IBM Cúram Social Program Management
Version 7.0.5

*Cúram Content Management
Interoperability Services Integration
Guide*

IBM

Note

Before using this information and the product it supports, read the information in "Notices" on page 19

Contents

List of Figures

List of Tables

Chapter 1. Integrating with a Content Management System

Use this information to register a target system so that you can use a content management system as a repository for documents. When integration with a content management system is enabled, attachments, Microsoft Word communications, and pro forma communications documents are stored and retrieved from the Content management system.

Introduction

Purpose

The purpose of this guide is to outline the available configuration options for integrating Cúram with a Content Management System (CMS) and to provide detail on what these configurations entail.

Prerequisites

The reader should be familiar with both the basic elements of case processing in the human services industry and with the case management functionality available in the application.

Before reading this guide the reader should be familiar with Cúram attachments and communications functionality described in the `Cúram Integrated Case Management Guide` and the `Cúram Communications Guide`.

Audience

This guide is intended for developers responsible for configuring Cúram to integrate with a Content Management System.

Background

Content management systems are used by organizations to store and manage various types of content. Using the Content Management Interoperability Services (CMIS) standard, the system provides the ability to integrate the documents associated with attachments, Microsoft Word communications, and pro forma communications with a CMS.

Cúram CMIS integration has been verified with IBM Filenet P8 v5.1 hosted on Windows Server 2008 R2 (64-bit edition).

For more information on CMIS standards, please consult http://docs.oasis-open.org/cmis/CMIS/v1.0/cmis-spec-v1.0.html.

Chapters in this Guide

The following list describes the chapters in this guide:

Configuring Cúram for use with a Content Management System
This chapter covers the administrative configuration options that are available to use a CMS as a repository for documents.

Integrating a Content Management System with Cúram
This chapter discusses the integration of documents with a CMS.

Metadata
This chapter gives an overview of how CMIS uses CMS metadata and how to configure the application to use metadata properties provided by default.

Customization
>This chapter outlines how to customize metadata, implement custom directory structure and file naming strategies, and Single Sign On.

Configuring Cúram for Use with a Content Management System

Introduction

Before a content management system can be used as a repository for documents, it must first be registered as a target system. Cúram must also be configured so that the application can communicate with it. This chapter describes how to go about both of these tasks.

Registering a Target System

In order to allow two-way communication between Cúram and a content management system, a system administrator first needs to set up and configure the details for the content management target system. To register a target system, log in to the system administration application, and select to create a new target system. Enter a unique name for the target system, and enter the target system URL.

Adding a Content Management Service to the Target System

When the target system is registered, the next step is to add the service. When you create a service, you must specify the following service settings:

Table 1: CMS Target System Service Settings.		
This table describes the CMS service settings that must be specified for a target system.		
Setting	**Description**	**Mandatory**
Service Name	Unique name of the service. Service Name must be set to 'Content Management Interoperability Service over Atompub'.	Yes
Extension	URL extension that is added to the target system address to give the full location of the CMIS repository.	Yes
Invoking User Username	Login user name is required by the CMS Atomic service. By default, this global account is required to connect to the CMS. It can be left blank if CMIS Single Sign On support is implemented by the organization and enabled in the application. For more information on Single Sign On, see the *CMIS Customization* Chapter.	No

Table 1: CMS Target System Service Settings.		
This table describes the CMS service settings that must be specified for a target system.		
(continued)		
Setting	**Description**	**Mandatory**
Invoking User Password	Login password is required by the CMS ATOM service. By default, this global account is required to connect to the CMS. It can be left blank if CMIS Single Sign On support is implemented by the organization and enabled in the application. For more information on Single Sign On, see *CMIS Customization* Chapter. The password is encrypted by Curam encryption algorithms during the creation of the target system service. The plain text password is never stored and Curam only ever compares the encrypted values for authentication. Subsequent editing of the service does not retrieve the password. If you edit the service, it displays an empty box for the **Invoking User Password** field. For more information, see *Cryptography in Curam*.	No

Note: It is only a valid configuration to have exactly one Target System service named "Content Management Interoperability Service over Atompub".

Related concepts

Cryptography in Cúram

Activating the Content Management Service

Once the CMS service has been added to the target system, it must then be configured and activated. Configuration is done by means of a group of system properties, 'Application Properties - Content Management Settings', which can be found in the Property Administration section of the System Administration application.

Update the CMIS properties listed below as follows:

Property	Description	Default Value
Table 2: CMIS Target System Service Settings. This table describes the CMIS application properties that must be configured.		
curam.cms.enable	Indicates if the Content Management Interoperability Service is enabled. True/false flag. True determines that the storage location for certain files is in a configured CMS. False determines that the storage location is in the Cúram database.	true
curam.cms.attachment.enable	Indicates if storing attachments to the CMS is enabled. True/false flag. True determines that the storage location for attachments is in a configured CMS. False determines that the storage location is in the Cúram database.	true
curam.cms.proforma.enable	Indicates if storing pro forma communications to the CMS is enabled. True/false flag. True determines that the storage location for communications is in a configured CMS. False determines that the storage location is in the database.	true
curam.cms.metadata.enable	Indicates if CMIS metadata is enabled. True/false flag. True determines that associated metadata will be uploaded alongside any file uploaded to the CMS. False determines that the file alone will be uploaded. Before setting this to true, the metadata properties must first be configured on the CMS as outlined in this guide.	false
curam.cms.curam.dir	The absolute path to the root directory for Cúram content within the global CMS repository used by the Cúram application. Must begin with a '/', must not end with a '/', and must not contain any unsupported characters. Refer to the CMS documentation for more specific information on what characters are unsupported.	Set based on the organization implementation, e.g., /Curam

This table describes the CMIS application properties that must be configured.

(continued)

Property	Description	Default Value
curam.cms.repository.name	The single repository for the global CMS account used by the Cúram application.	Set based on the organization implementation.
curam.cms.sso.enable	Indicates if Single Sign On (SSO) is enabled. True/false flag. True determines that SSO support is enabled. False determines that SSO support is disabled and that default authentication will be used. This should only be set to "true" when the organization has implemented the custom code for SSO. Further information can be found in the chapter on customization.	false
curam.cms.connectiontimeout	The number of milliseconds spent connecting with the configured CMS that will force a connection failure. Set to any positive integer or zero. Left as zero, this will be ignored.	0
curam.cms.readtimeout	The number of milliseconds spent reading from the configured CMS that will force a read failure. Set to any positive integer or zero. Left as zero, this will be ignored.	0

Note: The `curam.cms.curam.dir` property must begin with a '/', must not end with a '/', and must not contain any unsupported characters. Refer to the CMS documentation for more specific information on what characters are unsupported.

Note: The CMIS application properties are not designed to be changed once a system has gone live without extensive analysis and offline reconfiguration of the system.

Integrating a Content Management System with Cúram

Introduction

When integration with a CMS is enabled, attachments, Microsoft Word communications, and pro forma communications documents are stored and retrieved from the CMS, rather than being stored in the Cúram database.

Taking on this Functionality

Organizations that wish to utilize CMIS with attachments will need to check whether they have overridden Cúram's Attachment entity implementation. If it has been overridden, the customization can continue to be used as well as the CMIS with attachments functionality - with some impact. This involves updating

the custom version in line with Cúram's latest version. The custom implementation should be updated to implement the same functions as the Cúram implementation, and each custom function should be updated to call the Cúram implementation.

Note: CMIS integration is not supported for Cúram batch jobs.

Note: Cúram's CMIS integration only supports versionable document types; non-versionable document types are not supported.

In order to use the Cúram functionality, some manual configuration is required on the CMS. This involves adding a `CuramDocument` class as a subclass of `cmis:document`, and a `CuramAttachment` class as a subclass of this `CuramDocument` class. The following properties are required on the `CuramAttachment` document class on the CMS:

Table 3: CMIS Metadata Properties	
Property	**Type**
caseReference	String
communicationDate	Date
documentReceiptDate	Date
documentType	String
documentTypeCode	String
participantDOB	Date
participantFirstName	String
participantLastName	String
participantReference	String
applicationReference	String

Gradual Migration

Organizations that are already using the Cúram application and storing files in the application database can enable CMIS integration without performing an upfront migration of the content in the CMS repository. All of the documents stored in the Cúram application database will remain there and be fully accessible. Documents will only be saved to the CMS after any changes are made to a document or after a new document is created. After a document has been saved to the CMS, it will also be removed from the application database.

Important: Once CMIS is enabled, it should not be disabled because the documents stored on the CMS would no longer be accessible from within the application.

Custom CMIS Attachment integration points that read or modify content will need to be created or updated. The integration points that read will need to be updated to check whether the content exists on the CMS. The modify integration points will need to be updated according to the following pseudocode logic.

```
if CMIS is enabled {
  if content exists on the CMS for the record in question {
    modify the contents on the content management system
  } else {
    create the contents on the content management system
    blank the application database copy (optional)
  }
} else CMIS is not enabled {
  maintain the database copy
}
```

Figure 1: Pseudocode for Modify Integration Points

Note: Custom Pro Forma Communication integration points are not affected.

Securing Documents in a CMS

It is important to note that access to documents stored in a CMS will only be secured through existing Cúram security mechanisms, such as data based security and location based security. For more information on data and location based security, please consult the `Cúram System Configuration Guide` and the `Cúram Security Guide`. Customers will need to provide their own means of securing alternate forms of access to documents in the CMS, such as security for accessing documents directly from the CMS.

Attachments

Attachment records in the Cúram database no longer store the associated file content. Instead, the file along with any configured attachment metadata elements that currently exist for the attachment, are stored in the CMS. When a user subsequently selects to view the document, it is retrieved from the CMS. If the user selects to upload a new version of the file, the original document stored in the CMS is superseded by the new document. If the user updates any configured attachment metadata elements, they are also updated in the CMS. Reading this attachment will always return the latest version of the file on the CMS. In CMS repositories which support version control of documents, superseded versions of documents will still be available to view directly via the CMS application.

Microsoft Word Communications

For Microsoft Word communications, the Microsoft Word document is stored in the CMS when the communication is created, along with any configured attachment metadata elements that currently exist for the Microsoft Word document. It is retrieved from the CMS when a user selects to open the Microsoft Word document. When modifications are made to the Microsoft Word document, a new version of the document is stored in the CMS, with the latest version being the version retrieved when the Microsoft Word document is subsequently opened again.

Pro Forma Communications

For pro forma communications, the PDF file that is generated using a pro forma template is stored in the CMS when the status of the pro forma communication is set to 'Sent'. Once the PDF file is stored in the CMS, any requests to preview the pro forma communication retrieves the file from the CMS.

Note: No metadata is stored for Pro Forma Communications, other than the document title.

Metadata

Overview

It is possible to store metadata with a document that is stored in the CMS. All CMSs supporting CMIS must save some `cmis:document` metadata by default for all files. Of this system metadata, most CMS front ends only highlight the document title, but the rest of it is usually accessible.

Once CMIS is enabled in Cúram, metadata is stored for all Cúram Attachments by default. Custom Attachment integration points can be updated to use the `CMISAccess` API functions to store and modify additional metadata properties along with a file stored in a CMS. Attachments are stored as Class type `CuramAttachment`, which is a sub-class of `CuramDocument`, itself a sub-class of `cmis:document`. Pro Forma communication attachments are stored as `CuramDocuments` and do not support metadata.

Metadata is stored for all Attachments, including attachments associated to Recorded Communications and Microsoft Word Communications. By default, ten metadata properties for Attachments can be stored.

- `cmis:document`
  - `CuramDocument`
    - `CuramAttachment`
      - *caseReference*
      - *communicationDate*
      - *documentReceiptDate*
      - *documentType*
      - *documentTypeCode*
      - *participantDOB*
      - *participantFirstName*
      - *participantLastName*
      - *participantReference*
      - *applicationReference*

Figure 2: Class Type Tree

An administrative user can enable/disable the storage of individual metadata properties. Enabling/disabling a metadata property applies globally to all attachment integration points.

The metadata that is stored depends upon the Attachment business flow, e.g. if the Case Reference metadata property is enabled, it will be stored as metadata for a case attachment created within an Integrated Case, but it will not be stored for a participant attachment created for a Person as it is not relevant.

The following table provides a description of each of the ten default metadata properties, including information about when and how the metadata will be stored:

Table 4: CMIS Metadata Descriptions	
Metadata Element	**Description**
Case Reference	The case reference number of the case in which the attachment or recorded communication or MS Word communication is created, if created within a case. For example, if an attachment is created within a service plan, the case reference of the service plan is stored, not the case reference of the case in which the service plan was created.
Participant Reference	The participant reference number, as stored as the primary alternate ID for the participant. This is stored for all participant attachments, and also for attachments created in other contexts such as within a case if the attachment business flow allows for the user to select a specific participant while creating the attachment.
Participant First Name	The first name of a participant, as stored as part of the primary alternate name for the participant. This metadata property is only stored for person and prospect person type participants.
Participant Last Name	The last name of a participant, as stored as part of the primary alternate name for the participant. This metadata property is only stored for person and prospect person type participants.

Table 4: CMIS Metadata Descriptions (continued)	
Metadata Element	**Description**
Participant Date of Birth	The date of birth of a participant. This metadata property is only stored for person and prospect person type participants.
Document Type	The type of document, as captured in the Document Type field when creating an attachment. This metadata property is stored for all attachments for which this information is stored in the application database, but will not be stored if a particular attachment business flow does not capture this information.
Document Type Code	The code for the document type, as captured in the Document Type field when creating an attachment. This metadata property is stored for all attachments for which this information is stored in the application database, but will not be stored if a particular attachment business flow does not capture this information.
Document Receipt Date	The date on which the document was received, as captured in the Receipt Date field when creating an attachment. This metadata property is stored for all attachments for which this information is stored in the application database, but will not be stored if a particular attachment business flow does not capture this information.
Communication Date	The communication date of the communication to which the attachment is associated. This metadata property is only stored for attachments associated to Recorded Communications.
Application Reference	The reference number for the application case

Table 5: CMIS Metadata Descriptions	
Metadata Element	**Description**
Case Reference	The case reference number of the case in which the attachment or recorded communication or MS Word communication is created, if created within a case.
Participant Reference	The participant reference number, as stored as the primary alternate ID for the participant. This is stored for all participant attachments, and also for attachments created in other contexts such as within a case if the attachment business flow allows for the user to select a specific participant while creating the attachment.

Table 5: CMIS Metadata Descriptions (continued)	
Metadata Element	**Description**
Participant First Name	The first name of a participant, as stored as part of the primary alternate name for the participant. This metadata property is only stored for person and prospect person type participants.
Participant Last Name	The last name of a participant, as stored as part of the primary alternate name for the participant. This metadata property is only stored for person and prospect person type participants.
Participant Date of Birth	The date of birth of a participant. This metadata property is only stored for person and prospect person type participants.
Document Type	The type of document, as captured in the Document Type field when creating an attachment. This metadata property is stored for all attachments for which this information is stored in the application database, but will not be stored if a particular attachment business flow does not capture this information.
Document Type Code	The code for the document type, as captured in the Document Type field when creating an attachment. This metadata property is stored for all attachments for which this information is stored in the application database, but will not be stored if a particular attachment business flow does not capture this information.
Document Receipt Date	The date on which the document was received, as captured in the Receipt Date field when creating an attachment. This metadata property is stored for all attachments for which this information is stored in the application database, but will not be stored if a particular attachment business flow does not capture this information.
Communication Date	The communication date of the communication to which the attachment is associated. This metadata property is only stored for attachments associated to Recorded Communications.
Application Reference	The reference number for the application case

The metadata that is modified depends upon the Attachment business flow and what data that is stored as metadata can be modified in the application. For example, for Integrated Case Attachments, the document type for the attachment can be modified in the application and if it is it will be modified in the CMS; however the case reference of the attachment cannot be modified. If an attachment record is initially created without an associated file, no file and metadata will initially be stored in the CMS, but if the attachment is subsequently updated and a file is associated, all metadata elements that are enabled and available will then be stored along with the file in the CMS.

It is also possible to set up a custom implementation to include additional metadata properties through the use of the new APIs.

Note: The Infrastructure only updates the file content if it has changed and only updates the metadata properties if they are supplied.

Using Default Metadata

To use the metadata properties provided by default when CMIS is enabled, the CMS must be configured with a metadata schema to match the schema implemented on the application side. When a file is sent to the CMS through the CMIS API, contextual information including metadata is collected. The CMIS API does the following:

- Fetches all of the contextual information collected
- Derives whatever metadata it can
- Distinguishes metadata enabled via the system administration screen from other contextual information
- Calls a custom hook point which can wire up or derive additional, custom metadata
- Updates the metadata on the CMS

Configuring Metadata Properties on the CMS

To utilize the metadata support of CMIS, it is necessary to configure the CMS for metadata too. The following properties are required on the `CuramAttachment` document class on the CMS:

Table 6: CMIS Metadata Properties		
Property	**Type**	**Length**
caseReference	String	40
communicationDate	Date	n/a
documentReceiptDate	Date	n/a
documentType	String	500
documentTypeCode	String	10
participantDOB	Date	n/a
participantFirstName	String	65
participantLastName	String	65
participantReference	String	18
applicationReference	String	10

Configuring Metadata Properties in the Cúram Application

A user with system administrator privileges can manage metadata properties within the system administration application. Each individual property can be enabled or disable and multiple display names and descriptions can be specified for each property to provide multi-language support.

Note: These properties are enabled by default and should be checked, and modified if required, before going live with CMIS.

Note: Once CMIS is live, disabling a property through the application does not remove existing content from the CMS.

Customization

Overview

This chapter outlines how to customize:

1. Metadata
2. Directory Structures and File Naming Strategies
3. Single Sign On (SSO)

Customizing Metadata

In addition to the metadata properties that are stored by default, custom metadata properties can be added to the documents that are stored on the CMS.

In order to do this:

1. The additional metadata properties must be configured correctly on the CMS.
2. A custom hook point must be implemented and bound on the application side to derive or wire up the additional metadata.
3. Optionally, inputs for the custom hook point may be collected at any point along the business flow.

There are two ways of collecting extra metadata for a document, an explicit method and a transaction scoped method. Using the explicit method, it will be necessary to make modelling changes and to pass the metadata from function to function up to the integration point, where the custom code will add it to the document to be stored. This method makes it easier to follow information flow when reading or for debugging code. The transaction scoped method allows metadata to be collected anywhere along the business flow by using an object injected by Guice. As long as the metadata is collected by the Guice-injected object before the call to the CMIS infrastructure, then it will be available to wire up in the custom hook point. Metadata wired up in the custom hook point will be sent with the document to the CMS.

Configuring Additional Metadata Properties on the CMS

The Document Class on the CMS will need to be configured for the new custom metadata properties that are required. See the "Integrating a Content Management System with Cúram" and "Metadata" chapters for more information on setting up document classes and configuring metadata properties.

Note: Additional metadata properties added will be enabled and will not be configurable by the system administrator. It is assumed that if they are being implemented they are meant to be stored and not disabled.

Note: In the code samples in this chapter, we will show how to add the metadata *String* properties `xyz_contextualReference` and `xyz_updatedBy`, neither of which are included by default out of the box. These properties will need to be configured on the CMS.

Note: Some CMS implementations require recycling in order to begin using the updated metadata schema over CMIS.

Collecting Metadata Transactionally with Guice Injection

The following code sample shows how to collect metadata using a transaction-scoped object injected by Guice. In this example, the metadata is collected within the transaction that is associated with the creation/modification of a document. When the document is either added to the CMS or updated on the

CMS, the metadata stored in the transaction-scoped object will be available within the custom hook point in order to wire it up to the document.

```
@Inject
private Provider<CMSMetadataInterface> cmsMetadataProvider;

public SampleConstructor() {
  GuiceWrapper.getInjector().injectMembers(this);
}

public sampleMethod() {
  ...
  CMSMetadataInterface cmsMetadata = cmsMetadataProvider.get();
  cmsMetadata.add("xyz_contextualReference", contextualReference);
  cmsMetadata.add("xyz_updatedBy", curam.util.transaction.TransactionInfo.getProgramUser());
  ...
}
```

Figure 3: Transactional collection of contextual information or metadata

Note: The name of a custom input to the custom hook point, must, as above, be prefixed by a string such as "xyz_" where "xyz" is a customer specific prefix. Please refer to the *"Use Project-specific Prefixes in Artifact Names" section of the Curam Development Compliancy Guide* for more information.

```
// save the contents to the content management system
cmisAccess.create(details.attachmentID, CMSLINKRELATEDTYPEEntry.ATTACHMENT,
  attachmentDtls.attachmentContents.copyBytes(), attachmentDtls.attachmentName,
  CMISNAMINGTYPEEntry.ATTACHMENT, null, CMSMETADATACLASSTYPEEntry.ATTACHMENT);
```

Figure 4: Integration point for transactional method

Adding or Updating Metadata Explicitly

An alternative mechanism is to add the metadata explicitly. This method can be easier to follow the data flow when reading code or when debugging

In the method where the document is created or modified, if there is metadata to be added to the default set, the `CMISAccessImpl.addItemToExplicitMetadata` method should be called. This method takes in three parameters; the list that the metadata is to be added to, the metadata name and either a String value, or a date value.

The following code example shows how to use `CMISAccessImpl.addItemToExplicitMetadata` and how to call the overloaded `CMISAccessImpl.create` with this explicit metadata list. This will need to be replicated anywhere that custom metadata is to be added to a document that is being created.

```
CMSMetadataItemList metadataList = new CMSMetadataItemList();
cmisAccess.addItemToExplicitMetadata(metadataList, "xyz_contextualReference",
contextualReference);
```

Figure 5: Explicit collection of contextual information or metadata

```
// save the contents to the content management system
cmisAccess.create(details.attachmentID, CMSLINKRELATEDTYPEEntry.ATTACHMENT,
  attachmentDtls.attachmentContents.copyBytes(), attachmentDtls.attachmentName,
  CMISNAMINGTYPEEntry.ATTACHMENT, null, metadataList,
  CMSMETADATACLASSTYPEEntry.ATTACHMENT);
```

Figure 6: Integration point for explicit method

Wiring up and Deriving Custom Metadata

From the default set of metadata, it is possible to further derive additional custom metadata values.

To create a custom metadata hook, it is necessary to extend `curam.core.sl.infrastructure.cmis.impl.CMSMetadataClassCustomDefaultImpl`. The underlying interface itself is not to be implemented by organizations. This additional extension class must

then be bound, using a Guice module, to a codetable code on CMSMetadataClassType. The following Guice Module code sample shows how a sample implementation may be bound.

```
MapBinder<CMSMETADATACLASSTYPEEntry, CMSMetadataClassCustomInterface>
metadataStrategyCustomBinder
= MapBinder.newMapBinder(
      binder(), CMSMETADATACLASSTYPEEntry.class, CMSMetadataClassCustomInterface.class);

   metadataStrategyCustomBinder.addBinding(CMSMETADATACLASSTYPEEntry.ATTACHMENT)
      .to(SampleCustomMetadataImpl.class);
```

Figure 7: Sample Guice Module Binding

The following code sample shows how to extend the CMSMetadataClassCustomDefaultImpl class, which is used to derive and gather further metadata and then add this to the metadata to be sent to the CMS.

In the implementable functions, *properties* contains the enabled metadata properties and *otherData* contains any additional contextual information and any metadata that has been disabled.

```
public class SampleCustomMetadataImpl extends CMSMetadataClassCustomDefaultImpl {

  public void collectMetadataForNewFile(Map<String, Object> properties,
    Map<String, Object> otherData) throws AppException,
      InformationalException {

    // Manually add the metadata property to the Map
    properties.put("xyz_updatedBy",
      curam.util.transaction.TransactionInfo.getProgramUser());

    // Wire up additional metadata collected in custom facade
    properties.put("xyz_contextualReference",
      otherData.get("xyz_contextualReference"));

    // Potentially derive more custom metadata
    // properties from the data available
    ...

  }
  public void collectMetadataForUpdate(Map<String, Object> properties,
    Map<String, Object> otherData) throws AppException,
      InformationalException {

    // Manually add the metadata property to the Map
    properties.put("xyz_updatedBy",
      curam.util.transaction.TransactionInfo.getProgramUser());

    // Wire up additional metadata collected in custom facade
    properties.put("xyz_contextualReference",
      otherData.get("xyz_contextualReference"));

    // Potentially derive more custom metadata
    // properties from the data available
    ...
  }
}
```

Figure 8: Sample Custom Metadata Collection

Custom Directory Structures and File Naming Strategies

It is possible to customize the way that Cúram organizes and names the files that it stores in the CMS. By default, the following directory structures and file names are used (within the directory that is specified by the curam.cms.curam.dir application property):

- <ROOT>Default/<extension>/YYYY/MM/DD/HHMMSS_<Milliseconds>/
  <filename>.<extension>
- <ROOT>Attachment/YYYY/MM/DD/<filename>.<extension>
- <ROOT>ProForma/YYYY/MM/DD/<filename>_HHMMSS_<Milliseconds>.<extension>

The `CMISNamingType` codetable determines where the files are stored on the CMS. Attachments are stored in the Attachments directory. ProFormas are stored in the ProForma directory. Any other content types that are configured as the CuramDocument base class type are stored in the Default directory (they do not exist in the product ready for immediate use).

To create a custom naming strategy, it is necessary to extend `curam.core.sl.infrastructure.cmis.impl.CMISNamingDefaultImpl`. The `CMISNamingInterface` itself is not to be implemented by organizations. This additional extension class must then be bound, by using a Guice module, to a codetable code on `CMISNamingType`. The following Guice Module code sample shows how a sample implementation can be bound.

```
MapBinder<CMISNAMINGTYPEEntry, CMISNamingInterface> namerBinder = MapBinder.newMapBinder(
        binder(), CMISNAMINGTYPEEntry.class, CMISNamingInterface.class);

    namerBinder.addBinding(CMISNAMINGTYPEEntry.ATTACHMENT).to(
      SampleAttachmentCMISImpl.class);
```

Figure 9: Sample Guice Module Binding

The `CMISNamingInterface.getFilePath` method is where the directory structure and file name are composed. The method returns the filepath as an ordered list of folder names followed by the file name including extension. The wanted directory tree is built by adding the parts of the wanted naming strategy to the list in the correct order. The filename/extension is added last.

Note: The `curam.cms.curam.dir` application property is prepended to this list within the CMIS API.

```
public class SampleAttachmentCMISImpl extends CMISNamingDefaultImpl {

  @Override
  public List<String> getFilePath(String filename, long relatedID, Object relatedData)
    throws AppException, InformationalException {

    List<String> filePathBelowRoot = new ArrayList<String>();

    filePathBelowRoot.add(
      CodeTable.getOneItem(CMISNAMINGTYPE.TABLENAME, CMISNAMINGTYPE.ATTACHMENT));

    // Format current date time
    long currentTimeMillis = System.currentTimeMillis();
    DateTime currentDateTime = new DateTime(currentTimeMillis);

    String formattedDateTime = Locale.getFormattedDateTime(currentDateTime,
      Locale.Date_ISO8601_complete);

    filePathBelowRoot.add(formattedDateTime.substring(0, 4)); // YEAR
    filePathBelowRoot.add(formattedDateTime.substring(4, 6)); // MONTH
    filePathBelowRoot.add(formattedDateTime.substring(6, 8)); // DAY
    filePathBelowRoot.add(filename);

    return filePathBelowRoot;
  }
}
```

Figure 10: Sample Attachment Naming Strategy Implementation

The filename parameter of `getFilePath` is the full file name, including the extension but not prefixed by any directory structure. `getFilePath` is a method of `CMISNamingInterface`/ `CMISNamingDefaultImpl`.

The relatedID parameter of `getFilePath` is the identifier for the Cúram object in question on the Cúram database. For example, with the Attachment integration (with the product for immediate use), the related ID would be the attachmentID, and for Communications it would be communicationID.

The relatedData parameter of `getFilePath` is any object that the customer thinks can be useful to its naming strategy implementation. No integrations with the product for immediate use implement this feature but customers are free to use it in their own naming strategies. By default, the namingType and the current date are available for the naming strategy.

Single Sign On

Single Sign On (SSO) is a mechanism to allow an authenticated user to access multiple different systems and services without having to log in on each one. Typically, authentication is shared between the systems through a transmission of security tokens.

Implementing SSO with Cúram's CMIS functionality requires custom development in order to integrate with an organization's SSO system. It is also necessary to make some configuration updates to Cúram.

Custom Development

Organizations must first implement a hook point to add custom cookies to the CMIS Atompub invocations performed by Cúram - e.g. custom security tokens used by their SSO implementation. The organization should create a class which extends the CMISAuthenticationProviderDefaultImpl class which has the method getHTTPHeaders. This additional extension class must then be bound, using a Guice module, to CMISAuthenticationProviderDefaultImpl. The following Guice Module code sample shows how a sample implementation may be bound.

```
bind(CMISAuthenticationProviderDefaultImpl.class).to(
      CustomCMISAuthenticationProvider.class);
```

Figure 11: Sample Guice module linked binding

The following code sample illustrates how to create the custom SSO implementation.

```
public class CustomSSOImplementation extends CMISAuthenticationProviderDefaultImpl {

  @Override
  public Map<String, List<String>> getHTTPHeaders(String url) {
    String tokenName = // Get token name
    String tokenValue = // Get security token value

    Map<String, List<String>> tokenMap = new HashMap<String, List<String>>();
    List<String> tokenList = new List<String>();

    tokenList.add(tokenValue);
    tokenMap.put(tokenName, tokenList);

    return tokenMap;
  }
}
```

Figure 12: Sample SSO Hook Point

Configuration

Next, it is necessary to enable SSO support. This is administered within the application by use of a system administration application property, curam.cms.sso.enable. Enabling this application property effectively means that the CMIS API:

- Enables the transmission of cookies via CMIS to a CMS.
- Disables the transmission of the global CMS username and password credentials.
- Calls the custom hook point to populate custom cookies.

Appendix: Error messages

Validation Error Messages

The following table provides a list of possible validation error messages that may be encountered by a user when a CMS is used with Cúram.

Table 7: CMIS Validation Error Messages.

This table describes the CMIS Validation error messages.

Validation	When is this displayed?
An error occurred while interacting with the Content Management System. Please inform your system administrator.	A generic exception thrown from the CMIS infrastructure when there is no associated error handler.
No application property supplied for 'curam.cms.xxxx'. Please contact your administrator.	Upon storing a file to CMIS, when a required application property is empty.
The application property 'curam.cms.curam.dir' must be in the form '/x/y/z'. Please contact your administrator.	Upon storing a file to CMIS, when the path doesn't start with a '/', ends with a '/', or contains a folder name with an unsupported character.
The configured Content Management System is unavailable. Please contact your administrator.	Upon storing a file to CMIS, when a connection to the target system cannot be made. May be triggered by genuine networking issues, by timeouts being exceeded or by incorrect target system settings.
The file was not found on the Content Management System.	Upon retrieving a file at a specified location from CMIS within Cúram which has been deleted directly from the CMS application.
The file is no longer on the Content Management System. Please contact your administrator.	Upon retrieving a file from CMIS within Cúram which has been deleted directly from the CMS application.
The Content Management System has not been configured. Please contact your administrator.	Upon accessing a CMS, when the CMIS ATOM target system has not been configured.
Multiple Content Management Systems have been configured. Please contact your administrator.	Upon accessing a CMS, when more than one CMIS ATOM target system configured.
An error occurred gathering information about a file to send to the configured Content Management System. Please inform your system administrator.	When compiling metadata about a file, one or more properties have not been configured.
An error occurred gathering information about a file to send to the configured Content Management System. Please inform your system administrator.	When compiling metadata about a file, a required metadata property has not been provided.
An error occurred updating a file on the configured Content Management System. Please inform your system administrator.	When attempting to modify a file stored in the CMS, there are no changes to be made as all information is equal to the preexisting data.
	Or, when attempting to modify a file stored in the CMS, the file cannot be checked back in.
An error occurred reading a file on the configured Content Management System. Please inform your system administrator.	When attempting to read a file from the CMS, the file could not be found.
The file '%1s' could not be found within '%2s' on the Content Management System. Please inform your system administrator.	When attempting to replace a file on the CMS, the file could not be found on the CMS.

Table 7: CMIS Validation Error Messages. This table describes the CMIS Validation error messages. *(continued)*	
Validation	**When is this displayed?**
The file '%1s' was found in multiple locations within '%2s' on the Content Management System. Modification is not supported in this scenario.	When attempting to replace a file on the CMS, more than one location within the root folder is specified.

Notices

This information was developed for products and services offered in the United States.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan Ltd. 19-21, Nihonbashi-Hakozakicho, Chuo-ku Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBMproducts. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Privacy Policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies or other similar technologies that collect each user's name, user name, password, and/or other personally identifiable information for purposes of session management, authentication, enhanced user usability, single sign-on configuration and/or other usage tracking and/or functional purposes. These cookies or other similar technologies cannot be disabled.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at " Copyright and trademark information " at http://www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other names may be trademarks of their respective owners. Other company, product, and service names may be trademarks or service marks of others.

IBM®

Part Number: