignerGetDocumentPackTemplateV1.
- DesignerGetDocumentTemplate: Returns XML data that contains the definition of a Document Template. See DesignerGetDocumentTemplateV1.
- DesignerGetLetterbook: Returns XML data that contains the definition of a Letter Book. See DesignerGetLetterbookV1 and DesignerGetLetterbookV2.
- DesignerListLetterbooks: Returns XML data that contains a list of all Letter Books in a project. See DesignerListLetterbooksV1 and DesignerListLetterbooksV2.
- DesignerListDocumentTemplates: Returns XML data that contains a list of all Document Templates in a project. See DesignerListDocumentTemplatesV1.
- DesignerListProjects: Returns XML data that contains a list of all available projects. See DesignerListProjectsV1 and DesignerListProjectsV2.
- DesignerListTemplates: Returns XML data that contains a list of all available templates in a project (both Document Templates and Document Pack Templates). See DesignerListTemplatesV1.

Designer administration requests

To create new CCM Designer users and assign roles to the user, use the DesignerAddUserV2 and DesignerAddRoleV1 calls. See DesignerAddUserV2 and DesignerAddRoleV1, respectively.

The roles that are allowed to be assigned through the request are explicitly whitelisted on the server. The whitelist is defined by the CCM Core Designer.AllowedUserRoles setting, which can be specified on the General tab of the Environments node in CCM Core Administrator. For more information about CCM Core settings, see the *Kofax Customer Communications Manager Core Developer's Guide*.

These roles are allowed:

- Author
- Project Administrator
- Content Viewer
- Publisher
- Publishing Author
- Publishing Reviewer
- Viewer

For a description of the roles, see the section "Roles and permissions" in the Kofax CCM Designer online Help.

Composer requests

The Composer requests give you the ability to run templates and create documents and/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. There are separate calls for running interactive templates and non-interactive (on-demand) templates.
- V3: The compose calls import documents to be passed when composing a document pack, next to the functionality offered by V2.

Document pack composition

On-demand compose requests

The on-demand Compose calls give you the ability to run non-interactive templates.

To compose a document pack with one or more documents, use the non-interactive ComposeDocumentPack calls. The produced document packs contain a document for each Document Template in the pack. See ComposeDocumentPackV1 and ComposeDocumentPackV2.

Compose interactive requests

The interactive web service calls give you the ability to start CCM ComposerUI for HTML5 composition run and to obtain the result that the run produces.

Interactive requests require the implementation of a CCM ComposerUI for HTML5 web application contained in a default CCM installation. For details, see the *Kofax Customer Communications Manager ComposerUI for HTML5 JavaScript API Web Developer's Guide*.

If you installed CCM with the ExampleWebApp!Install deployment parameter, a default web application is available. You can reach this application through the following URL, where <ccm server> and <port> are the server and port where CCM Interactive is deployed.

```
http(s)://<ccm server>:<port>/start/home.html
```

This example link provides access to the login page of CCM Designer and also gives you the ability to run an example interactive template.

Start calls

You can start a new interactive run that produces a document pack with the ComposeDocumentPackInteractiveStart calls. See ComposeDocumentPackInteractiveStartV1 and ComposeDocumentPackInteractiveStartV2.

Suspend and resume calls

If the allowsuspend flag is passed on the ComposeDocumentPackInteractiveStart call, the user can suspend the session and resume it later. The CCMInteractive V2 contract type contains two calls that support this ability:

1. ComposeInteractiveGetSuspendedSession. See ComposeInteractiveGetSuspendedSessionV1.
2. ComposeInteractiveResumeSuspendedSession See ComposeInteractiveResumeSuspendedSessionV1.

Result

To obtain the result of an interactive run, use the ComposeDocumentPackGet call. See ComposeDocumentPackGetV1.

Document composition

On-demand compose request

The on-demand calls give you the ability to run non-interactive templates. The CCMInteractive V1 contract type has two non-interactive compose calls: one for generating a DOCX document and one for generating a PDF document.

See ComposeDocxV1 and ComposePdfV1, respectively.

Compose interactive requests

The interactive web service calls give you the ability to start a CCM ComposerUI for HTML5 composition run and obtain the result that the run produces.

Interactive requests require the implementation of a CCM ComposerUI for HTML5 web application contained in a default CCM installation. For details, see the *Kofax Customer Communications Manager ComposerUI for HTML5 JavaScript API Web Developer's Guide*.

If you installed CCM with the ExampleWebApp!Install deployment parameter, a web application is available. You can reach this application through the following URL, where <ccm server> and <port> are the server and port where CCM Interactive is deployed.

```
http(s)://<ccm server>:<port>/start/home.html
```

This example link provides access to the login page of CCM Designer and also gives you the ability to run an example interactive template.

InteractiveStart calls

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

See ComposeDocxInteractiveStartV1 and ComposePdfInteractiveStartV1, respectively.

InteractiveGet calls

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

See ComposeDocxInteractiveGetV1 and ComposePdfInteractiveGetV1, respectively.

InteractiveFinish calls

To remove any remaining server side data related to the composition run, use the ComposeInteractiveFinish call. The call also signals that interactive composition run is ended. See ComposeInteractiveFinishV1.

Document Pack processing requests

Both the CCMInteractive and CCMDistribution contract types offer functionality for processing Document Packs that have been generated earlier.

Review calls

The DocumentPackInteractiveReview call in the CCMInteractive V2 contract type gives you the ability to take an existing document pack and have the user review it by means of a reviewing tool. See DocumentPackInteractiveReviewV1.

Modification calls

If the user finds any errors during the review, use the DocumentPackInteractiveModify and DocumentPackInteractiveModifyFromReview calls to enable the 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, but with all answers of the initial composition already filled out in the interactive forms.

For the system to be able to retrieve the previously given answers from Forms and Content Wizards, you need to adjust the following settings:

- For Forms, select the "Default answer" option and select the Field that contains the desired answer. For more information, see the section "Create a new Form" in the Kofax CCM Designer online Help.
- For Content Wizards, store the answers in Status Field Sets. For more information, see the section "Manage Field Sets" in the Kofax CCM Designer online Help.

See the following sections about the calls to use: DocumentPackInteractiveModifyV1 and DocumentPackInteractiveModifyFromReviewV1.

Note You cannot modify a document pack created from a Document Pack Template that contains import documents in optional slots **if** at least one of these optional import documents is not part of the document pack.

Conversion calls

The CCMInteractive contract type contains a DocumentPackConvert call that gives you the ability to convert documents in the pack. The call converts all DOCX or DOC documents that are present in the document pack to the specified output format. The resulting document is placed next to the existing document in the document pack, so the document pack is augmented with the files in the requested format. See DocumentPackConvertV1.

Distribution calls

To distribute a previously created document pack, use the DocumentPackDistribute call. See DocumentPackDistributeV1.

The actual Distribution functionality is defined in the CCM Core DistributeDocumentPack exit point.

Distributing to CCM Batch & Output Management

To import composed document packs into CCM Batch & Output Management, use the DocumentPackDistributeOutputManagementV1 call of the CCMDistribution V2 contract type (see DocumentPackDistributeOutputManagementV1).

This call distributes a composed document pack along with an Import Request XML to the CCM Batch & Output Management Input Request folder. The Import Request XML contains an import request that conforms to CcmRequest.xsd. The import request is picked up by CCM Batch & Output Management for further distribution.

The call returns the name under which the document pack and import request have been imported into CCM Batch & Output Management, which means that the distribution to the Hotfolder is successful.

OutputManagementHotfolder

To use the DocumentPackDistributeOutputManagementV1 call, you need to configure the `/OutputManagementHotfolder=` parameter for a contract using the ManageCM tool. For more information, see the sections "Create a contract" and "Change properties of a contract" in the *Kofax Customer Communication Manager Getting Started Guide*.

Signing calls

To have the ability to hand off composed document packs to Kofax SignDoc, use the DocumentPackSign call, which resides in the CCMDistribution contract type. See DocumentPackSignV1.

This call creates a Kofax SignDoc signing package from the Microsoft Word DOCX files and the PDF documents in the document pack. Kofax SignDoc subsequently ensures that the documents in this signing package get signed.

Document Pack signing

Currently, Kofax Customer Communications Manager only supports signing Microsoft Word DOCX files and PDF files. The document pack must contain at least one DOCX file that contains a signature line. Kofax SignDoc uses these lines to determine the signer of this particular document.

Signing proceeds for all DOCX or DOC files and all PDF files in the document pack. If a document has both a DOCX or DOC version and a PDF version in the same document pack slot, only the DOCX version ends 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 verifies that:

- DOCX or DOC files that have a signature line for that signer get signed
- DOCX or DOC files that have no signature line for that signer get marked as viewed
- PDF files that have no DOCX or DOC equivalent get marked as viewed

See the SignDoc documentation for information on this signing process and how signature lines are related to signers.

DocumentPackSign reports an error if a document pack contains static or other documents that are not in DOCX, DOC, or PDF format. This may be the case for static documents. If that is the case, DocumentPackSign reports an error.

Authentication

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

SignDoc uses API keys to authenticate callers. This key is generated on 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 the user's individual account. There the user can find a separate section for API keys. See the Kofax SignDoc documentation for more information.

Signing package identification

Kofax SignDoc identifies signing packs through an ID. This ID is currently mandatory. If a caller does not specify an ID, an error is reported. The ID can contain alphanumerical characters, minus signs, and underscores. If the key already exists, the DocumentPackSign call reports an error. The DocumentPackSign call does not overwrite existing signing packages.

The passed ID is returned by the DocumentPackSing call. Although the returned ID is currently the same as the one passed as input, we advise the returned ID for use in your application for future reference to this document pack.

Resource handling requests

Various calls involve the server-side session state in CCM. The resources associated with this process are reclaimed automatically, which may take some time.

The FinishSession call signals that any server side resources associated with a particular session ID can be reclaimed. See FinishSessionV1.

Administration requests

The administration requests are placed in the CCMInteractive contract type.

The AdminGetLogs call provides access to a set of logs produced between a certain start date and end date. See AdminGetLogsV1 and AdminGetLogsV2.

System Check requests

The System Check functionality gives you the ability to verify that a deployed installation is up and running. These calls are placed in the CCMInteractive contract type:

- To check the non-interactive document composition functionality, use the SystemCheck call. It produces a test PDF document. See SystemCheckV1.
- To check the interactive composition functionality, use the SystemCheckInteractiveStart and SystemCheckInteractiveGet calls.

Note You need to set up a CCM ComposerUI for HTML5 web application prior to using the calls.

The SystemCheckInteractiveGet call initiates an interactive run that produces a PDF test document.

After the interactive run has completed, you can use the SystemCheckInteractiveGet call to retrieve the produced PDF document that was initiated earlier through SystemCheckInteractiveStartV1.

See SystemCheckInteractiveStartV1, SystemCheckInteractiveStartV2, SystemCheckInteractiveGetV1, and SystemCheckInteractiveGetV2.

Example requests and responses

DesignerGetWorkV1

Example request

This is an example of a DesignerGetWorkV1 request.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:v1="http://www.aiasoftware.com/cloud/v1">
   <soapenv:Header/>
   <soapenv:Body>
      <v1:DesignerGetWorkV1Request>
         <v1:partner>CCM</v1:partner>
         <v1:customer>Local</v1:customer>
         <v1:contracttypename>CCMContentApprovalWorkflow</v1:contracttypename>
         <v1:contracttypeversion>V1</v1:contracttypeversion>
         <v1:jobid>GetWork</v1:jobid>
         <v1:project>InstallationTest</v1:project>
         <v1:name>MyChangeset</v1:name>
      </v1:DesignerGetWorkV1Request>
   </soapenv:Body>
</soapenv:Envelope>
```

Example response

This is an example of a DesignerGetWorkV1 response.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <tns:DesignerGetWorkV1Response xmlns:tns="http://www.aiasoftware.com/cloud/v1">
         <tns:requestinfo>
            <tns:partner>CCM</tns:partner>
            <tns:customer>Local</tns:customer>
            <tns:contracttypename>CCMContentApprovalWorkflow</tns:contracttypename>
            <tns:contracttypeversion>V1</tns:contracttypeversion>
            <tns:jobid>GetWork</tns:jobid>
         </tns:requestinfo>
         <tns:work>PD94bWwgdmVyc2lvbj0iMS4wIiBlbm...NvZGluZz0iVVRGLTgiID8+==</tns:work>
      </tns:DesignerGetWorkV1Response>
   </soap:Body>
</soap:Envelope>
```

SystemCheckV1

Example request

This is an example of a SystemCheckV1 request.

```
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"
Authorization: Basic C29hcHV…..UFhEN0Y=
<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

This is an example of a SystemCheckV1 response.

```
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>
```

CCMCapabilitiesVerifyContractTypeAvailableV1

Example request

This is an example of a CCMCapabilitiesVerifyContractTypeAvailableV1 request. This request queries whether the contract type CCMInteractive version 5 is available for the contract CCM Local.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:v1="http://www.aiasoftware.com/cloud/v1">
   <soapenv:Header/>
   <soapenv:Body>
      <v1:CCMCapabilitiesVerifyContractTypeAvailableV1Request>
         <v1:partner>CCM</v1:partner>
         <v1:customer>Local</v1:customer>
         <v1:contracttypename>CCMInteractive</v1:contracttypename>
         <v1:contracttypeversion>V5</v1:contracttypeversion>
         <v1:jobid>cap</v1:jobid>
      </v1:CCMCapabilitiesVerifyContractTypeAvailableV1Request>
   </soapenv:Body>
</soapenv:Envelope
```

Example response

This is an example of a CCMCapabilitiesVerifyContractTypeAvailableV1 response. This response indicates that the contract type CCMInteractive is available, but the requested version 5 is unavailable. The maximum version that is available is version 3.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <tns:CCMCapabilitiesVerifyContractTypeAvailableV1Response 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>V5</tns:contracttypeversion>
            <tns:jobid>cap</tns:jobid>
         </tns:requestinfo>
         <tns:contracttypeisavailable>true</tns:contracttypeisavailable>
         <tns:requestedversionisavailable>false</tns:requestedversionisavailable>
         <tns:highestversionavailable>V3</tns:highestversionavailable>
      </tns:CCMCapabilitiesVerifyContractTypeAvailableV1Response>
   </soap:Body>
</soap:Envelope>
```

Chapter 3

CCM API description

Standard fields for web services requests

All web service requests contain at least the following fields. The partner, customer, contracttype, and contracttypeversion fields identify the contract type that you are using and are the same for all calls within the contract type. The jobid field is an ID that you provide to relate the execution of CCM jobs to the execution of your business application.

Field	Type	Description
partner	name	CCM
customer	name	Local
contracttypename	name	CCMInteractive, CCMDistribution, or CCMAdministration. This identifies the contract type.
contracttypeversion	version	V1, V2, or V3. This identifies the version of the contract type.
jobid	jobid	Custom identifier for diagnostic purposes. This value consists of ASCII alphanumeric characters and must not be empty. It can be any meaningful value of your choice.

For logging and traceability, the responses to the web services requests are nested in a requestinfo node.

For more information on the types, see the next section.

Field types for web services requests and responses

The following applies to all contract types.

Type	Format Description	Example
base64Binary	Standard XSD type base64Binary. A file encoded as an ASCII string according to the base-64 encoding specification	QSBCYXNlNjQgZW5jb2RlZ="

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 CCM Repository, consisting of text matching the regular expression: [^:?/<>*"\\()]+	My Document Template
sessionid	Text matching the regular expression: [A-Za-z0-9\-]+	3ff62c53-a2c6-4417-b1c5-379a0a8db0bf
url	Standard XSD type "anyURI". A relative or absolute URL according to standard URL specifications.	/ccm/login.jsf

Web services error handling

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

Field	Type	Description
errorcode	errorcode	A return code indicating the type of error
message	message	An error message. In some cases, the Contract Manager only returns an informational message referring to an error log. The log resides in: `CCM\Work\<version>\ContractManager\Logs` For more information, see the sections "ThrowError" and "[@*USER] prefix" in the Kofax Customer Communications Manager Core Scripting Language Developer's Guide.

Available calls

AdminGetLogsV1

The AdminGetLogsV1 call retrieves a .zip file with the CCM logs produced between a certain start date and end date. If no logs are available, nothing is returned.

The request requires the following fields next to the standard fields.

Field	Type	Description
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.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
url	message	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URLs that are returned are relative to this base URL. This URL gives access to a .zip file containing logs that are organized in a folder that reflects the internal components of CCM.

In case of an error, the response contains a SOAP Fault structure.

AdminGetLogsV2

The AdminGetLogsV2 call retrieves a .zip file with the CCM logs produced between a certain start date and end date.

The request requires the following fields next to the standard fields.

Field	Type	Description
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.

The response contains the following fields in addition to the standard fields.

Field	Type	Description

url	message	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URLs that are returned are relative to this base URL. This URL gives access to a .zip file containing logs that are organized in a folder that reflects the internal components of CCM.

DesignerStartSessionV1

The DesignerStartSessionV1 call starts a CCM Designer session. It returns a relative URL that provides access to CCM Designer.

The request requires the following fields next to the standard fields.

Field	Type	Description
user	message	Reserved. This parameter is currently being ignored.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
itpcloudid	sessionId	Legacy CCM Contract Manager session identifier.
url	url	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URLs that are returned are relative to this base URL.

In case of an error, the response contains a SOAP Fault structure.

DesignerStartSessionV2

The DesignerStartSessionV2 call starts a CCM Designer session. It returns a relative URL to the login page of CCM Designer.

This request only requires the standard fields.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier.

url	url	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URLs that are returned are relative to this base URL.

DesignerStartSessionV3

The DesignerStartSessionV3 call starts a CCM Designer session. It returns a relative URL that provides access to CCM Designer.

The request requires the following fields next to the standard fields.

Field	Type	Description
user	message	Optional. If a user name is passed, the user is logged in automatically, and the returned URL leads to the Dashboard of CCM Designer. **Note** When the LDAP mode is enabled in CCM Designer, you cannot use this field. For more information on LDAP, see the *Kofax Customer Communications Manager Getting Started Guide*.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
ccmsessionid	sessionId	CCM Contract Manager session identifier.
url	url	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URLs that are returned are relative to this base URL.

In case of an error, the response contains a SOAP Fault structure.

DesignerAddFieldsV1

The DesignerAddFielsV1 call gives you the ability to add Fields to an existing Field Set. This works for Field Sets in the root of the Field Sets folder.

The request requires the following fields next to the standard fields.

Field	Type	Description

project	repositoryobjectname	The name of the project to which the Fields should be added.
fieldset	repositoryobjectname	The name of the Field Set to which the Fields should be added. This Field Set has to be in the root of the Field Sets folder. Once the Fields are added to the Fields Set, it gets the status Published.
fieldlist	message	A comma-separated list of Fields that have to be added to the specified Field Set.

The response only contains the standard fields.

In case of an error, the response contains a SOAP Fault structure.

DesignerAddUserV1

The DesignerAddUserV1 call is deprecated. We recommend that you use DesignerAddUserV2 instead (see DesignerAddUserV2).

To keep using the call, set the minimum acceptable password length and the minimum safe password length for CCM Repository to 6. For more information, see "Configure the password policy" in the *Kofax Customer Communications Manager Repository Administrator's Guide*.

The DesignerAddUserV1 call gives you the ability to create new CCM Designer users.

Note When the LDAP mode is enabled in CCM Designer, you cannot use this call. For more information on LDAP, see the *Kofax Customer Communications Manager Getting Started Guide*.

The request requires the following fields next to the standard fields.

Field	Type	Description
user	message	The name of the user that is added. This name may contain spaces.
role	message	The role of the user. Only roles that are whitelisted on the CCM server are accepted. For more information on whitelisting, see Designer administration requests.

The response only contains the standard fields.

In case of an error, the response contains a SOAP Fault structure.

If the CCM Designer user with the given name already exists, this call has no effect. If the user exists but is marked for deletion, a new user is created.

DesignerAddUserV2

The DesignerAddUserV2 call gives you the ability to create new CCM Designer users. A business application can provide such a new user access to the CCM Designer with the DesignerStartSessionV3 call. The new user can create a password from this CCM Designer session. Once a password is created, the user can also log in through other mechanisms, such as CCM Designer for Windows.

If the CCM Designer user with the given name already exists, this call has no effect. If the user exists but is marked for deletion, a new user is created.

Note When the LDAP mode is enabled in CCM Designer, you cannot use this call. For more information on LDAP, see the *Kofax Customer Communications Manager Getting Started Guide*.

The request requires the following fields next to the standard fields.

Field	Type	Description
user	message	The name of the user that is added. This name may contain spaces.
role	message	Optional. The role of the user. If passed, it assigns this role to all existing projects. If the user with the given name already exists, the call has no effect, and the role is not assigned to the user. Use DesignerAddRoleV1 to ensure that the user account gets the specified role. See DesignerAddRoleV1. Only roles that are whitelisted on the CCM server are accepted. For more information on whitelisting, see Designer administration requests

The response only contains the standard fields.

In case of an error, the response contains a SOAP Fault structure.

Note If the user account already exists, the API call has no effect, but no error is generated. Also, the role specified in the call is not assigned to the user. To verify that the user account gets the role, use the DesignerAddRoleV1 call.

DesignerAddRoleV1

The DesignerAddRoleV1 call gives you the ability to assign a particular role to a user for a particular project. This must be an existing user. A business application can provide this user access to CCM Designer according to a particular role for a particular project.

Except for the Viewer role, the roles that are allowed to be assigned through the DesignerAddRole request must be explicitly whitelisted on the server. By default, only the Viewer role is allowed to be assigned. The

whitelist is defined by the CCM Core setting Designer.AllowedUserRoles, which can be specified on the General tab of the Environments node in CCM Core Administrator.

Multiple calls for the same user and the same project are cumulative, so adding a role for a particular user and project does not modify existing roles for that user.

Note When the LDAP mode is enabled in CCM Designer, you cannot use this call. For more information on LDAP, see the *Kofax Customer Communications Manager Getting Started Guide*.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	message	The name of the project for the given role is assigned to the given user.
user	message	The name of the user assigned the given role within the given project. This name may contain spaces.
role	message	The role of the user. Only roles that are whitelisted on the CCM server are accepted.

The response only contains the standard fields.

In case of an error, the response contains a SOAP Fault structure.

DesignerApproveWorkV1

Use the DesignerApproveWorkV1 call to move a Changeset from the *Under review* status to the *Accepted* status. Objects in the Changeset that have the *Draft* status receive the *Approved* status.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project with the Changeset to be approved.
name	repositoryobjectname	The name of the Changeset.
comment	message	The reason to approve the changes.

The response only contains the standard fields.

In case of an error, the response contains a SOAP Fault structure.

DesignerCreateWorkV1

The DesignerCreateWorkV1 call creates an empty Changeset that will be under the control of an external workflow. The Changeset receives the *In development* status. To add objects to it, use CCM Designer.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project with the Changeset to be created.
name	repositoryobjectname	The name of the new Changeset. Cannot contain the following characters: ?, /, \, <, >, *, ", and |.
description	message	Optional. The description of the new Changeset.

The response only contains the standard fields.

In case of an error, the response contains a SOAP Fault structure.

DesignerGetDataBackBoneDefinitionV1

DesignerGetDataBackBoneDefinitionV1 retrieves the Data Backbone definition. The result contains a base-64 encoded .zip file containing three .xsd files describing the Data Backbone. The Data Backbone definition is useful when building and filling a testdata.xml file to use in the composition process.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project from with the Data Backbone to retrieve.
status	name	Optional. The status of the Document Template that you want to use. The available statuses are Current, Accepted, or Published. If omitted, Published is used.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
databackbonedefinition	base64Binary	A base-64 encoded .zip file containing three .xsd files describing the Data Backbone.

DesignerGetDocumentPackTemplateV1

DesignerGetDocumentPackTemplateV1 returns a Document Pack Template as defined in CCM Designer as base-64 encoded XML. For example, you can use this XML to check which documents are produced for this document pack and which documents are optional.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project with the Document Pack Template to retrieve.

documentpacktemplate	repositoryobjectname	The name of the requested Document Pack Template.
status	name	Optional. The status of the Document Pack Template that you want to use. The available statuses are Current, Accepted, or Published. If omitted, Published is used.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
documentpacktemplate	base64Binary	A base-64 encoded XML describing the requested Document Pack Template. For more information on the format for this XML, see Format for the definitions.

DesignerGetDocumentTemplateV1

DesignerGetDocumentTemplateV1 returns a Document Template as defined in CCM Designer as base-64 encoded XML. You can use this XML to check which parameters this Document Template uses or on which CCM Repository object it is based.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project with the Document Template to retrieve.
documenttemplate	repositoryobjectname	The name of the requested Document Template.
status	name	Optional. The status of the Document Template that you want to use. The available statuses are Current, Accepted, or Published. If omitted, Published is used.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
documenttemplate	base64Binary	A base-64 encoded XML describing the requested Document Template. For more information on the format for this XML, see Format for the definitions.

DesignerGetLetterbookV1

The DesignerGetLetterbookV1 call is deprecated. We recommend that you use DesignerGetLetterbookV2 instead (see DesignerGetLetterbookV2).

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project with the Letter Book to be retrieved.
letterbook	repositoryobjectname	The name of a Letter Book.
status	message	Optional. The status of the listed Letter Book that is requested. The available statuses are Current, Accepted, or Published. If omitted, Published is used.

The response contains the following fields in addition to the standard 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 contains a SOAP Fault structure.

DesignerGetLetterbookV2

The DesignerGetLetterbookV2 call returns an XML structure that describes a particular Letter Book. A Letter Book is a collection of Document Templates. Letter Books can be used by business applications to provide users with a list of Document Templates to choose from.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project with the Letter Book to retrieve.
letterbook	repositoryobjectname	The name of the Letter Book to be retrieved.
status	name	Optional. The status of the Letter Book to retrieve. The available statuses are Current, Accepted, or Published. If omitted, Published is used.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
letterbook	base64Binary	A base-64 encoded XML structure that describes the specified Letter Book. For more information on the format for this XML, see Format for the definitions.

DesignerGetWorkV1

The DesignerGetWorkV1 call returns a detailed information on a Changeset as defined in CCM Designer as base-64 encoded XML.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project with the Changeset.
name	repositoryobjectname	The name of the Changeset.

The response contains the following field in addition to the standard fields.

Field	Type	Description
work	base64Binary	A base-64 encoded XML describing the requested Changeset. For more information on the format for this XML, see Format for the definitions.

In case of an error, the response contains a SOAP Fault structure.

DesignerGetWorkStateV1

With the DesignerGetWorkStateV1 call, you can retrieve the current status of a Changeset.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project with the Changeset.
name	repositoryobjectname	The name of the Changeset.

The response contains the following field in addition to the standard fields.

Field	Type	Description
state	workstate	The status of the Changeset. Can be one of the following statuses: *In development*, *Under review*, *Approved*, *Published*, or *Deactivated*.

In case of an error, the response contains a SOAP Fault structure.

DesignerListAllWorkV1

The DesignerListAllWorkV1 call returns a list of all Changesets in a project that are under the control of an external workflow as base-64 encoded XML, optionally showing inactive Changesets.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project.
listinactive	boolean	Optional. When set to true, the response also contains inactive (deactivated or published) Changesets. Defaults to false.

The response contains the following field in addition to the standard fields.

Field	Type	Description
allwork	base64Binary	A base-64 encoded XML listing all Changesets in the project, optionally showing inactive ones.

In case of an error, the response contains a SOAP Fault structure.

DesignerListLetterbooksV1

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

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project of which the Letter Books must be listed.
status	message	Reserved. This parameter is currently being ignored.

The response contains the following fields in addition to the standard 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 contains a SOAP Fault structure.

DesignerListLetterbooksV2

The DesignerListLetterbooksV2 call returns an XML structure that describes all available Letter Books in a particular project. Letter Books usually represent a category or collection of document templates.

For example, the list of Letter Books can be used by the integrating application to help in selecting a document template.

The request requires the following fields next to the standard fields.

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 are listed. The available statuses are Current, Accepted, or Published. If omitted, Published is used.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
letterbooks	base64Binary	A base-64 encoded XML structure that lists the available Letter Books in the project. For more information on the format for this XML, see Format for the definitions.

DesignerListDocumentTemplatesV1

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

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project with the Document Templates to 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 is used.

The response contains the following fields in addition to the standard 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 contains a SOAP Fault structure.

DesignerListProjectsV1

The DesignerListProjectsV1 call returns a base-64 encoded XML structure that describes all available projects.

The request requires the standard fields.

The response contains the following fields in addition to the standard fields.

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

In case of an error, the response contains a SOAP Fault structure.

DesignerListProjectsV2

DesignerListProjectsV2 returns a base-64 encoded XML structure that describes all available projects. The return XML structure differs from the structure returned for DesignerListProjectsV1.

This request only requires the standard parameters.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
projects	base64Binary	A base-64 encoded XML structure that lists the available projects. For more information on the format for this XML, see Format for the definitions.

DesignerListTemplatesV1

The DesignerListTemplatesV1 call retrieves a base-64 encoded XML representing a list of templates from CCM Designer. This list includes both Document Templates and Document Pack Templates which are distinguished by the type attribute. For example, you can use these templates in other calls, such as in ComposeDocumentPackV1 to create a document pack based on the template, or in GetDocumentTemplate to get more information about a certain template.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project with the templates to be listed.
status	name	Optional. Only templates that have the requested status are listed. The available statuses are Current, Accepted, or Published. If omitted, published is used.
ondemandonly	boolean	Optional. Filter applied to template list. When set to true, as of CCM version 5.0, the list contains only non-interactive templates.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
templatelist	base64Binary	A base-64 encoded XML structure that lists the available templates in the project. For more information on the format for this XML, see Format for the definitions.

DesignerPublishWorkV1

Use the DesignerPublishWorkV1 call to move a Changeset from the *Approved* status to the *Published* status. All *Accepted* objects in the Changeset become *Published*. The Changeset automatically becomes deactivated, and objects that belong to it can be used again outside of this Changeset.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project with the Changeset to be published.
name	repositoryobjectname	The name of the Changeset.
comment	message	The reason to publish the changes.

The response only contains the standard fields.

In case of an error, the response contains a SOAP Fault structure.

DesignerRejectWorkV1

Use the DesignerRejectWorkV1 call to reject changes made to a Changeset. This call moves the Changeset back to the *In development* status. The objects that belong to this Changeset keep their status.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The name of the project with the Changeset to be rejected.
name	repositoryobjectname	The name of the Changeset.
comment	message	The reason to reject the changes.

The response only contains the standard fields.

In case of an error, the response contains a SOAP Fault structure.

DocumentPackInteractiveReviewV1

Use the DocumentPackInteractiveReviewV1 call to review a previously 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, the user can call DocumentPackInteractiveModifyFromReviewV1 to make changes (see DocumentPackInteractiveModifyFromReviewV1).

The request requires the following fields next to the standard fields.

Field	Type	Description
documentpack	base64Binary	A base-64 encoded binary representation of a document pack data structure. This document pack contains the documents that can be reviewed.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier.
url	url	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URLs that are returned are relative to this base URL. With this URL, the interactive session can be started.

DocumentPackInteractiveModifyV1

You can use the DocumentPackInteractiveModifyV1 call when you need to make changes to the data entered interactively after its original composition has been finished.

The request requires the following fields next to the standard fields.

Field	Type	Description
documentpack	base64Binary	A base-64 encoded binary representation of a document pack data structure. This document pack contains the documents that can be modified.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier.

url	url	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URLs that are returned are relative to this base URL. With this URL, the interactive session can be started.

DocumentPackInteractiveModifyFromReviewV1

Use the DocumentPackInteractiveModifyFromReviewV1 call after a document pack review has been started, typically when a user finds 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.

The request requires the following fields next to the standard fields.

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 contains the following fields in addition to the standard fields.

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier.
url	url	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URLs that are returned are relative to this base URL. From this URL, the interactive session can be started.

DocumentPackConvertV1

DocumentPackConvertV1 converts the documents in a document pack to another file format, such as PDF. The new file format is added to the document pack. Existing documents are not removed.

The request requires the following fields next to the standard fields.

Field	Type	Description

documentpack	base64Binary	A base-64 encoded binary representation of a document pack data structure. This document pack contains the documents to be converted.
outputformat	message	The required output format. Currently, only PDF and PDF/A-1b are supported.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier. The identifier of the session that was used for the conversion.
documentpack	base64Binary	A base-64 encoded binary representation of a document pack data structure. This document pack contains the converted documents and updated references to them; otherwise, it is identical to the pack that was originally sent on the request. The conversion only converts the DOCX and DOC documents in the document pack.

DocumentPackDistributeV1

DocumentPackDistributeV1 distributes a previously created document pack. The actual Distribution functionality is defined in the CCM Core DistributeDocumentPack exit point.

Note There is no correlation between the Distribution channel used for DocumentPackDistributeV1 and the channels that can be configured as part of the Batch & Output Management to produce documents with different formats and purposes.

The request requires the following fields next to the standard fields.

Field	Type	Description
documentpack	base64Binary	A base-64 encoded binary representation of a Document Pack data structure. This is the document pack that is distributed.
channel	name	The Distribution channel to distribute to. Possible values are print, email, portal, and archive.
organizationalmetadata	base64Binary	A base-64 encoded binary representation of organizational metadata. This data is passed to the exit point as is.

The response only contains the standard fields.

DocumentPackDistributeOutputManagementV1

DocumentPackDistributeOutputManagementV1 submits a previously produced document pack to the CCM Batch & Output Management Hotfolder.

The request requires the following fields next to the standard fields.

Field	Type	Description
documentpack	Base64Binary	A base-64 encoded binary representation of a document pack data structure. This is the document pack to be distributed to CCM Batch & Output Management.
importrequest	Base64Binary	A base-64 encoded binary representation of an Import Request XML to be distributed to CCM Batch & Output Management.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
distributionname	message	The distribution name used to distribute the Import Request XML and document pack to CCM Batch & Output Management.

DocumentPackDistributeWorkplaceV1

Note Support for distribution to a Perceptive Workplace server has been removed.

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

The request requires the following fields next to the standard fields.

Field	Type	Description
documentpack	base64Binary	A base-64 encoded binary representation of a Document Pack data structure. This is the document pack submitted for distribution to the workplace.
user	message	The user name of the account used to upload documents.
password	message	The password of the account used to upload documents.

workplaceurl	url	URL of the workplace server to which the Document Pack is submitted. This URL should be formatted in the following way: https://<server>:<port>[/<service-context>]
location	message	A full path to the space or folder in which the documents are placed. This path must start with two separator characters. This separator determines where new subfolders are generated with this path and can be any character that does not occur in any of the object names (project or folder). Typically, the slash sign is the character. The location should not contain double quotation marks, tabs, or new lines. This path should be constructed as follows: `<path separator><path separator><projectname>` followed by 0 or more `<path separator><foldername>` **Example** `MyProject/MyFolder/MySubfolder`

outputformat	message	Optional. The requested output format for the documents placed in workplace. Currently, only native, PDF, and PDF/A-1b are supported: native: The original document format is uploaded to the workplace. For Document Templates based on a Master Template or a Quick Template, these are DOCX or DOC documents. For Static Documents, the document as it is stored in CCM Repository. Imported documents are uploaded as is. PDF: If a PDF version of a document is already present, it is uploaded. Otherwise, if a DOCX or DOC version is present, it is converted to PDF, and the result is uploaded. The remaining documents are uploaded as is. PDF/A-1b: If a PDF/A-1b version of a document is already present, it is uploaded. Otherwise, if a DOCX or DOC version is present, it is converted to PDF/A-1b, and the result is uploaded. The remaining documents are uploaded as is. This includes PDF documents that have no DOCX or DOC variant. The default value of this parameter is native.

The response contains the following fields in addition to the standard fields.

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

The documents within the Perceptive Workplace destination may be named in one of the following ways:

- If a slot identifier is defined for the Document Template, the name appears as *<n>-<slot identifier>.<extension>*, where *n* is the location of the document in the generated document pack, so *n* specifies the order.
- If no slot identifier is defined, the name appears as *<n>-<document template name>.<extension>*

Example

1-Welcome Letter Agricultural.pdf

2-Policy.pdf

3-Terms and Conditions 2016.pdf

DocumentPackSignV1

DocumentPackSignV1 submits a previously produced document pack to a Kofax SignDoc installation for signing and further handling. Use this call when the documents in a generated document pack have to be digitally signed.

The Document Pack submitted for signing should only contain Microsoft Word or PDF documents. If the document pack contains documents of other types, the document pack is rejected for signing by SignDoc.

The request requires the following fields next to the standard fields.

Field	Type	Description
documentpack	base64Binary	A base-64 encoded binary representation of a document pack data structure. This is the document pack submitted for signing and it should contain at least one Word document with signature lines that can be processed by Kofax SignDoc. It should only contain Word or PDF documents.
signdocurl	url	A URL of the SignDoc Rest API to which the document pack is submitted. This URL should be formatted in the following way: <signdocServerAddress>/<SignDocApplication>/rest/ **Example** http://signdocserver.com/cirrus/rest
signdocapikey	message	The API key used for authenticating calls to the SignDoc API. You can retrieve this key from the SignDoc installation.
signdocpackageid	message	The SignDoc ID (the name) of the signing package. This should be a new ID consisting of alphanumerical characters, underscores, and minus signs (see the Kofax SignDoc documentation for more information). If an existing ID or empty ID is passed, an error is generated.

The response contains the following fields in addition to the standard fields.

Field	Type	Description

signdocpackageid	message	The SignDoc ID of the created signing package. This is currently the same as signdocpackageid that is passed as an input argument. We advise that you use returned signdocpackageid for future reference, as future releases may return a modified ID.

ComposeDocumentPackV1

The ComposeDocumentPackV1 call creates a document pack from non-interactive Document Templates and Document Pack Templates. The document pack is returned in the response and can also be retrieved using ComposeDocumentPackGetV1 (see ComposeDocumentPackGetV1).

The request requires the following fields next to the standard fields.

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	A base-64 encoded representation of the Data Backbone XML.
status	name	Optional. The status of the template to use. The available statuses are Current, Accepted, or Published. If omitted, Published is used.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier.
documentpack	base64Binary	A base-64 encoded binary representation of the resulting document pack data structure.

ComposeDocumentPackV2

The ComposeDocumentPackV2 call creates a document pack from non-interactive Document Templates and Document Pack Templates. The document pack is returned in the response and can also be retrieved using ComposeDocumentPackGetV1 (see ComposeDocumentPackGetV1).

The request requires the following fields next to the standard fields.

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	A base-64 encoded representation of the Data Backbone XML.
status	name	Optional. The status of the template to use. The available statuses are Current, Accepted, or Published. If omitted, Published is used.
importdocuments	base64Binary	Optional. A base-64 encoded XML structure binary representation of an import documents data structure. For more information on the format for this XML, see Format for the definitions.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier.
documentpack	base64Binary	A base-64 encoded binary representation of the resulting document pack data structure.

ComposeDocumentPackInteractiveStartV1

ComposeDocumentPackInteractiveStartV1 creates a Document Pack based on either an interactive Document Template or an interactive Document Pack Template. The call results in a session ID and (partial) URL that can be used to start the interaction in ComposerUI. After the interaction has been completed, you can retrieve the resulting document pack using ComposeDocumentPackGetV1 (see ComposeDocumentPackGetV1).

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	Name of the CCM Repository project.
templatetype	name	Either documentpacktemplate, documenttemplate, or letterbook.
templatename	repositoryobjectname	Name of the template object to compose the document pack from.
databackbonexml	base64Binary	A base-64 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. The status of the selected template. The available statuses are Current, Accepted, or Published. If omitted, Published is used.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier.
url	url	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URL's that are returned are relative to this base URL. With this URL, the interactive session can be started.

ComposeDocumentPackInteractiveStartV2

ComposeDocumentPackInteractiveStartV2 creates a Document Pack based on either an interactive Document Template or an interactive Document Pack Template. The call results in a session ID and (partial) URL that can be used to start the interaction in ComposerUI. After the interaction has been completed, you can retrieve the resulting document pack using ComposeDocumentPackGetV1 (see ComposeDocumentPackGetV1).

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	Name of the CCM Repository project.
templatetype	name	Either documentpacktemplate, documenttemplate, or letterbook.
templatename	repositoryobjectname	Name of the template object to compose the document pack from.
databackbonexml	base64Binary	A base-64 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. The status of the selected template. The available statuses are Current, Accepted, or Published. If omitted, Published is used.

importdocuments	base64Binary	Optional. A base-64 encoded XML structure binary representation of an import documents data structure. For more information on the format for this XML, see Format for the definitions.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier.
url	url	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URLs that are returned are relative to this base URL. With this URL, the interactive session can be started.

ComposeInteractiveGetSuspendedSessionV1

You can use this functionality to pause longer interactive processes. ComposeInteractiveGetSuspendedSessionV1 retrieves a suspended session from <ccm_server>. It gets the session state that was returned by ComposeDocumenPackInteractiveStart as a parameter and returns new sessionfile that allows the CCM ComposerUI for HTML5 web application to proceed where it left off earlier.

This can only be done for interactive sessions that have been initiated with their allowsuspend field set to true.

The request requires the following fields next to the standard fields.

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 contains the following fields in addition to the standard fields.

Field	Type	Description
sessionfile	base64Binary	A base-64 encoded binary representation of a CCM session archive data structure. This file can be used in ComposeInteractiveResumeSuspendedSessionV1 to resume the suspended interaction.

ComposeInteractiveResumeSuspendedSessionV1

ComposeInteractiveResumeSuspendedSessionV1 resumes a suspended session retrieved by ComposeInteractiveGetSuspendedSessionV1. The returned values resume the interactive process at the point the user left.

The request requires the following fields next to the standard fields.

Field	Type	Description
sessionfile	base64Binary	A base-64 encoded binary representation of a CCM session archive data structure. This is the suspended session retrieved by ComposeInteractiveGetSuspendedSessionV1. It contains all information to continue the interactive process.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier. This is the CCM session identifier of the suspended session.
url	url	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URLs that are returned are relative to this base URL. With this URL, the interactive session can be resumed.

ComposeDocumentPackGetV1

You can use the ComposeDocumentPackGetV1 call after finishing the interactive part of a template run to retrieve the result of all composition calls that create a document pack.

The request requires the following fields next to the standard fields.

Field	Type	Description
ccmsessionid	sessionid	CCM Contract Manager session identifier. This is the CCM session identifier that was returned at the start of the composition call.

The response contains the following fields in addition to the standard fields.

Field	Type	Description

documentpack	base64Binary	A base-64 encoded binary representation of the document pack.

ComposeDocxV1

The ComposeDocxV1 call composes a DOCX document from a Document Template.

The request requires the following fields next to the standard fields.

Field	Type	Description
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 is 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.

The response contains the following fields in addition to the standard 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 contains a SOAP Fault structure.

ComposePdfV1

The ComposePdfV1 call composes a PDF document from a Document Template.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The Designer project that contains the Document Template to use.
documenttemplate	repositoryobjectname	The Document Template to use. Only the following formats are supported: DOC, DOCX, and PDF.

status	message	Optional. The status of the Document Template to use. The available statuses are Current, Accepted, or Published. If omitted, Published is 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.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
document	base64Binary	A base-64 encoded PDF document.
databackbonexml	url	A base-64 encoded Data Backbone XML.

In case of an error, the response contains a SOAP Fault structure.

ComposeDocxInteractiveStartV1

The ComposeDocxInteractiveStartV1 call starts an interactive run that composes a DOCX document from a Document Template or a Letter Book.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The Designer project that contains the Document Template or the Letter Book 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 Letter Book 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 is 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. HTML conversions proceed through the MakeHTMLDocument.dss Core scripting exit point. This exit point makes uses of the .msxsl transformation tool. 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.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
itpcloudid	sessionId	An identifier for the interactive run.
url	url	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URLs that are returned are relative to this base URL. This URL refers to JSON data that serves as input to your interactive ComposerUI web application.

In case of an error, the response contains a SOAP Fault structure.

ComposeDocxInteractiveGetV1

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

The request requires the following fields next to the standard fields.

Field	Type	Description
itpcloudid	sessionId	Specifies the identifier of the run started earlier.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
document	base64Binary	A base-64 encoded DOCX document.
databackbonexml	url	A base-64 encoded Data Backbone XML.

In case of an error, the response contains a SOAP Fault structure.

ComposePdfInteractiveStartV1

The ComposePdfInteractiveStartV1 call starts an interactive run that composes a PDF document from a Document Template or a Letter Book.

The request requires the following fields next to the standard fields.

Field	Type	Description
project	repositoryobjectname	The Designer project that contains the Document Template or the Letter Book 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. Only the following Document Template formats are supported: DOC, DOCX, and PDF.
lettterbook	repositoryobjectname	Optional. If specified, the Letter Book 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 is 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 is 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.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
itpcloudid	sessionId	An identifier for the interactive run.
url	url	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URLs that are returned are relative to this base URL. This URL refers to JSON data that serves as input to your interactive ComposerUI web application.

In case of an error, the response contains a SOAP Fault structure.

ComposePdfInteractiveGetV1

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

The request requires the following fields next to the standard fields.

Field	Type	Description
itpcloudid	sessionId	Specifies the identifier of the run started earlier.
securedocument	boolean	Optional: Indicates whether a secured PDF document must be produced. If omitted, the default value false is used.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
document	base64Binary	A base-64 encoded PDF document
databackbonexml	url	A base-64 encoded Data Backbone XML.

In case of an error, the response contains a SOAP Fault structure.

ComposeInteractiveFinishV1

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

The request requires the following fields next to the standard fields.

Field	Type	Description
itpcloudid	sessionId	Specifies the identifier of the run started earlier.

The response only contains the standard fields.

In case of an error, the response contains a SOAP Fault structure.

FinishSessionV1

You can use FinishSessionV1 to explicitly notify CCM that all interaction with the session has been finished.

The request requires the following fields next to the standard fields.

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

The response only contains the standard fields.

SystemCheckV1

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

The request requires the following fields next to the standard fields.

Field	Type	Description
text	message	A text that ends up in the produced test document.

The response contains the following fields in addition to the standard 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 contains a SOAP Fault structure.

SystemCheckInteractiveStartV1

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

The request requires the following fields next to the standard fields.

Field	Type	Description
text	message	A text that ends up in the produced test document.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
itpcloudid	sessionId	An identifier for the interactive run.
url	url	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URLs that are returned are relative to this base URL. This URL refers to JSON data that serves as input to your interactive ComposerUI web application.

In case of an error, the response contains a SOAP Fault structure.

SystemCheckInteractiveStartV2

SystemCheckInteractiveStartV2 starts an interactive test run. It produces a simple test document to verify that the CCM installation is up and running and can produce documents as intended. This document can be retrieved using SystemCheckInteractiveGetV2 with the returned ccmsessionid field (see SystemCheckInteractiveGetV1).

The request requires the following fields next to the standard fields.

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

The response contains the following fields in addition to the standard fields.

Field	Type	Description
ccmsessiondid	sessionid	The CCM Contract Manager session identifier to be used by SystemCheckInteractiveGetV2 to retrieve the resulting document.
url	url	A URL that is relative to the base URL of Kofax Customer Communications Manager. Example: https://<ccm server>:443/ccm/ All relative URLs that are returned are relative to this base URL. With this URL, the interactive process can be started.

SystemCheckInteractiveGetV1

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

The request requires the following fields next to the standard fields.

Field	Type	Description
itpcloudid	sessionId	Specifies the identifier of the run started earlier.

The response contains the following fields in addition to the standard fields.

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

In case of an error, the response contains a SOAP Fault structure.

SystemCheckInteractiveGetV2

SystemCheckInteractiveGetV2 obtains the PDF test document produced by the interactive run initiated through SystemCheckInteractiveStartV2.

The request requires the following fields next to the standard fields.

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

The response contains the following fields in addition to the standard fields.

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

CCMCapabilitiesVerifyContractTypeAvailableV1

This call returns information on the availability of a contract type for a contract.

This request only requires the standard fields.

The response contains the following fields in addition to the standard fields.

Field	Type	Description
contracttypeisavailable	boolean	Returns true if a version of the contracttype is available. It also returns true if the requested contracttype version is not available but another version is.

requestedversionisavailable	boolean	Returns true if both the requested contracttype and the contracttype version are available.
highestversionavailable	version	Optional. This field is returned in the following case: The requested contracttype is available. The requested contracttype version is not available. The returned value is the highest version that is available for the requested contracttype.

Format for the definitions

The following sections contain information on the available XML files and their usage. You can find their respective XSD files in the schemas directory that resides in: `<deploy root>\CCM\Documentation\5.2\Resources\Schemas`.

The B&OM XSD files are located in: `<deploy root>\CCM\Documentation\5.2\Resources\Schemas\Output Management`.

Document Pack Manifest XML

The Document Pack Manifest XML resides in the root of each Document Pack and describes the contents of the Document Pack. It specifies the slots where documents reside within the Document Pack.

Databackbone node

The databackbone node at the top level indicates either the result of the Data Preparation Template or the input databackbone when no Data Preparation Template is present in a Document Pack Template. Databackbone nodes for Document Templates indicate the resulting Data Backbone after the template is run for this item.

Item node

Each item node refers to a document within the Document Pack. For every Document Template 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 Kofax TotalAgility has passed a Closed Loop Identifier on the request and/or the Document Template has modified it.

Document Pack Template XML

A Document Pack Template XML is returned by DesignerGetDocumentPackTemplateV1. It describes the structure of a Document Pack Template by listing which slots the document pack consists of and how their

content should be created or added. The slot types are documentTemplate and importDocument. It also contains information about whether a slot is optional or conditional and marks the first template explicitly as cover letter.

Document Template XML

A Document Template XML is returned by DesignerGetDocumentTemplateV1. The Document Template XML describes a specific Document Template as defined in CCM Designer. It contains a name and description, information about which Master Template it is based on, and which parameters it takes.

Letter Book XML

A Letter Book XML describes a Letter Book and its contents, including a list of Document Templates and folders.

List Letter Books XML

The List Letter Books XML contains a list of Letter Books in a project, including descriptions.

List Projects XML

The List Projects XML contains a list of all projects in CCM Repository.

List Templates XML

The List Templates XML contains a list of the Document Templates and Document Pack Templates in a project.

Import Documents XML

Import Documents XML can be used to provide the required Import Documents for the composition of a Document Pack. It contains the base64 encoded files and an identification of the slot in the Document Pack Template. The name attribute of each importDocuments element specifies the slot identifier. The version attribute of the importDocuments element must be set to 1. No verification of MIME type or content takes place.

When an import document cannot be mapped to a slot in the Document Pack Template, it is ignored. If a document is not required, add it to the XML Optional Import Documents so that it is included in the Document Pack. Import documents are added to the Document Pack as they are. Import documents are not subject to conversion requests.

CCM Core exit points

This section contains information on the CCM Core exit point scripts that apply to the CCM API calls described in this guide. You can modify the exit points to provide custom functionality.

ContractDocToPDF.dss

This script is called when a call converts a document into the PDF format document. You can modify this script to alter parameters of the `DocToPDF` command or to use an alternative PDF converter.

Parameters

```
Parameter Text    Src;
Parameter Text    Dest;
Parameter Boolean ProducePDFA;
```

The parameter `Src` indicates the file name of the document to be converted to PDF. The parameter `Dest` indicates the name of the resulting PDF document after conversion. The `Boolean ProducePDFA` parameter indicates whether to produce a PDF/A-1b compliant document.

Example script

The script calls the `DocToPDF` command using the given parameters.

```
DocToPDF
  Src (Src)
  Dest (Dest)
  ProducePDFA (ProducePDFA);
```

You can add any additional parameter available to `DocToPDF`.

For example, to always embed all non-standard PostScript fonts, add the parameter `EmbedFonts(True)` as shown in the following example. To switch to the built-in Rendition technology, add the parameter `Processor("Rendition")`. These parameters override the global settings in the dp.ini file.

```
DocToPDF
  Src (Src)
  Dest (Dest)
  EmbedFont(TRUE)
  Processor("Rendition")
  ProducePDFA (ProducePDFA);
```

For more information on the command and available parameters, see the section "DocToPDF" in the *Kofax Customer Communications Manager Core Scripting Language Developer's Guide*.

DistributeDocumentPack.dss

You need to modify the DistributeDocumentPack exit point to script the distribution of Document Packs.

By default, the DistributeDocumentPack exit point does not perform any distribution. When no modifications are made and the exit point is called, an error message is shown.

Parameters

```
Parameter Text DocumentPack;
Parameter Text Manifest;
Parameter Text DistributionChannel;
Parameter Text OrganizationalMetaData;
```

The parameter `DocumentPack` indicates the location of the distributed Document Pack on the file system. The parameter `Manifest` indicates the location of the Manifest.xml. The Manifest.xml contains

information on the content of the Document Pack. Distribution channel can be used to distinguish the purpose of the distribution. Possible values are: print, email, portal, and archive. The parameter `OrganizationalMetaData` indicates the location of the Organizational Metadata XML on the file system. The Organizational Metadata XML can be used to pass key value pairs on for distribution.

Note There is no correlation between the Distribution channel and the channels that can be configured as part of the Batch & Output Management to produce documents with different formats and purposes.

KTADistributeDocumentPack.dss

You need to modify the KTADistributeDocumentPack exit point to script the distribution of Document Packs by Kofax TotalAgility (also known as KTA). By default, this exit point invokes the general CCM Core DistributeDocumentPack exit point, which means that distribution can be implemented in a general way through the DistributeDocumentPack exit point or through a KTA specific exit point.

Parameters

```
Parameter Text DocumentPack;
Parameter Text Manifest;
Parameter Text DistributionChannel;
Parameter Text OrganizationalMetaData;
```

The parameter `DocumentPack` indicates the location of the distributed Document Pack on the file system. The parameter `Manifest` indicates the location of the Manifest.xml. The Manifest.xml contains information on the content of the Document Pack. Distribution channel can be used to distinguish the purpose of the distribution. Possible values are: print, email, portal, and archive. The parameter `OrganizationalMetaData` indicates the location of the Organizational Metadata XML on the file system. The Organizational Metadata XML can be used to pass key value pairs on for distribution.

Note The DistributeDocumentPack exit point is invoked when the DocumentPackDistribute API call is invoked. The KTADistributeDocumentPack exit point is called when the DistributeDocumentPack exit point is called from one of the KTACCMDistribution contract types.