OTN OpenMail Technical Note

B.06.00 Changes to RTF Support

Summary

In this OTN, we look at the new extended support for messages containing RTF and, in particular, multibyte RTF. This affects several areas of OpenMail:

- Support for cc:Mail clients.
- Searching multibyte RTF items.
- Support for RTF messages at the Request Server.
- Filetyping RTF attachments and documents generated by Microsoft applications.

The OTN describes how these areas have changed.

This OTN will form part of the OM360 (B.06.00 upgrade) course.

Readership

Readers are assumed to be OpenMail support engineers, who have attended an advanced OpenMail course (AE351 or H1805s) or are familiar with the material covered in these courses.

I have also assumed that readers are familiar with basic file format and character set conversion mechanisms at release B.05.10. See the Multibyte OTN.

Revision History

August 1999: First issue

Comments Please!

I would welcome any comments you may have on this document. Please email them to joyce@pwd.hp.com.

Contents

Introduction

This OTN covers a variety of changes at B.06.00 mostly connected with OpenMail's support of RTF messages and attachments. The most significant improvement is the new inline RTF handling library, which provides fast RTF to text file format conversion, including reliable multibyte conversion.

As parts of the other changes described in the OTN result from this new library, we will start by looking at the new file format and character set conversion mechanisms and how you configure these.

Within the RTF handling library is a new converter to enable cc:Mail users to see text that has been highlighted in RTF messages they receive. So we will look next at how you configure this and other new features for supporting cc:Mail clients.

MAPI clients, such as Outlook, and the Web client allow you to perform searches on message items. In previous releases, because of the limitations in rendering RTF, this facility was only available for messages in ISO 8859_1 character set. The new inline RTF library provides routines for performing such searches on messages in other character sets, including multibyte character sets. We will take a brief look at this feature.

In the next section we will look at how the RTF handling library enable RTF messages to be sent to the Request Server. Until now, this server could only handle text messages. Also covered in this section is the new Request Server feature of multiple steering files, which can be tailored for specific request scripts.

The final section covers the improvements to filetyping RTF attachments and attachments generated by Microsoft applications.

RTF File Format And Character Set Conversion

At B.05.10, the process, `rtf.browse`, was used at gateways and client interfaces to convert RTF message bodyparts to plain text for clients and destinations which could not handle RTF. A major limitation was the browser's inability to convert accurately RTF in character sets other than ISO 8859_1. This meant that Asia Pacific users with clients such as OpenMail/Outlook could not reliably send RTF messages to RTF unaware clients.

In B.06.00, the conversion to text of RTF message bodyparts is handled by an inline RTF handling library. This library contains:

- New inline file format converters for converting RTF to text. Included are new routines to provide accurate multibyte RTF to text conversion.
- New inline file format converters for converting from RTF to cc:Mail Color Highlighted Text (CHT).
- Routines for performing word searches in RTF items.

The file format converters work together with the existing inline character set converters and conversion steering mechanisms. New inline routines allow you to configure RTF file format conversion and display character set conversion, using two steering file lines.

Note: For character set converters and file format converters, an inline converter, if it exists, will be used in preference to a configured process converter, unless the `general.cfg` option `CONV_USE_FILE_PROCESS` is set.

RTF File Format Conversion

The following inline file format converters are shipped with B.06.00:

```
RTF to Text: 2130 -> 1167
RTF to CHT: 2130 -> 2139              i.e. cc:Mail color highlighted text
```

An important function of the RTF to text converters is to find out the language and character set of the message, as this information is embedded in the message.

Note: The RTF handling library can only support multibyte properly if the RTF includes language or character set information. One example, that does not do this, is RTF generated by AMIPRO 3.1J in Japan. Therefore, the RTF library cannot be guaranteed to process this RTF correctly.

In previous releases, only character set converters were inline. The purpose of making converters inline is to speed up the conversion process. It also means that you don't need to call the converter explicitly as a process, so you will no longer see a reference to the `rtf.browse` process in the file `~openmail/sys/convert`.

You will still see an `rtf.browse` binary in `/opt/openmail/bin`. This just calls the RTF handling library. It is included for test purposes and to support any customer use of the binary other than through the OpenMail conversion mechanism.

Note: If you call the `rtf.browse` binary, the character set conversion is output on descriptor 3, so the command line should look something like this:

`rtf.browse < input_file > output_file 3> char_set_file`

In previous releases, there was a problem with browsing multibyte RTF messages, which resulted in characters being split between lines. Now the RTF handling library includes multibyte aware routines, which provide the logic needed when converting multibyte RTF. The file format converter attempts to format the following, ensuring, of course, that multibyte characters are not split across lines, of course:

- Lists
- Alignment and justification controls

- Strike through, by showing the multibyte character followed by a hyphen

But note these limitations:

- An assumption is made that 2 byte characters, when displayed, are twice the width of single byte characters. If the converted output is read on an interface where this is not the case, the formatting of lists and alignment will be incorrect.
- Spaces do not normally occur in most of the multibyte languages supported. Consequently, words may be broken across lines. This applies also to English words, if they occur within the multibyte text and without delimiting spaces.

Character Set Conversion

Inline character set converters are not new to B.06.00. The difference in this release is the extent to which character set conversion can be performed inline and in conjunction with inline RTF file format conversion.

For example, at the outgoing Internet Gateway, you might want RTF messages in the Japanese SJIS display character set, converted to plain text in UNIXJIS character set, which is widely used on the Internet as a transport character set. This involves file format conversion (from RTF to text) and character set conversion (from SJIS to UNIXJIS). All this can be driven by two steering file lines.

We will look at steering files in more detail in the next section. In this section, we are only looking at one of the lines needed - the one which drives the character set conversion.

The ability to convert between display character sets is provided by an extension to the existing inline converter mechanism, to perform a two stage character set conversion using the interchange character set (OMJIS in this case). This mechanism is only available where both parts of the conversion can be handled by inline file converters.

In a steering file, you can now specify:

```
2130.SJIS     1167.UNIXJIS R
```

and all the necessary intermediate conversions happen automatically, making the process much faster. The example steering line above would cause the following conversions to take place automatically (after the RTF file format converter has converted the RTF to SJIS text):

1167.SJIS -> 1736.OMJIS

1736.OMJIS to 1167.UNIXJIS

Just for information, here is a list of the inline character set converters shipped with OpenMail at B.06.00:

Japanese character set conversion routines:

```
1167.SJIS to 1736.OMJIS
1736.OMJIS to 1167.SJIS
1167.UJIS to 1736.OMJIS
1736.OMJIS to 1167.UJIS
1167.UNIXJIS to 1736.OMJIS
1736.OMJIS to 1167.UNIXJIS
1167.LMBCS_J to 1736.OMJIS
1736.OMJIS to 1167.LMBCS_J
1167.SJIS to 1736.OMJIS for Desk G/W
```

Korean character set conversion routines:

```
1736.OMKOREAN -> 1167.KSC5601
1167.KSC5601 -> 1736.OMKOREAN
1736.OMKOREAN -> 1167.ISO2022KR
1167.ISO2022KR -> 1736.OMKOREAN
1736.OMKOREAN -> 1167.LMBCS_K
1167.LMBCS_K -> 1736.OMKOREAN
```

Chinese character set conversion routines:

```
simpl-Ch: 1736.OMCHINESE -> 1167.HZGB2312
simpl-Ch: 1167.HZGB2312 -> 1736.OMCHINESE
simpl-Ch: 1736.OMCHINESE -> 1167.GB2312
simpl-Ch: 1167.GB2312 -> 1736.OMCHINESE
simpl-Ch: 1736.OMCHINESE -> 1167.ISO2022CN
simpl-Ch: 1167.ISO2022CN -> 1736.OMCHINESE
simpl-Ch: 1167.LMBCS_SCH -> 1736.OMCHINESE
simpl-Ch: 1736.OMCHINESE -> 1167.LMBCS_SCH
```

Steering Conversion

File format conversion and character set conversion at the gateways are controlled using the existing steering mechanism, that is, the files `~openmail/sys/mimeout.str`, `unixout.str`, `brwmime.str` etc. (`brwmime.str` is used for POP3/IMAP4, `unixout.str` and `mimeout.str` are used for the outgoing Internet gateway).

When an RTF message arrives at the gateway, the language and character set of the message is not known. This information is embedded in the RTF and is determined when the message is *browsed* by the RTF file format converter. To force this converter to process the message, you need to add one of the following lines to the appropriate steering file:

```
2130 1167 R
```

or

```
* 1167.ISO8859_1 R
```

It doesn't matter which of these lines you use, the effect is the same:

- The RTF converter is invoked to perform RTF to text file format conversion.

At the end of this file format conversion, the character set information picked up during this conversion is checked against the character set, if any, specified in the steering line. For example, in the second line above:

```
* 1167.ISO8859_1 R
```

the RTF to text file format conversion would be invoked by the "`* 1167`" part of the line. During this conversion, the converter discovers the language and character set of the message. When the file format conversion is complete, the character set of the message is compared with the "`ISO8859_1`" part of the steering line.

If they match, the required conversion is complete. If they differ, a second pass of the steering file is made to find the matching steering line for the *`filetype.char_set`* of the message. If no character set is specified, as in the line:

```
2130 1167 R
```

that is taken as "not matching" and a second pass of the steering file is made.

This is when a steering line, such as:

```
2130.SJIS    1167.UNIXJIS R
```

comes into effect, to drive subsequent character set conversion, as described in the previous section.

Some Examples

Just to summarize, let's work through the example, as a whole:

> At the outgoing Internet Gateway, we want to be able to convert RTF messages in the SJIS character set to plain text in the UNIXJIS character set.

In the steering file, `unixout.str`, we have added the lines:

```
2130.SJIS    1167.UNIXJIS R

* 1167.ISO8859_1 R
```

The diagram below just summarizes the conversion steps involved in this example.

- An RTF message in SJIS arrives at the Internet Gateway.
- The gateway reads the `unixout.str` steering file to find out what conversions are available.
- The character set of the RTF is not known at this point, so the first line to be executed is

  ```
  * 1167.ISO8859_1 R
  ```

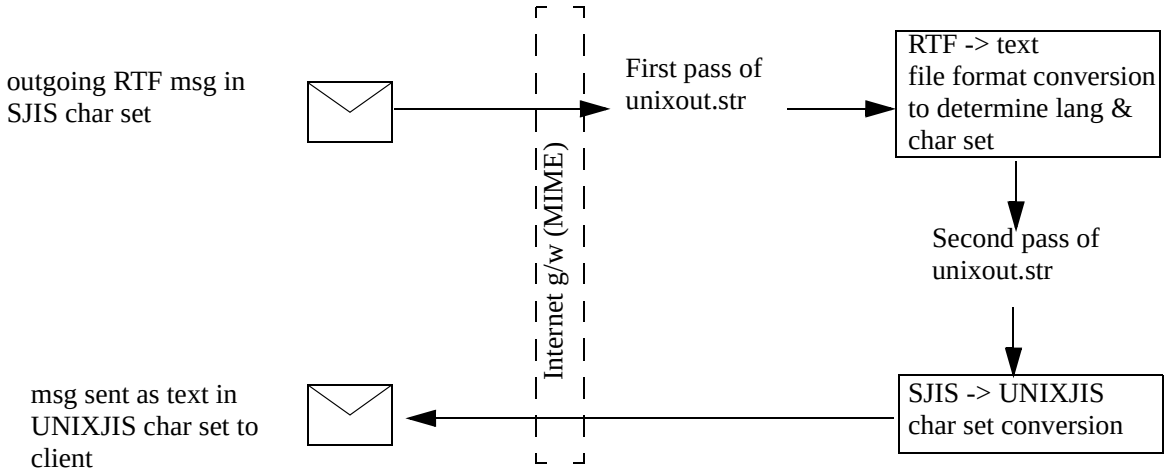
  which, first of all, invokes to RTF converter to convert to text (the * 1167 bit), without any character set conversion (i.e. it doesn't look at the .ISO8859_1 bit yet).

  During the file format conversion, the converter learns that the message is in SJIS.
- The character set of the message (SJIS) is compared with the character set specified on this steering line (ISO8859_1). It doesn't match, so a second pass is made of the steering file.
- We know now that the message is RTF (filetype 2130) in SJIS, which matches the input part of the steering line:

  ```
  2130.SJIS    1167.UNIXJIS R
  ```
- The inline character converters:

  ```
  1167.SJIS to 1736.OMJIS
  1736.OMJIS to 1167.UNIXJIS
  ```

  are invoked one after the other, to perform character set conversion on the text file output from the initial RTF to text conversion.
- The message in plain text UNIXJIS is sent to the client.



Let’s look briefly at another example:

> We want Japanese RTF messages going out through the MIME Internet Gateway to be converted to text but to remain in the SJIS character set.

For this, we would add the following lines to the steering file, `mimeout.str`:

```
2130.SJIS  1167.SJIS       R
*          1167.ISO8859_1  R
```

When an RTF SJIS message arrives at the gateway, the second line in your steering file is processed first. This line invokes the RTF converter to perform RTF to text file format conversion and to discover the language and character set information, which is embedded in the RTF.

When the character set of the message (SJIS) is then compared with the character set specified in this steering line (.ISO8859_1), the mismatch causes the steering file to be reread for a line with the input matching that of the message (2130.SJIS).

This second pass of the steering file invokes the line:

```
2130.SJIS  1167.SJIS       R
```

The "intermediate" text file, created by the initial file format conversion, is already in plain text SJIS, so no further conversion is done.

RTF Messages And cc:Mail Clients

In cc:Mail, the main text of the message is limited to plain text or color highlighted text. Therefore, RTF messages sent to cc:Mail clients need to be converted to text before they can be read by cc:Mail users.

In previous releases of OpenMail there were two main problems with the support of messages to cc:Mail clients:

- Color highlighted text was not supported by OpenMail.
- RTF messages sent to cc:Mail clients were converted to plain text but appeared as an attachment instead of the main message text. This meant that the user had to make further mouse clicks to open the RTF item, and could not make an annotated reply to the message.

At B.06.00, these limitations have been removed for cc:Mail R6.*x* clients. To support color highlighted text, the new RTF handling library includes an inline RTF to cc:Mail color highlighted text converter.

Enabling Support For Color Highlighted Text

To enable RTF, sent by an RTF client such as Outlook, to be displayed as cc:Mail color highlighted text in messages you receive, you need to set the option, `RTFConvertIfColoured`, in the `hpmail.ini` file on your PC:

```
RTFConvertIfColoured=1
```

If this option is set, the new RTF to CHT: 2130 -> 2139 inline file format converter will be invoked for RTF messages sent to this cc:Mail client.

To enable cc:Mail color highlighted text to be viewed as RTF by RTF aware recipients, you need to set the `hpmail.ini` option:

```
ConvertTextToRTF=1
```

This option invokes the `ccmht2rtf` process on the OpenMail server to convert the cc:Mail color highlighted text (filetype 2139) into a special, cut down version of RTF (filetype 1566), which can be read by full RTF clients, such as Outlook.

This file converter is not inline and therefore has the following entries in the `~openmail/sys/convert.cs` file:

```
2139 1566 /opt/openmail/bin/ccmht2rtf 2139 1566
1566 2139 /opt/openmail/bin/ccmht2rtf 1566 2139
```

Enabling Display And Editing Of RTF Messages

To ensure RTF messages are correctly displayed in cc:Mail and can be edited, follow the instructions in the following section of the `hpmail.ini` file on your PC and edit it to set `EditConvertedRTF=1`:

```
;  ------ EditConvertedRTF -----
; Use this entry to allow editing of converted RTF files within cc:Mail.
;
; If an RTF message (e.g. from Outlook) has been converted to plain text
; by including the appropriate conversion in the [Conversions] section
; of HPMAIL.INI ...
;    2130=1167 "Microsoft RTF"
; ... then enabling the EditConvertedRTF option causes it to be
; displayed as editable message text.
;
; 1/ON=Display converted RTF item as editable text.
; 0/OFF=(default) Display converted RTF as a text attachment
;
EditConvertedRTF=0
MailServerName=1
```

Searching RTF Message Content

With MAPI and the Web Client you can searching for words in RTF messages. Until B.06.00, this was limited to messages in the ISO 8859_1 character set. Now you can search RTF messages in other European and Asia Pacific character sets.

To perform such searches, quite a lot needs to happen in the background:

- The RTF characters must be decoded, and their character set determined, before making string comparisons.
- The supplied search pattern may be in a different character set to the RTF message, so the search pattern needs to be converted to the same character set as the message, i.e. comparisons are done in display character sets, rather than interchange character sets.

As we have seen, the necessary routines now exist in the RTF handling library to determine the language and character set of RTF and to convert between display character sets. The library also includes routines to perform the string matching.

Using an inline OpenMail library, as opposed to an invoked a process, ensures that searching RTF messages does not have an excessive effect on performance.

Request Server Support For RTF Messages

There are two enhancements to the Request Server at B.06.00:

- You can now configure steering files for particular scripts.
- Using the improved RTF to text conversion facilities, messages sent to the Request Server in RTF can now be supported.

This section explains these features.

Configuring Script Specific Steering Files

A new feature of the Request Server at B.06.00, is that you can configure additional steering files for particular request scripts. For example, let's assume I have three request scripts in `/opt/openmail/req`:

```
audit
info
admin
```

The input passed to the scripts, `audit` and `admin`, is controlled by the default steering file, `~openmail/sys/requestout.str`. However, I want additional processing of input to the `info` script, for example, RTF to text with character set conversion. To do this, I would set up all the processing required for this script in another steering file, `/opt/openmail/req/infoout.str` (*`<request_script_name>`*`out.str`). This steering file is associated with the `info` script only and will be used, instead of `requestout.str`, to process the input to the script.

Script specific steering files must be in `/opt/openmail/req`, which also holds the request scripts, and have a filename in the form:

`<request_script_name>``out.str`

Before invoking a script, the Request Server looks for and acts on steering files in the following order:

`/opt/openmail/req/`*`<request_script_name>`*`out.str`

`~openmail/sys/requestout.str`

The `requestout.str` file will be configured to provide the default B.05.10 action:

```
1734        T
*           R
```

which means, trash (T) any `WinMail.DAT` file (filetype 1734) and retain (R) everything else (*) without performing any conversion. You could add, in a similar way, any other content types which can be trashed.

Note: This makes the `general.cfg` option, `REQ_IGNORE_TYPES_OF_ATT=`*`value`*, redundant. It is retained in B.06.00, but we recommend you only use the steering file to configure content parts to be trashed, as the value set in the `general.cfg` option may conflict with the steering configuration.

Supporting RTF Messages Sent To The Request Server

In previous releases, all messages to the Request Server had to be in plain text. Now you can use the improved RTF to text conversion facilities, which we have discussed earlier in this OTN, in Request Server steering files to support request messages in RTF.

Steering files that process message content can include a line to ensure RTF content parts are converted to text and the character set converted to that of the request script. For example, if your request script requires plain text and the ISO8859_1 character set, you could add the following line to the start of the steering file:

```
2130    1167.ISO8859_1  R
```

Which means that all message content parts with a filetype of 2130 (RTF) will be converted to plain text with the ISO8859_1 character set. The original source will be retained (R). The processed message content is then passed to the script as in previous releases, that is, working from the end of the content records backwards, the first content part that is not discarded will be sent to the script.

Summary

To summarize the action taken when a message is sent to the Request Server:

1. While reading the message transaction file, save content file information about all the content parts that are not defined to be trashed by the `general.cfg` option, `REQ_IGNORE_TYPES_OF_ATT=`*`value`*.
2. Once the recipient is found, and therefore the request name is known, set up the steering information; if there is a `/opt/openmail/req/<`*`requestname`*`>out.str` file, use this, otherwise use the default instructions as given in `~openmail/sys/requestout.str`.
3. Work from the end of the content records backwards, and use the first one that is not configured to be trashed. Convert it if necessary and pass the input to the target script.

Filetyping Of RTF And Microsoft Documents

In this final section, we will look at the improvements in the way the file type of RTF attachments and Microsoft application files are determined at gateways and the Service Router (file coercion).

At The Internet Gateway and IMAP Interface

Documents mailed from Microsoft Word always have a content type of `application/msword` when they arrive at the Internet gateway or IMAP interface. This content type is automatically (and quite reasonably) mapped to the filecode for `.doc` files, which is fine when the document really has a file extension of `.doc`, but very irritating for Unix users if the document is, in fact, a `.rtf` document! They will not be able to open the attachment, even though OpenMail includes an RTF converter to enable Unix users to read RTF attachments in text.

Similarly, RTF documents created using other applications may have a misleading content type, `application/octet-stream`. When this arrives at the Internet gateway or IMAP interface, it is mapped to the binary filecode, which again renders the attachment unreadable.

At B.06.00, this problem has been solved by adding a new line to the beginning of the file, `~openmail/sys/mime.types`:

```
2130 application/*    rtf
```

By wildcarding the subtype part of the content type and giving the explicit filename extension, `.rtf`, any attachment with `application/` as the content type and a file extension of `.rtf` will be assigned the OpenMail RTF filetype, 2130, and therefore readable by the RTF converter.

The mapping of all other file types is not affected.

At The Service Router (File Coercion)

Just to recap on file coercion; the file type associated with a bodypart can be changed by the Service Router. For example, if a user or gateway identifies a body part as being of binary format when it is really a specific word processor file, the file code associated with that body part can be changed to reflect its true file type. The file `/var/opt/openmail/sys/map.types` specifies the file type coercions to be performed. The file `/var/opt/openmail/nls/<`*`language`*`>/filetype` lists available file types and their associated file codes.

When the Service Router processes a message, it examines the file types associated with the message body parts (except the active distribution list). If the content of a message body part and its file code matches a line in the `map.types` file, the file code of the body part is changed.

The file format of Microsoft Office documents changed at Office 97 to *OLE2* file format, which does not have a simple identifier. This meant that OLE2 documents could not be understood by the B.05.10 file coercion library.

In order to determine the type of an Office 97 file, OpenMail needs to understand the file structure, and read pointers within it to find where to look. The file coercion library has therefore been extended and new lines added to the `map.types` file to identify Word, Excel and PowerPoint documents:

```
0 2150 0xd0 0xcf 0x11 0xe0 @OLE2 "WordDocument"
0 2147 0xd0 0xcf 0x11 0xe0 @OLE2 "Workbook"
0 2148 0xd0 0xcf 0x11 0xe0 @OLE2 "PowerPoint Document"
```