A BRIEF INSIGHT INTO MESSAGINGAPP – THE EMAIL PROCESSING COMPONENT IN EMC® DOCUMENTUM® CONTENT SERVER

Abstract
This white paper provides a general overview of the MessagingApp component of Content Server. It briefly explains about the fundamental aspects of MessagingApp, packaging and deployment of MessagingApp. It also highlights some important customer reported issues and provides checklists for troubleshooting MessagingApp related issues.

October, 2013
Table of Contents

Executive summary ........................................................................................................... 4
   Audience ....................................................................................................................... 4

Introduction to MessagingApp ......................................................................................... 4
   What is MessagingApp? ............................................................................................... 4
   What are the capabilities of MessagingApp? .............................................................. 4
   Who are the users of MessagingApp? ......................................................................... 5
   Use cases for MessagingApp ..................................................................................... 5
      Use Case 1 – Bulk ingestion of email messages ..................................................... 5
      Use Case 2 – Efficient search capabilities for email messages .............................. 5
      Use Case 3 – Retention of email messages .............................................................. 5
      Use Case 4 – Efficient bulk disposition of email messages .................................... 6

MessagingApp Internals .................................................................................................. 6
   Data Model .................................................................................................................. 7
   Introduction to EMCMF .............................................................................................. 8
   Role of MRE ............................................................................................................... 9
   Types of Email Attachments ..................................................................................... 9
   Modes of operation supported by MessagingApp ...................................................... 10

Operations on Email Messages ...................................................................................... 13
   Operations supported by MessagingApp .................................................................. 13
   Operations not supported by MessagingApp ............................................................ 14
   Use case flow diagrams ............................................................................................ 14

Packaging ....................................................................................................................... 16

Deployment ..................................................................................................................... 16

Configuration ................................................................................................................ 16
   Enabling message archive support in Webtop .......................................................... 17
   Dropping unique index for dm_message_archive ................................................. 18

Important customer cases ............................................................................................ 18

Troubleshooting checklist ............................................................................................. 18
   General Checklist ..................................................................................................... 18
   Checklist for issues specific to inbound operations ............................................... 20
   Checklist for issues specific to outbound operations ............................................. 20
   Checklist for issues specific to copy, link and move operations ............................ 21

Conclusion ...................................................................................................................... 22

References ...................................................................................................................... 22
Executive summary

This white paper provides an overview of the MessagingApp component of Content Server. It briefly explains about the fundamental aspects and capabilities of MessagingApp along with some examples of their application in real time use cases.

This white paper briefly describes EMCMF format, the role that MRE tool plays during email message processing and the modes of operation supported by MessagingApp. It mentions about the list of operations that are supported/not supported on email messages by MessagingApp. It provides a complete picture of how the control flows within a Documentum system when an email message is imported and exported using webtop through MessagingApp.

This white paper briefly describes about the packaging and deployment aspects of MessagingApp. It mentions about some key configurations to be made while setting up MessagingApp. It also highlights some of the important customer reported issues and provides checklists for troubleshooting MessagingApp related issues.

Audience

This white paper is intended for Technical Support, engineering teams working on products that make use of MessagingApp for email processing such as My Documentum for Microsoft Outlook (MDMO), Webtop, etc along with customers who make use of the email processing capabilities offered by MDMO, Webtop, etc. This whitepaper assumes that readers possess a basic knowledge of DFC and Documentum Business Object Framework (BOF).

Introduction to MessagingApp

What is MessagingApp?

MessagingApp is the email processing component of Content Server. It provides a platform for storage and retrieval of email messages in a Documentum repository. It offers a means for unified message handling within a Documentum repository. It encapsulates all the logic for managing message objects such as message content, message attachments and all the associated metadata. It also takes care of handling different operations such as import, export, view, copy, move, etc on email messages within Documentum systems.

What are the capabilities of MessagingApp?

MessagingApp offers the following capabilities:

- Provides a unified way of handling messages within a Documentum repository.
- Provides efficient means for email archiving and retention.
- Allows collaboration of emails within an organization.
• Allows full text indexing of messages along with their attachments, thereby enabling efficient searching.

• Allows parsing of email messages such that the associated metadata can be queried and retrieved through one standard channel.

• Reduces database footprint by using registered tables wherever possible and leveraging lightweight sysobjects.

• Provides efficient means for bulk disposition of email messages.

• Leverages the batching and scoping capabilities whenever available.

Who are the users of MessagingApp?
The Documentum applications that make use of MessagingApp for processing email messages include

• My Documentum for Microsoft Outlook (MDMO)
• Webtop
• Record Management, etc.

Use cases for MessagingApp
This section lists some of the key real time use cases where the capabilities of MessagingApp can be made use of.

Use Case 1 – Bulk ingestion of email messages
The objective here is to allow ingestion of huge amounts of email messages into docbase on daily basis while keeping a relatively small database footprint.

This objective is achieved through

• Storing messages as lightweight sysobjects, which consume relatively less space in the underlying database tables than a standard sysobject type.

• Using registered tables to store a high volume of side objects that hold information about attachments, message routes, users, etc.

Use Case 2 – Efficient search capabilities for email messages
The objective here is to enable the users to search email messages based on the associated metadata.

This is achieved through full text indexing of message content and the associated metadata along with all its attachments, which offers efficient means for searching email messages in Documentum repositories.

Use Case 3 – Retention of email messages
The objective here is to enable retention of email messages present in docbase.
MessagingApp computes the retention start date for the email message while it is being imported and stores it in one of the attributes of the message object in docbase. This information can then be utilized by the applications such as Retention Policy Services to enforce retention policies on the email messages present in docbase.

**Use Case 4 – Efficient bulk disposition of email messages**

The objective here is to enable efficient bulk disposition of email messages in Documentum repositories based on the enforced retention policies.

This objective is achieved by storing email messages as lightweight sysobjects that allow efficient bulk disposition of objects.

**MessagingApp Internals**

MessagingApp is involved in the processing of email messages only when the user initiates any operation on email messages through an email-aware Documentum application such as MDMO. However, other Documentum applications such as Webtop require explicit configurations in order to become email-aware, so that they can then invoke MessagingApp for processing of email messages based on user requests.

Whenever user initiates any operation on email messages from end user machine, the request first reaches the email-aware Documentum application, which then invokes appropriate MessagingApp APIs to execute the operation requested by the end user. MessagingApp executes the requested operation and returns results to its caller. MessagingApp internally makes DFC calls wherever found necessary while processing the caller’s request. DFC in turn communicates with content server and helps MessagingApp perform its operation. Finally, the email-aware Documentum application takes care of appropriately displaying the results returned by MessagingApp to the end user.

The end-to-end block diagram depicting the position of MessagingApp in a Documentum system and the role that it plays while processing user requests on email messages is illustrated in Figure 1.
Data Model

From Documentum 6.5 onwards, MessagingApp follows a lightweight sysobject based model for storing email messages in docbase. Using lightweight sysobject based model for storing messages enables relatively less consumption of space in the underlying database tables than a standard sysobject based model.

MessagingApp has been designed to handle email messages in EMCMF format only; it cannot process email messages in any other native formats such as msg. More details on EMCMF format can be found in the subsequent sections.

MessagingApp stores email messages of EMCMF format as objects of lightweight sysobject type called ‘dm_message_archive’, with a shareable parent type of ‘dm_message_container’. The information that is specific to individual email messages such as message id, subject, size, sent date, received date, sensitivity, importance, etc goes into lightweight ‘dm_message_archive’ type object, whereas other information can be shared across multiple email messages such as mode of import, retention start date, etc goes into its shareable parent type - ‘dm_message_container’. It is the responsibility of the caller who invokes the import APIs of MessagingApp to specify whether the email message that is to be imported should be materialized or not while storing it in docbase.
In addition to the ‘dm_message_archive’ and ‘dm_message_container’ types, MessagingApp also makes use of other database objects such as

- **dm_message_attachment** – This is a registered table that stores information about attachments of email messages such as their names, attachment position, format, object id, etc.

- **dm_message_route** – This is a registered table that stores the routing information associated with an email message such as address ids, route types, etc.

- **dm_message_address** – This is a registered table that stores information about unique addresses associated with an email message such as address id, full email address, user name, address type, friendly name, primary domain, sub domain, etc.

- **dm_message_route_user_data** – It stores route-specific user information associated with the email message.

These database objects are used internally by MessagingApp to store and retrieve useful information related to email messages during its operations and their use is not supported for custom applications.

From Documentum 6.6 release onwards, the database objects such as ‘dm_message_attachment’, ‘dm_message_route’ and ‘dm_message_address’ were converted to registered tables from dm_type objects, which allowed further reduction in the database footprint.

**Introduction to EMCMF**

A number of applications would like to store and retrieve email messages into a Documentum repository. The need for a unified format for email message storage is paramount to ensure application interoperability. This led to the development of EMCMF format specification.

EMCMF is a proprietary binary format that encapsulates the content, metadata and attachment information all together as a single unit of information and thereby helps us in achieving standardization across different platforms.

The intent of EMCMF format is three-fold:

1. Define a storage format that allows for preserving message fidelity when archiving and restoring mail messages to and from a mail system.

2. Allow for rendering and transfer of messages to dissimilar mail system environments.

3. Provide large attachment extraction which allows for de-duplication of large content.

MessagingApp can handle email messages of EMCMF format only; hence it is the responsibility of the caller to first convert the messages from respective native formats to EMCMF format before invoking import APIs of MessagingApp. Similarly, the
caller should also take care of converting the messages from EMCMF format to respective native formats after invoking export-APIs of MessagingApp. This conversion between EMCMF and native email message formats is achieved with the help of MRE tool.

**Role of MRE**

MRE is a windows-based conversion tool that performs conversion between EMCMF and native message formats such as .msg. In case of applications such as Webtop, it is shipped along with UCF component so that it gets downloaded to individual client machines along with UCF client and is then installed on those client machines. After this, whenever users try to perform supported inbound and outbound operations on email messages, this MRE tool will be invoked in order to:

- Convert individual messages to be imported into EMCMF files, which will then be handed over to MessagingApp by the email-aware Documentum applications during inbound operations.
- Generate original messages in native formats from EMCMF files exported by MessagingApp during outbound operations.

**Types of Email Attachments**

When a message is converted from its native format to EMCMF by MRE, its attachments can be extracted out of the EMCMF or stored inside the EMCMF file itself, which depends upon several factors such as attachment size, whether attachment is another email message or a non-email document attachment, etc.

There are three kinds of attachments that can be generated by MRE during this conversion process, namely:

1. **Plain internal attachments or small attachments**
   - They are stored internally in EMCMF object itself after message to EMCMF conversion.
   - They are not available for de-duplication.
   - They are supported only in archive mode.

2. **Plain external attachments**
   - These are large attachment objects contained within an email message, which are extracted out from the email message and stored external to EMCMF object as separate files during message to EMCMF conversion.
   - The main intent of storing these large attachments external to EMCMF object as separate files is to enable de-duplication.
   - These large attachments are stored in docbase either as renditions of respective email messages in archive mode or
Separate objects inside dm_attachments_folder in collaboration mode. By default, their object type is set to dm_document, if not specified by the caller while invoking import APIs of MessagingApp.

Please refer to the subsequent sections to find more details on the modes of operations supported by MessagingApp.

3. Embedded email message attachments
   - These are the email messages present as attachments within another email message.
   - They are stored internally in EMCMF object itself after message to EMCMF conversion.
   - They are stored in docbase either as
     - separate dm_message_archive objects in the same location as that of the parent message forming a VDM structure in archive mode or
     - separate dm_message_archive objects inside dm_attachments_folder in collaboration mode

Please refer to the subsequent sections to find more details on the modes of operations supported by MessagingApp.

Whenever MRE performs message to EMCMF conversion, it generates a package containing the EMCMF file along with its external attachment files (if any present inside the email message). In order to be able to regenerate original email messages from these EMCMF files at any later point of time, MRE requires exactly the same set of files containing the EMCMF file along with all those large external attachment files with exactly same file names as generated by it during message to EMCMF conversion. Hence, it is the responsibility of the caller to provide the accurate details about the EMCMF file along with all its external attachments generated by MRE to MessagingApp during inbound operations. Similarly, it is the responsibility of the caller to pass the details of EMCMF file along with all such external attachments exported by MessagingApp to MRE during outbound operations for successful regeneration of original email messages in respective native formats.

**Modes of operation supported by MessagingApp**

Based on the use case of the application, messages are stored with characteristics more favorable for archiving versus for collaboration. MessagingApp supports two modes of operation namely archive mode and collaboration mode, more details on which are provided below:

1. Archive Mode – In this mode, the email message along with all its attachments together are processed as a single entity. The lightweight sysobject based model is found to be more beneficial for archive mode use cases, which involve bulk ingestion of messages into docbase at a time for archival.
Therefore, archive mode would be a better option in those cases that involve bulk ingestion of messages into docbase.

When an email message is imported into docbase in archive mode,

- The email message in EMCMF format is stored as an object of dm_message_archive type following the lightweight sysobject model described earlier.
- Its external attachments are stored as renditions of the email message itself.
- If the email message contains embedded message attachments, then each embedded message attachment is stored as a separate dm_message_archive object in the same docbase location as that of its parent message. In such cases, a VDM structure is formed with the embedded message attachments as the VDM children and their parent message as the VDM root.
- The same strategy described above is again followed while storing the attachments associated with these embedded messages.

Figure 2 illustrates the manner in which an email message is stored in docbase when imported in archive mode:

![Figure 2: Storage of email messages in archive mode](image)

2. Collaboration mode – In this mode, the root email message (which is being imported) along with all its attachments are processed as separate entities/objects. This mode would be a better option than archive mode in those cases where the volume of message input is relatively lower and more dynamic user interaction is involved.
When an email message is imported into docbase in collaboration mode,

- The root email message in EMCMF format is stored as an object of dm_message_archive type following the lightweight sysobject model described earlier.

- All its attachments are stored inside an attachments folder of dm_attachments_folder type, which is located in the same docbase location as that of the associated email message. This folder is created only if the email message contains any external or embedded attachments. It is hidden by default and is linked to the associated email message through a dm_relation by name ‘dmc_attachments_relation’.

- All external attachments are stored inside the attachments folder. The callers of MessagingApp, who invoke its import APIs, can specify the object type of these external attachments during invocation. However, if not specified, then MessagingApp stores all of them as objects of dm_document type.

- All its embedded message attachments are stored as dm_message_archive objects inside the attachments folder.

- The same strategy described above is again followed to store the attachments associated with the embedded email messages.

Figure 3 illustrates the manner in which an email message is stored in docbase when it is imported in collaboration mode.
Operations on Email Messages

Operations supported by MessagingApp

MessagingApp supports the following set of operations on email messages:

1. Importing email messages in EMCMF format into docbase through email-aware documentum applications such as MDMO, Webtop, etc,
2. Viewing email messages,
3. Exporting email messages from docbase in EMCMF format,
4. Copying email messages from one location to another in docbase,
5. Linking email messages from one location to another in docbase,
6. Moving email messages from one location to another in docbase,
7. Deleting email messages from docbase,
8. Full-text indexing based search for email messages along with associated attachments.
Operations not supported by MessagingApp

MessagingApp does not support the following set of operations on email messages imported into docbase:

1. Checkout of email messages,
2. Cancel checkout of email messages,
3. Checkin of email messages,
4. Editing email messages.

Use case flow diagrams

This section provides a complete picture on how the control flows within a Documentum system when an email message is imported and exported through MessagingApp.

Figure 4 illustrates the flow of control during import of email messages through webtop by means of MessagingApp.

Figure 4: Use case flow diagram for importing email messages using Webtop

1. User initiates request to import an email message into docbase through Webtop UI.
2. Webtop import component initiates content upload operation through UCF component.
3. UCF Server sends appropriate instructions to UCF client.
4. UCF client invokes MRE to convert the email message from native message format to emcmf format.

5. MRE generates emcmf file from the original email message and returns it along with all its external attachments to UCF client.

6. The emcmf file along with all the external attachments are uploaded from end-user client machine to application server machine.

7. The results are then returned to Webtop import component.

8. Webtop import component then invokes appropriate MessagingApp APIs to import the emcmf along with all its attachments into docbase.

9. MessagingApp takes care of importing the email message into docbase. It internally makes DFC calls whenever found necessary while processing the caller’s request.

10. DFC communicates with content server and helps MessagingApp in saving the emcmf file along with all its attachments into docbase.

Figure 5 illustrates the flow of control during export of email messages through webtop by means of MessagingApp.

![Use case flow diagram for exporting email messages using Webtop](image)

Figure 5: Use case flow diagram for exporting email messages using Webtop

1. User initiates request to export email message from docbase through Webtop UI.

2. Webtop export component initiates content download operation and invokes appropriate MessagingApp APIs for the same.
3. MessagingApp uses DFC to download the emcmf file along with all its attachments from docbase to application server machine.

4. DFC internally communicates with content server and helps MessagingApp in downloading the emcmf file along with its attachments from docbase to application server machine.

5. MessagingApp returns the results back to its caller (i.e. webtop).

6. Webtop export component then initiates the process of content download from application server machine to the end-user client machine with the help of UCF component.

7. UCF server then sends appropriate instructions to UCF client. UCF client will then download the emcmf file along with all the available attachments from application server machine to the end-user client machine.

8. UCF client invokes MRE to generate the original email message with the help of the downloaded emcmf file and all its attachments. MRE generates the original email message and returns it back to the UCF client, which is then made available to the end user in a manner as requested by the user.

Packaging

MessagingApp is delivered in the form of a dar file called ‘Messagingapp.dar’. This dar file consists of a set of various BOF modules, which contain logic for handling different operations that are supported on email messages by MessagingApp.

Based on the version of DFC being used by the email-aware documentum application, appropriate set of BOF modules are dynamically downloaded to the application server machine at runtime whenever the client application tries to download MessagingApp modules for invoking supported operations.

Deployment

The Messagingapp.dar is packaged along with Content Server installer and gets installed during initial docbase configuration itself. In case if necessary, the Messagingapp.dar can also be explicitly installed on docbase at any later point of time using DARInstaller.exe just like any other dar file.

Configuration

MessagingApp is involved in the processing of email messages only when the user initiates any operations on email messages through an email-aware Documentum application such as MDMO. However, other Documentum applications such as Webtop require explicit configurations in order to become email-aware, so that they can then invoke MessagingApp for processing of email messages based on user's requests.
In case of webtop, the message archive support is not enabled by default even after installing the MessagingApp.dar on docbase; it requires explicit configuration in the application server machine where webtop is deployed for enabling the message archive support.

**Enabling message archive support in Webtop**

In order to enable the message archive support, the following configuration changes are necessary:

1. Enable the `<messageArchive-support>` in `<WebAppRoot>/wdk/app.xml` file. Figure 6 contains a snippet extracted from the app.xml file and shows how to enable message archive support in webtop.

```xml
<messageArchive-support>
  <enabled>true</enabled>
  ...
  <store-emf-object-as-archive>false</store-emf-object-as-archive>
  ...
</messageArchive-support>
```

**Figure 6: Enabling message archive support in Webtop**

The above snippet also demonstrates how to configure the mode of operation for MessagingApp. For enabling collaboration mode for MessagingApp, set `<store-emf-object-as-archive>` setting to false and for enabling archive mode, set `<store-emf-object-as-archive>` setting to true. By default, Webtop uses the Collaboration mode while importing email messages through MessagingApp.

2. Uncomment the executables section for MRE installer in `<WebAppRoot>/wdk/contentXfer/ucf.installer.config.xml` under Windows platform setting as illustrated in Figure 7:

```xml
<platform os="windows" arch="x64">
  <runtime type="jre" version="1.5.0" href="win-jre1.5.0_06.zip">
    \(\text{The ESI_MRE.exe is used to convert dm_message_archive emmc format files to .msg files.}\)
    \</runtime>
  <executables>
    <executable version="6.5.0"
      href="ESI_MRE.exe"
      installationoptions="/qn ADDLOCAL=ALL"
      uninstallOptions="/s /qn REBOOT=REALLYSUPPRESS"/>
  \</executables>
  ...
</platform>
```

**Figure 7: Configuration for enabling download of MRE installer**

With this configuration change, UCF client will download the MRE installer and then trigger its installation on the end-user client machine. Only after the successful installation of MRE on the end-user client machine, user can initiate any of the MessagingApp supported operations on email messages.
A Brief Insight into MessagingApp – Email Processing Component in Documentum Content Server

through Webtop. The role played by MRE in any of the inbound and outbound operations supported by MessagingApp has already been explained earlier.

**Dropping unique index for dm_message_archive**

By default, MessagingApp does not allow a user to import an email message for the second time if it already exists in docbase, even if user attempts to import it to a different location in docbase. This default behavior can be overcome by dropping unique index for dm_message_archive type.

Dropping of unique index for dm_message_archive type will only enable users to import an email message to a location different than the one where it already exists. The user will still be not able to import the same email message twice to the same location in docbase even after dropping unique index for dm_message_archive.

**Important customer cases**

One of the important customer reported cases is about the problems faced while performing outbound operations on email messages present in docbase, such as

- Failing to export some email messages from docbase to end user client machines through webtop,
- Failing to view some email messages through Outlook using ‘View As Outlook’ option in webtop,
- Failing to view the content of some email messages through MDMO,
- Problems with accessing one or more attachments of collaboration mode messages from attachment section of Message Viewer in webtop, etc.

On analyzing such customer reported issues, it has been found that these problems usually occur whenever MRE fails to generate original messages from their emcwf files, one of the most common reasons for which being the unavailability of accurate information about all the attachments associated with the email message including the attachments associated with all its embedded messages.

We have developed separate utilities, which ensure that the accurate information about all the email attachments present in the docbase is made available to MessagingApp during outbound operations on email messages.

**Troubleshooting checklist**

**General Checklist**

1. Provide the description of the problem that customer is facing.
2. What is the business impact of the problem? At a high level, describe the business functionality affected on account of the problem.
3. How frequently is the problem occurring?
4. Is the problem consistently reproducible? Can the problem be reproduced on demand?

5. What are the steps to reproduce the problem?

6. Is the problem being faced in a production or test environment?

7. Was the affected functionality ever working? If yes, then which was the last version on which it was found working and what changes were made to the environment since then?

8. Collect the information about the set of relevant products involved in reproduction of the issue, their versions, patches and hotfixes applied, etc.
   - Content Server
   - DFC
   - MessagingApp
   - Webtop
   - My Documentum for Microsoft Outlook (MDMO)
   - MRE, etc.

9. Obtain the complete exception stack trace for the problem.

10. Collect the following set of logs generated whenever the issue is reproduced:
    - log4j.logs collected at DEBUG level on application server machine
    - MRE logs collected from the end-user client machine

11. Check if the DFC BOF cache located on application server machine contains correct messagingApp jar files. Setting ‘dfc.bof.cache.append_name’ to true in dfc.properties beforehand allows easier identification of the relevant jar files.

12. Collect the message-specific information, such as
    - The mode in which the message was imported into docbase – archive or collaboration,
    - The type and number of attachments present in the email message,
    - If possible, provide the source message and/or EMCMF file along with its attachments.
    - In case of webtop, please ensure that the message archive support is properly enabled on application server.
    - Make sure that the MRE installer packaged with the particular application is installed on the end-user client machine, from where user tries to perform inbound and outbound operations on email messages.
Checklist for issues specific to inbound operations

1. By default, MessagingApp does not allow a user to import an email message for the second time if it already exists in docbase, even if user attempts to import it to a different location in docbase. In order overcome this default behavior and enable users to import an email message to a location different than the one where it already exists, it is necessary to drop unique index for dm_message_archive type. Even after dropping unique index for dm_messagearchive type, user will still not be able to import the same email message twice to the same location in docbase.

2. For other problems faced during import of email messages into docbase, you may want to collect the following information:
   - If possible, collect the same set of email messages which are experiencing problems during their import.
   - Also collect other details about the email messages such as
     - Formatting details of email message (i.e. HTML or Plain Text or Rich Text formatting).
     - Does it have any attachments? If yes,
       - What is the total number of attachments that it has?
       - Are there any attachments whose names contain special characters that are not supported by the operating systems running on end-user client machines or application server machines?
       - Are there multiple attachments with same name? If yes, how many such sets of attachments with same names exist?
       - Provide details of each attachment present within the email message:
         - Attachment name with extension
         - Is it an embedded email attachment or external document attachment?
     - Collect the above mentioned set of information for each embedded message attachment present inside the email message.

3. In addition to above information, also collect all the information requested under General Checklist section.

Checklist for issues specific to outbound operations

1. Execute the following query and provide the results:

```sql
select * from dm_message_attachment where message_object_id='<r_object_id of affected message>'
```
2. Provide the IAPI dump of the affected email message.

3. Try to download the EMCMF file along with all its attachments present in docbase using the 'getfile' IAPI command and provide them to engineering.

4. Also collect other details about the email messages such as
   - Does it have attachments? If yes,
     - What is the total number of attachments that it has?
     - Are there any attachments whose names contain special characters that are not supported by the operating systems running on end-user client machines or application server machines?
     - Are there multiple attachments with same name? If yes, how many such sets of attachments with same names exist?
     - Provide details of each attachment present within the email message:
       - Attachment name with extension
       - Is it an embedded email attachment or external document attachment?
   - Collect the above mentioned set of information for each embedded message attachment present inside the email message.

5. In addition to above information, also collect all the information requested under General Checklist section.

**Checklist for issues specific to copy, link and move operations**

1. If possible, collect the set of email messages which are experiencing problems and provide them to engineering for analysis.

2. Also collect other details about the email messages such as
   - Does it have attachments? If yes,
     - What is the total number of attachments that it has?
     - Are there multiple attachments with same name? If yes, how many such sets of attachments with same names exist?
     - Provide details of each attachment present within the email message:
       - Attachment name with extension
       - Is it an embedded email attachment or external document attachment?
   - Collect the above mentioned set of information for each embedded message attachment present inside the email message.
3. In addition to above requested information, also collect all the information requested under General Checklist section.

Conclusion

This white paper provides an overview about the fundamental aspects of email processing in Content Server, which is achieved with the help of MessagingApp. This document helps the readers to understand the flow of control during import and export of email messages into Documentum repository through an email-aware Documentum application. This document also highlights some of the important customer reported issues and provides checklists for troubleshooting MessagingApp related issues.

References

- EMC® Documentum® Foundation Classes Version 7.0 Development Guide
- EMC® Documentum® Content Server Version 7.0 Fundamentals Guide
- EMC® Documentum® Content Server Version 7.0 System Object Reference
- EMC® Documentum® High-Volume Server Version 7.0 Development Guide
- EMC® Documentum® Web Development Kit Version 6.7 Development Guide