doForms™

Web Services API

Updated November 20, 2013

A Product of doForms Inc.

Table of Contents

Legal Notice

Your use of doForms, including online and offline components, is governed by the Terms of Use as specified at http://www.doforms.com/terms-of-use.

Copyright © 2011-2013 doForms Inc.
All Rights Reserved.

Contact Support

Email: support@doforms.com
Website: www.doforms.com/support

Please sign up to follow http://community.doforms.com/doforms/topics/using_web_services to be alerted to recent development to doForms web services.

Overview

doForms provides "smart-forms" for Android-powered smartphones and tablets - everything you need in a turn-key, all-in-one, reliable, secure, and fully hosted mobile data collection solution. Supported data types include:

- Section labels
- Textual data
- Numeric data
- Calculations
- Date:time
- Single choice answers
- Multiple choice answers
- Category scores
- Lookup tables
- Action buttons
- Barcode scanning
- NFC scanning
- Signatures
- Sketches
- Pictures
- Video recording
- Audio notes
- GPS locations
- Approvals
- Email reports

These data entry widgets can be displayed individually, or formatted using the following "containers":

- Page
- Table
- Grid
- Questionnaire
- Repeatable Section

Mobile Data Collection App

doForms mobile data collection software works with a wide selection of popular iOS and Android smartphones and tablets. Unlike all browser-based forms, our mobile data collection software, or "mobile forms app", enables your workers to operate in both connected and disconnected environments. This is critical for workers in rural areas or urban settings with cellular dead spots.

Imagine using your mobile data collection software to instantly take a picture, then sketch on top of it to illustrate something of interest; or to scan the barcodes of materials being delivered to a job site. Imagine recording audio notes and video clips and embedding them right in your electronic mobile data collection forms; or using the GPS to precisely record a location. Think about the increased speed and

accuracy of reporting.

doForms provides a flexible, fast, and easy solution to deploying mobile data collection forms to your workers - anywhere in the world. And the mobile forms on your workers' smartphone and tablet devices are automatically synchronized and remotely kept up-to-date. Control who gets which mobile data collection forms. Remotely control who can view or change the incoming data. doForms centralizes this control on a website dashboard to save time and money.
For paid accounts, your doForms mobile app can be branded using your company name and graphics.

doForms Website

doForms provides a fully integrated website for aggregating, sorting, querying, viewing and managing mobile forms data being collected by your workers. If GPS coordinates are included, your mobile forms data can be viewed on top of an interactive map. The website keeps track of which workers collected which mobile forms data. The website makes it easy to export mobile forms data to your other business applications, and to integrate your mobile forms data in real time with other IT systems by using web services.

Use our off-the-shelf mobile data collection forms library. Or create your own mobile forms using the most powerful, yet easy-to-use form creation software available. Our form builder provides simple and intuitive user interfaces for building mobile forms. You don't need to be a specialist to use our form creation software. You just need to have an understanding of what you want your mobile data collection software to do. With doForms, anyone with office software experience can create their own mobile data collection forms for smartphones and tablets. With doForms, there is no need for software programmers to build your mobile forms or IT support staff to deploy them.

doForms allows easy exporting of mobile forms information to other business applications such as Microsoft Excel, Open Office, and Google Docs. Export options are also provided for CSV, HTML and PDF file formats. Additionally, doForms also provides for direct integration of your mobile forms with CRM, database and GIS systems, such as Salesforce.com, Oracle, SAP, and ArcGIS, through the use of industry-standard SOAP web services. These web services can be set up and deployed in a matter of minutes with no software programmers involved.

For paid accounts, your doForms website and mobile app can be branded using your company name and graphics.

doForms Dispatch

doForms Dispatch provides powerful dispatch forms and work order forms functionality. These special purpose forms contain important information to tell mobile workers where to go (dispatch) and what to do when they get there (work order). As your mobile workers complete their assigned tasks, doForms let them fill out data fields in the form, take pictures, capture GPS locations, and collect signatures. The completed data records are then sent back to you.

In addition to sending and receiving forms from your workers, doForms Dispatch also tracks their current and past GPS locations, and these locations are plotted on an interactive map in the Dispatch tab. You can select which workers to view, as well as the time interval. The map also shows the locations

where forms were filled out.

Your doForms website account provides a specialized Dispatch tab where dispatch forms and work order forms can be filled out, managed, scheduled and sent to your mobile workers. The data sent back from your workers is also viewed in this tab, as well as the status of their assigned job. And your workers' past and present locations are shown on a map in this tab.

Dispatch data can also be sent from your existing dispatch and work order system, and forwarded via our Data Exchange Server to your doForms equipped mobile devices. Similarly, the competed forms can be sent from the mobile devices, and forwarded to your existing dispatch and work order system. Our Data Exchange Server makes this integration simple and quick.

Finally, Dispatch accounts provide the ability for mobile users to "**Retrieve**" previously sent data from the website, work on it, then send it back to the website. Imagine a situation where you have one mobile worker who fills out a form, and a different mobile worker who needs to approve the job record. Or imagine a field medicine scenario where different specialists need to add information to a patient record.

Security for Paid Accounts

The security of the doForms system is based on (i) data transmission encryption, and (ii) Google's App Engine IT infrastructure security.

Data transmission between your mobile devices and the doForms website is encrypted using Secure Socket Layers (SSL3). This protects your data while traveling over the airwaves or internet. Browsing of data on your doForms website may also be encrypted using SSL/HTTPS. Please be sure to use the **encrypted SSL3 connection** at https://mydoforms.appspot.com / **followed by the name of your doForms** website account.

You can also have peace of mind knowing that your data and forms are hosted on top of Google's App Engine IT infrastructure.   Google App Engine has successfully undergone annual SAS 70 Type II audits which have evolved into the SSAE 16 Type II attestation and its international counterpart, ISAE 3402 Type II. Google App Engine is one of the first major cloud providers to be certified for compliance to these new audit standards.

Third party audits are only part of the security and compliance benefits of Google App Engine products. Google protects our customers' data by employing some of the foremost security experts, by executing rigorous safety processes, and by implementing cutting-edge technology. These protections are highlighted in the data center video tour. For more information visit the Google Apps Trust page.

Source: http://googleenterprise.blogspot.com/2011/08/security-first-google-apps-and-google.html

SOAP API

IMPORTANT: Please sign up for http://community.doforms.com/doforms/topics/using_web_services to be alerted to recent developments in doForms web services.

IMPORTANT: The **Web Services** tab is available to doForms users with Manage and Admin privileges only. Web services are available for paid accounts only.

To access the doForms web services, go to the Accounts tab and click on Web Services. Web services are used to expose the data in your doForms account for use with other IT systems.doForms Web Services can be accessed using any program that supports SOAP web services (you MUST use SOAP in your client programs).

doForms web services are organized on a per-form basis. When a form is added to the Web Services tab, DoForms assigns a [WSID] and a [password] for accessing the form data. The form data served by the web services is described using Web Service Descriptive Language (WSDL).

Managing Services

Add service – To add a new web service, select **Add service** in the Row Menu. You will be prompted to first select a project, then select the form that you wish to publish. Then enter a password. The WS ID is a system-generated value. Press **Add** when done. A new row will be added to the Web Services tab containing the web service URL generated by the doForms system, [WSID], and the[Password] you entered (the use of these parameters is explained below).

Delete service – You can delete the web service by clicking **Delete service** in the Row Menu. A message will be displayed asking you to confirm that you wish to delete the web service. Press **OK** if you do. Otherwise, press **Cancel**.

Turn on/off – Existing web services can be turned off or on by clicking **Turn on/off** in the Row Menu.

Web services which are turned off will not be accessible.

Limits on Use

We monitor and restrict how many data records are read daily by each doForms web service WSID. All use limits are on a "per day" basis. Each "day" starts and ends at 00:00 UTC.

Limits on Data Record Reads: The limit is currently set to 50 times the number of mobile units in your account per day. So as an example, if you have 30 mobile units in your account, you can read 50X30=1500 records per day.

Limits on Lookup Table Uploads: The limit is currently set to 1,000,000 write operations per day.   A good rule of thumb is the number of write operations equals approximately 4 x number of records x number of fields that are uploaded. So for 10,000 records with 10 fields you would expect approximately 400,000 write ops.

Limits on Lookup Table Downloads: The limit is currently set to 1,000,000 read operations per day. A good rule of thumb is the number of read operations equals approximately just the number of records that are downloaded (i.e., read ops are independent of the number of fields). So for 10,000 records with 10 fields you would expect approximately 10,000 write ops.

Note that during the 7 days immediately following the creation of a web service, we provide higher limits than those above in order to permit adequate testing of your software. This is done on a WSID by WSID basis. So if you need to extend this 7 days you can do so by deleting the doforms web service, and creating a new one for the same project/form (this will generate a new WSID).

Customers requiring higher use limits should contact support@doforms.com. We reserve the right to change these limits at any time with or without notice in order to maintain the performance and reliability of your doForms website.

NOTICE: There are currently no use-limits on the BETA lookup table related web services. These use-limits will be implemented when the production version of these web services is released and additional charges may be applicable to these web services.

Connecting to a Web Service

doForms web services **MUST be connected to using SOAP**. There are known problems with connecting to the web services via HTTP and we do not support this method.

Detailed instructions on how to connect with a SOAP web service are beyond the scope of this manual, but well known to any IT professional experienced in SOAP. doForms Web Services strictly follow the W3C SOAP Specification Version 1.2 (http://www.w3.org/TR/soap/). We recommend using a tool such as **soapUI** (http://www.soapui.org/) for exploring and testing a doForms web service.

The WSDL file for a doForms web service looks something like the following, where the [WSID] and [Password] are provided in the Web Services tab (see above). The current WSDL can be obtained at http://www.mydoforms.com/wss?wsdl. The [URL] provide a quick view of the data in its XForms XML

format and can be used to verify that the web service is being used correctly.

Basic Methods for Reading Data

Most users of the doForms web services use them to retrieve data records submitted from mobile devices so that these data records can be integrated into another information system. In most cases this process centers around one method: "**getUnReadData**". This method is used for reading data records that have not been previously read by your client application. The "**DataFormat**" argument determines if data is returned in CSV or XML format. The "**NumberOfRec**" determines how many records are returned with each call. Null records values will be returned when there are no more data records to read. We strongly recommend making this value less than 100 for reliability.

In the simplest case, this method is used with the "isAutoUpdate" argument equal to "1". Doing so automatically removes any record that is read with the getUnReadData from the unread data queue so it will not be re-read with subsequent calls of this method. In other cases, you may want to have more control over when a record is marked as read. For example, if performing integrity checks on each returned record. In this case, use the getUnReadData method with the **"isAutoUpdate" argument equal to "0"**. Then use the **"markUnReadDataAsRead"** method to manually mark successfully read records from the unread record queue.

If you need more **control o**ver which records are read than described above, please consider using the **getRecordKeyByReceivedDateRang**e and **getDataByRecordKey** methods which will enable you to identify and get specific records based on the date:time in UTC when a record was received by the mydoForms website. Alternatively, use the **getRecordKeyByDateRange** method if you want to do this based on the "Date_Created" which is the date:time that the record was created on the mobile device(not recommended).

With all of the "get" methods above, you may also want to delete data records from the doForms website after they have been read.   In this case, use the "**deleteDatabyRecordKey**" method .   Note that even if the record is to be deleted, it must also be marked as "marked as read" to avoid an error if using the getUnReadData method.

IMPORTANT NOTE: In doForms there is a distinction between a record that is newly submitted by a mobile device to the doForms website and a previously submitted record that is edited on the doForms website. The getUnReadData method reads both newly submitted records and edited records.

IMPORTANT NOTE: See the Limits on Use section above before implementing any of the methods below.

Downloading Media Files

The output of the web services contains the html links for each media source file. With some coding, you can create a program sending an http request via the html links to download the media source files automatically.

Sample Code

Java - http://beta.mydoforms-hrd.appspot.com/support/doForms_WS_Java_Sample.zip
VB.Net - http://beta.mydoforms-hrd.appspot.com/support/doForms_WS_VBNet_Sample.zip
PHP - http://beta.mydoforms-hrd.appspot.com/support/doForms_WS_PHP_Sample.zip

API Support Forums

http://community.doforms.com/doforms/topics/using_web_services

Important Notes

Below are the most current recommended methods provided by the doForms web services API. Note that in all cases the "[WS ID]" and "[Password]" refer to a specific doForms project/form web service.

The API methods described herein use two different ways of referencing specific records: Please be sure to use to the correct reference when using a specific method:

RecordId – This is the same as the "Form_Record" field displayed in the View Data and Dispatch tabs and included in exports.

RecordKey – This is an internal database reference that is more efficient than RecordID.

Also, some of the API methods described here use and/or return two different date:time stamps:

Date_Created – This is the date:time when a record is first saved as complete on a mobile device (or on the website if it was first created there.)

Date_Recieved – This is the date:time when a record was first received by the website. Note that this is the most reliable date:time for specifying "non-overlapping" record ranges.

All date/time values are returned in the UTC/GMT time zone. However you can use the Time Zone parameter that is returned by the newest version of the getDataByRecordKey method to calculate the local Date_Created.

API Methods

Current API Methods

getWsIDList2: Return the list of active web service WSID's for an account which matching the password

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="services.wss.portal.doforms.mdt.com">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:getWsIDList2>
         <arg0_java_lang_String>[Account Name]</arg0_java_lang_String>
         <arg1_java_lang_String>[WS Password]</arg1_java_lang_String>
         <arg2_int>[ResponseFormat]</arg2_int>
      </ser:getWsIDList2>
   </soapenv:Body>
</soapenv:Envelope>


Where:
-  [Account Name] is the Account name or Website name
-  [WS Password] is the WS Password
-  [ResponseFormat]: 1=CSV, 2=XML, 3=JSON
```

getFormTemplate2: get the form structure in XML format

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="services.wss.portal.doforms.mdt.com">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:getFormTemplate2>
         <arg0_java_lang_String>[WS ID]</arg0_java_lang_String>
         <arg1_java_lang_String>[Password]</arg1_java_lang_String>
      </ser:getFormTemplate2>
   </soapenv:Body>
</soapenv:Envelope>
```

checkValidWebservice: Check if a WS is valid or not

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="services.wss.portal.doforms.mdt.com">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:checkValidWebservice>
         <arg0_java_lang_String>[WS ID]</arg0_java_lang_String>
         <arg1_java_lang_String>[Password]</arg1_java_lang_String>
</ser:checkValidWebservice>
   </soapenv:Body>
</soapenv:Envelope>


Return:
            true: the WS is valid
```

```
                                 false: the WS is invalid
```

getFormRecordCount: Get the total records for a specific form
```
      <soapenv:Envelope
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:ser="services.wss.portal.doforms.mdt.com">
          <soapenv:Header/>
          <soapenv:Body>
             <ser:getFormRecordCount>
                <arg0_java_lang_String>[WS ID]</arg0_java_lang_String>
                <arg1_java_lang_String>[Password]</arg1_java_lang_String>
      </ser:getFormRecordCount>
          </soapenv:Body>
      </soapenv:Envelope>


      Return: number of records
```

getUnReadData: Just download records that have not been previously sent
included the new records. Note that this method is subject to the Use Limits
described above.

```
      <soapenv:Envelope
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:ser="services.wss.portal.doforms.mdt.com">
          <soapenv:Header/>
          <soapenv:Body>
             <ser:getUnReadData>
                <arg0_java_lang_String>[WS ID]</arg0_java_lang_String>
                <arg1_java_lang_String>[Password]</arg1_java_lang_String>
                <arg2_int>[DataFormat]</arg2_int>
                <arg3_int>[NumberOfRec]</arg3_int>
                <arg4_int>[isAutoUpdate]</arg4_int>
             </ser:getUnReadData>
          </soapenv:Body>
      </soapenv:Envelope>


      Where:
      -   [DataFormat]: 1=CSV, 2=XML
      -   [NumberOfRec]: the number of records to be downloaded. Due to the
          existing limits from GAE, we recommend to just read less than 100
          records for each call.
      -   [isAutoUpdate]: 1-the downloaded records will be automatically set
          to "read" and will not be returned in next calls; 0-the downloaded
          records are still kept as "unread"
      -   Note: If using isAutoUpdate=0, you should use the
          markUnReadDataAsRead method to manually mark the records as read in
          order to avoid service limitations on number of records read daily.
```

markUnReadDataAsRead: Mark the unread data as "read" to remove the
corresponding record from the getUnReadData call
```
      <soapenv:Envelope
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:ser="services.wss.portal.doforms.mdt.com">
```

```
        <soapenv:Header/>
        <soapenv:Body>
            <ser:markUnReadDataAsRead>
                <arg0_java_lang_String>[RecordKey]</arg0_java_lang_String>
            </ser:markUnReadDataAsRead>
        </soapenv:Body>
    </soapenv:Envelope>
```

Where:
-   [RecordKey]: The RecordKey which is extracted from the "@recordKey"
    field returned getUnReadData method.

getRecordKeyByReceivedDateRange: Return a list of the RecordKeys in a given
received date range which is the date:time that the record is received by the
website.

```
    <soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ser="services.wss.portal.doforms.mdt.com">
        <soapenv:Header/>
        <soapenv:Body>
            <ser:getRecordKeyByReceivedDateRange>
                <arg0_java_lang_String>[WS ID]</arg0_java_lang_String>
                <arg1_java_lang_String>[Password]</arg1_java_lang_String>
                <arg2_java_lang_String>[From Date]</arg2_java_lang_String>
                <arg3_java_lang_String>[To Date]</arg3_java_lang_String>
            </ser:getRecordKeyByReceivedDateRange>
        </soapenv:Body>
    </soapenv:Envelope>
```

Where:
-   [From Date]: the UTC date:time in MM/dd/yyyy HH:mm:ss format
-   [To Date]: the UTC date:time in MM/dd/yyyy HH:mm:ss format

getRecordKeyByEditedDateRange: Return a list of the RecordKeys in a given
date range when the records are edited/updated.

```
    <soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ser="services.wss.portal.doforms.mdt.com">
        <soapenv:Header/>
        <soapenv:Body>
            <ser:getRecordKeyByEditedDateRange>
                <arg0_java_lang_String>[WS ID]</arg0_java_lang_String>
                <arg1_java_lang_String>[Password]</arg1_java_lang_String>
                <arg2_java_lang_String>[From Date]</arg2_java_lang_String>
                <arg3_java_lang_String>[To Date]</arg3_java_lang_String>
            </ser:getRecordKeyByEditedDateRange>
        </soapenv:Body>
    </soapenv:Envelope>
```

Where:
-   [From Date]: the UTC date:time in MM/dd/yyyy HH:mm:ss format
-   [To Date]: the UTC date:time in MM/dd/yyyy HH:mm:ss format

getRecordKeyByDateRange: Return a list of the RecordKeys in a given date range.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="services.wss.portal.doforms.mdt.com">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:getRecordKeyByDateRange>
         <arg0_java_lang_String>[WS ID]</arg0_java_lang_String>
         <arg1_java_lang_String>[Password]</arg1_java_lang_String>
         <arg2_java_lang_String>[From Date]</arg2_java_lang_String>
         <arg3_java_lang_String>[To Date]</arg3_java_lang_String>
      </ser:getRecordKeyByDateRange>
   </soapenv:Body>
</soapenv:Envelope>
```

Where:
- [From Date]: the UTC date:time in MM/dd/yyyy HH:mm:ss format
- [To Date]: the UTC date:time in MM/dd/yyyy HH:mm:ss format

getDataByRecordKey4: Get the detail of a record with the specified RecordKey in a specified format. This newest version of the getDataByRecordKey method also returns the Time Zone for the record Date_Created (which is always returned in UTC/GMT). Note that this method is subject to the Use Limits described above.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="services.wss.portal.doforms.mdt.com">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:getDataByRecordKey>
         <arg0_java_lang_String>[WS ID]</arg0_java_lang_String>
         <arg1_java_lang_String>[Password]</arg1_java_lang_String>
         <arg2_java_lang_String>[RecordKey]</arg2_java_lang_String>
         <arg3_int>[ResponseFormat]</arg3_int>
         <arg4_int>[DatetimeFormat]</arg4_int>
      </ser:getDataByRecordKey>
   </soapenv:Body>
</soapenv:Envelope>
```

Where:
- [RecordKey]: The RecordKey which is extracted from the "@recordKey" field returned by the "getRecordKeyByDateRange" method
- [ResponseFormat]: 1=CSV, 2=XML, 3=JSON
- [DatetimeFormat]:
-     + "0" : yyyy-MM-ddTHH:mm:ss (UTC/GMT)
-     + "1" : MM/dd/yyyy HH:mm:ss (UTC/GMT)

submissionDispatch2: Submit a record to given WebService.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="services.wss.portal.doforms.mdt.com">
   <soapenv:Header/>
   <soapenv:Body>
```

```
        <ser:submissionDispatch2>
            <arg0_java_lang_String>[WS ID]</arg0_java_lang_String>
            <arg1_java_lang_String>[Password]</arg1_java_lang_String>
          <arg2_java_lang_String>[recordsToSubmit]</arg2_java_lang_String>
        </ser:submissionDispatch2>
    </soapenv:Body>
</soapenv:Envelope>
```

Where:
- [recordsToSubmit]is a CSV data string including headers and corresponding values concatenated by @END_LINE; For example: "@mobileNumber","string_question","integer_question","decimal_questi on","date_question","Time_question","Date_Time_question","select_mul tiple_question","select_one_question","geopoint_question","barcode_q uestion"@END_LINE;"0909707606","sent using wss to the mobile device 0909707606 #9","11","12.5","2012-07-06","23:59:59","2012-07- 21T23:59:59","option_a option_c","option_3","10.687 106.23 54 20","898777907";

  The "@mobileNumber" is a system field. If this field is blank or omitted, the submitted record will be set to "pending". Otherwise, this record will be automatically set to "sent" and delivered to the mobile unit whose mobile number is set in this field.

listDispatch: Return a list of Records headers based On Status Flag, only included RecordKey, and base header fields(MobileNumber, Date Completed, Date Submitted).
```
    <soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ser="services.wss.portal.doforms.mdt.com">
        <soapenv:Header/>
        <soapenv:Body>
            <ser:listDispatch>
                <arg0_java_lang_String>[WS ID]</arg0_java_lang_String>
                <arg1_java_lang_String>[Password]</arg1_java_lang_String>
                <arg2_int>[DispatchStatus]</arg2_int>
                <arg3_int>[ResponseFormat]</arg3_int>
            </ser:listDispatch>
        </soapenv:Body>
    </soapenv:Envelope>
```

Where:
- [DispatchStatus]: 1=Pending, 2=Scheduled, 3=Sent, 4=Received, 5=Viewed, 6=Rejected, 7=Completed, 100=All
- [ResponseFormat]: 1=CSV, 2=XML, 3=JSON

deleteDataByRecordKey: Delete records by the *RecordKey*
```
    <soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ser="services.wss.portal.doforms.mdt.com">
        <soapenv:Header/>
        <soapenv:Body>
            <ser:deleteDataByRecordKey>
                <arg0_java_lang_String>[RecordKeys]</arg0_java_lang_String>
```

```
            </ser:deleteDataByRecordKey>
        </soapenv:Body>
    </soapenv:Envelope>
```

Where:
- [RecordKeys]: The list of the RecordKey separated by a comma (",").
  The RecordKey which is extracted from the "@recordKey" field
  returned by the "getRecordKeyByDateRange" method.

Return:
> true: the WS is valid
> false: the WS is invalid

deleteData: Delete records by the *record Id*
```
    <soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ser="services.wss.portal.doforms.mdt.com">
        <soapenv:Header/>
        <soapenv:Body>
            <ser:deleteData>
                <arg0_java_lang_String>[Record IDs]</arg0_java_lang_String>
            </ser:deleteData>
        </soapenv:Body>
    </soapenv:Envelope>
```

Where:
- [Record IDs] The list of the record ID separated by a comma (",").
  Note that the record ID is the "Form_Record" field on the View Data
  tab.

Return:
> 0: Success
> -1: System Error
> -2: No record found

deleteDispatch: Deletes Dispatch records based on Ids
```
    <soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ser="services.wss.portal.doforms.mdt.com">
        <soapenv:Header/>
        <soapenv:Body>
            <ser:deleteDispatch>
                <arg0_java_lang_String>[WS ID]</arg0_java_lang_String>
                <arg1_java_lang_String>[Password]</arg1_java_lang_String>
                <arg2_java_lang_String>[RecordKeys]</arg2_java_lang_String>
                <arg3_int>[ResponseFormat]</arg3_int>
            </ser:deleteDispatch>
        </soapenv:Body>
    </soapenv:Envelope>
```

Where:
- [RecordKeys]: the list of the RecordKey concatenated by a comma
- [ResponseFormat]: 1=CSV, 2=XML, 3=JSON

getLookupTableList: Return the Lookup Table list that includes only the Lookup Table key, name, description and the column list.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="services.wss.portal.doforms.mdt.com">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:getLookupTableList>
         <arg0_java_lang_String>[ACCOUNT NAME]</arg0_java_lang_String>
         <arg1_java_lang_String>[EMAIL]</arg1_java_lang_String>
         <arg2_java_lang_String>[PASSWORD]</arg2_java_lang_String>
         <arg3_int>[ResponseFormat]</arg3_int>
      </ser:getLookupTableList>
   </soapenv:Body>
</soapenv:Envelope>
```

Where:
- [ACCOUNT NAME]: your doForms account name
- [EMAIL]: The email address which used to login into mydoforms
- [PASSWORD]: The password which used to login into mydoforms
- [ResponseFormat]: 1=CSV, 2=XML, 3=JSON

An example of a reponse in CSV format

```
[<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Root version="1.0">
<StatusCode>200</StatusCode>
<ErrorMessage>Success</ErrorMessage>
<Data>
<error_message/>
<error_code/>
<Result><![CDATA[key, tableName, description, columnsName
aglteWRvZm9ybXNyFQsSDkxvb2t1cFRhYmxlTXN0GIIGDA,"DS1","Sample
1","STATION,STATION_NAME,ELEVATION,LATITUDE,LONGITUDE,DATE,HLY-CLDH-
NORMAL,Completeness Flag,HLY-HTDH-NORMAL,Completeness Flag"
aglteWRvZm9ybXNyFQsSDkxvb2t1cFRhYmxlTXN0GJEGDA,"DS2","Sample
2","STATION,STATION_NAME,ELEVATION,LATITUDE,LONGITUDE,DATE,HLY-CLDH-
NORMAL,Completeness Flag,HLY-HTDH-NORMAL,Completeness Flag"
</Data>
</Root>]
```

getLookupTableData: Get the content of a Lookup Table. Note that this method is subject to the Use Limits described above.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="services.wss.portal.doforms.mdt.com">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:getLookupTableData>
         <arg0_java_lang_String>[ACCOUNT NAME]</arg0_java_lang_String>
         <arg1_java_lang_String>[EMAIL]</arg1_java_lang_String>
         <arg2_java_lang_String>[PASSWORD]</arg2_java_lang_String>
         <arg3_java_lang_String>[LU KEY]</arg3_java_lang_String>
         <arg4_int>[OFFSET]</arg4_int>
         <arg5_int>[LIMIT]</arg5_int>
```

```
                <arg6_int>[ResponseFormat]</arg6_int>
            </ser:getLookupTableData>
        </soapenv:Body>
    </soapenv:Envelope>
    Where:
    -   [ACCOUNT NAME]: your doForms account name
    -   [EMAIL]: The email address which used to login into mydoforms
    -   [PASSWORD]: The password which used to login into mydoforms
    -   [LU KEY]: The Lookup Table key which returned by getLookupTableList
    -   [OFFSET]: The row position will be returned
    -   [LIMIT]: The number of rows will be returned
        Note: return all rows if [LIMIT] is -1.
    -   [ResponseFormat]: 1=CSV, 2=XML, 3=JSON
    An example of a reponse in CSV format
    [<?xml version="1.0" encoding="UTF-8" standalone="no"?>
    <Root version="1.0">
    <StatusCode>200</StatusCode>
    <ErrorMessage>Success</ErrorMessage>
    <Data>
    <error_message/>
    <error_code/>
    <Result><![CDATA["STATION","STATION_NAME","ELEVATION","LATITUDE","LONGI
    TUDE","DATE","HLY-CLDH-NORMAL","Completeness Flag","HLY-HTDH-
    NORMAL","Completeness Flag"
    "GHCND:USW00003947","KANSAS CITY INTERNATIONAL AIRPORT MO
    US","306.3","39.2972","-94.7306","20100101 00:00","0","C","375","C"
    "GHCND:USW00003947","KANSAS CITY INTERNATIONAL AIRPORT MO
    US","306.3","39.2972","-94.7306","20100101 01:00","0","C","380","C"
    "GHCND:USW00003947","KANSAS CITY INTERNATIONAL AIRPORT MO
    US","306.3","39.2972","-94.7306","20100101
    02:00","0","C","384","C"]]></Result>
    </Data>
    </Root>]
```

addLookupTableData: Add a new Lookup Table. Note that this method is subject to the Use Limits described above.

```
    <soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ser="services.wss.portal.doforms.mdt.com">
        <soapenv:Header/>
        <soapenv:Body>
            <ser:addLookupTableData>
                <arg0_java_lang_String>[ACCOUNT NAME]</arg0_java_lang_String>
                <arg1_java_lang_String>[EMAIL]</arg1_java_lang_String>
                <arg2_java_lang_String>[PASSWORD]</arg2_java_lang_String>
                <arg3_java_lang_String>[LU NAME]</arg3_java_lang_String>
                <arg4_java_lang_String>[DESCRIPTION]</arg4_java_lang_String>
                <arg5_java_lang_String>[DATA SOURCE]</arg5_java_lang_String>
                <arg6_int>[ResponseFormat]</arg6_int>
            </ser:addLookupTableData>
        </soapenv:Body>
    </soapenv:Envelope>

    Where:
```

- [ACCOUNT NAME]: your doForms account name
- [EMAIL]: The email address which used to login into mydoforms
- [PASSWORD]: The password which used to login into mydoforms
- [LU NAME]: The Lookup Table name
- [DESCRIPTION]: The Lookup Table description
- [DATA SOURCE]: The datasource in CSV format. The @END_LINE; is used as a carriage return for each row and the first row is the header.
- For example:
  "STATION","STATION_NAME","ELEVATION","LATITUDE","LONGITUDE","DATE","HLY-CLDH-NORMAL","Completeness Flag","HLY-HTDH-NORMAL","Completeness Flag"@END_LINE;
  "GHCND:USW00003947","KANSAS CITY INTERNATIONAL AIRPORT MO US","306.3","39.2972","-94.7306","20100101 00:00","0","C","375","C"@END_LINE;
  "GHCND:USW00003947","KANSAS CITY INTERNATIONAL AIRPORT MO US","306.3","39.2972","-94.7306","20100101 01:00","0","C","380","C"
- [ResponseFormat]: 1=CSV, 2=XML, 3=JSON

updateLookupTableData: Update an existing Lookup Table. Note that this method is subject to the Use Limits described above.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="services.wss.portal.doforms.mdt.com">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:updateLookupTableData>
         <arg0_java_lang_String>[ACCOUNT NAME]</arg0_java_lang_String>
         <arg1_java_lang_String>[EMAIL]</arg1_java_lang_String>
         <arg2_java_lang_String>[PASSWORD]</arg2_java_lang_String>
         <arg3_java_lang_String>[LU NAME]</arg3_java_lang_String>
         <arg4_java_lang_String>[DESCRIPTION]</arg4_java_lang_String>
         <arg5_java_lang_String>[LU KEY]</arg5_java_lang_String>
         <arg6_java_lang_String>[DATA SOURCE]</arg6_java_lang_String>
         <arg7_int>[ResponseFormat]</arg7_int>
      </ser:updateLookupTableData>
   </soapenv:Body>
</soapenv:Envelope>
```
Where:
- [ACCOUNT NAME]: your doForms account name
- [EMAIL]: The email address which used to login into mydoforms
- [PASSWORD]: The password which used to login into mydoforms
- [LU NAME]: The Lookup Table name
- [DESCRIPTION]: The Lookup Table description
- [LU KEY]: The Lookup Table key which returned by getLookupTableList
- [DATA SOURCE]: The datasource in CSV format. The @END_LINE; is used as a carriage return for each row and the first row is the header. Note: the number of fields in the new datasource must be same with the current datasource.
  For example:
  "STATION","STATION_NAME","ELEVATION","LATITUDE","LONGITUDE","DATE","HLY-CLDH-NORMAL","Completeness Flag","HLY-HTDH-NORMAL","Completeness Flag"@END_LINE;

```
        "GHCND:USW00003947","KANSAS CITY INTERNATIONAL AIRPORT MO
        US","306.3","39.2972","-94.7306","20100101
        00:00","0","C","375","C"@END_LINE;
        "GHCND:USW00003947","KANSAS CITY INTERNATIONAL AIRPORT MO
        US","306.3","39.2972","-94.7306","20100101 01:00","0","C","380","C"
    -   [ResponseFormat]: 1=CSV, 2=XML, 3=JSON
```

appendLookupTableData: Append new rows to an existing Lookup Table. Note that this method is subject to the Use Limits described above.

```
    <soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ser="services.wss.portal.doforms.mdt.com">
        <soapenv:Header/>
        <soapenv:Body>
            <ser:appendLookupTableData>
                <arg0_java_lang_String>[ACCOUNT NAME]</arg0_java_lang_String>
                <arg1_java_lang_String>[EMAIL]</arg1_java_lang_String>
                <arg2_java_lang_String>[PASSWORD]</arg2_java_lang_String>
                <arg3_java_lang_String>[LU NAME]</arg3_java_lang_String>
                <arg4_java_lang_String>[DESCRIPTION]</arg4_java_lang_String>
                <arg5_java_lang_String>[LU KEY]</arg5_java_lang_String>
                <arg6_java_lang_String>[DATA SOURCE]</arg6_java_lang_String>
                <arg7_int>[ResponseFormat]</arg7_int>
            </ser:appendLookupTableData>
        </soapenv:Body>
    </soapenv:Envelope>
    Where:
    -   [ACCOUNT NAME]: your doForms account name
    -   [EMAIL]: The email address which used to login into mydoforms
    -   [PASSWORD]: The password which used to login into mydoforms
    -   [LU NAME]: The Lookup Table name
    -   [DESCRIPTION]: The Lookup Table description
    -   [LU KEY]: The Lookup Table key which returned by getLookupTableList
    -   [DATA SOURCE]: The datasource in CSV format. The @END_LINE; is used
        as a carriage return for each row and the first row is the header.
        Note: the number of fields in the new datasource must be same with
        the current datasource.
        For example:
        "STATION","STATION_NAME","ELEVATION","LATITUDE","LONGITUDE","DATE","
        HLY-CLDH-NORMAL","Completeness Flag","HLY-HTDH-NORMAL","Completeness
        Flag"@END_LINE;
        "GHCND:USW00003947","KANSAS CITY INTERNATIONAL AIRPORT MO
        US","306.3","39.2972","-94.7306","20100101
        00:00","0","C","375","C"@END_LINE;
        "GHCND:USW00003947","KANSAS CITY INTERNATIONAL AIRPORT MO
        US","306.3","39.2972","-94.7306","20100101 01:00","0","C","380","C"
    -   [ResponseFormat]: 1=CSV, 2=XML, 3=JSON
```

deleteLookupTableData: Delete all rows of an existing Lookup Table.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="services.wss.portal.doforms.mdt.com">
    <soapenv:Header/>
    <soapenv:Body>
        <ser:deleteLookupTableData>
            <arg0_java_lang_String>[ACCOUNT NAME]</arg0_java_lang_String>
            <arg1_java_lang_String>[EMAIL]</arg1_java_lang_String>
            <arg2_java_lang_String>[PASSWORD]</arg2_java_lang_String>
            <arg3_java_lang_String>[LU KEY]</arg3_java_lang_String>
            <arg4_int>[ResponseFormat]</arg4_int>
        </ser:deleteLookupTableData>
    </soapenv:Body>
</soapenv:Envelope>
Where:
```
- [ACCOUNT NAME]: your doForms account name
- [EMAIL]: The email address which used to login into mydoforms
- [PASSWORD]: The password which used to login into mydoforms
- [LU KEY]: The Lookup Table key which returned by getLookupTableList
- [ResponseFormat]: 1=CSV, 2=XML, 3=JSON

[BETA – access via http://7.mydoforms.appspot.com/wss?wsdl]
getMobileUnits: Get a list of all the mobile units connected to an account

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="services.wss.portal.doforms.mdt.com">
    <soapenv:Header/>
    <soapenv:Body>
        <ser:getMobileUnits>
            <arg0_java_lang_String>>[ACCOUNT NAME]</arg0_java_lang_String>
            <arg1_java_lang_String>>[EMAIL]</arg1_java_lang_String>
            <arg2_java_lang_String>[PASSWORD]</</arg2_java_lang_String>
            <arg3_int>[ResponseFormat]</arg3_int>
        </ser:getMobileUnits>
    </soapenv:Body>
</soapenv:Envelope>
Where:
```
- [ACCOUNT NAME]: your doForms account name
- [EMAIL]: The email address which used to login into mydoforms
- [PASSWORD]: The password which used to login into mydoforms
  [ResponseFormat]: 1=CSV, 2=XML, 3=JSON

[BETA – access via http://7.mydoforms.appspot.com/wss?wsdl]
getGPSTrackingPoints: Get all the tracking points that are within a certain date range (created and received on the mobile devices) for a specific mobile unit.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="services.wss.portal.doforms.mdt.com">
    <soapenv:Header/>
    <soapenv:Body>
        <ser:getGPSTrackingPoints>
            <arg0_java_lang_String>[ACCOUNT NAME]</arg0_java_lang_String>
            <arg1_java_lang_String>[EMAIL]</arg1_java_lang_String>
```

```
            <arg2_java_lang_String>[PASSWORD]</arg2_java_lang_String>
            <arg3_java_lang_String>[MOBILE KEY]</arg3_java_lang_String>
            <arg4_java_lang_String>["FROM" DATE]</arg4_java_lang_String>
            <arg5_java_lang_String>["TO" DATE]</arg5_java_lang_String>
            <arg6_int>[ResponseFormat]</arg6_int>
        </ser:getGPSTrackingPoints>
    </soapenv:Body>
</soapenv:Envelope>
Where:
```

- [ACCOUNT NAME]: your doForms account name
- [EMAIL]: The email address which used to login into mydoforms
- [PASSWORD]: The password which used to login into mydoforms
- [MOBILE KEY]: The mobile key which returned by the getMobileUnits method
- ["FROM" DATE]: the date:time when the GPS values are created and received on the mobile device
- ["FROM" DATE]: the date:time when the GPS values are created and received on the mobile device
- [ResponseFormat]: 1=CSV, 2=XML, 3=JSON, 4=KML, 5=GPX

Depreciated Methods

Although deprecated methods remain in the API, their use is discouraged, and deprecation may indicate that the feature will be removed in the future. Features are deprecated—rather than immediately removed—in order to provide backward compatibility, and give programmers who have used the feature enough time to bring their code into compliance with the new standard.

The following methods are depreciated:

```
getDataByRecordKey2
getDataByRecordKey3
```

Custom Development Services

Need programming services to help you with your integration?

Our software developers are experts at smartphone application development and related backends. If you have used doForms, you know how good their applications are.. Now this expertise is at your disposal for developing customized mobile business applications for your organization's specific needs. Our software development services start at $65 per hour.

Contact us for a free consultation.