Method for Requesting and Viewing a Preview of a Table Attachment on a Mobile Communication Device

[0001] The following is directed in general to displaying content on mobile communication devices, and more particularly to a method for previewing a table attachment on a mobile communication device.

[0002] Mobile communication devices are becoming increasingly popular for business and personal use due to a relatively recent increase in number of services and features that the devices and mobile infrastructures support. Handheld mobile communication devices, sometimes referred to as mobile stations, are essentially portable computers having wireless capability, and come in various forms. These comprise Personal Digital Assistants (PDAs), cellular phones and smart phones. While their reduced size is an advantage to portability, limitations in download bandwidth and device screen size give rise to challenges in viewing large documents with multiple embedded tables, having a large number of columns and rows for presenting information in a tabular format.

[0003] For wireless devices that support viewing of attachments, this represents a large amount of information for downloading, requiring a large amount of bandwidth and the associated cost thereof. Additionally, the user must wait for an extended period of time while the device is downloading the complete table.

[0004] Since mobile communication devices have limited screen real estate, it is known in the art to provide an Attachment Server (AS) for delivering "on-demand" content to wireless device users in order to minimize bandwidth, and device CPU/memory usage. When the user of a wireless device views a document attachment that uses tables, the Attachment Server typically downloads the document text to the Attachment Viewer (AV) on the mobile device along with a selectable link (e.g. [Table: N]), rather than the entire table contents. The user can then retrieve the entire table, if desired, by selecting the link. By downloading only the text and a link to the table, and providing the user with an option for retrieving or viewing the entire table attachment, such existing architectures thereby adhere to the on-demand design principle set forth above.

[0005] Unfortunately, the selectable link (e.g. [Table: N]) provides no information relating to the actual content of the table. Some documents use tables extensively to present information. Indeed, in some cases the entire document is created using only tables containing the entire textual content. These types of table-centric documents, when viewed on the wireless device through the foregoing on-demand architecture, are

represented only by a large number of selectable table links, without any other textual information. For these types of documents the user is unable to discern which of the table(s) contain information that is relevant to them and thus should be retrieved. The user is therefore required to retrieve some, if not all, of the tables in the document in order to identify any table(s) of interest. Consequently, the design principle of offering on-demand retrieval, wherever possible, is defeated by such multiple downloads of irrelevant attachment parts.

Summary

[0006] According to one aspect, a method is provided for previewing a table attachment on a mobile communication device, and upon identifying table content of interest viewing and/or retrieving the entire table.

[0007] In one embodiment, a small 4x4 row/column preview is presented on the device of every table found in a document attachment, with the cell content textual information limited to the requesting device display capabilities (e.g. typically 8-32 characters per cell). This limitation avoids any requirement to download large amounts of individual cell content to the device, which otherwise would defeat the on-demand architectural design principle.

[0008] Thus, the embodiments set forth herein allow the user to quickly identify which table(s) is (are) of interest and then request full table download from the Attachment Server in an on-demand fashion. In this regard, the preview table itself preferably also functions as a link to the full table stored at the server to facilitate easy retrieval to the device for viewing in a spreadsheet view.

[0009] Additional aspects and advantages will be apparent to a person of ordinary skill in the art, residing in the details of construction and operation as more fully hereinafter described and claimed, reference being had to the accompanying drawings.

Brief Description of the Drawings

[0010] A detailed description of the preferred embodiment is set forth in detail below, with reference to the following drawings, in which:

[0011] Figure 1 is a block diagram of a network environment in which the preferred embodiment may be practiced;

[0012] Figure 2 is a device screen display of a plurality of table attachments, according to the prior art;

[0013] Figure 3 is a device screen display showing preview of a plurality of table attachments as in Figure 2, according to an embodiment;

[0014] Figure 4A and 4B are device screen displays showing menu command options for retrieving and viewing, respectively, the previewed table attachment of Figure 3;

[0015] Figure 5 is a device screen display showing the retrieved table of Figures 4A and 4B;

[0016] Figure 6 is a tree diagram showing the basic structure of a Document Object Model (DOM);

[0017] Figure 7 shows the top-level of the DOM structure in Figure 6;

[0018] Figure 8 shows an exemplary DOM structure for a word processing document;

[0019] Figure 9 shows an exemplary DOM structure for a table document;

[0020] Figure 10 shows an exemplary DOM structure for a word processing document containing an image subdocument;

[0021] Figure 11 is a flowchart showing a client-side process for previewing a table attachment using an attachment viewed on a mobile communication device;

[0022] Figure 12 is a DOM structure graph presentation of a sample document containing a referenced table component added to the DOM structure as an attribute of an external component reference (ECR) command;

[0023] Figure 13 is a flowchart showing a server-side process for constructing a table component as a preview for the referenced table component of Figure 12, and adding the preview table component as an attribute to the ECR command for previewing a table attachment;

[0024] Figure 14 is a DOM structure graph presentation of the document in Figure 12 following the process of Figure 13; and

[0025] Figure 15 is a flowchart showing a server-side process for persisting the ECR command and the attached preview table component with its formatting properties to construct the preview table.

Details

[0026] With reference to Figure 1, network environment 10 is shown in which the preferred embodiment may be practiced. Network environment 10 comprises at least one mobile communication device 12 communicating via a wireless network/Internet 14 to a server 28, via a firewall security server 29, for downloading document attachments to the devices 12. While only one server 28 is shown for illustration purposes, a person of skill in the art will understand that network environment 10 could have many such servers for hosting web sites or graphic download sites, providing access to picture files such as JPEG, TIFF, BMP, PNG, SGI, MP4, MOV, GIF, SVG, etc. As would be understood by one of ordinary skill in the art, wireless networks 14 comprise GSM/GPRS, CDPD, TDMA, iDEN Mobitex, DataTAC networks, or future networks such as EDGE or UMTS, and broadband networks like Bluetooth and variants of 802.11.

[0027] As discussed above, for documents that use embedded tables, or for documents that are constructed using only tables the existing "on-demand" architecture displays a series of embedded links to the tables, as shown in Figure 2. This forces the user to retrieve each and every table in order to view the table content, which is a time/bandwidth consuming operation if the user only needs information from one or a few of the tables.

[0028] The table preview feature set forth herein, presents the user with a small preview of the table content on a mobile device 12 in "Document View", as show in Figure 3. Specifically, the table preview feature displays the first 4 columns/rows of an embedded table in order to quickly give the user an idea of what the table content is. In order to view the full table, the "Table Link" header in the preview acts as a selectable link and "Retrieve" is available from the menu when clicking the track wheel of device 12, as shown in Figure 4A. The full table can then be viewed in "Spreadsheet View" using the "View" command, as shown in Figure 4B.

[0029] Finally, the full table in the document can be viewed in Spreadsheet View after it has been successfully been retrieved from the Attachment Service, as shown in Figure 5.

[0030] Before turning to the specifics of the client-side and server-side applications for the table preview feature, a description is provided herein of the structure of a Document Object Model (DOM) for a document attachment to be viewed on the mobile communication device 12.

[0031] The attachment server 28 uses a file-parsing distiller in the preferred

embodiment, for a specific document type, to build an in-memory Document Object Model (DOM) structure representing an attachment of that document type. The document DOM structure is stored in a memory cache of server 28, and can be iterated bi-directionally.

[0032] As shown in Figure 6, the graph-based document DOM structure consists of nodes and leaves. The nodes serve as the parents of leaves and nodes, while leaves are end points of a branch in the graph. Each node and leaf can have a set of attributes to specify its own characteristics. For example, a paragraph node can contain attributes to specify its alignment, style, entry of document TOC, etc. In addition, each of the nodes and the leaves has a unique identifier, called a DOM ID, to identify itself in the document DOM structure.

[0033] The document DOM structure is divided into three parts: top-level, component and references. The top level refers to the document root structure, while the main document is constructed in the component and the references represent document references to either internal or external sub-document parts. The following paragraphs examine each part in detail.

[0034] The root node of a document DOM structure, referred to as "Document", contains several children nodes, referred to as "Contents", which represent different aspects of the document contents. Each "Contents" node contains one or multiple "Container" nodes used to store various document global attributes. The children of the "Container" nodes are components, which store the document structural and navigational information. When the attachment server 28 builds the DOM structure for an attachment file for the first time, the top-level structure is a single parent-child chain as shown in Figure 7:

[0035] Three types of components are defined by the attachment server 28: text components, table components and image components, which represent text, tables and images in a document, respectively. The text and table components are described in detail below, and the image component structure is identical.

[0036] A component consists of a hierarchy of command nodes. Each command represents a physical entity, a property, or a reference defined in a document. For the text component, the physical entity commands are page, section, paragraph, text segments, comments, footnote and endnote commands, which by name define the corresponding entity contained in a document. The property commands for the text component are font, text color, text background color, hyperlink start/end and bookmark commands. The text component has only one reference command, referred to as the text reference command,

which is used to reference a subdocument defined in the main body of a document. Usually, the children of a text component are page or section command nodes that, in turn, comprise a set of paragraph command nodes. The paragraph command can contain one or multiple nodes for the remaining command types.

[0037] Using the following sample text document, the corresponding document DOM structure is shown in Figure 8:

First paragraph. Second paragraph with **bold** and red text.

[0038] As Figure 8 demonstrates, the section command, which is the child of the text component, consists of two paragraph commands. The first paragraph command contains one text segment command and the text content for that paragraph is added as an attribute to the text segment command. The second paragraph command has a relatively more complex structure, as the text properties in the paragraph are much richer. Each time a text property (font, text color, etc) changes, a corresponding text property command is created and the change value is added to that command as an attribute. The subsequent text segment command records the text with the same text property as an attribute. As document structure gets richer and more complex, more commands of corresponding types are created and the document properties are added as attributes to those commands.

[0039] The table component has the same three types of commands as the text component, but different command names. The document DOM structure for the sample table document below is shown in Figure 9:

Cell One	Cell Two
Cell Three	Cell Four

[0040] As shown in the Figure 9, the table component has physical entity type commands of table, tablerow and tablecell, where the tablecell command can contain all available commands for the text component. In the example above, the first child TableRow command of the table command has an attribute "Index" defined by value of 0. This indicates that the indicated table row is the first one defined in the table. The attribute of the leftmost table cell command in Figure 9 has the same meaning.

[0041] A document sometimes contains subdocuments, for example images, tables, text boxes etc. The DOM structure set forth herein uses a reference command to point to the graph of such subdocuments. Thus, for the following sample document, the

attachment server 28 generates the DOM structure shown in Figure 10:

This document has subdocument of images like this one .

Second paragraph contains the same image .

The structure shown in Figure 10 is identical to that discussed above in connection with Figures 8 and 9, except for the attributes of the two reference commands. The attachment server 28 constructs the image in "Sample Three" as a separate image component, which contains all of the image data in its own DOM hierarchy. In the DOM structure for the main document, the values of the "Ref" attributes of those two reference commands point to the image component, as indicated by the dashed lines, such that the DOM structure connects together all parts of the document.

[0042] Returning now to the preferred embodiment, the client component (Attachment Viewer (AV) application on device 12), when requesting a document attachment conversion from the server component (Attachment Server (AS) 28), informs the server of its device and feature specific capabilities such as screen size, color capabilities and functionality support when requesting an operation. This allows the server 28 to tailor its response to the requesting device 12 and optimize bandwidth utilization and minimize device CPU/storage usage while preserving the device presentation capabilities (e.g. color presentation data will not be returned to a monochrome device and vice versa).

[0043] For the table preview capability, the client (AV) will as part of the attachment conversion request inform the server 28 that it is capable of showing table previews as well as supplying the requesting device 12 screen resolution and color information.

[0044] This information enables the server 28 to return UCS (Universal Content Stream) data back to the device 12 containing preview table information preferably for a 4x4 row/column table with individual cell content limited to between 8-32 characters based on the requesting device screen information as well as preview table color information if supported by the device. A DOM ID for the full table is also returned to the AV on device 12 that allows the user to request the full table for viewing in spreadsheet view, if desired.

[0045] A flow chart of the device side request process is shown in Figure 11. At step 110, the user requests the full content of an attachment (e.g. an attachment to an email), by appropriate operation of the user interface for device 12. In response, at step 115, the device AV issues a document request to the AS server 28 for the first chunk (e.g. typically

3K) of the document attachment configured for the device display width, height and number of colors the device can display.

[0046] If the download request is successful (a YES branch at step 120), the client-side AV determines whether a table preview is comprised in the first portion (e.g. 3K of downloaded data. If not (step 125), the device displays the normal text content (step 130).

[0047] The AV then issues any further requests for additional chunks of the document attachment (a YES at step 135), and process flow returns to step 120. Otherwise, if the entire document has been downloaded (a NO at step 135), then the process terminates (step 140).

[0048] If a request to download a chunk of data from the AS on server 28 is unsuccessful (i.e. a NO at step 120), then an error flag is generated and an error message displayed (step 145), following which the process terminates (step 140).

[0049] If, as step 125, a table preview is comprised in the chunk of downloaded data the AV displays the table preview, as shown in Figure 3, and associates the DOM ID for the full table (stored in server 28) in the table link (step 150). If additional content (beyond the table preview) is available in the data chunk for viewing (step 155), the additional content is displayed (step 130). Otherwise, process flow branches to step 135.

[0050] Turning now to the server-side process, when the attachment server 28 receives a document containing embedded tables, it first constructs the DOM structure for the document if it does not already exist in the in-memory DOM structure cache. Each embedded table in the file is constructed as an external table component and is added to the DOM structure as an attribute of an external component reference (ECR) command, as described in greater detail below.

[0051] The corresponding DOM structure for the following sample document containing an embedded table, is shown in Figure 12 where, in order to simplify the graph, the DOM structure does not contain the detailed graph presentation for the embedded table:

Table Test

Cell One	Cell Two
Cell Three	Cell Four

[0052] As shown in Figure12, the table component is added as attribute "Ref" to the external component reference command (ECR), which is a node in the DOM graph structure for the main document.

[0053] The server-side process of Figure 13 begins at step 135 to iterate through the entire document DOM structure 130. Provided the un-iterated remaining portion of the DOM structure continues to contain commands (step 140), the next command is iterated (step 145). Otherwise, the process ends (step 150). If the process encounters an ECR command (step 155) and the component being referenced is a table (step 160), the process constructs a table component (step 165), which is a preview for the referenced table component. This preview table component is added as an attribute to the ECR command (step 170), as shown in Figure 14.

[0054] The preview table component contains only partial information about the referenced table. By default, as discussed above, the preview table preferably has only a maximum of 4 rows and 4 columns of the original table, and each table cell comprises up to 16 characters. However, a person skilled in the art will understand that the numbers of table rows, columns and table cell characters are configurable and can be dynamically set in the conversion request. In addition the table formatting information is preserved in the preview table component.

[0055] After the preview table component is constructed, it is added as an attribute to the original ECR command, as discussed above. During the response and resulting DOM graph construction for a conversion request, the attachment server 28 persists the ECR command and the attached preview table component with its formatting properties. The flow chart of Figure 15 sets forth the steps of persisting the ECR command and attached preview table component in the document DOM structure of Figure 14 with the required formatting properties.

[0056] After initialization of variables (140), the server process iterates the referenced table component (step 145) extracted from the ECR command shown in Figure 14. Provided that there are commands remaining to be iterated (step 150), a determination is made as to whether the next command is a Table Row command (step 160), Table Cell command (step 165), Text Segment command (step 170), or whether (step 175) the table cell character index (m_iCharIndex) is less than the maximum number of character in each table cell (kiMaxCellChar). If the result of each of these determinations is NO, then process flow returns to step 150. On the other hand, if there are no further commands then the process ends (step 155).

[0057] If the next command is a Table Row command (step 160), a determination is made at step 180 as to whether the Table row index (m_iRowIndex) exceeds the maximum number of rows in the preview table (kiMaxRow). If no, then a table row command is created and the attributes from the original table row command are copied to

the new table row command (step 185), the table row index (m_iRowIndex) is incremented (step 190), the table cell index (m_iCellIndex) and Table cell character index (m_iCharIndex) are reset (step 195), and the new command is added to the preview table DOM structure hierarchy (step 200).

[0058] If the next command is a Table Cell command (step 165), a determination is made at step 210 as to whether the table cell index (m_iCellIndex) exceeds the maximum number of cells in the preview table (kiMaxCell). If no, then a table cell command is created and the attributes from the original table cell command are copied to the new table cell command (step 215), the table cell index (m_iCellIndex) is incremented (step 220), the table cell character index (m_iCharIndex) is reset (step 225), and the new command is added to the preview table DOM structure hierarchy (step 200). If at step 210 the table cell index (m_iCellIndex) does not exceed the maximum number of cells in the preview table (kiMaxCell), then process flow returns to step 150.

[0059] If the next command is a Text Segment command (step 170), a determination is made at step 230 as to whether the table cell character index (m_iCharIndex) is less than the maximum number of characters in each cell (kiMaxCellChar). If yes, then a text segment command is created (step 235), then at step 140 the text from the original text segment command is copied, where the amount of text to be copied is the lesser of the number of characters in the original text segment and (kiMaxCellChar - m_iCharIndex). At step 245, the copied text is added to the new text segment command as an attribute, where m_iCellIndex. += number of character copied. The new command is then added to the preview table DOM structure hierarchy (step 200). If at step 230 the table cell character index (m_iCharIndex) is not less than the maximum number of characters in each cell (kiMaxCellChar), then process flow returns to step 150.

[0060] If at step 175 the table cell character index (m_iCharIndex) is less than the maximum number of character in each table cell (kiMaxCellChar), then a command is created (step 250) that is of the same type as in the original document. The new command is then added to the preview table DOM structure hierarchy (step 200).

[0061] In summary, since the preview table for a table component is cached in the DOM structure as an attribute of the corresponding ECR command, memory consumption is slightly increased. However the processing time is significantly reduced by eliminating extra iterations of the table component on repeated file conversion requests.

[0062] A person skilled in the art, having read this description of the preferred embodiment, may conceive of variations and alternative embodiments.

[0063] All such variations and alternative embodiments are believed to be within the ambit of the claims appended hereto.

Claims:

1. A process for previewing at least one table embedded in a document stored in a server, comprising:

building a graph structure within said server representing a map of said document, comprising a table component representing each said table;

issuing a request for said document from a mobile communication device to said server comprising an indication of display capabilities for said device;

iterating said graph structure within said server and for each said table component constructing and adding a preview table component as an attribute to an external component reference command, wherein said preview table component comprises a predetermined number of rows and columns of said table with cell content textual information limited to said display capabilities; and

downloading and displaying said document from said server to said mobile communication device comprising said preview table component.

2. The process of claim 1, wherein said predetermined number of rows does not exceed four, said predetermined number of columns does not exceed four, and said cell content textual information is limited to no more than thirty-two characters.

3. The process of claim 1 or claim 2, wherein said iterating further comprises iterating said preview table component and for each iterated table row command, table cell command and text segment command within said predetermined number of rows and columns and limited cell content textual information, creating a new table row command, new table cell command, and new text segment command respectively with attributes copied from each said iterated table row command table, cell command and text segment command and adding said new table row command, new table cell command, and new text segment command to said preview table component in said graph structure.

4. The process of claim 3, further comprising incrementing an index associated with each of said new table row command, new table cell command, and new text segment command, and repeating said iterating until any said index exceeds said predetermined number of rows and columns and limited cell content textual information.

5. The process of any one of claims 1 to 4, wherein said downloading and displaying further comprises downloading a document ID associated with said table in said graph structure from said server to said device.

6. The process of claim 5, further comprising generating a menu selection at said mobile communication device for initiating one of either user retrieval or viewing of said table in response to which said device transmits said document ID to said server and said server downloads said table to said device.

7. A process in a server of representing a document, comprising:

building a graph structure representing a map of a document having at least one embedded table;

iterating said graph structure and for each said table constructing and adding a preview table component as an attribute to an external component reference command, wherein said preview table component comprises a predetermined number of rows and columns of said table with limited cell content textual information relative to said embedded table; and

transmitting said document comprising said preview table component for external viewing.

8. The process of claim 7, wherein said predetermined number of rows does not exceed four, said predetermined number of columns does not exceed four, and said cell content textual information is limited to no more than thirty-two characters.

9. The process of claim 7 or claim 8, wherein said iterating further comprises iterating said preview table component and for each iterated table row command, table cell command and text segment command within said predetermined number of rows and columns and limited cell content textual information, creating a new table row command, new table cell command, and new text segment command respectively with attributes copied from each said iterated table row command table, cell command and text segment command and adding said new table row command, new table cell command, and new text segment command to said preview table component in said graph structure.

10. The process of claim 9, further comprising incrementing an index associated with each of said new table row command, new table cell command, and new text segment command, and repeating said iterating until any said index exceeds said predetermined number of rows and columns and limited cell content textual information.

11. The process of any one of claims 7 to 10, wherein said downloading and displaying further comprises downloading a document ID associated with said table in said graph structure.

12. A process in a mobile communication device of previewing a document comprising:

issuing a request for a remotely stored document having at least one embedded table, said request comprising an indication of display capabilities for said mobile communication device; and

receiving and displaying said remotely stored document comprising a preview table component having a predetermined number of rows and columns of said table with cell content textual information limited to said display capabilities for said mobile communication device.

13. The process of claim 12, wherein said predetermined number of rows does not exceed four, said predetermined number of columns does not exceed four, and said cell content textual information is limited to no more than thirty-two characters.

14. The process of claim 12 or claim 13, wherein said receiving and displaying further comprises receiving a document ID associated with said table in said remotely stored document.

15. The process of claim 14, further comprising generating a menu selection at said mobile communication device for initiating one of either user retrieval or viewing of said table in response to which said device transmits said document ID to identify said table in said remotely stored document.

16. A system for previewing at least one table embedded in a document stored in a server, comprising:

means for building a graph structure within said server representing a map of said document, comprising a table component representing each said table;

means for issuing a request for said document from a mobile communication device to said server comprising an indication of display capabilities for said device;

means for iterating said graph structure within said server and for each said table component constructing and adding a preview table component as an attribute to an external component reference command, wherein said preview table component comprises a predetermined number of rows and columns of said table with cell content textual information limited to said display capabilities; and

means for downloading and displaying said document from said server to said mobile communication device comprising said preview table component.

17. A server comprising:

means for building a graph structure representing a map of a document having at least one embedded table;

means for iterating said graph structure and for each said table constructing and adding a preview table component as an attribute to an external component reference command, wherein said preview table component comprises a predetermined number of rows and columns of said table with limited cell content textual information relative to said embedded table; and

means for transmitting said document comprising said preview table component for external viewing.

18. A mobile communication device comprising:

means for issuing a request for a remotely stored document having at least one embedded table, said request comprising an indication of display capabilities for said mobile communication device; and

means for receiving and displaying said remotely stored document comprising a preview table component having a predetermined number of rows and columns of said table with cell content textual information limited to said display capabilities for said mobile communication device.

19. A machine readable medium comprising program code means executable in a processor of the server of claim 17 and/or the mobile communication device of claim 18 for implementing the process of any one of claims 1 to 6 and/or the process of any one of claims 7 to 11 and/or the process of any one of claims 12 to 15.

20. A communications system comprising at least one mobile communication device according to claim 18 and/or a server according to claim 17.