

Oracle® Documaker

Mobile **User Guide**

13.0.0

Part number: G18689-01

December 2024

Copyright © 2024 Oracle and/or its affiliates. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Oracle, JD Edwards, and PeopleSoft are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

CONTENTS

Preface	5
Audience	5
Reference Documents	5
Conventions	6
Documentation Accessibility	6
Customer Support	6
Contact	7
Oracle Documaker Mobile Overview	8
Understanding Mobile	11
Understanding Mobile Processing	12
Snippets	13
Understanding Snippet Keywords	22
Creating a Cascading Style Sheet	27
Choosing the Right Mobile Output Format	27
Delivering Mobile Output	29
Mobile Output Process	30
Mobile Output Management Workflow	35
Configuring and Enabling Mobile	37
Configuring Mobile for Oracle Documaker Enterprise Edition	38
Validating Documaker Mobile Installation	39
Configuring Mobile for Oracle Documaker Standard Edition	41
Validating Documaker Mobile Installation	42
Best Practices	44
Troubleshooting	45
Layout and Design	47
Layout Example	48
Mockup Design for Mobile	48
Navigation	49
Content Data based Interactions	50
XSLT Techniques	51

Mobile Responsive Output Format	54
Mobile Document Authoring Tools	57
Adobe's Dreamweaver	57
Oracle JDeveloper	59
Eclipse's IDE for Java EE Developers	61
Notepad++	64
Microsoft's Expression	64
Packaging and Deployment of Mobile	65
References	66
Mobilizing Existing Formsets	68

Preface

The purpose of this document is to provide a high level overview supported by step by step instructions to configure and use Oracle Documaker Mobile (ODM). This document also provides guidance on the mobile implementation process for Documaker with information on how to develop mobile content.

AUDIENCE

This document is intended for those who will utilize Oracle Documaker Mobile to create a web presentation of documents generated by the Documaker system. Oracle Documaker Mobile provides an ability to generate the type of responsive output required by today's mobile landscape. Readers of this document are:

- Documaker Document Authors
- Mobile Document Authors- web designer, web author
- Development Managers
- Quality Assurance Managers
- IT / Infrastructure Managers
- Senior Management

REFERENCE DOCUMENTS

Refer to the following documents for information on steps to install and maintain Oracle Documaker Mobile:

- Documaker Mobile Installation Guide
- Documaker Word Add In Help
- Documaker Studio User Guide and Online Help
- Documaker System Requirements Guide
- Documaker Administrator Help
- Documaker Enterprise Administration Guide
- Documaker Output Management Guide
- DAL Reference
- INI Reference

To view and download the latest Oracle Documaker Mobile documentation, visit the Oracle Technology Network:

<http://www.oracle.com/technetwork/documentation/insurance-097481.html>

CONVENTIONS

The following text conventions are used in this document:

Convention	Description
bold	Indicates information you enter.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
<code>monospace</code>	Monospace type indicates commands, URLs, code in examples, and text that appears on the screen.

Tips, Notes, and Warnings

- A *Tip* provides a better way to use the software.
- A *Note* contains special information and reminders.
- A *Warning* contains critical information that if ignored, may cause errors or result in the loss of information.

DOCUMENTATION ACCESSIBILITY

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

CUSTOMER SUPPORT

If you have any questions about the installation or use of our products, please call +1.800.223.1711 or visit the My Oracle Support web site:

<http://www.oracle.com/us/support/index.html>.

Go to My Oracle Support to find answers in the Oracle Support Knowledge Base, submit, update or review your Service Requests, engage the My Oracle Support Community, download software updates, and tap into Oracle proactive support tools and best practices.

Hearing impaired customers in the U.S. who need to speak with an Oracle Support representative may use a telecommunications relay service (TRS); information about TRS is available at <http://www.fcc.gov/cgb/consumerfacts/trs.html>. International hearing impaired customers should use the TRS at 1.605.224.1837.

CONTACT

USA:+1.800.223.1711

Canada: 1.800.668.8921 or +1.905.890.6690

Latin America: 877.767.2253

For other regions including Latin America, Europe, Middle East, Africa, and Asia Pacific regions: Visit- <http://www.oracle.com/us/support/contact/index.html>.

Copyright ©2024 Oracle and/or its affiliates. All rights reserved.

Follow us



<https://blogs.oracle.com/documakertech/category/dt-odee>

Chapter 1

Oracle Documaker Mobile Overview

Documaker Mobile allows businesses to utilize the same content created for paper-style output for responsive, device aware, mobile delivery. Documaker Mobile provides the ability to render output for iOS, Android, laptop, and desktop devices by leveraging existing Documaker capabilities including content, triggers, and data mapping.

Businesses have traditionally delivered complex documents to their customers using print delivery or electronic versions of that same printed output. Consumers are demanding more interactive documents that are searchable, easy to navigate, and responsive to their reading device including smaller mobile devices such as phones and tablets.

With Documaker Mobile you can:

- Provide mobile context to existing content
- Reformat print page-styled content into mobile device responsive HTML
- Reformat print page-styled content into XML that can be transformed with industry standard mobile device responsive style-sheets

Documaker Mobile enables content from Documaker to be formatted as device responsive HTML5 and in other in UTF-8 based text formats such as Extensible Markup Language (XML) or Comma Separated Values (CSV), XHTML, JSON, etc. and enables user defined data schemas to define the output nomenclature.

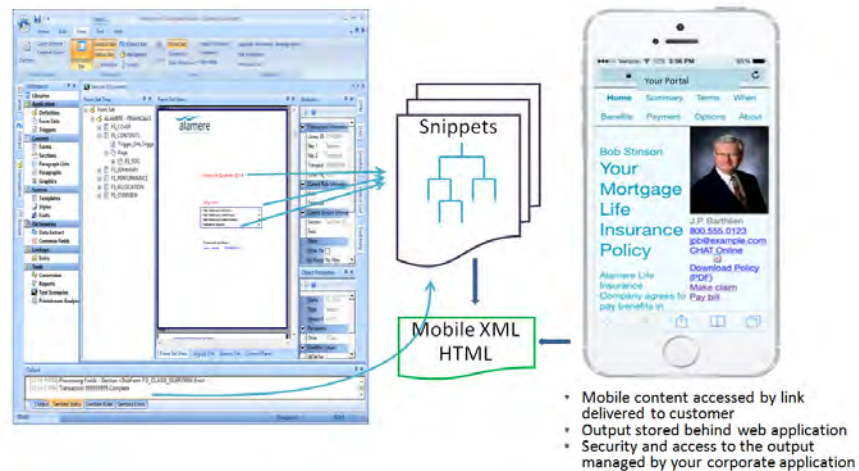
Oracle Documaker Mobile produces output in a format easily targeted for use on mobile devices. Traditionally Documaker has provided high fidelity output that is device independent. This ensures that regardless of output target (printer or PDF) the recipients will receive the exact content, structure, and layout defined in Documaker. Documaker Mobile does not follow a print page fidelity paradigm as other output targets have in the past. The output from Documaker Mobile diverges from the printed format and varies its presentation depending on the (mobile) device that it is viewed on. This type of output is known as responsive because it responds by restructuring its presentation to support the device on which it is being viewed. The objective is for the Documaker Mobile output to be mobile device responsive by supporting features such as changing presentation styles depending on the size of the viewport of the device on which the presentation is being viewed.

Documaker Mobile output is a merge of the formset content as triggered for a transaction through Documaker with specially prepared files for mobile such as XML snippets, XSL style sheets, CSS style sheets, JavaScript files, etc. Documaker Mobile is responsible for performing the merger of Documaker formset content with a set of mobile definitions to produce mobile targeted results based on rules defined in Documaker Studio.

Documaker Mobile repurposes the hard earned content knowledge embedded in your legacy paper based publishing systems with state-of-the-art support for web based mobile document publishing.

To know more about the operating system and infrastructure software which is needed to run Oracle Documaker Mobile See [Oracle Documaker System Requirements guide](#) section “*Oracle Documaker Mobile*”.

A high level view is presented here.



User Roles and Responsibilities

Generating documents relies on someone to draft the content of the output sent to customers. This activity is typically done by users of Documaker Studio, Word, InDesign, or other content tools that allow these users to identify the content, variable information, and intended document recipients. Individuals responsible for developing the document content may also be referred as template designers, business analysts, or various other terms but for purposes of this document, they are called **Document Authors**. There is an additional role introduced with Documaker Mobile and that is the role of the **Mobile Author**, who designs, develops, and implements the output intended for a mobile user. The Mobile Author works with the mobile strategy team to identify a content layout for optimal user experience on mobile devices.

Mobile Author

To implement a mobile strategy for document output, the Mobile Author designs and develops output intended for a web or mobile user. This user plays the role of a web user interface developer and should be skilled in the following:

-
- Web based technologies and tools such as Adobe's Dreamweaver, Eclipse Foundation's IDE for Java EE Developers, Notepad++ or other tools for manipulating text based syntax conforming to Web Standards.
 - Ability to implement web pages via HTML/HTML5, CSS/CSS3, JavaScript and other Standards based technologies such as XML and XSLT.
 - Experience in developing Single Page Architecture (SPA) based interactive web pages.
 - An understanding of the company's existing web standards that can be used to leverage Web assets a company already has in place, including style standards and methodologies for securely distributing content to customers over the Web including knowledge about company Web portals.

The company that implements a mobile strategy with Documaker must also pull in dedicated representatives from their Web group or source these resources from within their group so the output from Documaker can be designed to work with the Web presence already established. If no Web presence is already in place Oracle offers tools such as WebCenter Sites and Portal that provide a complete tool set for creating and managing Web presence.

Chapter 2

Understanding Mobile

This chapter provides detailed information on the architecture and processing capabilities of Oracle Documaker Mobile.

This chapter discusses the following topics:

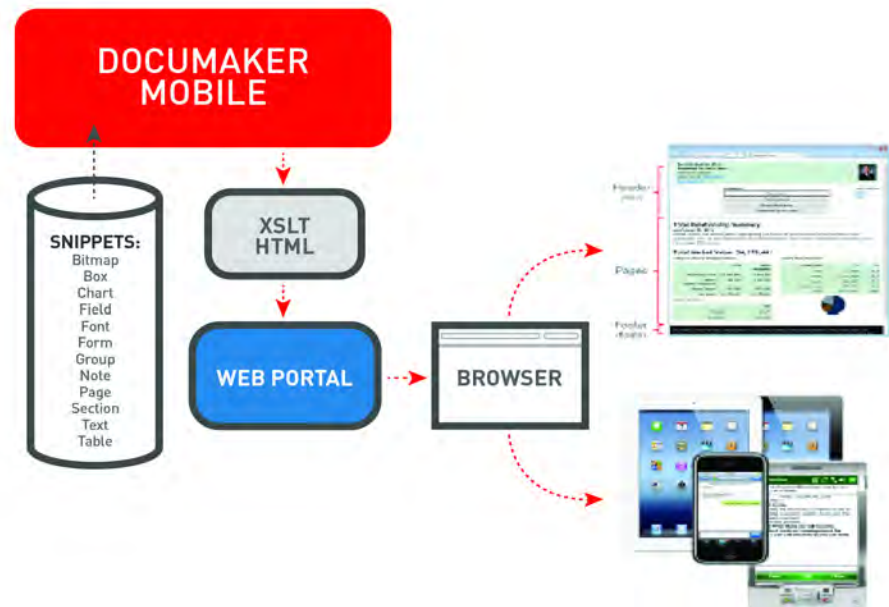
- *Understanding Mobile Processing* on page 12
- *Snippets* on page 13

UNDERSTANDING MOBILE PROCESSING

Oracle Documaker Mobile provides the ability to render output for devices such as iOS, Android, and laptops. It does so by leveraging the existing capabilities in Documaker to define content, triggers, and data mapping elements while adding in new elements that allow Documaker to render output for various device targets, including mobile, instead of page based targets such as print or PDF.

These new elements, snippets, scripts, widgets, etc., combined with the document content provide the responsive user experience.

Documaker Mobile processes each object of document content through a snippet

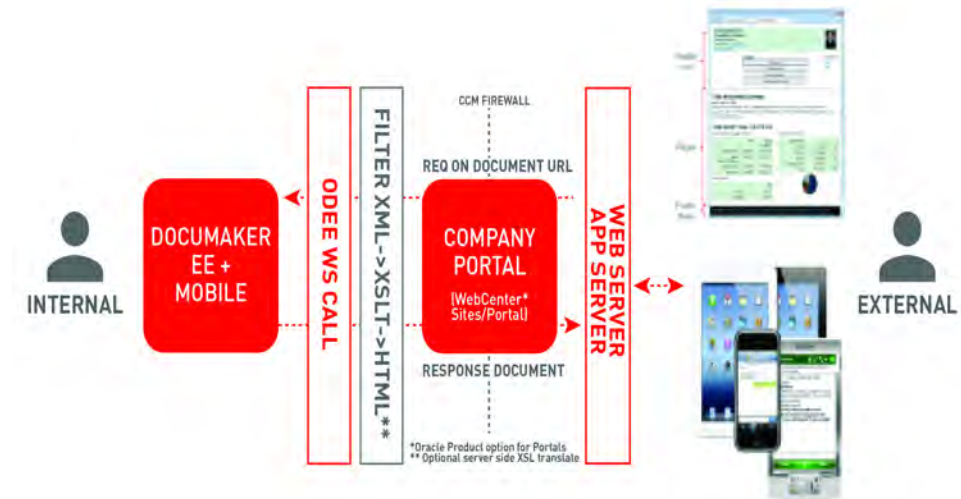


file. The snippet, or layout file, contains presentation markup and keywords that are replaced by the properties and values of the document content being processed. For example - each field in the document content will be processed with the field snippet designed to output the name of the field, the font used for the field, and the value populated into the field using keywords as shown here:

```
<FLD N="documaker.name" F="documaker.font.id">documaker.text</FLD>
```

When a field in the document content is processed through this snippet by the Mobile Responsive Output (MRO), all the snippet content is written as output and the keywords, also known as replacement tags, such as: "documaker.name", "documaker.font.id" and "documaker.text". These will be replaced with field's name, font ID and content text respectively. By default, similar snippets are defined for objects other than fields and the output from Documaker Mobile represents processing of all objects within the document via this snippet transformation technique. Note that the format of the final mobile output is dependent upon the snippets used to process the document content.

Once the mobile output is ready, it can be delivered to the recipient through one of many channels. Mobile output can be placed / stored on a file system, pushed to the Oracle WebCenter product or to other targets such as FTP, or returned to a calling application using Documaker Enterprise's DWS web service interfaces. For more information about possible delivery options, see the topic on *Delivering Mobile Output*.



As discussed in the overview, generating mobile output relies on the contents of snippet files to determine the final output format. Let's take a closer examination of how snippets are used by the system.

SNIPPETS

A snippet is a programming term for a small file of re-usable text or source logic written to allow substitution of variables. Snippets are generally small files written with a text editor or program source code editors.

Mobile output processing begins by loading a “master” snippet that directs the output process. Each object included by the output generation has a default snippet that is named for that particular object type. For example, text area uses “textarea.xml” as the default snippet name. Fields use “field.xml” as the default snippet name. Sections use “section.xml” as the default snippet name. See the table below for a full list.

While the Documaker Mobile installer includes both xml and json snippets, Documaker Mobile assumes snippet files have an .xml extension. If you will be using snippets with a .json extension to produce mobile output, set the SnippetExt to .json in the <PrtType:MRO> configuration group as shown below:

```
<PRTOTYPE:MRO>
```

```
SnippetExt = .JSON
```

The following table identifies the default snippet name associated with objects that might be included in mobile responsive output. All snippet names are required to be lowercase file names.

Snippet Name (without extension)	Object Type Using That Snippet
master	The controlling snippet that drives the process and directs the layout of the responsive output. This snippet is assumed related to the entire document.
group	Forms Lists, also know as "Line of Business", "Form Groups" or references to KEY2 or KEY3
form	Forms (.FOR)
page	Pages
section	Sections
box	Box, line, and shaded areas defined within sections
bitmap	Graphic bitmaps and logos
textarea	Text Areas
stext	Text Labels, also sometimes referred to as "static text". This object snippet is also used for individual text elements found within text areas.
field	Fields that are not barcode or multiline text fields. May also be used for barcode and multiline fields if a specific snippet is not available for those objects.
field_barcode	Fields that are barcode type
field_textarea	Fields that are multiline text fields
barcode	Barcodes
font	Fonts
note	Note objects where the type is sticky note <ul style="list-style-type: none"> • note_bookmark - bookmark notes • note_memo - note objects where the type is memo
mtformat	End of paragraph marks found within textareas
chart	Charts <p>Note: The axis values will be output only for chart text labels that have names beginning with POINTAXIS or SCALEAXIS. The label's text value should be set to the content you wish to display on the chart.</p>
vector	Vectors
signature	Signatures
table	Tables (Note that components within a table will reference other object snippets like textarea and stext.

Default Snippets

By default, the Documaker mobile solution provides object level snippets and stylesheets for rendering content for mobile use. However, these resources are meant to only provide a generic mobile enabled solution that is not designed for use as final output within a company's mobile strategy. It is up to the company to build upon the snippets provided to achieve the look and feel desired for mobile documents.

Overriding Snippets with Custom Definitions

To support the use of non-default snippets, the Documaker author can directly update the default snippets provided, thus changing the snippets used for all documents, or they can override the default definitions on a document per document basis, using context tags. See the *Customizing Mobile Output with Context Tag and Omit Properties* topic for more information on using context tags.

Snippet Contents

Documaker's Mobile installer includes both XML and JSON snippets. MRO assumes snippet files have an extension of .xml. Should you want to produce JSON mobile output, set the SnippetExt setting to .json in the <PrtType: MRO> configuration group. However, exactly what a snippet contains depends on what type of object is referenced by the snippet, the intended output format, and the desired behavior. Documaker does not validate the content of the snippet, it only looks for variable replacement tags to replace with content from the transaction being processed. For more information on the variable replacements or supported keywords, see the topic *Understanding Snippet Keywords*.

Snippet Location

All snippets used by a given MRL will be found in the same location defined within the MasterResource INI group option MROLIB, as shown here:

```
<MasterResource>
```

```
MROLIB=
```

MasterResource settings option can be redirected to the CONFIG group that controls other MRL options.

Master Snippet

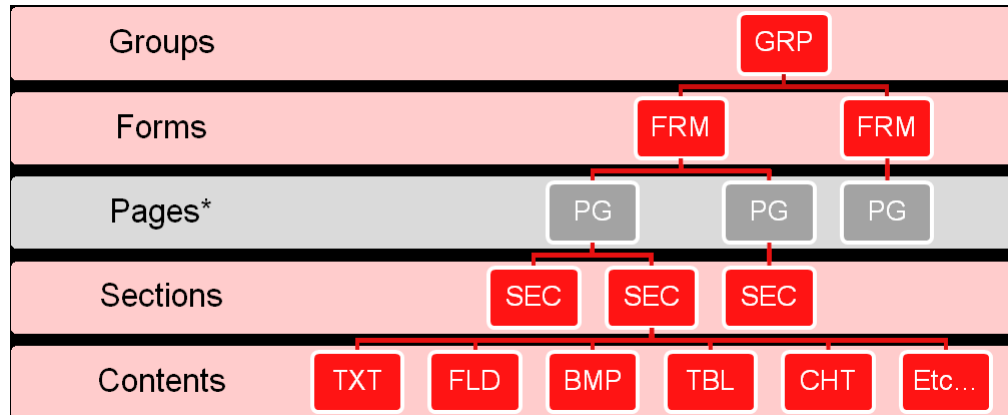
Each mobile generation begins by looking for the master snippet. The master can be thought of as the shell that drives the mobile transformation process. The master snippet name for XML snippets is 'master.xml'. The master snippet name for JSON snippets is 'master.json'.

The master snippet must also be located in the location defined by the MROLIB= option within the MasterResource group – as are all other snippets.

Information will be presented to show that you can have multiple master snippets to drive the mobile transformation process for different situations. See *Global Context Tag* for more information.

Object Specific Snippets

The default snippets provided with Documaker Mobile demonstrate a hierarchy of object processing to help you understand the sequence of processing content that follows from the master snippet. The following diagram shows the layers of objects within the system as it relates to the provided snippets.



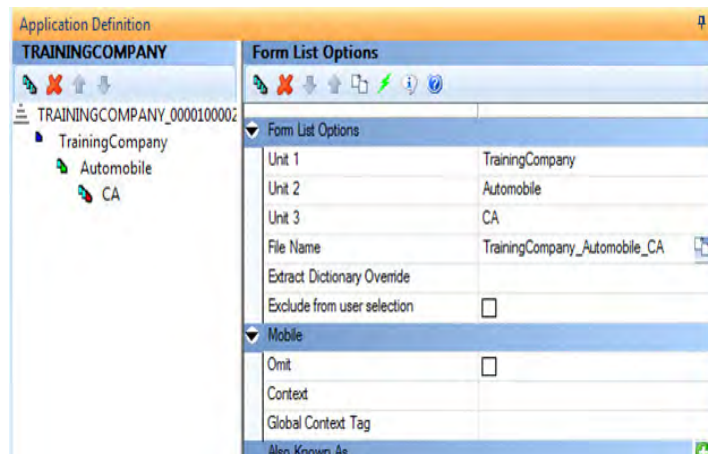
In the default output there is a DocSet element which serves as the document root and contains Groups. Groups have Forms. Forms have Pages. Pages have Sections. Sections have low level contents that include all the publishing objects in the section in reading order including text content, field content, images, tables, charts and all other publishing objects available.

In the default snippets the Page layer is omitted. If you want to have pages in your output, you can change your Form snippet to list Pages instead of Sections. This will cause Pages to be inserted into your Mobile output.

Customizing Mobile Output with Context Tag and Omit Properties

An object property can be used to define a tag that will be appended to the default snippet name for the object, generating a “custom” named snippet for that particular object.

Within Documaker Studio and the Microsoft Word Add-In, each object that may be included in mobile output has two new properties available for use: Omit and Context.



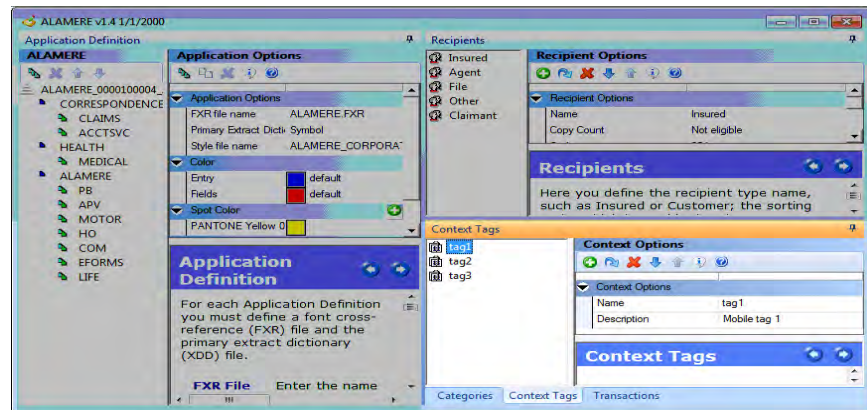
The Context property is used to define a value you wish to append to the default snippet name and thereby generate a “custom” snippet name to use for this particular object. For example, the field object by default will look for a snippet with the name “field.xml”. If you associated a tag of “green” on a particular field object, this would expect to locate a snippet with the name “field_green.xml”. Any field not specifying a context tag will continue to look for the normal “field.xml”. See details below on Snippet Search Behavior for more information.

The Omit check box is used to suppress a particular object from being included in the responsive output. Depending upon the exact nature of the desired output, it may not be necessary to include an object. For instance, when generating a responsive HTML output, you may wish to suppress certain boxes or large graphics that may not format in the desired output or cause that output to become overly large; therefore the object's Omit property can be enabled. In addition, Omit might be used to suppress certain sensitive field or text information that you don't want to include when output in a given format might be considered a security risk in a mobile landscape. Omitting an object does not omit embedded objects within that object. For example, omitting a text area does not omit embedded fields, boxes or graphics within the text area. You must also omit the field, box or graphic itself.

Application-wide Context Tags For User Selection

A predefined list of context tags can be defined for user selection within Documaker Studio and the Word Add-In. These are defined and maintained in the Application Definition (BDF).

When Oracle Documaker Mobile is installed in your development environment, a new tab appears in the same area as the Category and Transactions tabs. This tab is called Context Tags.



The Description property of the Context option is simply to associate some descriptive text to assist the Document Author in choosing the correct context. Note, since the tag name ultimately becomes part of the file name for the snippet, the Context tag name follows file naming restrictions. Therefore, the characters : / \ > < ? * cannot be used in the name field. Characters with ASCII scan codes lower than 32 or higher than 127 are also not permitted. Uppercase letters will be lowercased. A name with all spaces is not valid. The Context Tag list simply holds those tags that you would like the Document Author to be able to select when creating content. It is possible to assign security restrictions to users that will force selection from this list or to allow users to create a custom tag not defined in the Application Definition.

The application defined Context Tag list is also included in the Workspace Definition file Studio creates and that is consumed by Word Add-in users. See below for an example of the file contents.

```
- <Definition Name="Alamere" TimeStamp="2014-10-10 12:06:14" Version="12.4">
+ <DALTriggers>
- <ContextTags>
  - <ContextTag Name="tag1">
    <Description>Mobile tag 1</Description>
  </ContextTag>
  - <ContextTag Name="tag2">
    <Description>Mobile tag 2</Description>
  </ContextTag>
  - <ContextTag Name="tag3">
    <Description>Mobile tag 3</Description>
  </ContextTag>
</ContextTags>
- <Recipients>
```

Security option for choosing from available tags

A security setting determines if a user is required to select from the application defined context tag list. This is similar to other security attributes that determine whether a user is limited to choosing predefined fields, triggers, and templates. The context tag restriction attribute can be found on the User Rights page of the Security Wizard.

When checked, that user is only allowed to select from the predefined tag list specified by the application definition. Users without this rights restriction are allowed to define a mobile context tag not defined within the application list. The default restriction attribute for a new user is not selected, but if the user is created based on another user, this property will be carried forward.

In the drop-down selection box the user sees the available tags as well as the descriptions defined for each. Only the selected tag name appears in the actual control after selection. The description is not maintained in the individual object property.

Snippet Search Behavior

As mentioned, each object type that can be included in mobile responsive output has a default snippet named for that type of object. As identified in the previous section, you may customize a snippet for a given object for any reason where the output for that particular object should be different than other similar objects of the same 'type' - i.e. fields, sections, forms. The custom snippet is named starting with default snippet name for that object and followed with an underscore and appended with additional text.

When a snippet name contains underscores, the mobile output generation will attempt multiple searches looking first for more highly qualified snippet names and working back to less qualified snippet names until a snippet with that name is found. For instance, in the snippet name reference table (shown earlier) you will find that fields of a barcode type expect to use the default snippet name "field_barcode.xml". If a file with that name is not found, the search will continue and look for "field.xml". A new search attempt will occur at each underscore, reversing through the names until a matching snippet file is found.

For example, if starting with a snippet name of "object_one_two_three_four.xml", the following names will be searched for order until one was found.

- object_one_two_three_four.xml
- object_one_two_three.xml
- object_one_two.xml
- object_one.xml
- object.xml

This is an important behavior to note since objects allow for the definition of a mobile tag to be appended to the names. If the custom snippet name is not found, the system will then continue parsing backwards until eventually the default snippet name is used.

Remember a previous example where a field defined a mobile context tag "green", when that particular field object is output the first attempt will be to locate a snippet with the name "field_green.xml". If that is not found, then the name "field.xml" (the default for fields) will be used. If that field happened to have been a barcode type, the first search would have been to look for "field_barcode_green.xml". If that file is not found, then "field_barcode.xml" will be next. If that is not found, then simply "field.xml" will be used.

If a snippet of a given name is not located, this is not considered an error. As a default, a warning message will be output that takes the following form where the name of the missing snippet and its extension is listed as shown:

Unable to open file snippetname.ext

This warning message causes no harm and will generally only be output once per run. A subsequent reference to the same file will simply shortcut to the name that was eventually used for that object snippet request. You can suppress the output messages if desired by using this INI option within the Mobile options group defining the responsive output.

```
<PRTTYPE:MRO>
```

```
Verbose = No
```

As mentioned, the default is Yes, and messages will be generated when snippets of a given name are not found.

Global Context Tag

It is possible to assign a global context tag to a document on its application definition. When a global context tag is defined, all objects will first look for the associated snippet with the global tag appended to the name, just as if the context tag had been set on each and every instance of an object being evaluated.

For example, if a global context tag "claim" is assigned at the transaction level, then a search for a field's snippet will first look for "field_claim.xml". This occurs even through the field did not specify a mobile context.

In fact, if the field does have a custom mobile context tag defined, that tag is still added into the initial search name with the global tag appended at the end. Returning to an earlier example where a field defines a mobile tag "green", assume that field is part of a transaction that defines a global tag "claim". The first snippet name searched for this field will be "field_green_claim.xml". (Remember, that snippet searches iterate by shortening the name at each underscore until a snippet is found.)

This global context tag feature is to be used in situations where it is desirable to generate customized output for a given transactions.

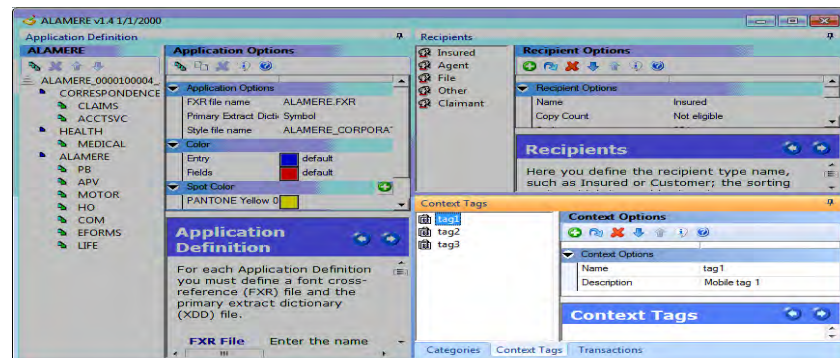
This different output can start with the master snippet. The global context tag defined for a document will also apply to determining the master snippet name. This means if a transaction has "claim" assigned as the global context tag, the initial master snippet searched for will be "master_claim.xml". (Just as any other snippet search, if this name is not found, then the "master.xml" will be searched next.)

Since the master snippet drives the start of the mobile generation, having a global tag modify the master snippet name offers the ability to generate customized by doing things such as referencing different style sheets or other layout resources not used in the normal output case.

Although the global context tag is first appended to all object snippet searches, there is no requirement that you have a custom snippet for every object. The normal snippet search method will continue looking for a snippet by truncating the name at the underscores. You need only define custom snippets for those objects that require custom output for that tag, which very often will just be the master snippet.

Assigning a Global Context Tag to a Transaction Document

A global context tag can be defined for particular lines of business or Key1/Key2 combinations in the application definition (BDF). Within the definition of the Forms Lists options for a Key1/Key2 combination, you will see an additional option in the Mobile properties. The Global Context Tag is not the same as the Context tag defined for any other object - seen here just above this option.



Remember, a Mobile Context tag definition only applies to the given object at run time. Conversely a Global Context Tag defines a tag that will be applied to every object, including the master snippet, when generating output for the entire transaction.

The Global Context Tag is associated with the document via the normal triggering or key selection process. The first Forms List assigned to the transaction will have its Global Context Tag setup to the document level.

If multiple Forms Lists are included in a transaction document, only the first Forms List definition determines the Global Context Tag for the document.

Alternatively, it is possible to assign a document Global Context Tag without defining the tag within the Forms List options of the Application Definition. Instead, you can use a DAL script to assign or override this document level (Global) tag. You just have to ensure the script is executed prior to the start of mobile responsive output generation.

DAL Functions for Global Context Tag Support

Two DAL functions are available that determine or assign the document Global Context Tag. These new functions appear under the Miscellaneous grouping in the Trigger Manager Keyword List.

GetGlobalMROTag() - This function can be used to obtain the value of the Global Context Tag that is assigned to the transaction. There are no parameters expected. Usage in a script must pick up the return value or else a runtime syntax error will be generated.

Example:

```
Tag = GetGlobalMROTag()  
If (Tag = "")  
// No tag assigned yet
```

The Global Context Tag determines the initial snippet names used by mobile responsive output (MRO). If a tag is defined, then it is assumed that the transaction will attempt to use specific snippet names before reverting to defaults.

SetGlobalMROTag(tag) - This function assigns the given parameter string as the Global Context Tag associated with the active transaction document. There is one expected parameter which is assumed to be a string that defines the Global Context Tag to assign.

Example:

```
SetGlobalMROTag("claim")
```

Using this function will override any previously assigned Global Context Tag assigned to the document whether assigned implicitly via the triggering process or by previous script assignment. If you pass an empty string to this function, example: Passing an empty string to the SetGlobalMROTag function, for example, SetGlobalMROTag(""), is the equivalent of removing the documents assigned Global Context Tag and allows the mobile responsive generation to use normal defaults for all snippet references.

UNDERSTANDING SNIPPET KEYWORDS

As discussed, snippet files include keywords to pull the values of object properties and data and write those values into the mobile output. The table below shows variable tag replacement keywords available within the system, what they are used for, and where they may be used. The target use is not necessarily limited but keep in mind that not all keywords make sense to use in all snippets. If a keyword does not have meaning for the particular object but is used in the object's snippet, the mobile output will display the keyword.

Special note on keyword processing, A keyword followed by two periods (..) indicates the end of a keyword. This syntax is required when you want to append the result of a keyword with other text without an intervening space. For example, suppose you place something like this in your snippet:

```
src ="documaker.bitmap.filename...jpg"
```

Notice in this case that although there are three periods in a row, the keyword processing stops at the second period. Since the keyword is followed by two periods in sequence, these will be stripped from the output and the remainder of the line will stay intact. So in this example if the bitmap has the filename of "alamere", the result written from snippet will be `src="alamere.jpg"`. Note that the period in the snippet output is actually the third one in the snippet definition above.

Here is another example, Suppose you wish to append an object's name content together with the suffix "MAIN".

`docmaker.nameMAIN` - results in unknown keyword results in no output.

Without a space, this looks like a single keyword and will be misinterpreted. With a space between the keyword and suffix, the space is maintained but you still do not end up with the desired results:

`docmaker.nameMAIN` - this unknown keyword results in no output, which is not the desired result.

A single period will be misintepreted as a keyword (as in `docmaker.name.MAIN`) since we allow single periods between the words in keywords. Therefore, use two periods to indicate the end of a keyword as in this example:

`docmaker.name..MAIN` results in "ObjectNameMAIN" with no spacing.

Finally, note that this double-period method is only required when you are trying to append the keyword to text that is alphanumeric and/or a period. Any other non alpha character will not be included in the keyword and would be assumed as text.

For example, `docmaker.name+MAIN` will result in "ObjectName+MAIN" as the plus-sign is not alphanumeric or a period and is considered normal text.

Keyword	Description	Default Snippet	Target Use
<code>docmaker.groups</code>	Process through each group within the formset and then when processing execute the default snippet or the named context snippet	<code>group.xml</code>	<code>master.xml</code>
<code>docmaker.forms</code>	Process through each form within the formset and then when processing execute the default snippet or the named context snippet	<code>form.xml</code>	<code>group.xml</code> <code>master.xml</code>
<code>docmaker.pages</code>	Process through each page within the formset and then when processing execute the default snippet or the named context snippet	<code>page.xml</code>	<code>form.xml</code> <code>group.xml</code> <code>master.xml</code>
<code>docmaker.sections</code>	Process through each section within the formset and then when processing execute the default snippet or the named context snippet	<code>section.xml</code>	<code>form.xml</code> <code>group.xml</code> <code>master.xml</code> <code>page.xml</code>
<code>docmaker.section.type</code>	The section type attribute indicates the type of the section being used. Returned values are either: Group Begin, Group End, Subform Begin, Subform End, Subform, or Placeholder. If no value is returned, the section listed is a standard section in the form.	<code>section.xml</code>	<code>section.xml</code> or where processing section contents
<code>docmaker.recip</code>	Output the name of the recipient for the current generation of output.	n/a	Any (usually <code>master.xml</code>)
<code>docmaker.fonts</code>	Process through each font used within the formset and then when processing execute the default snippet or the named context snippet	<code>font.xml</code>	<code>master.xml</code>
<code>docmaker.fields</code>	Process through each field within the formset and then when processing execute the default snippet or the named context snippet	<code>field.xml</code>	<code>section.xml</code> <code>page.xml</code> <code>form.xml</code> <code>group.xml</code> <code>master.xml</code>

Keyword	Description	Default Snippet	Target Use
documaker.field.type	Output the field type as an abbreviated word string.	n/a	field.xml
documaker.field.length	Output the defined field entry length.	n/a	field.xml
documaker.field.displayonly	Will return 1 if the field attribute for display only is set. Otherwise 0 is returned.	n/a	field.xml
documaker.field.required	Will return 1 if the field attribute for required is set. Otherwise 0 is returned.	n/a	field.xml
documaker.name	Used within an object level snippet, returns the name of the object	n/a	Any
documaker.description	Used with any object snippet that allows the object to define a description.	n/a	master.xml form.xml field.xml and any other object that can define a description.
documaker.descname	If the object has a description attribute, it is returned. If the description is empty or undefined, the name of the object is returned. The theory is that if you are building a list of descriptive results, you might want the item's name if it did not have any other description defined.	n/a	Any
documaker.name.parent	This will return the name of the parent that controls the object being processed by the snippet. For instance, SText elements often will not have names, so the documaker.name keyword would return no value. However since a text element generally "belongs" to a section or is defined within a text area, the documaker.name.parent keyword will return that parent name. This will make it possible to find all similar objects with the same parent name.	n/a	Any
documaker.name.section	This will return the name section that contains the object being processed. If you include this keyword at an object level above sections – like page or form – no value will be returned. If you include this keyword in a section snippet, the result is the same response as the documaker.name keyword.	n/a	Any object snippet descendant descendant to a section.
documaker.name.form	This will return the name form that contains the object being processed. If you include this keyword at an object level above forms – like group or formset – no value will be returned. If you include this keyword in a section snippet, the result is the same response as the documaker.name keyword.	n/a	Any object snippet descendant to a form.
documaker.name.group	This will return the name forms list grouping that contains the object being processed. If you include this keyword at an object level above groups – like formset – no value will be returned. If you include this keyword in a group snippet, the result is the same response as the documaker.name keyword.	n/a	Any object snippet descendant to a group.
documaker.text	Used within an object level snippet, returns the text content of the object	n/a	field.xml stext.xml note.xml, note_memo.xml, note_bookmark.xml
documaker.base64	Used within a bitmap, barcode, or vector object level snippet, returns the a base-64 encoded bitmap representation of the object	n/a	bitmap.xml barcode.xml
documaker.font.id	Used within an object level snippet, returns the font ID of the object.	n/a	font.xml Any object snippet descendant to a section that utilizes a font.
documaker.font.style	Used within an object level snippet, returns the font style name of the font associated with that object.	n/a	font.xml Any object snippet descendant to a section that utilizes a font.
documaker.font.weight	Used within an object level snippet, returns the font weight of the font associated with that object.	n/a	font.xml Any object snippet descendant to a section that utilizes a font.
documaker.bitmaps	Iterate and output each bitmap/logo on a section by utilizing the bitmap snippet.	bitmap.xml	section.xml
documaker.bitmap.filename	Output the filename associated with the bitmap object.		bitmap.xml

Keyword	Description	Default Snippet	Target Use
documaker.charts	Iterate and output each chart on a section by utilizing the chart snippet.	chart.xml	section.xml
documaker.chart.id	Returns an ID comprised of the chart name plus a sequence to make unique. The name should be compatible as an HTML/XML entity name with illegal characters changed to underscore.	n/a	chart.xml
documaker.chart.json.v1	JSON (tree) of chart attributes and data contents.	n/a	chart.xml
documaker.chart.json.v1.escaped	This will return the same data as the documaker.table.json.v1 except that HTML character entity replacements will be performed on the data. Where regular JSON data might appear as this: "Description": "My description here", The keyword will result in this output. "Description": "My description here";	n/a	chart.xml
documaker.tables	Iteration of tables calling related snippet.	table.xml	section.xml
documaker.table.json.v1	JSON (tree) of table attributes and cell contents	n/a	table.xml
documaker.table.json.v1.escaped	This will return the same data as the documaker.chart.json.v1 except that HTML character entity replacements will be performed on the data. Where regular JSON data might appear as this: "Title": "My Chart here", The keyword will result in this output. "Title": "My Chart here";	n/a	table.xml
documaker.contents	Processes through all of the objects in the object being processed by the snippet. For example, if called from the documaker.section snippet, processes through all of the objects on the section and calls the default snippet for each object accordingly.	Any descendant snippet from page or section	page.xml section.xml
documaker.stexts	Process through each independent stext or text labels within the formset (or within the particular object being processed)_ and ignore the text elements within a text area.	stext.xml	section.xml
documaker.textareas	Process through each text area and output their contents by calling the snippets associated with the children contained within. So a text area output will use the same stext snippet for text elements that was utilized for individual stext snippets.	textarea.xml	section.xml
documaker.id	Returns a unique ID/name for the object. Replaces all invalid characters with an underscore and makes each ID unique by adding an _# where # is a sequence value.	n/a	Any
documaker.version	Returns the Documaker version and build number as a string of text.	n/a	master.xml
documaker.height	Returns the height (in FAP units) of the object currently being addressed.	n/a	Any object snippet descendant to a form where the object has a height.
documaker.width	Returns the width (in FAP units) of the object currently being addressed.	n/a	Any object snippet descendant to a form where the object has a width.
documaker.color	Returns the color of the object currently being addressed. The color is expressed as a 6-digit hexadecimal value of the form #000000.	n/a	Any snippet utilized by an object that defines a color attribute.
documaker.alttext	Returns any accessibility alternate text string associated with object that can define such. The alternate text is meant to be used by reader applications for objects that might not be text objects.	n/a	Any snippet utilized by an object that defines alternate text.
documaker.skipread	Returns 1 for objects that define the accessibility attribute for skip reading. This is used by reader applications to know not to read anything related to this object.	n/a	Any snippet utilized by an object that defines the skip reading attribute. section.xml

Keyword	Description	Default Snippet	Target Use
documaker.memo.date	The date associated with the note memo	n/a	note_memo.xml, note.xml
documaker.memo.to	The recipient line associated with the note memo	n/a	note_memo.xml, note.xml
documaker.memo.from	The sender line associated with the note memo.	n/a	note_memo.xml, note.xml
documaker.memo.subject	The subject line associated with the note memo	n/a	note_memo.xml, note.xml
documaker.bookmark.level	The level number associated with the bookmark.	n/a	note_bookmark.xml, note.xml
documaker.bookmark.toc	1 or 0 to indicate if the bookmark Table of Contents property is checked.	n/a	note_bookmark.xml, note.xml
documaker.bookmark.figure	1 or 0 to indicate if the bookmark Table of Figures property is checked.	n/a	note_bookmark.xml, note.xml
documaker.bookmark.index	1 or 0 to indicate if the bookmark Index property is checked.	n/a	note_bookmark.xml, note.xml
documaker.bookmark.index.main	if the bookmark Index property is checked, this outputs the text associated with the main subject property.	n/a	note_bookmark.xml, note.xml
documaker.bookmark.index.subentry	if the bookmark Index property is checked, this outputs the text associated with the subentry property	n/a	note_bookmark.xml, note.xml
documaker.link.name	Contains the hyperlink name (used by Target and hyperlinks).	n/a	bitmap.xml, stext.xml, field.xml
documaker.link.type	Contains a number representing the type of hyperlink (1 = target, 2 = external, 4 = internal).	n/a	bitmap.xml, stext.xml, field.xml
documaker.link.options	Contains a number representing the border style (0 = none, 1 = solid box, 2 = dashed box, 3 = underline).	n/a	bitmap.xml, stext.xml, field.xml
documaker.link.href	This will reflect the URL of the link. Possible values: An absolute URL - points to another web site (like href="http://www.example.com/default.htm") A relative URL - points to a file within a web site (like href="default.htm") Link to an element with a specified id within the page (like href="#top") Other protocols (like https://, ftp://, mailto:, file:, etc.) A script (like href="javascript:alert('Hello');")	n/a	bitmap.xml, stext.xml, field.xml
documaker.link.params	Use this field when producing HTML output to specify additional parameters to an external (HREF) type link, such as a target frame or mouse-over behavior. For example: target="new"	n/a	bitmap.xml, stext.xml, field.xml
documaker.form.metadata	Process through user attribute collection within form and then when processing execute the default snippet or the named context snippet.	n/a	form.xml
documaker.metadata.key	This will return the key from user attribute collection element.	metadata_form.xml	form.xml group.xml master.xml
documaker.metadata.value	This will return the value from user attribute collection element.	metadata_form.xml	form.xml group.xml master.xml
documaker.readonce	1 indicates the section has been set for Read Once 0 indicates the section has not been set for Read Once	n/a	section.xml

CREATING A CASCADING STYLE SHEET

The look and feel of mobile output is critical to your customer's satisfaction. In many cases, great efforts have been put into designing the look and feel of printed output as well. In the event that you want to use font information from the printed format to guide the look of the mobile format, Documaker Studio provides an option to create a CSS from the fonts within the resource library. To produce the CSS, open the workspace as a user with permission to access Font Manager. Open the Fonts Manager and select Export CSS File from the Fonts, Export menu. From here you can select the possible fonts to reference, the name of the file to be created and the type of entity, in this case CSS, to export and click Next. On the Export - Step 2 of 2 dialog, click Select All, then Finish to complete the export. The resulting file can then be manipulated as needed in tools for managing HTML and CSS and distributed for use with mobile output.

CHOOSING THE RIGHT MOBILE OUTPUT FORMAT

Oracle Documaker Mobile provides default snippets in an XML format. However the final targeted delivery format is often HTML. The XML, which serves to capture raw data content from a Documaker document transaction, is then often transformed to HTML through industry standard XSL. The XSL provides the rules for transforming the raw content XML into device responsive HTML5. The snippet output format for your implementation may be XML, HTML or may be another format based on your company's needs. However, when choosing between output formats from your snippets, consider the following points below.

Output Format Considerations

The default Snippets produce an XML output format that is transformed to HTML via EXtensible Stylesheet Language (XSL). However, Snippets can be composed in any UTF8 based text script including HTML, HTML5 and Comma Separated Values (CSV), etc. To create a decision point for the team to determine what snippet output format is to be used, your company may have standards and requirements for the targeted formats. Evaluate other factors like:

1. Skill set of the team
2. Requirements of the delivery and the end users including:
 - Intended supported browsers
 - Intended devices or platforms
 - Speed and responsiveness
3. Ability of the output format to deliver the intended customer experience

A popular choice for mobile output is HTML5. Mobile output rendered in HTML5 from Documaker can leverage all parts of the HTML5 standards including Cascading Style Sheets (CSS/CSS3), JavaScript (and JS based frameworks such as jQuery), AJAX, Web Fonts and all other HTML/HTML5 based features. This allows you to design a mobile experience using all of the latest industry standard capabilities.

While selecting the format of snippet contents, consider the following when choosing between producing XML that must be transformed (client or server side) when compared with producing HTML directly from Documaker Mobile.

Option 1: XML Output	Option 2: HTML Output
Payload depends on location of transformation. If transformation occurs on the server, the ultimate payload may be small	The client may have a large payload
Follows industry XSLT standards to transform output to responsive HTML/JavaScript	Raw output from Documaker may be usable in the browser, does not require further transformation.
Tools such as Dreamweaver is capable of understanding XSLT	

Chapter 3

Delivering Mobile Output

This chapter provides detailed information on the mobile delivering process.

This chapter discusses the following topics:

- *Mobile Output Process* on page 30
- *Mobile Output Transformation* on page 34
- *Mobile Output Management Workflow* on page 35

MOBILE OUTPUT PROCESS

Mobile output generation, like all other output formats, may occur on demand or in a batch process. Part of the document automation team's role is to work with a cross-functional team to determine how your company will make mobile output from Documaker available to your customers. We will explore possible work flows for mobile output management for both on demand document access and generation as well as batch creation for access.

Regardless of the delivery methodology, secure access to mobile content must be established through your existing customer access systems. Generally this means that the customer would login to your company's web portal to gain access to mobile content. The web portal is therefore responsible for managing client identification and client profile information including the customer's content delivery preferences and authorization for web content delivery. If your company does not already have a web portal strategy in place Oracle has a WebCenter product that can be considered.

The scenarios provided below offer possible options for mobile delivery that may suit the needs of your requirements. These scenarios cover the basic flow of request, processing, and presentation; however, there may be additional considerations such as transformation or filtering of the generated content.

Scenario 1 - Real Time Generation and Display

In this scenario the system creates mobile output and returns it to the user at the time of request. In this case the output is created at the point in time when the customer wants access to the output.

1. Customer requests access to a mobile document via portal or online application
2. The Portal or application request with the available data is routed to Documaker
3. Documaker returns the personalized content to targeted location and provides feedback to calling application
4. Portal or application streams the content to the customer

This methodology is suitable for situations where a defined set of customer data has already been processed in a prior Documaker request. For example, your system generates monthly billing statements that have historically been mailed to the customer. The customer typically receives a monthly statement in the mail and has not signed up for electronic delivery. However, they now are online, logged in to your company portal, and wish to view last month's statement electronically. In a traditional PC browser they could view a generated PDF (assuming one was created when the paper statement was generated); however, you wish to provide them with a richer mobile experience on their mobile device. To support this effort, the company portal can provide an option that will allow the user to view their prior month's statement in mobile form. The web portal must be able to identify the transaction identifier for the content to be delivered to the end user. The portal would be responsible for sending a web service request with that identifier to Documaker's `doPublishFromFactory` web service method and for processing the response. Using values returned in the `doPublishFromFactory` and `doGetPublishingInfo` web services response, the portal would then call `doGetPublishingInfo` using the Heavy setting to return the mobile output in the response so that it can be streamed back to the user's device. For more information on configuring the `doPublishFromImport` and `doGetPublishingInfo` web services to respond with the needed output, see the [Documaker Enterprise Edition Administration's Guide](#) topic "Using Documaker Web Services".

Note that this method assumes the content can be rendered independently of other files or that the files are contained/referenced elsewhere. The `doPublishFromImport` response only contains the mobile output contents.

Scenario 2 - Repurpose and Display

In this scenario the system creates mobile output from previously processed data and returns it to the user at the time requested. In this case the data necessary for the output was used in advance of the customer's request for generation and delivery in a non-mobile format. The prior request is repurposed to create the mobile output.

1. Back office system routes a request with data to Documaker
2. Documaker generates and delivers content
 - a. Customer locates prior transaction/activity through the portal or application and requests access to a mobile document
 - b. System locates prior request and requests mobile output
 - c. Documaker uses the prior request to generate personalized content
 - d. System streams the content to the customer

This methodology is suitable for situations where a defined set of customer data has already been processed in a prior Documaker request. For example, your system generates monthly billing statements that have historically been mailed to the customer. The customer typically receives a monthly statement in the mail and has not signed up for electronic delivery. However, they now are online, logged in to your company portal, and wish to view last month's statement electronically. In a traditional PC browser they could view a generated PDF (assuming one was created when the paper statement was generated); however, you wish to provide them with a richer mobile experience on their mobile device. To support this effort, the company portal can provide an option that will allow the user to view their prior month's statement in mobile form. The web portal must be able to identify the transaction identifier for the content to be delivered to the end user. The portal would be responsible for sending a web service request with that identifier to Documaker's `doPublishFromFactory` web service method and for processing the response. Using values returned in the `doPublishFromFactory` and `doGetPublishingInfo` web services response, the portal would then call `doGetPublishingInfo` using the heavy setting to return the mobile output in the response so that it can be streamed back to the user's device. For more information on configuring the `doPublishFromFactory` web service to respond with the needed output, see the [Documaker Enterprise Edition Administration's Guide](#) topic "Documaker Web Services".

This methodology is suitable for situations where you will not generate mobile output in advance but have created the output in a traditional format. In this scenario mobile output is generated based on a request, either on demand or scheduled batch and the output is retained within the Documaker processing tables. Documaker only retains prior requests for a temporary period established by your corporate guidelines and configured with the Historian tasks. The application that generated the request must know the correct index information to locate the content, allowing the portal to provide a direct link to the staged content and the portal is also responsible for managing access to that link. While it is always possible to reference an invalid link in other Use Cases noted, it is important to consider accessing the links back to the Documaker processing tables during temporary storage.

Scenario 3 - Archive and Display

In this scenario the system creates mobile output during a scheduled process, archives it in a repository, and returns it to the user at the time requested. In this case the data necessary for the output was used in advance of the customer's request display to generate the content and stage it for future display.

1. Back office system routes a request with data to Documaker
2. Documaker generates and stores the mobile content into a repository (e.g. WebCenter Content)
 - a. Customer requests access to a mobile document
 - b. System locates stored content in the repository
 - c. System streams the content to the customer

This methodology is suitable for situations where a defined set of customer data is to be processed in advance of its access and use. For example, you may wish to prepare periodic statements for customers that have opted-in to electronic delivery. A customer's monthly banking statement for example where the data is generated on a scheduled basis, processed by Documaker, and stored in a content management system for later access. In this case, a customer may receive a notification that their statement is available (see scenario 4) or happens to log in to your customer portal and can access the content there. In this case the customer selects a link presented in the portal which ties back to the content repository. The content repository returns the content and the web portal would then be responsible for transformation (if needed) and presentation to the customer.

In this scenario Documaker is not invoked at the time of delivery. This scenario differs from scenario 2 in that the mobile content is pre-created and stored in a repository for eventual delivery.

Scenario 4 - Mail Notification and Delivery

In this scenario the system creates mobile output during a scheduled or on demand request and delivers it via email. In this case the output is pushed to the user.

1. Back office system routes a request with data to Documaker
2. Documaker generates the mobile content
3. Documaker attaches the content to an email and delivers it to the customer

This methodology is suitable for situations where the mobile content should be delivered via email and is generated while the user is not waiting for the request. For example - a personalized notification to complete forms online for the customer (patient's) upcoming appointment. On a scheduled basis, content from the originating application would be sent to Documaker, Documaker would process the data to create mobile output and then generate an email with that content to send to the intended recipient. In this use case, it is important that sensitive information be sent in a secure fashion or not sent at all. With this scenario, it is also important to keep in mind the limitations of mail clients to view certain file types and perform certain transformations. As always, be sure to validate that your mobile output displays appropriate for your intended mobile devices.

Additional Considerations

The Scenarios "Repurpose and Display" and "Archive and Display", require the calling application to have some identifying information linking the previously generated data or previously processed output. There are two possible ways for the calling application to have this information:

- a. Documaker returns a unique identifier to the calling application when the original request to process the data is made.
- b. The calling application provides a unique identifier when originally requesting the content creation from Documaker.

In the On Demand scenarios listed above, the mobile output is provided back to the web portal via the calling web service. While this is a convenient method for providing access to the content, it is possible to generate the content on demand, have Documaker store the content in a content repository and provide the locating details back to the calling application. None of the methodologies presented dictate where the mobile content ultimately resides. Where you want the content to reside should guide how the mobile content is accessed. Additionally, the two concepts do not need to be related. Your company may choose to provide mobile output back to the web portal within the Documaker Web Service response but to also store the output in WebCenter Content repository for subsequent access and retention considerations. The decisions on where and for how long to retain the output in this format depend on your company's requirements to retain and reuse the output.

Mobile supporting files

Mobile output produced by Documaker may be self-contained but it's likely that your implementation will have external resources needed for Mobile output display. Items such as bitmaps and JavaScripts can in that case be versioned and stored separately from the Documaker Mobile content. Therefore your infrastructure will need to include a web server or provide sufficient access to this static content. The Documaker Mobile output generated must be able to properly reference to the static content in the selected location.

Mobile Output Transformation

Depending upon the output format created by Documaker Mobile (XML, HTML5, etc.) transformation may be needed to produce the final presentation composition of a mobile document.

The mobile output format generated from Documaker Mobile is determined by your company and the needs of the content being presented. The "Mobile Output Format" topic reviews some considerations for the format of mobile output. If XML is selected as the target output format for mobile content from your snippets, you will need to determine where the resulting XML for mobile will be transformed to HTML for presentation in a browser. The transformation may occur at one of many points within the mobile document production workflow, including on your web server or in the browser of mobile document recipients. Here are some things to consider when selecting either server side or client side transformation from XML to HTML.

Considerations for each Server Side and Client Side transformation

Below are some items to consider when choosing between client side and server side transformation activities within the mobile delivery workflow.

- Server Side provides consistent transformation by running all mobile documents through the same transformation engine on the server.
- Server Side, may reduce the size of the payload delivered from your Web server to the browser of your mobile document recipients when compared with client side transformation.

- Server Side, offers several options for where the transformation can be implemented in a typical mobile delivery workflow:
 - For example it may be executed on the XML payload delivered from Documaker Web services (DWS) calls before the results are provided to your Web Portal or Web Content Management System.
 - It can be executed in your Web Portal when retrieving content from the Content Management System feeding your Portal, or it may possibly be executed in your Web Portal just before delivering content through the Portal to your HTTP server that will feed the content to the browsers of mobile document recipients.
- Server Side uses resources on your Web servers to perform the XML to HTML transformation.
- Server Side becomes a step in your mobile delivery work flow implementation that must be implemented by your technical team.
- Client Side does not consume resources on your Web servers to perform the transformation. i.e. Moves the CPU load required to perform XML to HTML transformation from your Web servers to the browsers of your mobile document recipients.
- Client Side may increase the size of the payload delivered from the Web server to the browser of mobile document recipients when compared with server side transformation.
- Client Side produces inconsistent transformation as different browsers implement transformation engines uniquely on each client. This can be a particular challenge with IE browsers where some features of the W3C specification for handling client side XML transformation appear to not be completely supported (specifically support appears to be missing for embedded style sheets in an XML document through IE11 as of the writing of this document).

MOBILE OUTPUT MANAGEMENT WORKFLOW

We have reviewed a number of possible use cases for managing mobile output. The following presents the possible workflows from the end user experience:

1. Customer signs on to the company portal.
 - a. This could be triggered by a notification that a document is available.
 - b. This notification can be sent by or triggered by Oracle Documaker and can occur when the mobile output is generated or when the data is available to generate the mobile output
2. Portal authenticates customer and presents customer with options to view available mobile documents
3. Customer selects a mobile document, or activity if the mobile document is not yet created, for viewing
4. The portal can then do one of three things:

- a.** If the mobile output is already generated and stored in the content repository serving the portal, the portal can display the mobile output from its own repository.
- b.** If the mobile output is already generated and still available in Documaker's Assembly Line tables the portal can use Documaker Web Services.
- c.** If the mobile output is not generated, the portal can use Documaker Web Services to generate the mobile output on demand.

Chapter 4

Configuring and Enabling Mobile

This chapter provides detailed information on configuring, installation and validating installation of Oracle Documaker Mobile.

This chapter discusses the following topics:

- *Configuring Mobile for Oracle Documaker Enterprise Edition* on page 38
- *Validating Documaker Mobile Installation* on page 40
- *Configuring Mobile for Oracle Documaker Standard Edition* on page 42
- *Validating Documaker Mobile Installation* on page 43

CONFIGURING MOBILE FOR ORACLE DOCUMAKER ENTERPRISE EDITION

Oracle Documaker Mobile is enabled in Oracle Documaker Enterprise Edition (ODEE) using the Documaker Administrator. Before configuring Oracle Documaker Mobile within ODEE, you should first do the following:

1. Install Oracle Documaker Mobile on the Oracle Documaker Enterprise Edition application server according to the steps within the Oracle Documaker Mobile Installation Guide.
2. Ensure that the default and custom snippet files along with the needed mobile supporting files - adapts, javascripts, etc. - are located in the *mrolib* location defined for your system.
3. To process the policy, quote and financial statement documents into mobile ready, user friendly content you are recommended to use the sample snippets and supporting files available through [Oracle Technology Network \(OTN\)](#). The default snippets within the ODEE installation will only provide the contents that are useful for data exchange. The sample files are located on OTN with the Help, Tutorials, and Samples pages for the current version of Documaker. Download these samples and extract the files to the *mrolib* location defined for your system.
4. Ensure that the mobile supporting files - jpps, javascripts, etc. - are located wherever they may be referenced based on the selected delivery methods to be used in your implementation. For example, if you will be sending content to WebCenter and your users will be accessing mobile documents stored within WebCenter, ensure that the files needed to render the mobile output are available in the referenced location in WebCenter.
5. Identify any notification content needed to support the selected delivery methods to be used in your implementation. If you plan to generate notifications from the Oracle Documaker Enterprise Edition, collect the required language for the notification.
6. Identify the conditions for producing Mobile output for a given transaction. This aspect is an extension of the selected delivery methods used for the implementation. For example, if you are generating Mobile output for users who have opted for a 'paperless' initiative, determine the identifier for this process within the incoming data feed. In the samples provided with Oracle Documaker Mobile, mobile output is generated based on the 'SMS' distribution value. Ensure that this condition is routed to a targeted batch for processing.

To enable a batch to generate mobile output, navigate to the batch definition to be mobile enabled. Select the MRO output format for that batch. If Oracle Documaker Mobile has not been installed on the ODEE server, the mobile option will be enabled, but mobile output will not be generated.

Oracle Documaker Enterprise Edition contains a script that determines how Oracle Documaker Mobile output will be delivered. Use the Oracle Documaker Administrator's Archive destinations to configure where the mobile output should be delivered. The possible options to include the archive destinations are defined in the 'Configuring the Archiver' Section of the Oracle Documaker Enterprise Edition Administrator's Guide. You can also choose your custom destination to create a Mobile batch, configure the batch for File system output, and establish a notification message to alert the recipient of the mobile output.

This script, *dmkr_asline_mobile.sql*, is stored in the */documaker/database/databasetype/mobile* directory. You can run this script with appropriate permissions to get an example established within your system. The specific script is stored by database type in the following locations based on the database selected during installation:

Database Type	Location
Oracle	odee_1/documaker/database/oracle11g/mobile
DB2	odee_1/documaker/database/db2v97/mobile
SQL Server	odee_1/documaker/database/sqlserver2012/mobile

Note: When running the SQL *dmkr_asline_mobile.sql* for SQL Server that the file *genericnotificationmessage.txt* has to be in a location accessible by the SQL Server Database server and referenced as such in the *dmkr_asline_mobile.sql* BULK command so that the SQL Server run successfully. See note in *dmkr_asline_mobile.sql*

Alternatively, you may use the script and notification content as a template to modify and deploy the mobile output. If you don't want to run the script, you may also choose to not to run the script. In such case, you can simply use the Administrator tool to define the batch and the delivered configuration for the implementation.

The base installer for Oracle Documaker Enterprise Edition creates the default output folder for the MRO Batch's FileSystem Archiver output to be *ODEE factory root installer + .\filesystem-archive*

For example, if you install in *c:\oracle\odee_1*, the Mobile created Batch will get the output location as *c:\oracle\odee_1\documaker\filesystem-archive*

The batch, the output location, and the notification content are provided only as samples and are expected to be modified or used solely as a reference for your implementation. You should choose a location and a file destination output which meets your delivery needs.

The Oracle Documaker Enterprise Edition installation comes with a reference implementation. The reference implementation has sample files for processing that may be used to demonstrate mobile delivery post installation of Oracle Documaker Mobile. The implementation also includes the settings needed for Mobile processing; be sure to copy over the default snippet and resource files to your *mrolib* location as instructed in the [Oracle Documaker Mobile Installation Guide](#).

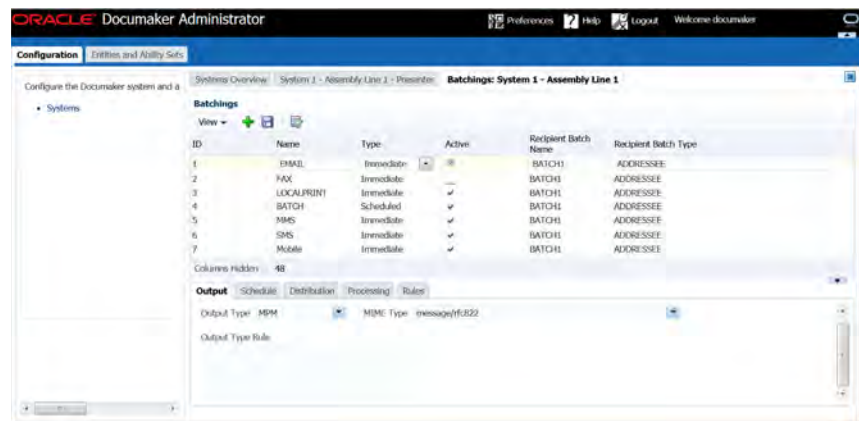
If your delivery scenario calls for a notification message, you must define the notification message using a database tool to access the PUBNTFS table. For more information on the contents of this table, see the [Oracle Documaker Enterprise Edition Administration's Guide](#).

VALIDATING DOCUMAKER MOBILE INSTALLATION

Once you have installed the ODEE and Oracle Documaker Mobile setups and wish to validate your installation, follow the steps below:

To Enable Mobile output, use the Documaker Administrator web application to configure the output type, destination, and handling.

1. Login to Documaker Administrator and enable the Mobile Batchings. To know more about the Mobile Batchings see the 'Understand Batches' section in ODEE Admin Guide.



2. In Mobile Batchings, activate the Archive Distribution and activate the Notifications property.
3. In the left panel, click *Systems* link. and expand *System node* and *Assembly Line node* in the right panel.
4. Select the row containing the newly-installed assembly line for Mobile, select *Archiver*, and click *Configure*.
5. From the Archiver tab, select *Filesystem-Mobile* category within DESTINATION context name, and select Mappings.
6. Create an *mrolib* directory in your workspace. Copy the files and resources folder from *Oracle\odee_1\mobile_install\mstrres\mobile* to the following location:
`C:\Oracle\Middleware\user_projects\domain\idocumaker_domain\servers\idm_server\stage\WipEdit\WipEdit\mrolib`
7. Select the *destination.file.name.pattern* property and edit the value to point to *C:\Oracle\Middleware\user_projects\domain\idocumaker_domain\servers\idm_server\stage\WipEdit\WipEdit\mrolib* directory.

This enables the mobile output.

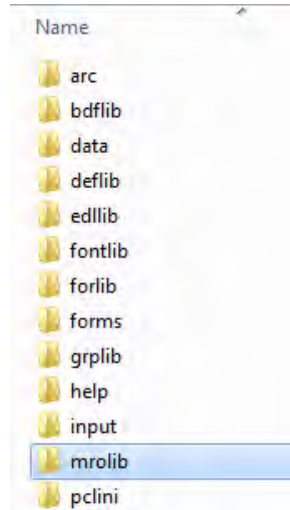
To validate the Documaker Mobile Installation

1. Back up the *default.xml* and *default.json* files located in *odee_1\documaker\mstrres\dmres\input* folder.
2. Drop the *default.xml* and *default.json* files in the hotfolder directory on the Document Factory server.

CONFIGURING MOBILE FOR ORACLE DOCUMAKER STANDARD EDITION

To configure the Oracle Documaker Mobile within ODSE, you should first do the following:

1. Install Oracle Documaker Mobile on the Oracle Documaker Standard Edition application according to the steps within the Oracle Documaker Mobile Installation Guide.
2. Create an *mrolib* directory in your workspace, for example:



3. Copy the files and resources folder from *Oracle\odee_1\mobile_install\mstrres\mobile* to the *mrolib* folder in your workspace. Make sure that the default and custom snippet files along with the needed mobile supporting files - adapts, javascripts, etc. - should be located in the *mrolib* location defined in your system.
4. Open the workspaces *fsisys.ini* file using any text editor. Add `PrtType = MRO` to the `<Printers>` group, for example:


```
< Printers >
    PrtType           = PCL
    PrtType           = MRO
```
5. Add `<PrtType: MRO>` group and settings:


```
< PrtType:MRO >
    Class             = MRO
    Device            = mrolib\default.xml
    Module            = MROLIB
    PrintFunc         = MROPrint
    TermFunc          = MROTerm
    SnippetExt        = .xml
```
6. Save your *fsisys.ini* file.
7. Ensure that the mobile supporting files - jpps, javascripts, etc. - are located wherever they may be referenced based on the selected delivery methods to be used in your implementation.

VALIDATING DOCUMAKER MOBILE INSTALLATION

Perform the following steps to validate an Oracle Documaker Standard Edition and Oracle Documaker Mobile installation:

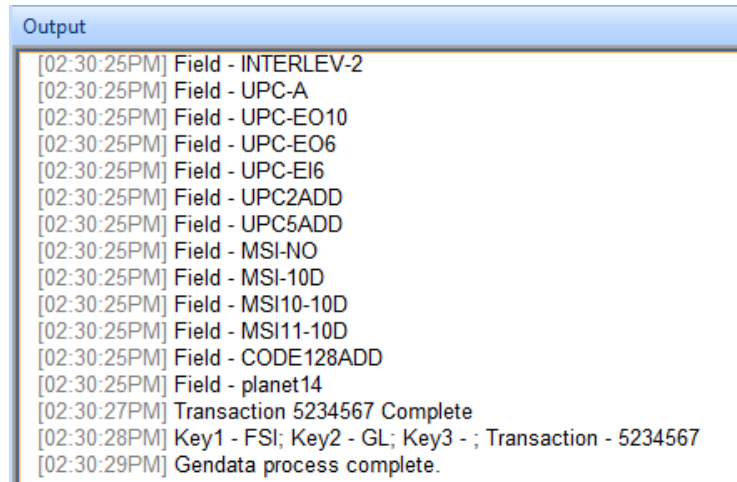
1. Open the workspace in Studio, then open Test Manager.

Note If you do not already have a test profile, you will be prompted to create one. Run the test profile.

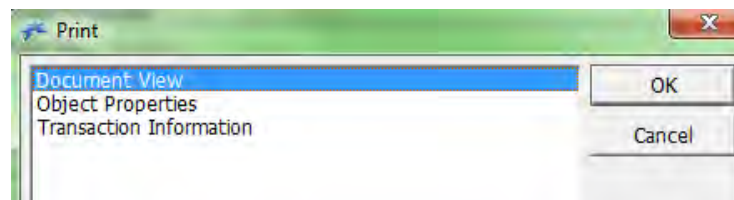
2. Click *Run* button to start the test run.



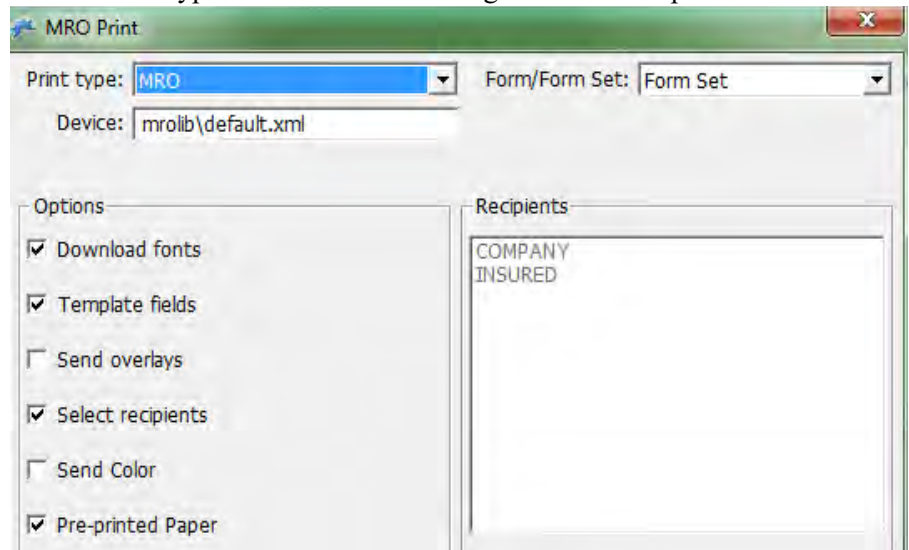
3. Click *Step* button several times to process further until *Gendata Process Complete* status appears in the output area.



4. When the transaction completes, select *File, Print, Document View*.



5. Set the Print type *MRO* and click *Ok* to generate the output file.



To validate the Documaker Mobile Installation

Go to *mrolib* folder of your MRL and open the *default.xml* file to view the MRO.

Chapter 5

Best Practices

This identifies the implementation considerations and best practice approaches to guide you through the options available with Documaker Mobile.

1. Secure a Mobile Author for the Documaker team from the outset.
2. Emphasize usability in the layout of the mobile experience. We recommend a mobile (phone) first approach to layout design.
3. Define mobile content divisions for navigation of mobile “pages”.
4. Define mobile content interactions such as drill in or field level inter activity.
5. Map page based content to the mobile presentation (we recommend via XSLT) and supply relevant Names to Documaker publishing objects in your MRL as required to map page content to the mobile presentation.
6. Ensure that all needed infrastructure is available such as WebCenter, iOS and Android Emulator as well as access to real physical mobile devices (one each for Smart Phone, Tablet covering iOS and Android).
7. We recommend XML be used for defining snippets contents and XSLT be used to transform the XML to HTML for a mobile presentation, but this is not a requirement.
8. Identify the approach your company will use for delivery of mobile content and engage other resources in your company as needed as early as possible in your mobile document development process. Delivery of mobile content from Documaker to your customers will usually require your company’s web technical team to create calls to Documaker’s Web Services (DWS) interfaces. Our recommendation is to use something like a Web Portal to control customer access to documents from Documaker and to manage content referenced by mobile XML/HTML documents in particular from Documaker.

Chapter 6

Troubleshooting

This chapter provides troubleshooting tips for issues you might encounter during the Mobile Implementation.

1. I have installed Documaker Mobile, but cannot access the Mobile options when using the Documaker Add-In for Microsoft Word

After you have installed Documaker Mobile, you will need to re-export your Workspace Definition. See the topic "GENERATING A WORKSPACE DEFINITION FILE" in the Documaker Studio user guide for details on exporting this file.

2. Cannot access Mobile options Context Tag or Omit in Documaker Studio either at the resource level, for example, Application Definition, Sections, Forms or the object level, for example Charts, Fields, Text Areas, etc.)

Mobile options requires Documaker Mobile. If you have not purchased and installed Documaker Mobile the required mobile options will be disabled. To purchase Documaker Mobile, visit My Oracle Support website or contact your Oracle Sales representative.

3. Cannot define a new context tag in a resource, such as a Section or a Field

Check the user right "Limit to pre-defined context tags". When this option is turned on, the user is limited to selecting pre-defined context tags specified in the Application Definition file.

4. Is there a way to write snippet contents to the trace file at runtime?

The Debug_Switches INI group contains the option MRO_Debug which when active, includes snippet information in the trace file.

To activate, set both Enable_Debug_Options and MRO_Debug to Yes

For example,

- Opening C:\FAP\mstrres\enterprise\MROLIB\master_financials.xml *
UTLDefWarnNotify * PrtType:MRO: Unable to open file
C:\FAP\mstrres\enterprise\MROLIB\master_financials.xml

Unable to open file C:\FAP\mstrres\enterprise\MROLIB\master_financials.xml

Opening C:\FAP\mstrres\enterprise\MROLIB\master.xml

-
- Opening C:\FAP\mstrres\enterprise\MROLIB\group_financials.xml *
UTLDefWarnNotify * PrtType:MRO: Unable to open file
C:\FAP\mstrres\enterprise\MROLIB\group_financials.xml

Unable to open file C:\FAP\mstrres\enterprise\MROLIB\group_financials.xml

Opening C:\FAP\mstrres\enterprise\MROLIB\group.xml

Form name:FS_COVER
 - Opening C:\FAP\mstrresenterprise\MROLIB\form_financials.xml *
UTLDefWarnNotify * PrtType:MRO: Unable to open file
C:\FAP\mstrres\enterprise\MROLIB\form_financials.xml

Note: Be sure to turn debugging off by setting Enable_Debug_Switches to No once you no longer need to log MRO or other debug information to the trace file.

5. I am getting results that look fine but a field (or other) value is missing which I expected from the Documaker output.

Generate Mobile output using the Documaker Mobile default snippets, then check for that value in the output. Using the Documaker Mobile default snippets provides a clean way to analyze the output from Documaker.

6. I would like to omit groups from my mobile output. How can I accomplish this?

To omit groups from mobile output, you would change the master snippet to use documaker.forms instead of documaker.groups

For example,

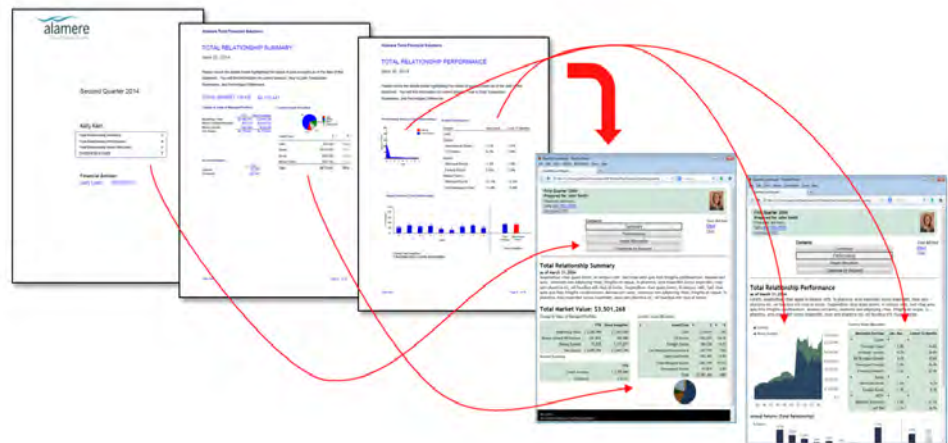
```
<?xml version="1.0" encoding="UTF-8" ?>
<DOCSET NAME="documaker.name" DESC="documaker.description">
  <RECIPIENT NAME="documaker.recip"/>
  <FONTS>
    documaker.fonts
  </FONTS>
  documaker.forms
</DOCSET>
```

Appendix A

Layout and Design

Whether you are developing new documents for use with 'paper' and mobile presentation or mobilizing existing documents for mobile delivery, user experience should be at the forefront of the effort. Work with a Mobile Author that has experience in SPA based interactive web pages to create a prototype of the targeted mobile experience of the document and then match that to the paper based layout. Keep in mind that not all content in the paper layout needs to be included in the mobile content and vice versa.

An example paper to mobile project is shown here:



For the interactive SPA based web page:

- Layout for mobile
 - Columns or divisions of areas where content is displayed
 - Styling for content (fonts, colors)
 - Content layout changes for various devices such as full size browsers running on desktops or laptop, Tablet devices such as iPad and Android based devices, and Phones such as iPhone and Android based phones.
- Navigation

- Content segmentation for navigation (pages, sections, other divisions that make sense for the content)
- Interactive controls used for navigation (buttons, tabs, index sliders, etc)
- Content based interactive navigation (table or chart drill in, content hyperlinks)
- Content data based interactions
 - Table sorting

Data interactions such as “what if” type controls (data entry elements, sliders, other controls)

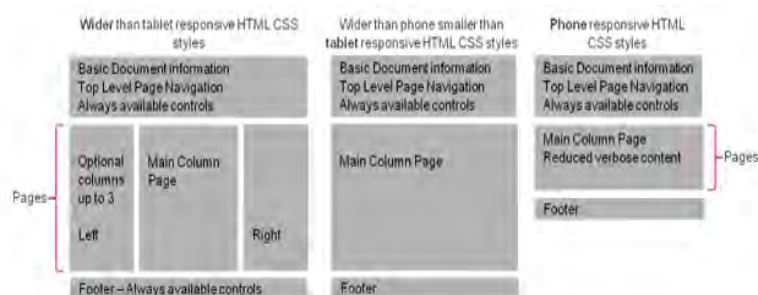
LAYOUT EXAMPLE

Let us consider the following pages to design an implementation of a financial statement document.

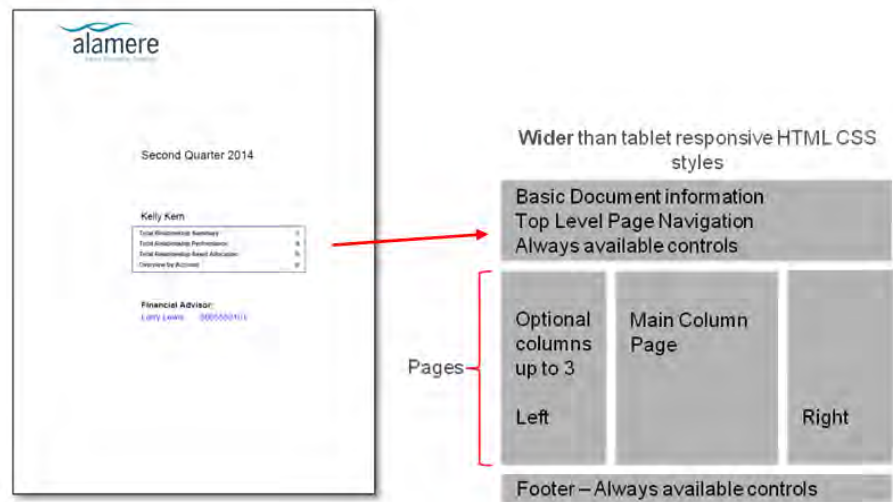


MOCKUP DESIGN FOR MOBILE

Mobile Author generates an initial mockup design “HTML Wireframe” layout for mobile. See, below:

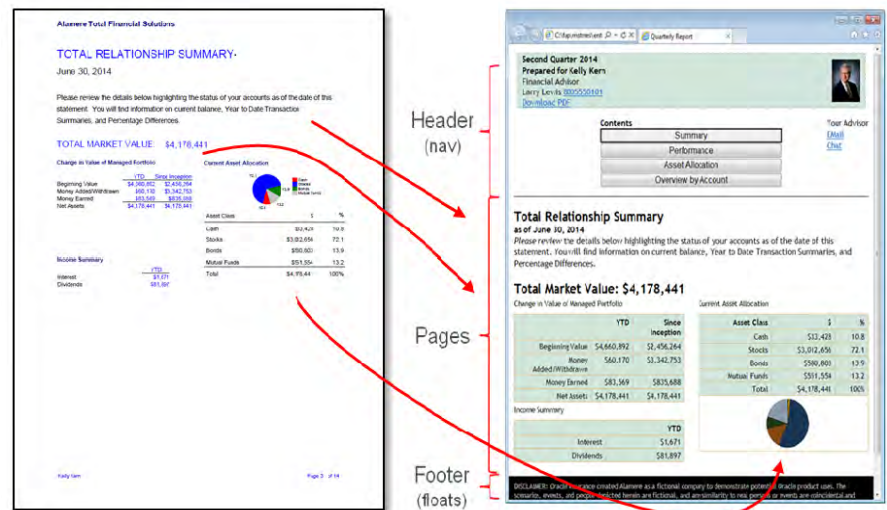


Based on the initial mockup mobile design and the contents of the example page based document, the first page has a simple table of contents as shown below:



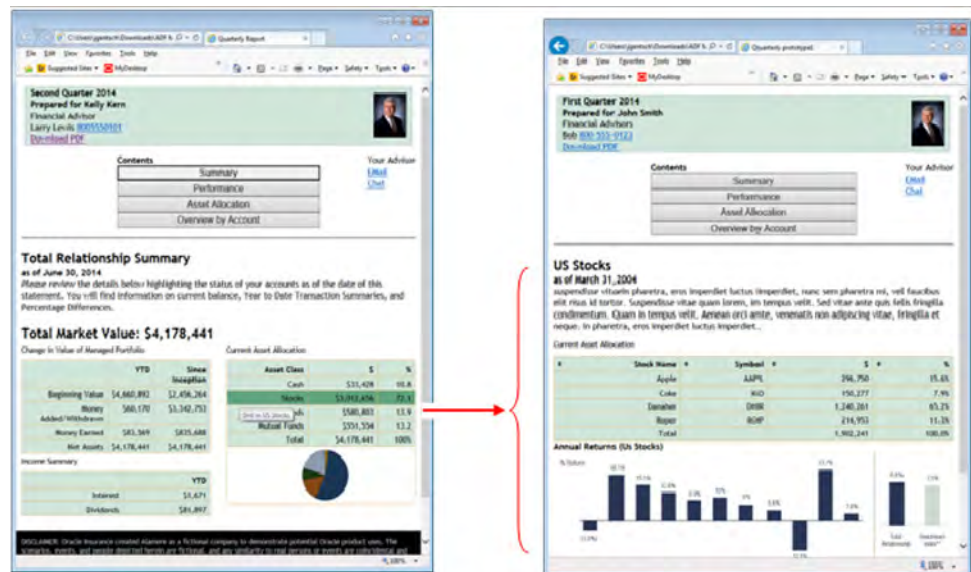
NAVIGATION

Once you decide the basic division of content that should appear in the mobile view (e.g. mobile content pages), you can begin mapping the content from the page based document to the mobile layout:



CONTENT DATA BASED INTERACTIONS

Once the content is generally mapped to the mobile layout, you can begin looking at content interactions in the mobile view beyond just moving between pages, such as content drill down. For example, in the financial statement, the Relationship Summary table holds rows of totals from other content pages. These rows become a point of navigation to be able to jump from the table row with the totals to the mobile page with the details for that table row as shown in the following figure:



In the example above, the "drill in" navigation is accomplished with styling on the table through Cascading Style Sheets (CSS) and actual navigation triggered through JavaScript. The table row will be highlighted in dark green when the user hovers their mouse over the row. This green highlighting provides a visual clue of drill in support along with a corresponding drill in tooltip.

A wide range of content interactions are possible through HTML5 based mobile content output including controls on the mobile web page that can update embedded data and re-compute values providing mild online analytical processing (OLAP) results.

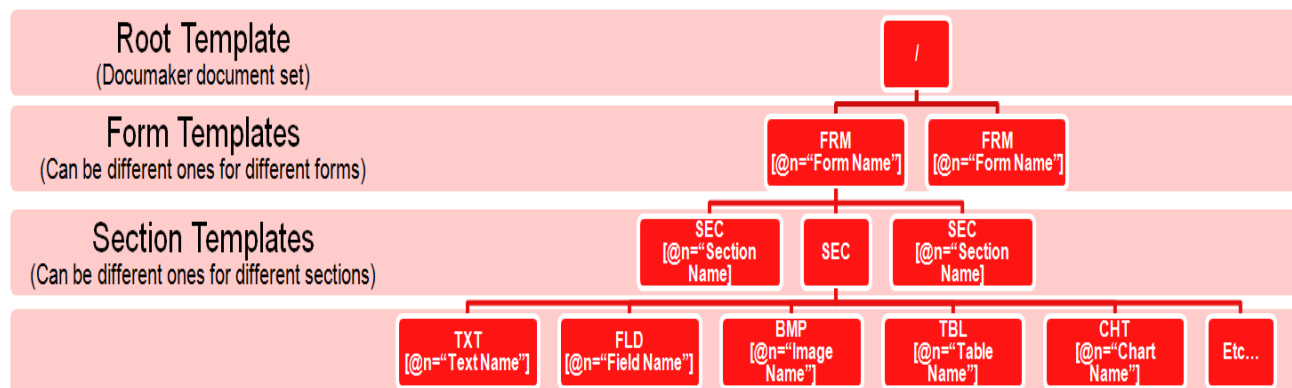
In the figure below, the death benefit amount is updated and the estimated policy premium values are recomputed as the slider bar under Desired Death Benefit text is moved from side to side.



XSLT TECHNIQUES

XSLT supports a recursive processing model to break up the transformation process into blocks of instructions called Templates, where a top most Template is triggered when the XML document root is found in the XML being transformed. That Template can in turn then trigger other more specialized Templates to process child XML elements it contains.

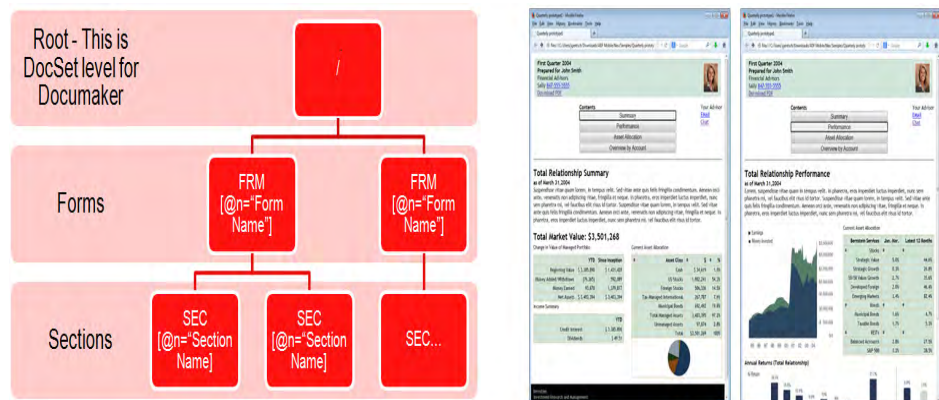
An XSLT Template can be thought of as a subroutine setup to handle the XML transformation which matches an XPATH selection expression defined for the Template. Here is a conceptual example of an organization for an XSL style sheet with layers of Templates for processing output from the default snippets shipping with Documaker:



In the example above, the XSL Template associated with the XML root (“/”) first performs any processing desired at the Documaker document set level and then calls more specific XSL Templates to process the child contents of the XML root element via an apply-templates statement. Those Templates then in turn do the same thereby breaking up the XSL transformation process into smaller and smaller specialized chunks of transformation instructions where the transformation details for each chunk are defined in the lower level Templates.

In the example above, the lower level Templates define an XSL match definition (which is an XPATH expression) that, in this example, also contain XPATH predicate syntax to filter XML elements eligible to be processed by the Template via the value of an “N” attribute. The “N” attribute is populated by the default snippets from Documaker with the name of their publishing object. This capability enables specific XSL Templates to then be defined for processing publishing objects by both object type (e.g. Form, Section, etc.) and by the name of the publishing object (e.g. Form name or Section name that was defined in the Documaker Studio for the MRL).

In the example below, the lowest level section Templates are used to produce mobile aware HTML for specific document sections found in the XML formed by the default snippets.



This mobile aware HTML could have been authored in external tools and then pasted into the XSL Template syntax. The HTML syntax in the XSL Template can be further updated with other XSL syntax such as `<xsl:value-of select='FLD[@N='SomeFieldName']' />` which will cause values from the XML being transformed to be inserted directly into the resulting HTML output that forms the mobile presentation end users will see in their browser.

There are also times when a formset will contain repeating data for transformation. The XSL sheet that process the XML to HTML can use techniques such as XSL Templates that are described above to recursively process repeating data, or it can use syntax such as `<xsl:for-each select="myExpresion">` in Templates to process repeating data. XSL supports syntax that groups duplicates of repeating data so only unique occurrences are processed such as `<xsl:key name="Grouped-Forms" match="FRM" use="@N"/>` and `<xsl:apply-templates select="FRM[generate-id() = generate-id(key('Grouped-Forms', @N)[1])]" />`, or syntax that can look for specific occurrences of repeating data such as: `<xsl:value-of select="FRM[@N='FrmName'][1]/SEC/FLD[@N='FldName']" />`. Both of the preceding can be used to process only the first occurrence of a form either at the whole XML document level or at the level of Forms under a Group, respectively.

The preceding paragraphs are an example of a few techniques that are available for use through XSLT. Using recursively called XSL Templates is not a requirement for processing MRO output from Documaker, but is an option worth consideration. There are no restrictions on any XSLT techniques that can be used with Documaker mobile.

Appendix B

Mobile Responsive Output Format

There are many tools available for composing HTML that can be used to form mobile responsive output from Documaker. These include Adobe's Dreamweaver, Eclipse Foundation's IDE for Java EE Developers, Notepad++ and many others. This section provides more insight on techniques for forming the HTML composition desired from Documaker mobile.

We recommend these steps:

1. Start with setting up a directory with separate files for HTML, CSS, JS, IMG (images) and XSL. Place the files other than the HTML in separate sub directories under the HTML location. It is recommend to use all lowercase for all file and directory names.
2. Build prototype HTML with hardcoded values as placeholders for variable data from Documaker. This prototype HTML should follow the design of one of the wireframe layouts defined in *Mockup Design for Mobile* section. The prototype HTML can then be distributed for further review and adjustment. Mark hard coded values in the HTML with a special CSS style class, such as "hardcode_val". This will be used later in the variable content mapping process.
3. Add @media rules to the CSS used with the prototype to adjust the style rules for various max-width values. So in step 2, if you started with the thinnest layout (mobile first design) add rules to implement the wireframe and handle styling for the larger layout(s). The idea here is to make as few updates possible in the @media rules to achieve the desired results. Doing this allows you to treat the @media rules as limited overrides to the base CSS rules and it allows changes to the base CSS rules to flow through to all layout sizes without requiring special attention in the @media definitions.
4. Add JavaScript automation to the prototype that supports your desired mobile page navigation. This may include handling click events on a document menu or other navigation UI paradigm. Place the JavaScript syntax required in a file with a *.js extension in the directory setup for JS files. You may consider using jQuery as a framework for implementing mobile document automation and UI event handling.
5. Add JavaScript automation to the prototype that supports your desired content data based interactions (things such as table row drill in). Again place the JavaScript syntax in a file (can be the same file as step 4) with a *.js extension in the directory setup for JS files.

-
6. Merge the prototype HTML just created with the XSL Stylesheet developed previously to process the XML produced by the Documaker MRO and snippets. Depending on the tools you are using to compose HTML, either chunks of HTML can be pasted into various XSL Templates that will push out the HTML depending on XML contents from Documaker, or the HTML can mostly be placed in the root XSL Template (where match="/") and then other XSL techniques can be used to control the HTML output (such as `<xsl:for-each select="myExpresion">`). Also find the variable fields that were hardcoded in the prototype HTML syntax and replace their hardcoded values with XSL `<xsl:value-of select=" FLD[@N='FldName']"/>` syntax so that live XML values from Documaker will flow through to the HTML output for testing.

Note Change the HTML CSS style class used on the values that are now mapped from XSL to a new value such as "variable_val". It helps you identify what values are live and what values are hard coded.

7. Optionally add support for rendering charts and tables from Documaker in your mobile output. There are examples of both table and chart rendering logic contained the default snippets supplied with Documaker mobile for this purpose. This logic can be taken from the Documaker mobile defaults and reused in your own output to form charts and tables based on the JSON data produced by the Documaker MRO.
8. Create a master snippet in your snippet library deployed with Documaker. You can copy the default master.xml snippet and save it with a new name by adding a context value to the file name with an underscore (for example: "master_benefits.xml", the context value is "benefits"). In the new master snippet update the XSL Stylesheet reference to equate to the XSL Stylesheet you have created by updating the href attribute of line "`<?xml-stylesheet type="text/xsl" href="resources/xsl/default.xml" ?>`" replacing the "resources/xsl/default.xml" entry so it equates to the location of your XSL Stylesheet.
9. In Documaker Studio, update the Documaker Application Definition for the formset (KEY1/KEY2) whose composition output will be used to drive this mobile output with a global context value. This value will be used to locate the desired master snippet. For example, if the global context value is set to "benefits" in Studio then the Documaker MRO will look for snippets that have "*_benefits.xml" in their file name, starting with the master snippet: "master_benefits.xml". Usually the only snippet you will need to customize is the master, so you will not need to copy and create any other snippets with a "*_benefits.xml" name. In the example above the Documaker MRO will first look for snippets named "*_benefits.xml" and then it will look for the default snippet names that do not have the "_benefits" in their name.
10. Generate Mobile Output by printing with the MRO or Mobile Responsive Output delivery channel. This will drop an XML dataset file (assuming XML based snippets) that you can use for testing in a browser by directly opening the XML file with a browser. The browser (IE9+, Firefox, Chrome, Safari, iOS, Android) will run an XML to HTML transform on the XML file that was directly opened so you can test.

As testing continues, most of the updates required will usually be to the XSL Stylesheet, either adjusting XSLT logic or adjusting the HTML being written by the XSLT to achieve the mobile presentation desired. As changes are made to the XSL, CSS, JS or XML data file the change impacts can be viewed and retested by refreshing the browser window. Once testing looks solid with an initial transaction, create several Documaker transactions to test with. Make sure to touch on various transaction configurations from Documaker to test as many possible combinations and ordering of XML from the Documaker MRO as possible for your document.

11. When development testing is complete, it's time to reorganize the document's mobile components for production deployment. Review the CSS, JS, IMG and XSL files for looking for referenced content that can be embedded in the XSL for the document. For example, CSS rules that were developed just for this mobile document can be placed in the XSL as part of the root XSL Template so they are emitted as in-line styles with the HTML results. Placing the CSS rules in the XSL eliminates management and delivery (via HTTP/HTTPS) of an external file along with the mobile document.

All CSS and JS files that are not shared resources between other documents or web pages should be embedded in the XSL as part of the root XSL Template.

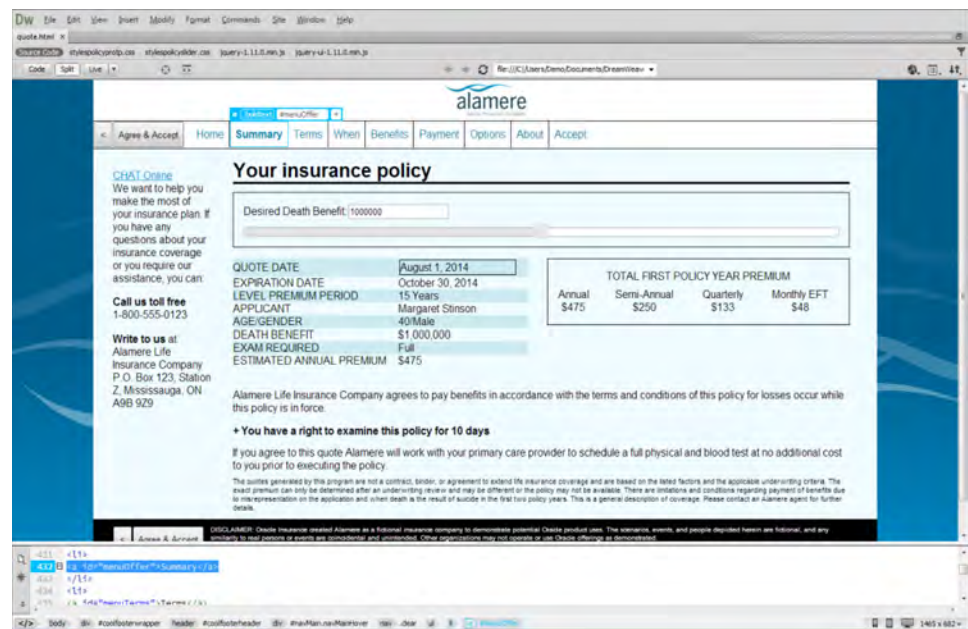
12. Optionally move the contents of the (now probably quite a bit larger) XSL document into your master snippet XML replacing the XSL Stylesheet reference in the master snippet XML. This removes the XSL document an external file that must be managed (and delivered via HTTP/HTTPS) along with the mobile document. Now all the mobile composition instructions for this document are housed in the master snippet, and the resulting mobile output requires a minimum delivery footprint of files from the Web server. The last step in this process is to involve your company's web masters to discuss mobile document delivery options such as your company web portal.

Appendix C

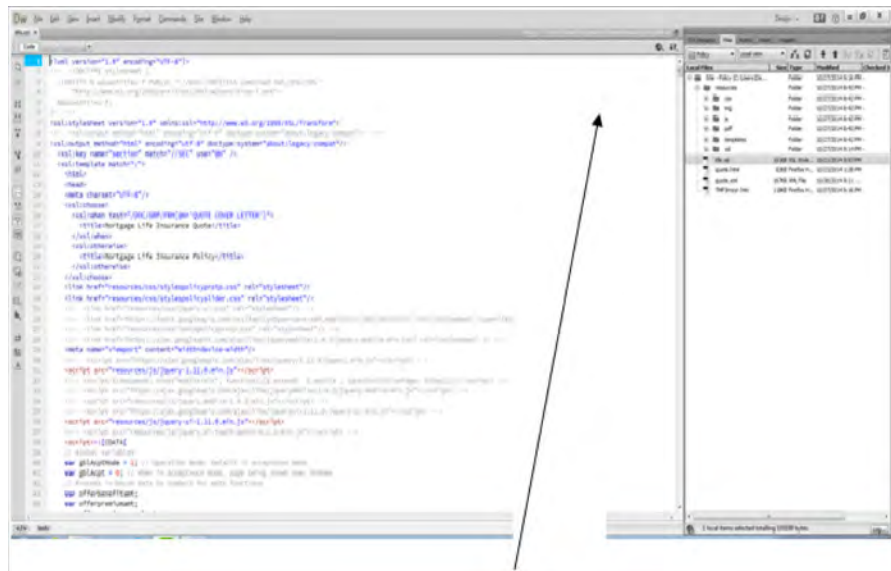
Mobile Document Authoring Tools

ADOBE'S DREAMWEAVER

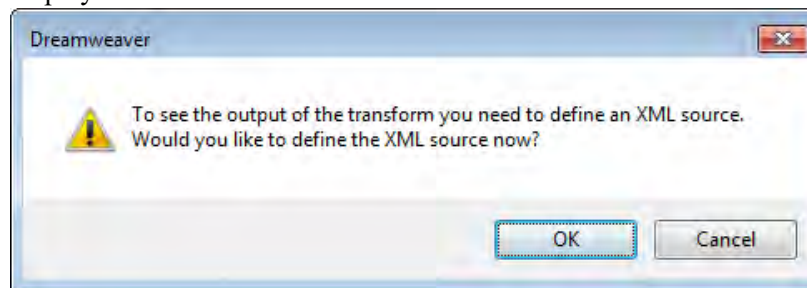
Dreamweaver is a commercial tool that does HTML and CSS composition. It also supports composing XSL with syntax coloring and help and has a feature to run XSL with an XML data file for testing from the editor tools. Here is an example of the Life quote as an HTML document under development in Dreamweaver:



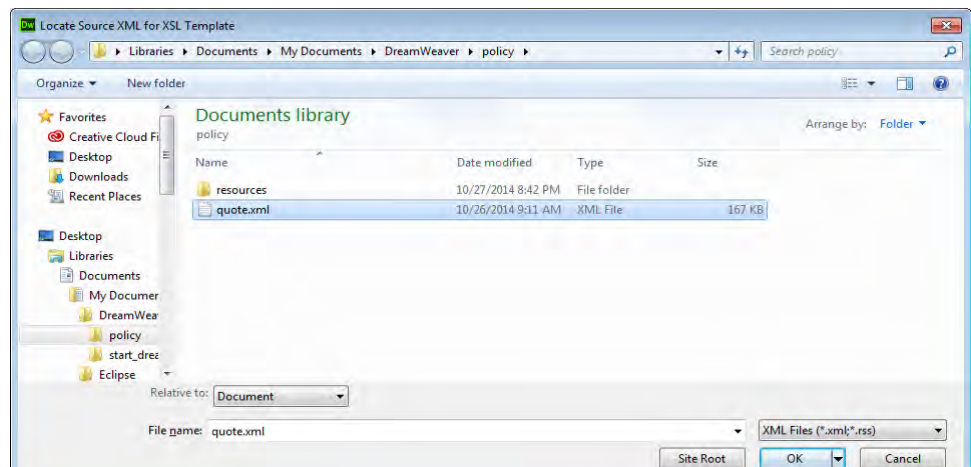
Here is an example of Dreamweaver's coloring of XSL syntax for the Life example mobile document shipping with Documaker where the HTML from above has been pasted in to XSL Templates for transformation testing:



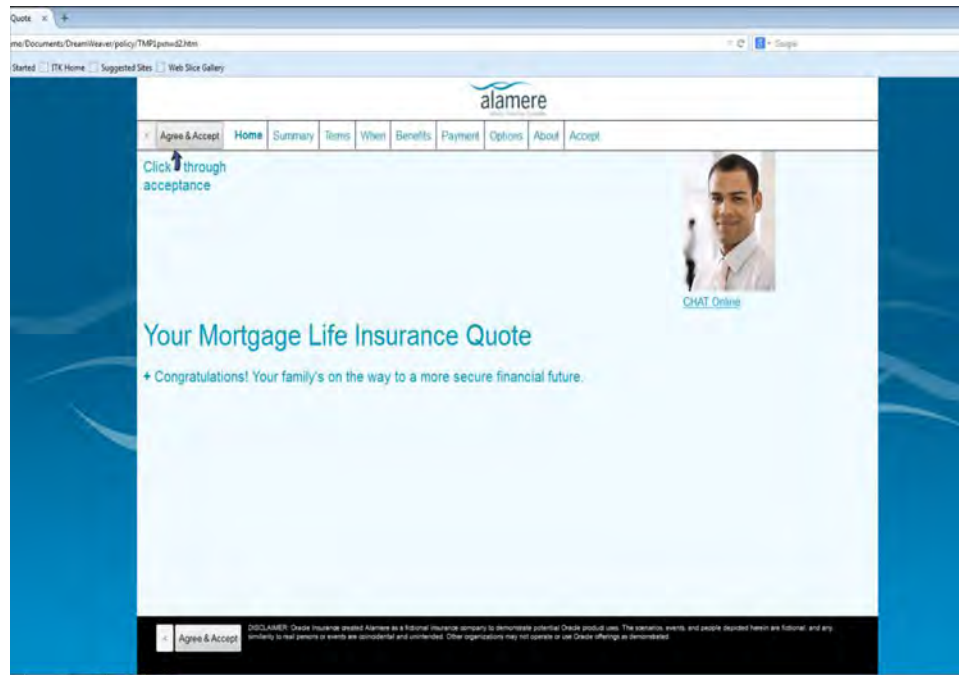
If you select the test tool from the XSL source, the following warning message is displayed:



Select an XML source:



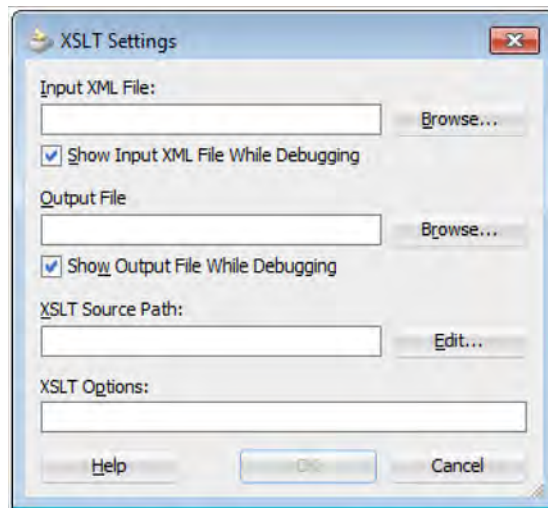
The output for testing in the default browser is shown in the following figure:



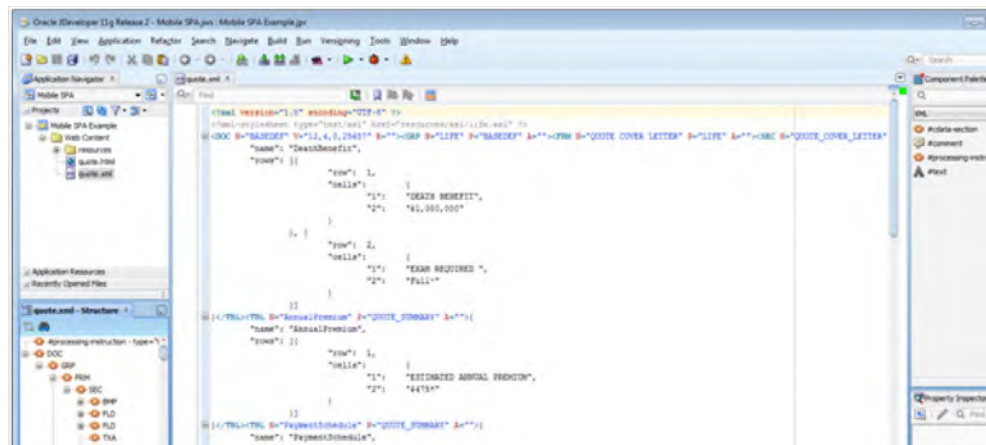
Note Oracle does not require and neither endorses or discourages the use of **Dreamweaver** when creating Snippets for Documaker Mobile use.

ORACLE JDEVELOPER

JDeveloper is a tool available from Oracle as a general integrated development environment (IDE) for development with Oracle products including deep capabilities for Web development. It has features for editing HTML, XSL, CSS, and XML including rich syntax color coding, with intelligence on the structure of XSL, HTML, and CSS including statement completion help and an outline view of the documents. Here is the prototype Quote output in the HTML editor.



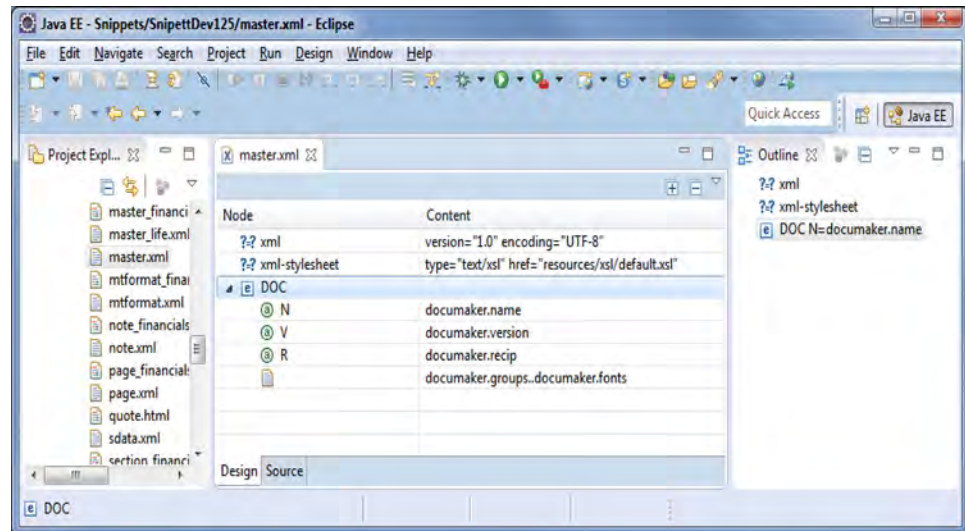
And support for XML editing:



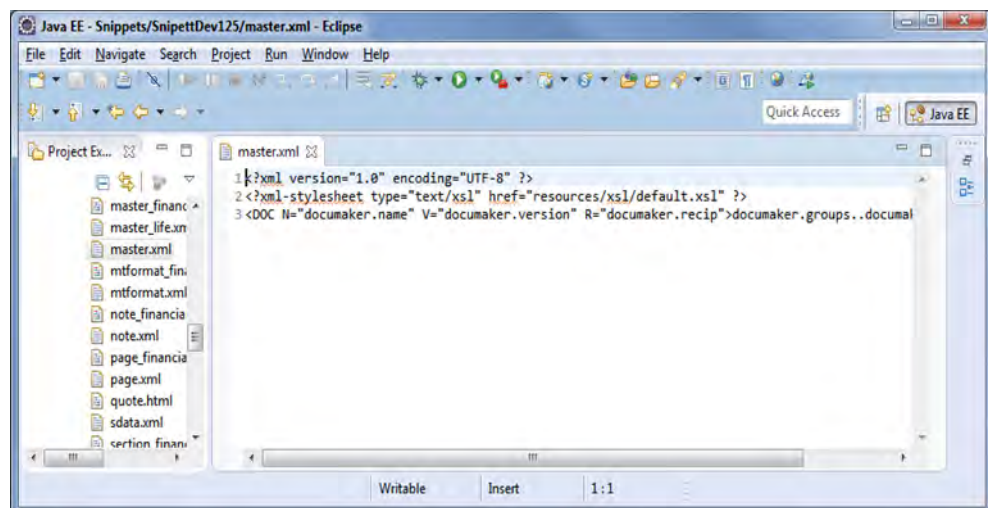
Note Oracle does not require and neither endorses or discourages the use of JDeveloper when creating Snippets for Documaker Mobile use.

ECLIPSE'S IDE FOR JAVA EE DEVELOPERS

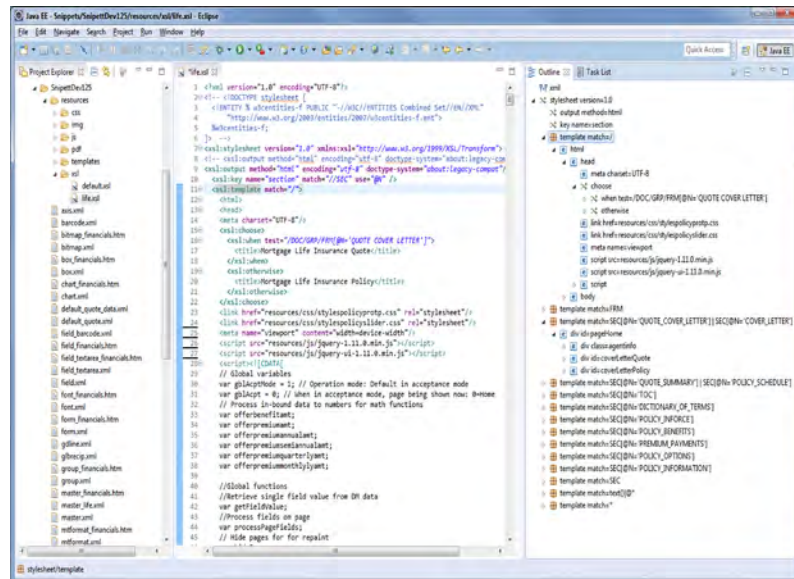
The Eclipse packaging for Java EE developers is a widely used free open source product that provides almost as much functionality as Dreamweaver does for Documaker mobile document development. This includes a dedicated XML editor.



Ability to also view content through a text editor:

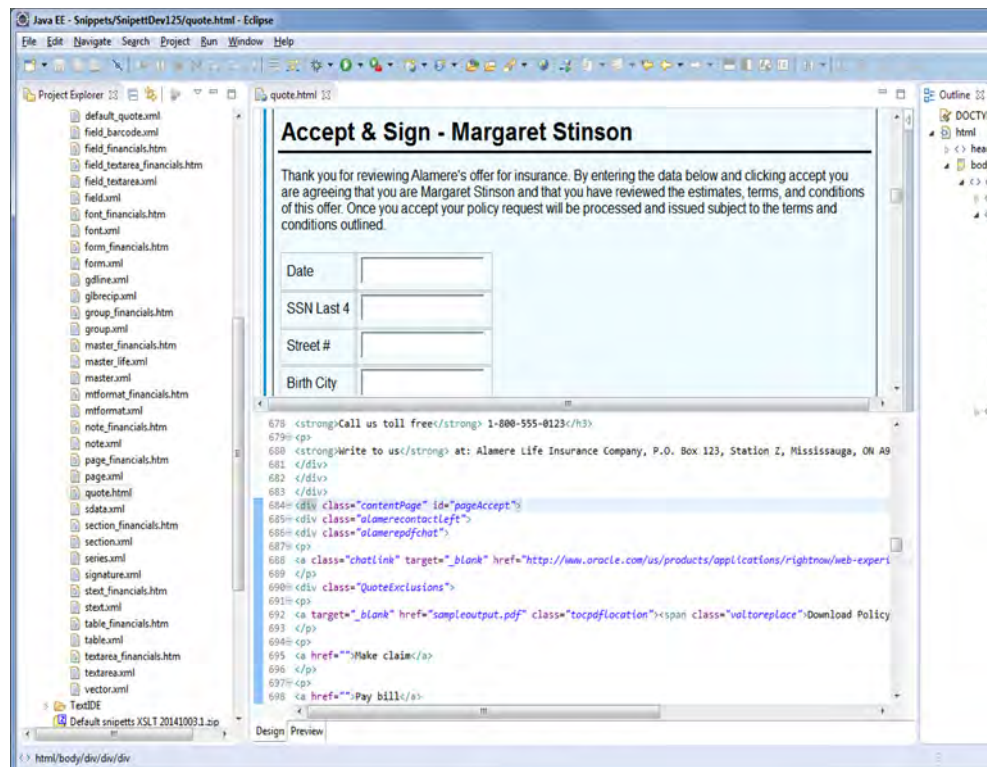


It has a dedicated editor for XSL and HTML (this is the Life quote example document):



Eclipse provides rich syntax color coding, with intelligence on the structure of XSL, HTML, and CSS including statement completion help and an outline view of the HTML document.

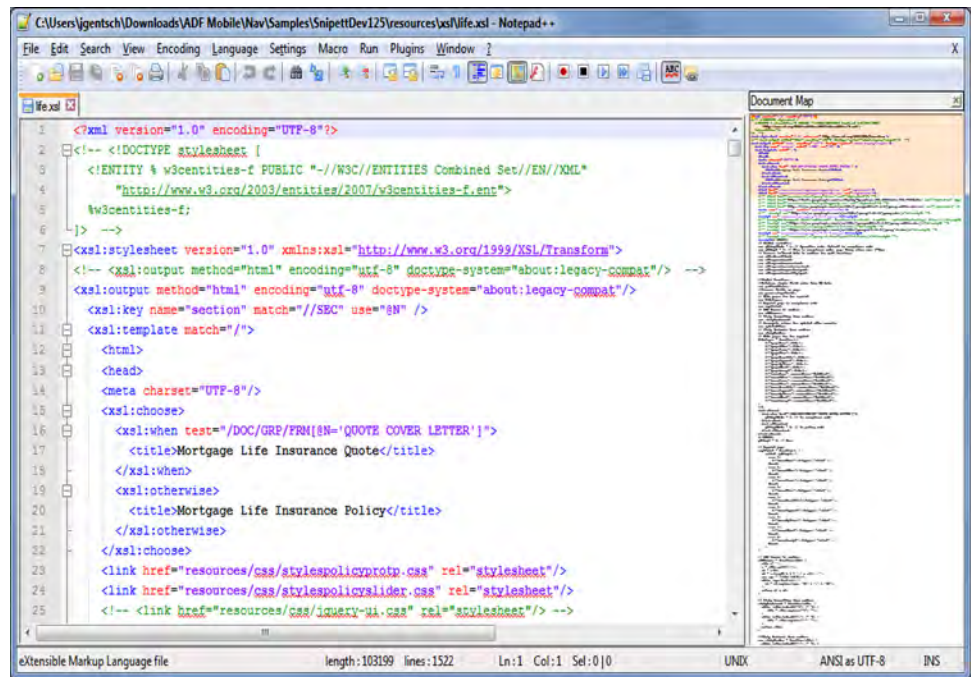
Eclipse also provides a special editor for HTML syntax authoring with a level of web preview.



Note Oracle does not require and neither endorses or discourages the use of **Eclipse** when creating Snippets for Documaker Mobile use.

NOTEPAD++

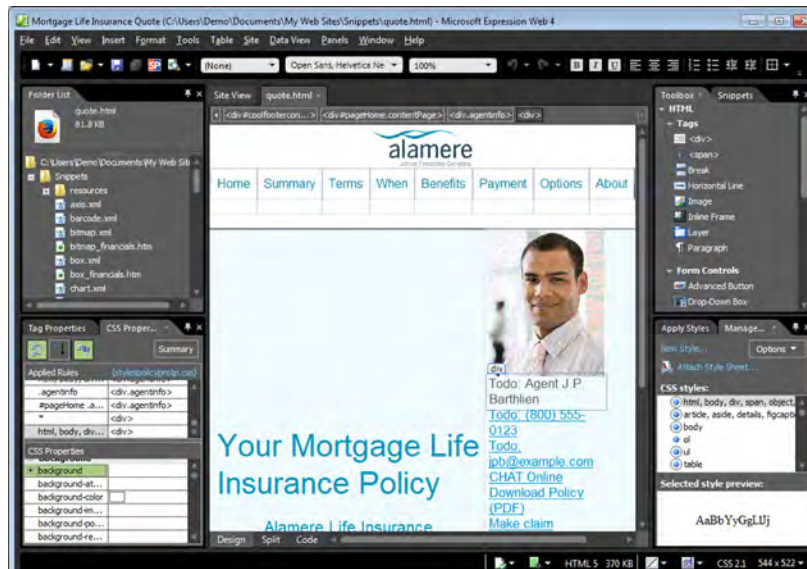
Notepad++ is a Windows only free pure text editor with rich support for syntax highlighting. It has no preview capability or deep syntax help like Dreamweaver or Eclipse:



Note Oracle does not require and neither endorses or discourages the use of **Notepad++** when creating Snippets for Documaker Mobile use.

MICROSOFT'S EXPRESSION

Expression is a commercial product available for free download from Microsoft®. This is a pure HTML and CSS editor with tuning for Microsoft's NET environments. It includes both preview and syntax helpers for HTML and CSS. There is no functionality within Expression for handling XSL or XML.



Note Oracle does not require and neither endorses or discourages the use of **Microsoft's Expression** when creating Snippets for Documaker Mobile use.

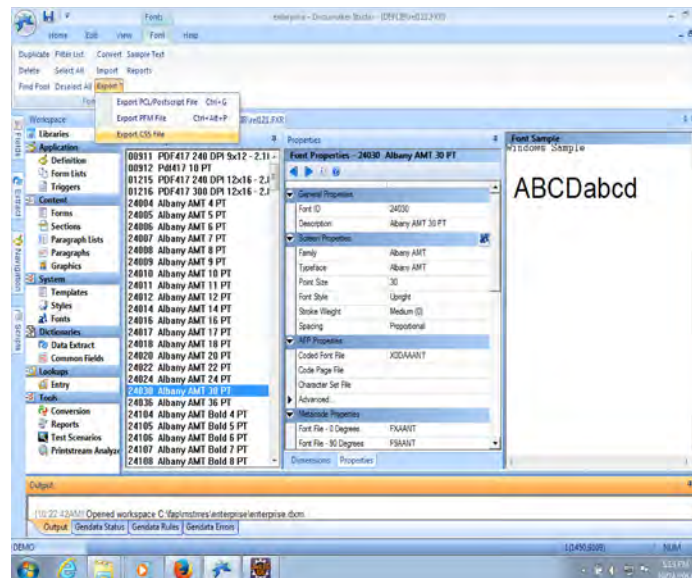
PACKAGING AND DEPLOYMENT OF MOBILE

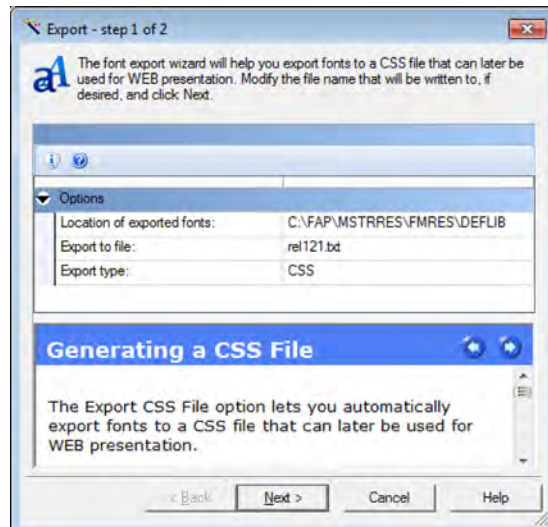
When using Documaker Mobile, a single MRL (Assembly Line) may generate output in more than one layout. You may choose to have a single layout (shell or master snippet) for all forms in the Master Resource Library, or, more likely you may define a different master snippet per formset (KEY/KEY2). This is done using Documaker Studio to override the master snippet name associated with the Key 2 value in the Application Definition. Each resource also can be associated with a custom snippet but most often there will be no need to create custom definitions for any snippet types other than the master. Each master snippet will typically reference an XSLT definition when XML is used to create output, as needed per formset associated to the master snippet so that a unique mobile presentation can be defined via the referenced XSLT for the formset (KEY1/KEY2).

REFERENCES

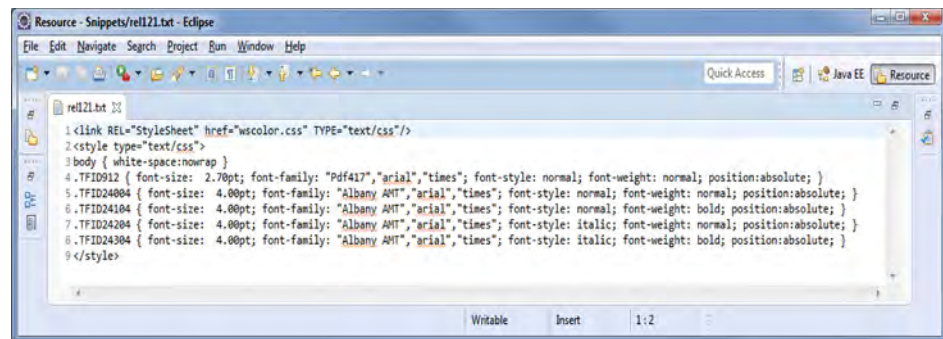
The master and individual object snippet files are stored in a location referenced by the MROLIB path definition. These files must be included in the deploy of the master resource library as part of the overall configuration. These files are NOT stored within the database tables for the MRL. Additionally, some of these files may also need to be made accessible to the web portal used to display/render the mobile output. Which means, any files referenced in mobile presentation output (e.g. CSS, WebFonts, Images, JavaScript frameworks, etc) need to be placed in a location that can be resolved to from the mobile output web server. The mobile output generated from Documaker should also be designed to reference external items that are managed and version controlled by the web portal, or its content server, being used to distribute the mobile document. This way the content management system (CMS) under the web portal will be responsible for making sure the correct versions of reference external items are delivered with the mobile content to consumers.

It is recommended to use existing resources which are already part of your web or mobile offering. If you need a style sheet to reference fonts used in Documaker for the mobile presentation, you can use Studio's Generate CSS option to have the Studio drop a text file that is really an HTML formatted file with embedded CSS:





Some assembly will be required to pull the CSS rules you want to use from the file produced by Studio into your mobile presentation. Consider the file below as an example:



Typically the first two lines of HTML syntax and also the third line including the “body” rule can be removed from the file to form pure CSS. Also, the last line containing HTML syntax can be removed. The remaining lines are CSS rules where the font-size and position attributes can be adjusted or removed leaving font faces, styles and weight attributes such as this:

```
.TFID912 {font-family: "Pdf417","arial","times"; font-style: normal; font-weight: normal;}
.TFID24004 {font-family: "Albany AMT","arial","times"; font-style: normal; font-weight: normal;}
.TFID24104 {font-family: "Albany AMT","arial","times"; font-style: normal; font-weight: bold;}
.TFID24204 {font-family: "Albany AMT","arial","times"; font-style: italic; font-weight: normal;}
.TFID24304 {font-family: "Albany AMT","arial","times"; font-style: italic; font-weight: bold;}
```


The CSS rule names listed by the Documaker Studio font export are matched to the values produced by the MRO for the “documaker.font.id” replacement tags by taking the documaker.font.id value and appending “TFID”. The default snippets place the value of the documaker.font.id replacement tag in an “FNTID” attribute that is emitted for text snippets to form CSS style class name references that match the style rule names exported by the Documaker Studio’s font CSS export function.

MOBILIZING EXISTING FORMSETS

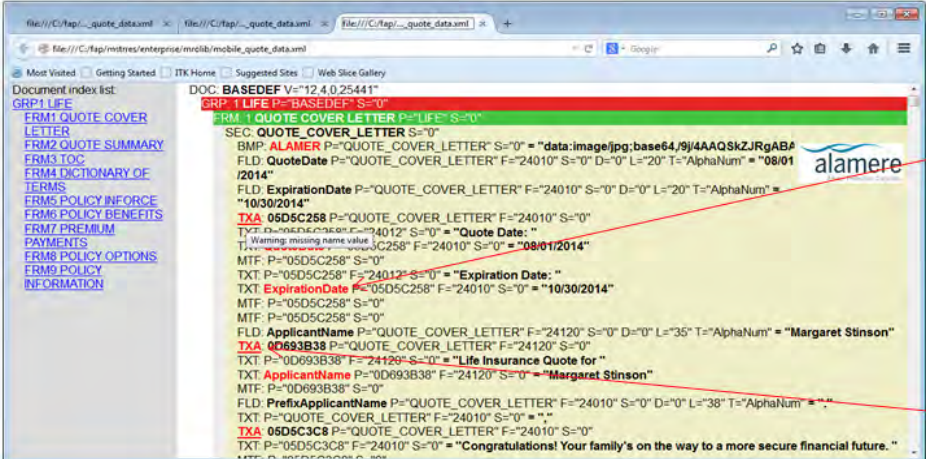
Existing Documaker formsets contain publishing objects that are already named with a meaning that supports mapping and editing. However, for Documaker Mobile use, it is advisable for the Document Author and the Mobile Author to work together to identify other objects that must be named for reference within the mobile context. These items may be:

- Charts
- Images
- Tables
- TextAreas

Adding names to objects will allow the Mobile Author to layout the data mapping logic (XSLT or JavaScript/jQuery) to consistently locate needed page based document content in the assembled output produced by Documaker’s Mobile Responsive Output (MRO) driver.

The default snippets delivered with Documaker mobile will work with any existing Documaker formset to produce a mobile presentation that helps to identify unnamed publishing objects, so they can be named for use in mobile output, and objects with names that are not unique, so they can be renamed for use in mobile output. For instance, here is the default mobile output for the mortgage life insurance example that is used earlier in this document:

The Document Index List shows the overall document hierarchy



Items in RED indicate repeating items.

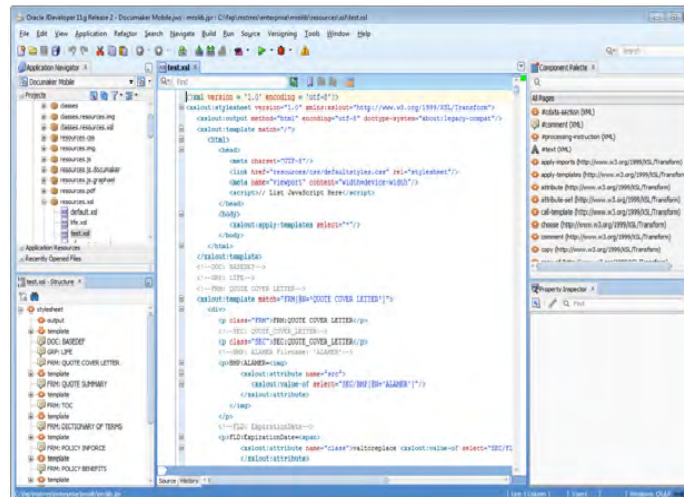
Items with an underline indicate either a missing name or a non-unique name.

The screenshot shows a web browser window with a document index list on the left and a document preview on the right. The document preview shows a form with various fields and text. Red annotations highlight specific items in the document preview:

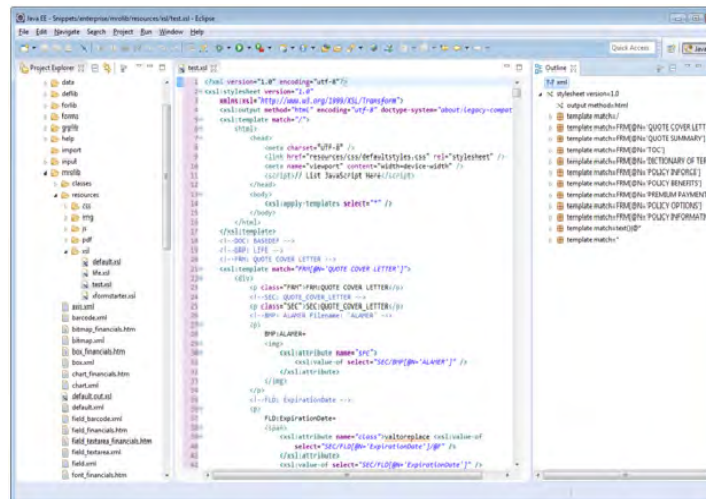
- Red text: **GRP 1 LIFE P="BASEDEF" S="0"**
- Red text: **TXA 05D5C258 P="QUOTE_COVER_LETTER" F="24010" S="0"**
- Red text: **TXA 0D693B38 P="QUOTE_COVER_LETTER" F="24120" S="0"**
- Red text: **TXA 05D5C3C8 P="QUOTE_COVER_LETTER" F="24010" S="0"**

When enabling existing documents for use in mobile, there is a step where the mobile design for the document is implemented in XSL that consumes the XML results produced by the MRO via the snippets to compose the document's mobile representation. To assist the process, an accelerator tool is provided that consumes the default XML results produced by the MRO via the default snippets and generates a starter XSL stylesheet. The starter XSL stylesheet has template syntax defined for each form that is represented in the default XML results. Syntax for accessing each named field, text, and bitmap element is found in the default XML results. The accelerator is an XSL stylesheet that produces an XSL stylesheet. It is named "xformstarter.xml" and can be found in the resources directory under the MRO snippet directory.

To create a starter XSL stylesheet, run the xformstarter.xml providing it the default XML results from the default snippets. Here is what the resulting starter XSL stylesheet looks like for the example mortgage life insurance documents from the enterprise MRL shipping with Documaker in JDeveloper:



Here is what the starter XSL stylesheet looks like in Eclipse:



This starter XSL stylesheet's structure triggers its transformation templates based on encountering named forms in the XML output:



This is a starter XML structure designed to accelerate the development of an XSL stylesheet for mobile output creation. It helps by providing XPATH syntax for processing the data hierarchy produced by the default snippets with syntax for locating Forms, Fields, Texts, Bitmaps, Tables and Charts in the MRO output. Once the XSL stylesheet has been created the next step is to load it with the HTML required for your mobile design as described in Step 6 of the *Delivering Mobile Output* topic in this document.

If the resulting starter XSL stylesheet is used without modification to transform the default snippet output into HTML, it produces a mobile document as seen in the image below. This image was produced for the example Mortgage Live Insurance documents from the enterprise MRL that ships with Documaker.



Index

A

Adobe's Dreamweaver **57**
Architecture Overview **12**

B

Best Practices **44**

C

Cascading Style Sheet **27**
conventions **6**
CSS **9**

D

databasetype **39**
Delivering Mobile Output **29**
dobe **10**
Documaker **8**
Documaker Add-In **45**
Documaker Mobile **8**
Document Authoring Tools **57**
Document Authors **9**
doPublishFromFactory **32**

I

implementation **38**

J

JavaScript files, **9**

M

Mappings **40**
Mobile Author **9**
Mobile options **45**
Mobile output **13**
MRO **46**

O

Object Specific Snippets **16**

P

preface **5**

S

Single Page Architecture **10**
Snippet Contents **15**
Snippets **13**

T

task flow **49**
template designers **9**

W

Web Standards **10**
web user interface developer **9**

X

XSL **57**
XSL style sheets **9**