



# **E Team Admin Guide**

## **Appendix H – REST API**

**E Team** **Software Admin Guide**

Product Version 9.7 GA | 08/31/2016

## NC4 Disclaimer

*The written and visual contents of this manual are the sole and exclusive property of NC4 Inc., and/or one of its wholly owned subsidiaries (collectively as "NC4"), and is issued to the customer solely for its own internal business purposes in connection with use of the products or services provided by NC4. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, taping, recording or information storage and retrieval systems, without the prior written consent of NC4 Inc. No part of this manual may therefore be copied, loaned or otherwise disclosed to any third party without the prior written consent of NC4 Inc. Copyright protection claimed includes all forms and matters of copyrightable material and information now allowed by applicable statutory or judicial law or hereinafter granted, including without limitation, material generated from the corresponding software programs which are displayed on the screen such as icons, screen displays, looks, etc.*

*While NC4 has exercised reasonable skill and care in producing this manual, its accuracy cannot be guaranteed. Information in this manual is subject to change without notice and does not represent a commitment on the part of NC4.*

*NC4 and the NC4 logo are trademarks of NC4 Inc.*

*ActivTravel and ActivPoint are trademarks of NC4 Inc.*

*E Team and the E Team logo are trademarks of NC4 Public Sector LLC.*

*E•SPONDER, E•SPONDER Express, and the E•SPONDER and E•SPONDER Express logos are trademarks of NC4 Public Sector LLC.*

*NC4 Street Smart and the NC4 Street Smart logo are trademarks of NC4 Public Sector LLC.*

*NC4 Signal and the NC4 Signal logo are trademarks of NC4 Public Sector LLC.*

*NC4 Risk Center and NC4 Mission Center are trademarks of NC4 Inc.*

*All other brand and product names and logos are the trademarks of their respective holders.*

*NC4 Inc.*

*100 N. Sepulveda Blvd., Suite 200*

*El Segundo, CA 90245*

*© 2011-2016 NC4 Inc. All Rights Reserved.*

## Table of Contents

<b>TABLE OF CONTENTS.....</b>	<b>3</b>
<b>SECTION 1 – INTRODUCTION.....</b>	<b>4</b>
<b>NC4 Support Center .....</b>	<b>4</b>
<b>Get API Version .....</b>	<b>5</b>
<b>Since: 0.2.2.....</b>	<b>5</b>
<b>Authentication .....</b>	<b>7</b>
<b>Login .....</b>	<b>7</b>
<b>Logout.....</b>	<b>8</b>
<b>Get a List of a User's Reports .....</b>	<b>8</b>
<b>See a List of Specific Reports .....</b>	<b>10</b>
<b>See a Specific Report.....</b>	<b>12</b>
<b>Create a New Report .....</b>	<b>13</b>
<b>Create a New Custom Form Report .....</b>	<b>15</b>
<b>Update an Existing Report.....</b>	<b>16</b>
<b>Delete a Report .....</b>	<b>17</b>
<b>Failures.....</b>	<b>17</b>
<b>Typical Uses.....</b>	<b>17</b>
<b>Upgrading from Previous Version .....</b>	<b>18</b>
<b>Login .....</b>	<b>18</b>
Old .....	18
New.....	18
<b>Get list of reports .....</b>	<b>19</b>
Old .....	19
New.....	19

## Section 1 – Introduction

This document is intended for use by all individuals deploying or using the REST tool. Direct questions regarding the REST API to [support@NC4.us](mailto:support@NC4.us).

### NC4 Support Center

---

The Support Center is available to answer any questions regarding the E Team application.

The Support Center is available 24 hours a day, 7 days a week.

Phone: 800-209-2312

Email: [support@NC4.us](mailto:support@NC4.us)

Support Website: <https://supportcenter.nc4.com>

## Section 2 – Overview

A REST API request is made of 4 pieces: the Method, the Resource, the Authentication, and the Entry. Not all requests need an Entry and Authentication, but all requests need the first two.

- The REST API Request is an HTTP Request.
- The Method is a parameter: `_m` (method)
- The Resource is a URI parameter: `_r` (resource)
- The Authentication is two parameters: `_u` (username) & `_p` (password)
- The Entry is a set of key-value paired parameters.
- Every REST request is made to same URL using the HTTP POST Method.



*GET is acceptable as well, but is inherently limited and is not recommended.*

*All other Methods will be rejected.*

- Everything else is not passed in an HTTP header but rather in URL parameters - `page?key=value&parameter=value&...etc.`

## Get API Version

Since: 0.2.2

Using this method you will be able to retrieve the REST API version as well as the E Team version that the REST API hits.

### Headers :

```
POST /REST HTTP/1.1
Host: <rest-server-uri>
```

### Parameters :

<code>_m</code>	GET
<code>_r</code>	version
<code>_url</code>	<code>http(s)://&lt;etteam-server&gt;/&lt;instance-name&gt;</code>

If your request was successful, you should receive 2 data points:

1. E Team version
2. REST API version

\_f=j

```
[
  {
    "ETeamVersion": "XYZ" , "RESTVersion": "0.2.2"
  }
]
```

\_f=x

```
<items>
  <item ETeamVersion ="XYZ" RESTVersion ="0.2.2" />
</ items >
```

\_f=c

```
"ETeamVersion"," RESTVersion"
"XYZ"," 0.2.2"
```

\_f=t

```
<table>
  <tr>
    <th> ETeamVersion </th><th> RESTVersion </th>
  </tr>
  <tr>
    <td> XYZ </td><td> 0.2.2 </td>
  </tr>
</table>
```

## Authentication

---

Complete every REST API request, except to get API Version, is authenticated by using custom server-side authentication over SSL.

You can pass in the E Team URL, username and password on every request, and the system will try to authenticate every time. Another strategy is to login once to create a session, and never pass in the username and password again, that way the REST API server will use the authentication stored in the session.

## Login

This logs you in and creates a session cookie. Once logged in, you no longer need to login on every request. This is the only kind of "state" that is kept.

**Headers :**

```
POST /REST HTTP/1.1
Host: <rest-server-uri>
```

**Parameters :**

<code>_m</code>	GET
<code>_r</code>	login
<code>_url</code>	http(s)://<etteam-server>/<instance-name>
<code>_u</code>	<username>
<code>_p</code>	<password>

If your request was successful, you should receive:

- HTTP Response Status Code: 200 OK

## Logout

It is not required, but recommended that you log out at the end of your session.

### Headers :

```
POST /REST HTTP/1.1
Host: <rest-server-uri>
```

### Parameters :

<code>_m</code>	DELETE
<code>_r</code>	login

If your request was successful, you should receive:

- HTTP Response Status Code: 200 OK

## Get a List of a User's Reports

This returns a list of all reports that are available to this user over the REST API. This may not return the complete list of reports available to the user if not all are enabled under the REST API. This respects the user's rights and permissions in the system. Additionally, a user will not be able to view all reports of a specific type, even if they are allowed to see that report type if they have been banned from viewing specific reports of that type.

Here we introduce a new optional parameter: `_f` (format) which represents the user's desired data output format. This parameter applies for the GET / list, and GET / report requests. The possible values are j (json [default]), c (csv), t (html <table>) and x (XML).

### Headers :

```
POST /REST HTTP/1.1
Host: <rest-server-uri>
```

### Parameters :

<code>_m</code>	GET
<code>_r</code>	report
<code>[_f]</code>	{j c t x}



If your request was successful, you should receive 3 data points:

1. Internal report type name (e.g. “incident”)
  - This is the value that will later be needed when requesting reports of specific incident type
2. External report type name (e.g. “Incident”)
3. Count of reports of such type accessible to user (e.g. “5”)

And the results will be formatted per the `_f` parameter

`_f=j`

```
[
  {
    "internal": "agency_sitrep", "external": "Agency SitRep",
    "count": "0"
  },
  {
    "internal": "personnel", "external": "Personnel", "count": "7"
  }
]
```

`_f=x`

```
<items>
  <item internal="agency_sitrep" external="Agency SitRep" count="0" />
  < item internal="personnel" external="Personnel" count="7" />
</ items >
```

`_f=c`

```
"internal","external","count"
"agency_sitrep","Agency SitRep","0"
"personnel", "Personnel", "7"
```

`_f=t`

```
<table>
  <tr>
    <th>internal</th><th>external</th><th>count</th>
```

```

</tr>

<tr>

    <td>agency_sitrep</td><td> Agency SitRep </td><td>0</td>

</tr>

<tr>

    <td> personnel </td><td> Personnel </td><td>7</td>

</tr>

</table>

```

## See a List of Specific Reports

### Headers :

```

POST /REST HTTP/1.1
Host: <rest-server-uri>

```

### Parameters :

<code>_m</code>	GET
<code>_r</code>	report
<code>_t</code>	<internal_report_type_name>
<code>[_f]</code>	{ j   c   t   x }
<code>[_k]</code>	{ <empty>   <comma-separated list of keys> }

Here we introduce a new optional parameter: `_k` (keys) which is a comma-separated list of report keys to be returned. So if the report is a collection of key-value pairs, to get ALL the data inside the report, leave the `_k` parameter blank. To see a very very short version of the report, set `_k` to be “reportId,id,reportType,status”. And you will get only those 4 columns. This replaces the previously static implementation of “short” vs. “long” formats and it is up to the caller to specify which columns (keys) they require. This parameter is valid when requesting to see a list of specific reports and when requesting to see a specific report.

If your request was successful, you should receive:

```

_f=j

```

```

[

```

```

{"currentObjectives":"Life Safety Crime Scene Preservation Crime Scene
Investigation", "catBNoOfSites":"3", "reportType":"agency_sitrep",
"modifiedBy":"eswerw -117953833425905025870", "entityMap":"{}",
"cityName":"Orlando", "relatedReportId":"", "resFiveDep":"0",
"catCNoOfSites":"0", "isCenterActive":"Y", "modifierProfileId":.....},

{"currentObjectives":"","catBNoOfSites":"0",
"reportType":"agency_sitrep", "modifiedBy":"ETR6PV21120wewer5609-eteam-
12sdfsrrwe2287616237507701507", "entityMap":"{}", "cityName":"Orlando",
"relatedReportId":"ET12008135609-eteam-122806804360911301310",
"resFiveDep":"0", "catCNoOfSites":"0", .....
}
]
_f=x

<items>

<item currentObjectives="Life...." " catBNoOfSites="3"
reportType="agency_sitrep" modifiedBy="ehk62sql15142007115549-
r62demo-117953833425905025870" entityMap="{}" cityName="Orlando"
relatedReportId="" resFiveDep="0" catCNoOfSites="0"
isCenterActive="Y" modif... />

<item currentObjectives="" catBNoOfSites="0"
reportType="agency_sitrep" modifiedBy="ETR6PV2112008135609-eteam-
12287616237507701507" entityMap="{}" cityName="Orlando"
relatedReportId="ETR6PV2112008135609-eteam-122806804360911301310"...
catANoOfSites="0"/>

</items>

_f=c

reportType,currentStatus,reportId,...
agency_sitrep,70014,eh5549-r62demo-118178581402207102183,...
agency_sitrep,70014,ehk62sq7115549-r62demo-118178557347608548560,...
agency_sitrep,70015,ETR6PV211609-eteam-123101244180301060290,...

_f=t

<table>

<tr>

<th> reportType</th><th>currentStatus</th><th> reportId </th>

</tr>

<tr>

<td>agency_sitrep</td><td>70014</td><td> eh5549-r62demo-

```

```

118178581402207102183</td>

</tr>

<tr>

<td>agency_sitrep</td><td>70014</td><td>ehk62sq7115549-r62demo-
118178557347608548560</td>

</tr>

<tr>

<td>agency_sitrep</td><td>70016</td> <td> ETR6PV211609-eteam-
123101244180301060290</td>

</tr>

</table>

```

## See a Specific Report

### Headers:

```

POST /REST HTTP/1.1
Host: <rest-server-uri>

```

### Parameters:

<code>_m</code>	GET
<code>_r</code>	report
<code>_t</code>	<internal_report_type_name>
<code>[_f]</code>	{ j   c   t   x }
<code>[_k]</code>	{ <empty>   <comma-separated list of keys> }
<code>_i</code>	<report_id>

If your request was successful, you should receive:

```

_f=j

[
  {
    "currentObjectives": "Life Safety Crime Scene Preservation Crime Scene
Investigation",
    "catBNoOfSites": "3",
    "reportType": "agency_sitrep",

```

```

    "modifiedBy":"eswerw -117953833425905025870", "entityMap":"{}",
    "cityName":"Orlando", "relatedReportId":"", "resFiveDep":"0",
    "catCNoOfSites":"0", "isCenterActive":"Y", "modifierProfileId":.....}
  ]
_f=x

<items>

  <item currentObjectives="Life...." " catBNoOfSites="3"
  reportType="agency_sitrep" modifiedBy="ehk62sql15142007115549-
  r62demo-117953833425905025870" entityMap="{" cityName="Orlando"
  relatedReportId="" resFiveDep="0" catCNoOfSites="0"
  isCenterActive="Y" modif... />

</items>

_f=c

reportType,currentStatus,reportId,...
agency_sitrep,70014,eh5549-r62demo-118178581402207102183,...

_f=t

<table>

  <tr>

    <th> reportType</th><th>currentStatus</th><th>reportId</th>

  </tr>

  <tr>

    <td>agency_sitrep</td><td>70014</td><td> eh5549-r62demo-
    118178581402207102183</td>

  </tr>

</table>

```

## Create a New Report

### Headers:

```

POST /REST HTTP/1.1
Host: <rest-server-uri>

```

#### Parameters:

<code>_m</code>	PUT
<code>_r</code>	report
<code>_t</code>	<internal_report_type_name>
<KVPs>	<key-value pairs that describe the report>

Here we introduce yet another kind of parameter - <KVP> - which denotes a set of key-value pairs – key1=value1&key2=value2&key3=value3&keyN=valueN. This set of parameters is required when creating a new report or updating an existing one. And each “key” is a report field, and each “value” is a value for that field. Not every field is required to create a report, and when you create a report without a required field you will receive a server error. Required fields are documented in the ETeam Web Services specification. You can also use the sample client to create a report and see what errors are produced by the server.

When updating a report, it is only necessary to specify the fields you want to update with their new value.



*These are the only request parameters that do not start with an underscore (“\_”) because they are essentially free-text and vary depending on the particular report type you are working with.*

These parameters are valid when requesting to create a new report and when requesting to update a specific report.

If your request was successful, you should receive the id of the newly created report.

```
_f=j
[
  { "id": "abc-123-xyz" }
]

_f=x
<items>
  <item id =" abc-123-xyz" />
</ items >

_f=c
```

```
"id"

"abc-123-xyz"

_f=t

<table>

  <tr>

    <th>id</th>

  </tr>

  <tr>

    <td> abc-123-xyz</td>

  </tr>

</table>
```

## Create a New Custom Form Report

---

Custom Form data is kept in an XML format. The data is described in the XSD document that can be obtained through the preview page of E Team Form Builder Manager. A sample XML message can also be obtained on that same page. When a customer wants to use web services they will need to extract those 2 pieces of information to give to whoever will implement the web service. Also, the implementer will need to have knowledge of any rules that the form follows. For instance, if a field called `custom_form_v1` is set to the field value `Lastname` through a rule, this will have to be done in the XML string sent in. The XML will not be going through the Frevvo application and therefore no rules are applied to the fields unless the form is updated at a later time through the E Team application using a browser.

Web services is set up to accept a hashmap of fields/values. In the Custom Form case, those values will be only the XML string and the Custom Form's name. Upon accepting the data, the application will check the data against the Custom Form's schema held in the database. If it passes, it will follow standard code to create the form. In the event that the system fails to find the Custom Form or it does not pass the schema check, the form will be rejected and the error will be returned.

**Headers :**

```
POST /REST HTTP/1.1
Host: <rest-server-uri>
```

**Parameters :**

<b>_m</b>	PUT
<b>_r</b>	report
<b>_t</b>	custom_form
<b>&lt;KVPs&gt;</b>	<key-value pairs that describe the report> (see below)

KVP for a Custom Form report consist of only 2 values: formName and frevvoData. The formName is the name of the Custom Form. frevvoData is a properly formatted XML string that contains the data in the Custom Form. This would be built by using the XSD and sample data extracted from the form builder as stated above.

## Update an Existing Report

---

**Headers :**

```
POST /REST HTTP/1.1
Host: <rest-server-uri>
```

**Parameters :**

<b>_m</b>	POST
<b>_r</b>	report
<b>_t</b>	<internal_report_type_name>
<b>_i</b>	<report_id>
<b>&lt;KVPs&gt;</b>	<key-value pairs that describe the report>

If your request was successful, you should receive:

- HTTP/1.1 200 OK



## Delete a Report

---

### Headers:

```
POST /REST HTTP/1.1
Host: <rest-server-uri>
```

### Parameters:

<code>_m</code>	POST
<code>_r</code>	report
<code>_t</code>	<internal_report_type_name>
<code>_i</code>	<report_id>

If your request was successful, you should receive:

- HTTP/1.1 204 No Content

## Failures

---

If unable to authenticate your request, the server will respond with:

- HTTP/1.1 401 Unauthorized Your username/password is invalid.

After you're done debugging your client, the most likely culprit of this error will be user error. For example, trying to delete a report that doesn't belong to them but they can still see.

## Typical Uses

---

Most consumer facing clients will probably act this way:

1. A user enters their username/password combination.
  - The client software sends a request to with the credentials
  - E Team authenticates the user
2. The client software sends a request for list of report types
  - E Team returns list of report types

3. The user picks the report type they want to drill down on
  - The client sends a request for reports of specific type
  - E Team returns list of reports of that type
4. The user picks a report they want to view
  - The client sends a request to view a specific report
  - E Team returns all the data for a specific report
5. The user edits a report
  - The client sends the updates
6. The user decides to create a new report
  - The client sends a new entry

## Upgrading from Previous Version

---

The previous version of the REST-like API used separate end-points for the various CRUD operations. Now, there's a single end-point with varying parameters. Here's a sampling of how to convert previous calls from version 0.1 to version 0.2:

### Login

#### Old

`http://<server>:<port>/<service>/servlet/login?etteamurl=http://preview.nc4.us/etteam&etteamuser=vbokin&etteampass=*****`

#### New

`http://<server>:<port>/<service>/servlet/REST?_m=GET&_r=login&_url=<url>&_p=<password>&_u=<username>`

## Get list of reports

Old

`http://<server>:<port>/<service>/servlet/getReport?t=agency_sitrep&l=long&f=csv`

New

`http://<server>:<port>/<service>/servlet/REST?_m=GET&_r=report&_t=agency_sitrep&_f=c`



*Examples are shown as a GET request (parameters attached to the URL after ? question mark) but we highly recommend that you submit all requests with POST, with the parameters attached in the body of the post.*