

Mobenzi. RESEARCHER **R**



API GUIDE

CONTENTS

| | |
|---|----|
| Overview | 3 |
| API | 4 |
| API Entities..... | 4 |
| Query String Options | 5 |
| Expression Syntax | 7 |
| Authentication | 11 |
| Supported Data Formats | 11 |
| Updating and Deleting API entities..... | 12 |
| Triggers | 14 |
| Overview..... | 14 |
| Configuration | 14 |
| Notification Procedure | 14 |
| Notification Metadata | 15 |
| Survey Override | 17 |
| Overview..... | 17 |
| Process..... | 18 |
| Configuration | 19 |
| External Identifiers | 21 |
| Document History | 22 |
| References | 23 |

OVERVIEW

Mobenzi Researcher supports a read-only [RESTful](#) API (Application Programming Interface). The API leverages [Microsoft ADO .NET Data Services](#).

The goal of the ADO.NET Data Services framework is to facilitate the creation of flexible data services that are naturally integrated with the web. As such, ADO.NET Data Services use URIs to point to pieces of data and simple, well-known formats to represent that data, such as JSON and ATOM (XML-based feed format). This results in the data service being surfaced as a REST-style resource collection that is addressable with URIs and that agents can interact with using standard HTTP verbs such as GET, POST, PUT or DELETE. *

The Mobenzi Researcher implementation of Microsoft ADO .NET Data Services can be located at the following URI: <https://www.researchconsole.com/secure/api.svc/>

* Mobenzi Researcher's implementation of the ADO .NET Data Services does not support POST, PUT and DELETE for all entities.

API

API ENTITIES

To manually review entities accessible via the Mobenzi Researcher API you first need to log into your [research console](#) and then access the API URI (<https://www.researchconsole.com/secure/api.svc/>). The following steps outline the procedure to do this utilising Internet Explorer 7+. Please note that the steps include configuring your browser to disable feed reading view since this browser feature obfuscates the data returned by the API.

1. Launch Internet Explorer
2. Once launched, click the Tools menu option (if no menu is shown, click ALT+T), then select Internet Options.
3. Navigate to the Content tab
4. Under the “Feeds and Web Slices” section, click Settings
5. Un-check “Turn on feed reading view” (if you utilise this capability you can turn it back on after reviewing the API)
6. Click OK
7. Close and re-launch your browser
8. Navigate to www.researchconsole.com and login
9. Copy and paste this into your browser address bar <https://www.researchconsole.com/secure/api.svc/>

After following these steps you should be able to manually review a list of entities returned by the API. The following entities should be returned:

- Devices
- Models
- Countries
- Manufacturers
- Surveys
- Studies
- Submissions
- Responses (A response entity represents a field’s value in a submission)
- ResponseOptions
- Questions
- DeviceSurveyRelationships
- SubmissionsWithLocation

If you wish to review data returned for any particular entity, you would access the entity as follows: <https://www.researchconsole.com/secure/api.svc/{Entity}> – please note that {Entity} should be replaced with any particular entity you are interested in reviewing, also note that the API is case-sensitive. For example, if you wished to review all Question entities for your console, you would navigate to <https://www.researchconsole.com/secure/api.svc/Questions>.

Please be aware that when accessing records for a particular entity in the fashion described above a query to retrieve all data (i.e. no filtering is applied) will be produced. If you access the Response or Submission entities in this manner, all Responses/Submissions will be returned. This may cause your browser to freeze or crash as there are usually a large number of these entities accessible through your console.

Appropriately, ADO .NET Data Services provides a rich query protocol which allows you to search and filter the data you may be interested in.

For example, to review the first 10 Submissions for 2010, you could so as follows:

[https://www.researchconsole.com/secure/api.svc/Submissions?\\$stop=10&\\$filter=year\(uploaddate\) eq 2010&\\$orderby=uploaddate](https://www.researchconsole.com/secure/api.svc/Submissions?$stop=10&$filter=year(uploaddate) eq 2010&$orderby=uploaddate)

If you wished to include the actual responses for each of the first 10 Submissions for 2010, you could so as follows:

[https://www.researchconsole.com/secure/api.svc/Submissions?\\$stop=10&\\$filter=year\(uploaddate\) eq 2010&\\$orderby=uploaddate&\\$expand=Responses](https://www.researchconsole.com/secure/api.svc/Submissions?$stop=10&$filter=year(uploaddate) eq 2010&$orderby=uploaddate&$expand=Responses)

The default data serialisation format for data returned by the API is [ATOM](#) (XML-based feed format). When accessing the API programmatically, you may specify that the data be returned in [JSON](#) format. To change the default serialisation format to JSON, you would simply need to set the “Accept” HTTP header to “application/json” when accessing the API.

QUERY STRING OPTIONS

Please find a summary of the available query string parameters below:

| Option | Description | Example |
|---------|---|---|
| expand | The ‘expand’ option allows you to embed one or more sets of related entities in the results. For example, if you want to display a Submission and its Responses, you could execute two requests, one for <code>/Submissions(guid'42d4de6e-38b9-4b88-bcb1-df70227aff0a')</code> and one for <code>/Submissions(guid'42d4de6e-38b9-4b88-bcb1-df70227aff0a')/Responses</code> . The ‘expand’ option on the other hand allows you to return the related entities in-line with the response of the parent in a single HTTP request. You may specify multiple navigation properties to expand by separating them with commas, and you may traverse more than one relationship by using a dot to jump to the next navigation property. | A Submission with related Responses: <code>/Submissions(guid'42d4de6e-38b9-4b88-bcb1-df70227aff0a')?\$expand=Responses</code> . A Submission with related Responses and Response Options related to each Response: <code>/Submissions(guid'42d4de6e-38b9-4b88-bcb1-df70227aff0a')?\$expand=Responses/ResponseOptions</code> . A Submission with related Responses and Survey: Orders with related <code>/Submissions(guid'42d4de6e-38b9-4b88-bcb1-df70227aff0a')?\$expand=Responses,Survey</code> |
| orderby | Sort the results by the criteria given in this value. Multiple properties can be indicated by separating them with a comma. The sort order can be controlled by using the “asc” (default) and “desc” modifiers. | <code>/Submissions?\$orderby=uploaddate</code> <code>/Submissions?\$orderby=uploaddate desc</code> <code>/Submissions?\$orderby=uploaddate,survey_id desc</code> |
| skip | Skip a given number of rows when returning results. This is useful in combination with “top” to implement paging (e.g. if using 10-entity pages, saying <code>\$skip=30&top=\$10</code> would return | Return all Submissions except the first 10: <code>/Submissions?\$skip=10</code> Return the 4th page where each page shows only 10 |

| | | |
|--------|--|--|
| | <p>the fourth page). NOTE: Skip only makes sense on sorted sets; if an orderby option is included, 'skip' will skip entities in the order given by that option. If no orderby option is given, 'skip' will sort the entities by their primary key and then perform the skip operation.</p> | <p>Submissions: /Submissions?\$skip=30&\$top=10</p> |
| top | <p>Restrict the maximum number of entities to be returned. This option is useful both by itself and in combination with skip, where it can be used to implement paging as discussed in the description of 'skip'.</p> | <p>Top 10 Submissions: /Submissions?\$top=10</p> <p>Get the last 10 Submissions: /Submissions?\$orderby=uploaddate desc&\$top=10</p> |
| filter | <p>Restrict the entities returned from a query by applying the expression specified in this operator to the entity set identified by the last segment of the URI path.</p> | <p>Get all Submissions for Survey with Id 101 : /Submissions?\$filter=survey_id eq 101</p> <p>Get all Questions where the question name contains "health": /Questions?\$filter=substringof('health', name)</p> |

EXPRESSION SYNTAX

The simple expression language that is used in filter operators (and also supported in *orderby* operations) supports references to columns and literals. The literal values can be strings enclosed in single quotes, numbers and boolean values (true or false) or any of the additional literal representations shown in the 'Data Type Literal Representations' section below. The operators in the expression language use abbreviations of the names rather than symbols to reduce the amount of escaping necessary in the URL.

LOGICAL OPERATORS:

| Operator | Description | Example |
|----------|-----------------------|--|
| eq | Equal | /Questions?filter=name eq 'Health district' |
| ne | Not equal | /Questions?filter=name ne 'Health district' |
| gt | Greater than | /Submissions?\$filter=year(uploaddate) gt 2009 |
| ge | Greater than or equal | /Submissions?\$filter=year(uploaddate) ge 2010 |
| lt | Less than | /Submissions?\$filter=year(uploaddate) lt 2009 |
| le | Less than or equal | /Submissions?\$filter=year(uploaddate) le 2010 |
| and | Logical and | /Submissions?\$filter=year(uploaddate) le 2010 and survey_id = 101 |
| or | Logical or | /Submissions?\$filter=year(uploaddate) eq 2008 or year(uploaddate) eq 2010 |
| not | Logical negation | /Questions?\$filter=not endswith(name,'test.') |

ARITHMETIC OPERATORS:

| Operator | Description | Example |
|----------|----------------|---|
| add | Addition | /Submissions?\$filter=hour(uploaddate) add 5 gt 10 |
| sub | Subtraction | /Submissions?\$filter=hour(uploaddate) sub 5 gt 3 |
| mul | Multiplication | /Submissions?\$filter=minute(uploaddate) mul 0.5 lt 3 |
| div | Division | /Submissions?\$filter=minute(uploaddate) div 2 lt 3 |
| mod | Modulo | /Submissions?\$filter=minute(uploaddate) mod 2 eq 3 |

GROUPING OPERATORS:

| Operator | Description | Example |
|----------|---------------------|---|
| () | Precedence grouping | /Responses?filter=(year(Submission/uploaddate) sub 5) lt 2005 |

STRING FUNCTIONS:

| String functions |
|--|
| bool substringof(string p0, string p1) |
| bool endswith(string p0, string p1) |
| bool startswith(string p0, string p1) |
| int length(string p0) |
| int indexof(string arg) |
| string insert(string p0, int pos, string p1) |
| string remove(string p0, int pos) |
| string remove(string p0, int pos, int length) |
| string replace(string p0, string find, string replace) |
| string substring(string p0, int pos) |
| string substring(string p0, int pos, int length) |
| string tolower(string p0) |
| string toupper(string p0) |
| string trim(string p0) |
| string concat(string p0, string p1) |

DATE FUNCTIONS:

| Date Functions |
|-------------------------|
| int day(DateTime p0) |
| int hour(DateTime p0) |
| int minute(DateTime p0) |
| int month(DateTime p0) |

| |
|-------------------------|
| int second(DateTime p0) |
| int year(DateTime p0) |

Please note that it is possible to apply expressions on related entities as per the following example:

[/Responses?\\$filter=year\(Submission/uploaddate\) gt 2010](#)

To perform a date query for a specific date please review the example below:

[/Submissions?\\$filter=survey_id eq 123 and uploaddate gt datetime'2009-12-10T14:00:00.000'](#)

This query above returns all submissions for survey Id '123' where the submission was received after 2PM 10th December 2009.

| Math functions |
|-----------------------------|
| double round(double p0) |
| decimal round(decimal p0) |
| double floor(double p0) |
| decimal floor(decimal p0) |
| double ceiling(double p0) |
| decimal ceiling(decimal p0) |

AUTHENTICATION

The API should always be accessed via the SSL/HTTPS endpoint. This ensures that any data accessed through the API is encrypted while being transferred. The API supports [basic HTTPAuthentication](#).

To access the API programmatically, you will need to set the “Authorization” HTTP header when executing the web request to the relevant API endpoint. Please note that the credentials used should be that of a valid user setup in your console. To access the Submission and Response data, the user should be setup with the *View Responses* permission.

Please note that if the user account used to access the API is linked to more than one console, a custom HTTP header is required to be set. The custom HTTP header is “AccountID”. The value of this header should be set to the unique identifier for the account/console you are targeting. If you are unsure what your unique console/account ID is, please send an email requesting your account ID to support@mobileresearcher.com.

SUPPORTED DATA FORMATS

The API currently supports exchanging entities in JSON and Atom (an XML- based feed format). The mechanism used to specify in which format information is sent to a data service is the “Content-Type” HTTP header. In terms of specifying what format to receive data in, the "Accept" HTTP header should be set.

| Requested Mime Type | Response Mime Type | Serialization Format |
|------------------------|----------------------|----------------------|
| Grouping Media Types | | |
| */* | application/atom+xml | AtomPub |
| text/* | Not supported | N/A |
| application/* | Not supported | N/A |
| Individual Media Types | | |
| text/xml | text/xml | AtomPub |
| application/xml | application/xml | AtomPub |
| application/atom+xml | application/atom+xml | AtomPub |

| | | |
|------------------|------------------|------|
| application/json | application/json | JSON |
|------------------|------------------|------|

UPDATING AND DELETING API ENTITIES

Right now it is only possible to update and delete *Response* entities via the API. In future, the API will be upgraded to support modification and deletes of other entities such as *Submissions*. Please review the section on request authentication above to ensure that all requests to update/delete entities via the API carry the correct authentication tokens.

DELETING A RESPONSE

To delete a response, the response ID is required. The following HTTP DELETE action will delete a response with the ID 'E271F157-4356-4E16-B63B-AD681AED7BEE':

URL: [https://www.researchconsole.com/secure/api.svc/Responses\(guid'E971F157-4356-4E06-B63B-AD681AED7BEE'\)](https://www.researchconsole.com/secure/api.svc/Responses(guid'E971F157-4356-4E06-B63B-AD681AED7BEE'))

Please note that the HTTP method must be set to DELETE. Also, the account being used to delete a response should have the *ModifyResponses* permission in the relevant study.

UPDATING A RESPONSE

To update a response, the response ID is required. The HTTP MERGE method action and payload will update a response with ID 'E271F157-4356-4E16-B63B-AD681AED7BEE':

URL: [https://www.researchconsole.com/secure/api.svc/Responses\(guid'E971F157-4356-4E06-B63B-AD681AED7BEE'\)](https://www.researchconsole.com/secure/api.svc/Responses(guid'E971F157-4356-4E06-B63B-AD681AED7BEE'))

PAYLOAD:

```
<entry xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
        xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
        xmlns="http://www.w3.org/2005/Atom">
  <content type="application/xml">
    <m:properties>
      <d:responsevalue>Yes</d:responsevalue>
    </m:properties>
  </content>
</entry>
```

In the example above, a response's value is being updated to 'Yes'. Please note that the HTTP method must be set to MERGE and that the account being used to update the response should have the *ModifyResponses* permission in the relevant study.

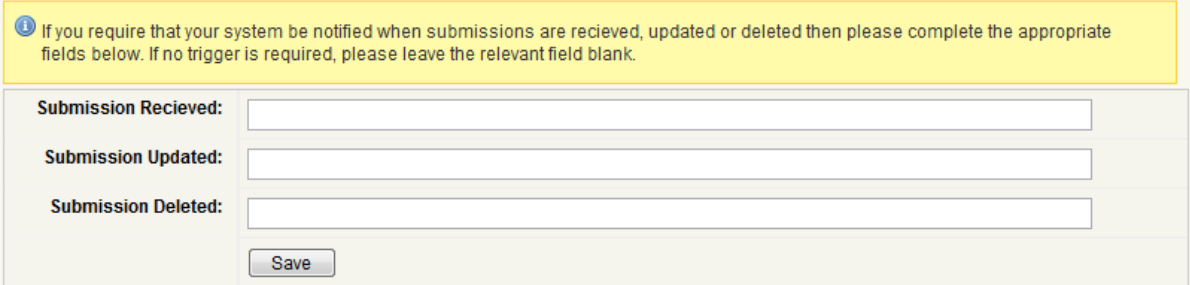
TRIGGERS

OVERVIEW

The trigger mechanism allows third party applications to be notified when a submission for a survey is **received, updated or deleted**. Triggers facilitate real time updates to third party systems and ensure data integrity. Notifications are pushed to third party systems via simple HTTP GET requests.

CONFIGURATION

Triggers are configured at a survey level. Log into www.researchconsole.com and navigate to the relevant survey that requires triggers to be setup. Once a survey is selected, click the *Design* tab and from the ribbon select *Options*. Once the survey options screen has loaded, complete the trigger section as highlighted in the screenshot below.



Submission Recieved:

Submission Updated:

Submission Deleted:

It is not necessary to setup all three trigger types. To remove an existing trigger simply clear the trigger URL field and click *Save*.

Once one or more trigger URL's have been configured, the triggers are activated immediately for all future events.

NOTIFICATION PROCEDURE

The submission trigger mechanism will reprocess a notification in the event of a delivery failure (such as network failure or a third party server being offline). This means that the notification system is resilient in scenarios where a third party system is offline (or incurs an error) while a submission is received, updated or deleted. In scenarios where a third party system is offline the processing mechanism will make up to 6 attempts to notify the system as per the following schedule (assuming failure of the first and subsequent requests):

- Retry 1 minute after first failure
- Retry 5 minutes after second failure
- Retry 20 minutes after third failure
- Retry 1 hour after fourth failure
- Retry 4 hours after fifth failure

If the result of the 6th attempt is a failure, the Mobenzi Researcher support team will notify you via email and wait on a resolution before re-processing the undelivered notification(s).

The successful delivery of a notification is defined as an attempt that results in a “200 OK” [HTTP status code](#) being returned by the relevant web server. If any other code is returned Mobenzi Researcher will deem the delivery a failure and enter the reprocessing workflow as described above.

NOTIFICATION METADATA

When a notification is processed, submission metadata is supplied to the third party system to indicate the context of the notification. The following information is passed through via [GET](#) parameters to the relevant trigger URL's configured.

New Submission:

| PARAMETER | Type | VALUE |
|---------------------|----------|---|
| surveyid | Integer | ID of the survey the submission received is relevant to. |
| submissionid | Guid | The unique ID of the submission record. |
| customid | String | If the survey override mechanism was used in provisioning the relevant survey instance to a device, the custom identifier used will be made available through this parameter. |
| createdon | DateTime | The date and time the submission was received. |
| createdby | String | The name of the fieldworker who captured the submission. |

Submission modified/deleted:

| PARAMETER | Type | VALUE |
|---------------------|---------|---|
| surveyid | Integer | ID of the survey the submission updated is relevant to. |
| submissionid | Guid | The unique ID of the submission record. |
| customid | String | If the survey override mechanism was used in |

| | | |
|-------------------|----------|--|
| | | provisioning the relevant survey instance to a device, the custom identifier used will be made available through this parameter. |
| modifiedon | DateTime | The date and time the submission was modified. |
| modifiedby | String | The name of the user who modified the submission. |

SURVEY OVERRIDE

OVERVIEW

The survey override mechanism enables the ability to dynamically create multiple instances of a survey, rename survey instances, inject custom data into survey question fields, and override field default values at runtime. The mechanism is implemented at the survey level. If configured, the override mechanism is executed before any survey updates are pushed to a Mobenzi Researcher device. It does **not** affect the survey data configured through the Mobenzi Researcher Survey Designer.

Similar to the trigger mechanism, the survey override mechanism works by using HTTP requests to communicate with a third party system.

This feature enables a third party system to provide context to surveys that are sent to a mobile device. There are a number of features that the survey override mechanism supplies:

| Feature | Description |
|--------------------------------|--|
| Instance count override | This feature enables a third party system to control whether a survey should be sent to a fieldworker. It also enables a third party system to control the number of survey instances to be sent to a fieldworker. The default instance count is 1. If the instance count is reduced to 0, then the relevant survey is not sent to the fieldworker even though they are assigned the survey. If a greater number of instances of a survey are to be sent to a device, then the features outlined below may apply to each instance. |
| Survey title override | For each instance of a survey that is sent to a fieldworker, the title of the survey (as it appears on the mobile device) may be updated by a third party system. |
| Question text override | The question text of any field (for any survey instance) may be overridden. This includes instruction fields. The feature enables contextual question text to be injected into survey fields by a third party system. |
| Option label override | Any option label (for any instance) may be overridden. The feature enables a third party system to update existing option labels, remove option labels or add new options. |
| Default value override | The default values of any field (for any survey instance) may be overridden. This is useful in scenarios where information is known and being verified by a fieldworker out in the field. The available data (for example, previously captured field values) may be injected into the |

survey as default values. For repeating regions, it is possible to define multiple default values to support surveys with repeating regions.

Required field flag override

This feature allows a third party system to alter the required flag for any field (for any survey instance). It is thus possible to dynamically set whether a field requires a response or not given the context.

Survey removal override

This feature allows a third party system to dynamically configure whether a survey instance is removed from a mobile device once the survey instance is completed.

Survey expiry override

This feature allows a third party system to dynamically configure whether a survey instance should expire after a set period of time. The time is configured in minutes and should be set to '-1' to disable expiry completely.

PROCESS

The following sequence of events outlines survey override process:

1. A fieldworker using Mobenzi Researcher checks for updates.
2. Mobenzi Researcher determines which surveys the relevant fieldworker is assigned.
3. For surveys that are assigned, Mobenzi Researcher checks whether survey override has been configured for any of them.
4. For each survey configured with a survey override, Mobenzi Researcher requests the relevant third party system to apply any changes to the survey to be sent to the mobile device. This is achieved by transmitting a serialised version of the survey (in XML format) to the third party system. The third party responds with modifications by sending back a modified version of the XML file posted. A unique ID is expected to be injected for each survey instance modified by the third party. If all survey instances are removed (i.e. the survey should not be sent to the mobile device, then no ID is required). The ID is required to uniquely define the version of the instance as modified by the third party system. If there are no changes that are required to be made, the same XML posted to the third party system should be posted back with a static custom ID (i.e. for all future update requests where there are no modifications to the instances downloaded, the same custom ID needs to be used). The snapshot of XML data is outlined below outlining which ID is being referred to by 'Custom ID':

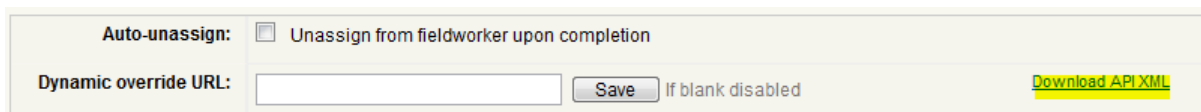
```
<?xml version="1.0" encoding="utf-16"?>
<survey id="675" name="Household Survey" revision="42">
  <instance language="EN" id="INSERT_CUSTOM_KEY_HERE" removeUponCompletion="0" expireminutes="-1">
    <fields>
      <field id="33715" required="1">
        <text>Please enter the assigned household ID:</text>
      <defaults>
```

5. The modified XML documents sent back from a third party system are inspected by Mobile Researcher. Mobenzi Researcher inspects the custom ID's sent back for each modified instance and determines whether the mobile device needs to download the modified instance or whether it

already has it. If no instances are returned by the third party system (i.e. all instances are removed), then Mobenzi Researcher instructs the mobile device to remove all local instances. If there are changes, Mobenzi Researcher instructs the Mobile Device to download the relevant changes.

CONFIGURATION

To facilitate development of the third party process handler, it possible to download sample XML for any survey. To do so, log into www.researchconsole.com and navigate to the relevant survey that requires the survey override mechanism to be setup. Once a survey is selected, click the *Design* tab and from the ribbon select *Options*. Once the survey options screen has loaded, locate the *Dynamic Override* field. As per the screenshot below, there is a link to download sample API XML to facilitate development of the override handler.



The screenshot shows a configuration panel with the following elements:

- Auto-unassign:** A checkbox labeled "Unassign from fieldworker upon completion" which is currently unchecked.
- Dynamic override URL:** A text input field that is currently empty.
- Save:** A button next to the URL field with the text "if blank disabled" to its right.
- Download API XML:** A yellow link located to the right of the Save button.

To configure the survey override mechanism once ready to be deployed, the Dynamic override URL field needs to be populated with the HTTP endpoint responsible for processing survey override requests.

When a fieldworker checks for updates and the survey override mechanism has been activated (by completing the field above and clicking save), Mobenzi Researcher will POST the serialised XML document to the URL configured. There are a few other values posted to provide context, all POST fields are documented below. The HTTP request's content type is 'application/x-www-form-urlencoded'.

| PARAMETER | Type | VALUE |
|-----------------------|---------|---|
| survey_xml | String | The survey override XML. |
| languages | String | A comma separated list of languages that are available for the fieldworker. |
| fieldworker_id | Guid | The fieldworker ID as defined by Mobile Researcher. |
| device_id | Guid | The device ID as defined by Mobile Researcher. |
| survey_id | Integer | The relevant survey ID. |

If the request to the third party system fails for whatever reason, then the update check process fails and the fieldworker receives an error notification on their device.

The table below outlines which XML elements may be updated and extended to facilitate the override mechanism:

| ELEMENT | DETAILS |
|---|---|
| Survey | The <i>name</i> attribute may be override, this overwrites the title as displayed on the mobile device |
| Survey>Instance | <p>The <i>id</i> attribute is required to be overridden in all cases.</p> <p>If an instance should be removed (i.e. not sent to a fieldworker), then the entire instance element may be removed.</p> <p>The <i>removeUponCompletion</i> attribute may be updated. A value of 0 indicates that the survey instance should not be removed after completion while a value of 1 indicates that it should.</p> <p>The <i>expireminutes</i> attribute may be updated. -1 indicates no expiry, any value greater than 0 indicates that the survey instance should expire on the fieldworker's device after the defined period.</p> |
| Survey>Instance>Fields>Field | The <i>required</i> attribute may be updated to dynamically set whether a response is required for the field or not. |
| Survey>Instance>Fields>Field>Text | The <i>text</i> element value can be overridden with custom text. |
| Survey>Instance>Fields>Field>Defaults>Default and Survey>Instance>Fields>Field>Options>Option>Defaults>Default | <p>The <i>default</i> element value can be override with a custom value, it is the responsibility of the third party service to ensure the type of the default value matches that of the field type.</p> <p>Default elements may be created to support multiple default values for repeating region. For example, if a field is contained within a repeating region, and the repeating region root field has been defaulted to 5 (i.e. repeat 5 times), then 5 default elements may be created to support a default value for each iteration.</p> |

Survey>Instance>Fields>Field> The *label* attribute may be updated.

Options>Option

The *value* attribute may be updated.

Option elements may be removed or added depending on the scenario.

To return the modified data, the modified XML should simply be returned to Mobile Researcher. The following content types are supported: text/plain, text/xml. The returned XML will only be parsed if the override attempt results in a “200 OK” [HTTP status code](#) being returned. If a response is not received within 30 seconds, the request will time out and the update process will fail.

EXTERNAL IDENTIFIERS

For a given survey, external identifiers may be configured to identify response fields that are made available through the API and the survey override mechanism. This is useful as it allows a third party system to identify entities within Mobenzi Researcher in their own way. Unique constraints are not applied against third party External Id definitions which means copies of a survey may assume duplicate identifiers which negates the need to maintain a field mapping in the third party system. In terms of the survey override mechanism, defined External Id’s may be included as attributes to the following elements:

- Survey
- Field
- Option

External identifiers are not meant to be overridden and are provided as a means to identify fields when a third party system processes a survey override request. Please note, external identifiers are not required to be setup in order to process a survey override request (field Id’s may be used to identify fields for example), however it is recommended best practise that they be configured and used to identify fields.

DOCUMENT HISTORY

| Version | Changes |
|---------|---|
| 1.0 | Initial documentation of the core Mobenzi Researcher API features. |
| 2.0 | Included documentation on how to configure and use triggers. Included documentation on how to setup and use the survey override feature. Included documentation on updating and deleting responses through the API. |
| 2.1 | Included documentation on external identifiers and the information made available as part of the survey override process. Updated documentation on the trigger notification meta-data 'customid' field. |

REFERENCES

- Overview: Microsoft ADO .NET Data Services: <http://msdn.microsoft.com/en-us/library/cc956153.aspx>
- Using Microsoft ADO .NET Data Services : <http://msdn.microsoft.com/en-us/library/cc907912.aspx>