

A Brief Introduction to DOTGO

CMRL Version 1.0



Contents

1	Introduction	3
2	Implement Your Own Mobile Service	4
3	A Simple Example	5
3.1	Create the Document	5
3.2	Install the Document	7
3.3	Validate the Document	7
3.4	Test the Document	8
4	Something a Little Fancier	8
4.1	Prepare the Document	8
4.2	Null Match	10
4.3	Unambiguous Match	10
4.4	Ambiguous Match	11
4.5	Unambiguous Match Skipping a Level	11
4.6	Mis-spellings and Abbreviations	11
4.7	The Engine Tag	11
4.8	The RSS Tag	12
4.9	Default Match	12
5	So What Next?	12

1 Introduction

DOTGO is a brand new way to send and receive information to and from any individual, company, or organization with an internet domain name by cell-phone text messaging. To use DOTGO, just send a text message starting with an internet domain name to the phone number DOTCOM (368266)—or to one of the phone numbers DOTEDU (368338), DOTGOV (368468), DOTNET (368638), or DOTORG (368674) as appropriate. Try it now on your mobile phone in three simple steps:

1. Compose a new text message to the phone number 368266, which spells out “D-O-T-C-O-M” on most mobile-phone keypads.
2. Type the internet domain name “cnn” as the body of the message.
3. Send the message. You should receive a response within a few seconds.

In this case, the response is a list of news topics from cnn.com. Reply with the text message “1” to select the first topic, “Top Stories.” You should again receive a response within a few seconds—in this case, a list of stories within “Top Stories” from cnn.com. Now reply again with the text message “1” to select the first story, “2” to select the second story, etc.

As another example, text “nytimes” to the phone number DOTCOM (368266). In this case, the response is a list of news stories from nytimes.com. You can reply to browse the stories as in the previous example.

Many services also accept additional input, allowing you to specify exactly what information you are seeking. For example, text “wikipedia dog” to the phone number DOTORG (368674) to learn about dogs from wikipedia.org. Or text “weather” + your zip code to the phone number DOTCOM (368266) to get the latest weather forecast for your area from weather.com. Or text “ezipsky” + your zip code + something in the night sky (like the name of a planet, star, or constellation) to the phone number DOTCOM (368266) to get up-to-the-minute instruction on how to see a celestial object from ezipsky.com. (This one works best at night!) Try a few of these examples now.

By now it is probably clear how DOTGO works: the first word of your message specifies the internet domain name, and the phone number specifies the top-level domain. So does this mean that DOTGO is simply the internet running over cell-phone text messaging? Well, not exactly. Certainly DOTGO replicates the organizational structure of the internet. That makes it easy for you to find what you are looking for. Looking for weather.com? Find it by texting “weather” to the phone number DOTCOM (368266). Looking for wikipedia.org? Find it by texting “wikipedia” to the phone number DOTORG (368674). But the information offered by DOTGO is generally a *subset* of the information available on the internet—the subset that is most useful and relevant in a mobile setting. Text messages are short, and DOTGO is designed to make it easy—and fast—to send and receive small amounts of information (say one or a few text messages worth) using a mobile phone. It is perhaps most useful to think of DOTGO as an *extension* of the internet to cell-phone text messaging.

This document provides a brief introduction to DOTGO from the perspective of a publisher seeking to implement a mobile service.

2 Implement Your Own Mobile Service

So where does the content offered by DOTGO come from? This is where things begin to get interesting. If you own an internet domain name, then you (and only you) have the authority to customize the content returned by DOTGO under that internet domain name. How? By placing a file called “index.cmrl” onto the root directory of your web server. The file index.cmrl contains instructions that describe how to distribute content via cell-phone text messaging, much as the file “index.html” contains instructions that describe how to distribute content via the web. These instructions can be as simple or as sophisticated as you like: return static or dynamic content, provide or collect information, sell or buy products—the possibilities are limited only by your imagination.

The instructions contained in the file index.cmrl are written in a simple language called the “Concise Message Routing Language” or “CMRL.” Here CMRL plays the same role to DOTGO that HTML plays to the internet—it describes content. But CMRL is specifically designed to “route” the short queries and responses that are appropriate to cell-phone text messaging. That makes it ideal for describing content that is intended to be accessed in a mobile setting, where slow data transfer rates often make browsing the internet on a mobile-phone browser a frustrating experience.

What if there is no file index.cmrl? In this case, DOTGO seeks to render the file index.html as well as is possible within the limitations of cell-phone text messaging. Sometimes the result is surprisingly useful—for example, any page that references one or more RSS feeds is rendered in such a way as to make those RSS feeds available. Other times the result is less useful. But this feature ensures that DOTGO will always respond with *something* to any request for any internet domain name. In fact, DOTGO will respond with something to any request for any URL. Try it now by texting the URL of any HTML file to the phone number DOTCOM (368266).

What about cost? All DOTGO services are offered free of charge—to both users and publishers. In particular, DOTGO charges no setup fees, no monthly fees, and no per-message fees. (Of course, users pay their standard mobile carrier text messaging rates to send or receive text messages.) The bottom line is that DOTGO is free and easy to use—just like the internet.

The advantages of DOTGO over other methods of implementing your own mobile service (e.g. by obtaining a dedicated or shared short code) are substantial: (1) You instantly carry over the brand-name recognition of your internet domain name to your mobile service. If your users know where to find you on the internet, then they know where to find you on DOTGO. (2) You save money—there are no setup fees, no monthly fees, and no per-message fees. (3) You reduce or eliminate expensive hardware—your mobile service runs on DOTGO servers. And (4) you can immediately take advantage of the powerful and easy-to-use DOTGO software infrastructure to compose innovative mobile content. There is simply no quicker, easier, or cheaper way to implement your own mobile service.

3 A Simple Example

Ready to try it out with a simple example? The source file for the example described in this section is available online at <http://dotgo.com/Support/Documentation/example1/index.cmrl>.

This section assumes that you own an internet domain name and have access to the root directory of your web server.

3.1 Create the Document

The first step is to create the document. To do this, open a text editor and enter the following text (or download the source file referenced above):

```
<?xml version="1.0" encoding="UTF-8"?>
<cmrl xmlns:dotgo="http://dotgo.com/cmrl/1.0">
  <match pattern="*">
    <message>
      <content>Hello world!</content>
    </message>
  </match>
</cmrl>
```

Save the document as “index.cmrl.”

So what does it all mean? Let’s have a look at the document, line by line.

Line 1: `<?xml version="1.0" encoding="UTF-8"?>`

This line indicates that the document is an XML document, which is necessary because CMRL is a variety of XML. An opening line like this is required of any XML document and hence of any CMRL document.

Line 2: `<cmrl xmlns:dotgo="http://dotgo.com/cmrl/1.0">`

This line indicates that the document is a CMRL document. A line like this is required of any CMRL document.

Line 3: `<match pattern="*">`

This line introduces the *match* tag, which is a fundamental aspect of CMRL. A match specifies a *pattern* against which the query is compared to determine how the query is to be routed or resolved. In this case, the match pattern is the “default” match pattern (represented by the wild-card character “*”) and there are no other match patterns specified in the document, so any query that is presented to the document will satisfy this match.

Line 4: `<message>`

This line introduces the *message* tag, which is another fundamental aspect of CMRL. Every terminating match must contain one (and only one) terminating node, of which a message is the simplest example. A message must contain *content* (i.e. the body of the message) and may in addition contain other constructs (which are beyond the scope of this document).

Line 5: `<content>Hello world!</content>`

This line contains the content of the message. In this case, the content is the simple text string “Hello world!” enclosed by the opening `<content>` and closing `</content>` tags.

Line 6: </message>

This line closes the <message> tag opened in line 4.

Line 7: </match>

This line closes the <match> tag opened in line 3.

Line 8: </cmr1>

This line closes the <cmr1> tag opened in line 2.

3.2 Install the Document

The next step is to install the document. To do this, transfer the document to the root directory of your web server, i.e. to the same directory that contains the root “index.html” of your web site. Make sure that the document is named “index.cmr1” and that permissions are set to allow appropriate access to the document, and test that the document is accessible by opening it in a web browser (e.g. at <http://example.com/index.cmr1>).

NOTE: Some web servers (such as Microsoft IIS) will not allow access to .cmr1 files until a proper MIME type is added to their configuration. For Microsoft IIS, you can add a MIME type for .cmr1 files by following instructions at

<http://support.microsoft.com/kb/326965>

Use “.cmr1” (without the quotes) for the extension and “text/xml” (without the quotes) for the MIME type. Similar steps can be taken when using web configuration control panels such as Plesk. If you do not have access to your web server configuration, then contact your web hosting service to request this change.

3.3 Validate the Document

The next step is to validate the document. To do this, first direct your web browser to <http://dotgo.com/Publishers/Resources/CMRLValidator> and then type your domain name into the input box under “Validate by URI” and click the button “Validate.” You should receive a response indicating that your index.cmr1 file is valid. If not, then verify that the document exactly matches the listing of section 3.1, that the document is named “index.cmr1,” and that permissions are set to allow appropriate access to the document.

3.4 Test the Document

The final step is to test the document. To do this, text your internet domain name to the phone number DOTCOM (368266)—or to one of the the phone numbers DOTEDU (368338), DOTGOV (368468), DOTNET (368638), or DOTORG (368674) as appropriate. For example, to access the internet domain “example.com,” text “example” to the phone number DOTCOM (3682266). (Of course, you should substitute your actual internet domain name in place of “example,” and you should text to the phone number corresponding to your actual top-level domain.) You should receive the response “Hello world!” Congratulations—you have implemented your own mobile service.

So how did it work? First, the system used the first word of the query “example” and the phone number DOTCOM (368266) through which the query was received to determine that the query should be routed to example.com. Next, the system retrieved the file index.cmrl from example.com. Finally, the system compared the remainder of the query (i.e. everything after “example,” which in this case was an empty string) against the match patterns specified in the file index.cmrl in order to resolve the query. In this case, there was only one match pattern specified in the file index.cmrl—the default match pattern—which resolved to the message “Hello world!” returned by the system.

4 Something a Little Fancier

Ready to try something a little fancier? The source file for the example described in this section is available online at <http://dotgo.com/Support/Documentation/example2/index.cmrl>. Again, this section assumes that you own an internet domain name and have access to the root directory of your web server.

4.1 Prepare the Document

Open a text editor and enter the following text (or download the source file referenced above):

```
<?xml version="1.0" encoding="UTF-8"?>
<cmrl xmlns:dotgo="http://dotgo.com/cmrl/1.0">

  <match pattern="">
    <message>
      <content>Welcome to the DOTGO example.</content>
    </message>
  </match>

  <match pattern="red">

    <match pattern="circle">
      <message>
        <content>You have selected a red circle.</content>
      </message>
    </match>

    <match pattern="square">
      <message>
        <content>You have selected a red square.</content>
      </message>
    </match>

  </match>
```

```

<match pattern="green">

  <match pattern="circle">
    <message>
      <content>You have selected a green circle.</content>
    </message>
  </match>

</match>

<match pattern="engine">
  <engine href="http://dotgo.com/cgi-bin/examples/current_time.cgi"/>
</match>

<match pattern="rss">
  <rss href="http://dotgo.com/support/documentation/example2/example.rss"/>
</match>

<match pattern="*">
  <message>
    <content>Nothing relevant here.</content>
  </message>
</match>

</cmrl>

```

Again save the document as "index.cmrl," and again install and validate the document as in the previous example.

Now let's see what this document does. We will again take as an example the internet domain "example.com," although of course you should substitute your actual internet domain name in place of "example," and you should text to the phone number corresponding to your actual top-level domain.

4.2 Null Match

The query "example" produces the response "Welcome to the DOTGO example." The system again used the first word of the query "example" and the phone number DOTCOM (368266) through which the query was received to determine that the query should be routed to example.com, and the system again retrieved the file index.cmrl from example.com, and the system again compared the remainder of the query (which was again an empty string) against the match patterns specified in the file index.cmrl to resolve the query. But in this case, the empty string was matched to the "null" match pattern (i.e. the empty string), which resolved to the message "Welcome to the DOTGO example."

4.3 Unambiguous Match

The query “example red circle” produces the response “You have selected a red circle.” In this case, the second token of the query is matched to the match pattern “red,” and the third token of the query is matched to the match pattern “circle,” so the match is unambiguous.

4.4 Ambiguous Match

The query “example circle” produces the response “Did you mean ‘red circle’(1) or ‘green circle’(2)?” In this case, the second token of the query is matched to the lower-level match pattern “circle” but is ambiguous between the higher-level match patterns “red” and “green,” so the match is ambiguous, and the system seeks to resolve the ambiguity.

This query also illustrates the “hyperlink” feature of DOTGO. In reply to the above response, the query “1” is equivalent to the query “example red circle.” Similarly, the query “2” is equivalent to the query “example green circle.” In this case, the hyperlinks were generated automatically by the system, although CMRL allows arbitrary hyperlinks to be specified.

4.5 Unambiguous Match Skipping a Level

The query “example square” produces the response “You have selected a red square.” In this case, the second token of the query is matched to the match pattern “square,” skipping the higher-level match pattern “red” since the match to “square” is unambiguous.

4.6 Mis-spellings and Abbreviations

Both the queries “example grean cirkel” and “example g c” produce the response “You have selected a green circle.” In these cases, the queries are matched to the match patterns “green” and “circle.” In general, the system seeks to obtain matches using phonetics and abbreviations as well as by direct comparison (although it is possible to turn this feature off, requiring exact matches, which is beyond the scope of this document).

4.7 The Engine Tag

The query “example engine” produces a response similar to “It is 9:18 UTC on April 12, 2008.” (but with the current time and date). In this case, the query is matched to the match pattern “engine,” which terminates in an “engine” tag that references a cgi program running on the DOTGO servers. An engine is any web-based application program and can return arbitrary output in response to arbitrary input. Here the engine simply returns the current time and date.

4.8 The RSS Tag

The query “example rss” produces the response “DOTGO News and Blog. Reply (1) for latest news entry and (2) for latest blog entry.” In this case, the query is matched to the match pattern “rss,” which terminates in an “rss” tag that references RSS feeds hosted on the DOTGO servers.

4.9 Default Match

The query “example encyclopedia” produces the response “Nothing relevant here.” In this case, the query does not match any of the specified match patterns so is matched to the default match pattern. A document does not have to have a default match, but in general it is a good idea to include one, to handle queries that cannot otherwise be resolved.

5 So What Next?

By now you know enough to get started. You create a document, install the document, and test the document.

So what next? You can use the framework described here here to write rich content: you can add arbitrary matches, nest matches within matches, and return dynamic content using the `<engine>` and `<rss>` tags. But there is much more to CMRL beyond the scope of this document: You can designate arbitrary hyperlinks. You can accept user input. You can set and get “session variables,” which store information on a per-user basis. And you can specify “keywords,” which “correct” user input to a list of standard tokens. These other features are described in other DOTGO reference documents.