

# NCheck Bio Attendance Integration Manual



9/23/2020

---

# CONTENTS

---

1	INTRODUCTION .....	1
2	NCHECK API .....	2
2.1	API credentials.....	2
2.2	Authentication token.....	2
2.3	API's .....	3
2.3.1	User employee data .....	3
2.3.2	Biometric data .....	9
2.3.3	Attendance events.....	13
3	PERIPHERAL INTEGRATION API .....	24
3.1	Third Party sensor Integration with Android Standard client .....	25
3.1.1	Components of the application .....	25
3.1.2	Create Scanner integration application.....	25
3.1.3	Testing scanner service .....	29
4	ACCESS CONTROL INTEGRATION.....	30
4.1	Parameters .....	31
4.2	Trigger access control.....	31
5	CUSTOM FIELDS CONFIGURATION .....	33
5.1	Add custom fields .....	33
6	IMPORT EXPORT .....	34
6.1	Import total work hour report in Tally ERP system.....	34
6.2	Import total work hour report in Quickbooks.....	37
6.2.1	Import work hour data (Daily) from NCheck Bio Attendance as a CSV file.....	37
6.2.2	Manipulate work hour report to format data and remove unnecessary data .....	39
6.2.3	Download sample IIF file .....	39
6.2.4	Update correct header information using <i>Timer list</i> IIF file .....	40
6.2.5	Merge manipulated report data.....	40
6.2.6	Import to QuickBooks.....	41
7	CONTACT .....	43

---

# 1 INTRODUCTION

---

NCheck Bio Attendance supports integrating with other systems. This document explains available integration mechanisms.

1. NCheck API

Programmers can use NCheck API to integrate attendance system data from/to other systems.

2. Peripheral integration API

Programmers can write peripheral integration module using peripheral integration module to use peripherals such as temperature sensors, displays, fingerprint sensor, door control device etc.

3. Access control integration

Administrators can conditionally trigger door controller executable or URL to control access gates, doors etc.

4. Custom field configuration

Administrators can request configuration of additional data fields for attendance system data to send/receive to/from other systems.

5. Import export

Administrators can export/import attendance system data to/from other systems such as Payroll

---

## 2 NCHECK API

---

NCheck API can be used by programmers to retrieve and update attendance system data in NCheck Bio Attendance. It can manage following information.

- Users/Employees
- Biometrics data
- Attendance events

### 2.1 API credentials

API user name and password can be generate by NCheck Bio Attendance server administrator as mentioned in [generate API access credentials](#) section. Programmers can use the username and password to generate a session authentication token to use in API method calls.

Authentication token expires in one week. Once it expired, the new authentication token can be generated using the same username and password.

Refer section [Authentication token](#) for more detail about generating authentication token.

### 2.2 Authentication token

Session authentication token is used to authenticate API method calls. Authentication token retrieval request call using C# is shown in [Figure 2.1](#) Getting authentication token using C# code

```
WebRequest req = WebRequest.Create(server_url + "/oauth/token");
req.Method = "POST";
req.Headers["Authorization"] = "Basic " + Convert.ToBase64String(Encoding.Default.GetBytes("ncheck:"));
req.ContentType = "application/x-www-form-urlencoded";
var postData = "grant_type=password&username=" + user_name + "&password=" + password;
var data = Encoding.ASCII.GetBytes(postData);
req.ContentLength = data.Length;
using (var stream = req.GetRequestStream())
{
    stream.Write(data, 0, data.Length);
}
HttpWebResponse resp = req.GetResponse() as HttpWebResponse;
if (resp.StatusCode == HttpStatusCode.OK)
{
    var responseString = new StreamReader(resp.GetResponseStream()).ReadToEnd();
    JObject json = JObject.Parse(responseString);
    String access_token = json["access_token"].Value<string>();
}
```

*Figure 2.1 Getting authentication token using C# code*

In the code following variables should be replaced with the given values

Table 2.1 Variables of C# code for getting access token using

Variable	Value
Server_url	The URL of the NCheck Bio Attendance server
User_name	Username for the API
password	Password for the API

If the response is successful, the access token will be extracted from the *access\_token* variable.

## 2.3 API's

Once the access token has retrieved, API's can be used to manage data as follows.

1. *User employee data*  
NCheck Bio Attendance APIs allows to add, update, delete user/employee and retrieve user/employee data from the system
2. *Biometric data*  
Add/delete biometric images (Face, finger, and iris).
3. *Attendance events*  
Retrieve/delete user/employee attendance events.

### 2.3.1 User employee data

NCheck Bio Attendance is providing following APIs to manage user employee data

1. *Add/update user*
2. *Delete User*
3. *Get user*

#### 2.3.1.1 Add/update user

This API can be used for add new user or change user details as shown in [Table 2.2](#) parameters of the JSON object.

- Request: */api/ncheck/user*
- Method: *POST*
- Body  
User/employee as a *JSON object* with following information. [Table 2.2](#) parameters of the JSON object is showing the parameters required in the json object. [Figure 2.2](#) JSON object for the body of add/update user request is showing the sample *JSON object* with parameters.

Table 2.2 parameters of the JSON object

Parameter	Type	Description	Availability
employeeCode	string	<ul style="list-style-type: none"> <li>Unique identification code for a user.</li> </ul>	Required

		<ul style="list-style-type: none"> <li>To update the details, add the employee code of the existing user.</li> </ul>	
<b>firstName</b>	string	First name of the user	Required
<b>lastName</b>	string	Last name of the user	Required
<b>email</b>	string	Email address	Optional
<b>loginName</b>	string	Login name for the user. Login cannot change on update	Optional
<b>password</b>	string	Password of the user. Password cannot be changed on update	Optional
<b>status</b>	string	Status of the user as <ol style="list-style-type: none"> <li>0 for Active</li> <li>1 for Disabled</li> </ol>	Optional

```
{
  "employeeCode": "13312",
  "firstName": "Peter",
  "lastName": "Hanzward",
  "loginName": "peter",
  "password": "hasnsward",
  "email": "peter@hanzward.com",
  "status": 0
}
```

Figure 2.2 JSON object for the body of add/update user request

- Response

Response is a *JSON Object* with following information.

- statusCode

*String* type parameter to show the status of the request. This is a required field. Available status code is shown in below.

Table 2.3 Status codes in the API response for add/update user

Status code	Description
<b>INVALID_EMAIL</b>	Email address is invalid
<b>FIRST_NAME_EMPTY</b>	First name is empty
<b>Last_NAME_EMPTY</b>	Last name is empty
<b>INVALID_EMPCODE</b>	Employee code is already used
<b>ERROR</b>	System error (Need to check server logs for more details)
<b>SUCCESS</b>	Successful

- statusDescription  
*String* parameter to show the description of the status code. This is an optional field.
- returnValue  
Added/edited user details as *json object* if the status code is *success* as shown in [Figure 2.3](#) Json response for add/update user

```
{
  "statusCode": "SUCCESS",
  "statusDescription": "Successfully updated the person",
  "returnValue": {
    "employeeCode": "13312",
    "firstName": "Peter",
    "lastName": "Hanzward",
    "email": "Peter@Hanzward.com",
    "status": 0,
    "loginName": "Peter",
    "password": "Hanzward"
  }
}
```

Figure 2.3 Json response for add/update user

[Figure 2.4](#) C# code to add an employee is showing sample C# code to add new user.

```

WebRequest req = WebRequest.Create(server_url + "/api/ncheck/user");
SetSSLValidator();
req.Method = "POST";
//add access token
req.Headers["Authorization"] = "Bearer " + access_token;
req.ContentType = "application/json; charset=utf-8";

//employee details as json object
JsonObject json = new JsonObject();
json.Add("employeeCode", "13312");
json.Add("firstName", "Peter");
json.Add("lastName", "Hanzward");
json.Add("loginName", "Peter");
json.Add("password", "Hanzward");
json.Add("email", "Peter@Hanzward.com");
json.Add("status", "0");

var postData = JsonConvert.SerializeObject(json);
var data = Encoding.UTF8.GetBytes(postData);
req.ContentLength = data.Length;
using (var stream = req.GetRequestStream())
{
    stream.Write(data, 0, data.Length);
}
HttpWebResponse resp = req.GetResponse() as HttpWebResponse;
if (resp.StatusCode == HttpStatusCode.OK)
{
    var responseString = new StreamReader(resp.GetResponseStream()).ReadToEnd();
    JsonObject jsonResponse = JsonObject.Parse(responseString);
}

```

Figure 2.4 C# code to add an employee

### 2.3.1.2 Delete User

This API is using to delete an existing user.

- Request: `/api/ncheck/user/<employeeCode>/`
- Method: `DELETE`
- Parameters  
API parameters to delete user

Table 2.4 API parameters for delete user

Parameter	Type	Description	Availability
<b>employeeCode</b>	string	Unique identification code for a user	Required



- Response

Status codes are shown in [Table 2.5](#) Status codes in the API response for delete user. If the status code is success, a JSON object will be received with the deleted person details as shown in [Figure 2.5](#) JSON response for delete user.

*Table 2.5 Status codes in the API response for delete user API*

Status code	Description
USER_NOT_AVAILABLE	User is not found.
ERROR	System error (Need to check server logs for more information)
SUCCESS	Successful

```
{
  "statusCode": "SUCCESS",
  "statusDescription": "Successfully deleted the person",
  "returnValue": {
    "employeeCode": "13312",
    "firstName": "Peter",
    "lastName": "Hanzward",
    "email": "peter@hanzward.com",
    "status": 0,
    "loginName": null,
    "password": null
  }
}
```

*Figure 2.5 JSON response for delete user*

[Figure 2.6](#) C# code sample to delete an employee Is showing the C# code sample to delete an

```
WebRequest req = WebRequest.Create(server_url + "/api/ncheck/user/13312/");
req.Method = "DELETE";
//add access token
req.Headers["Authorization"] = "Bearer " + access_token;
req.ContentType = "application/json; charset=utf-8";
HttpWebResponse resp = req.GetResponse() as HttpWebResponse;
if (resp.StatusCode == HttpStatusCode.OK)
{
    var responseString = new StreamReader(resp.GetResponseStream()).ReadToEnd();
    JObject jsonResponse = JObject.Parse(responseString);
}
```

*Figure 2.6 C# code sample to delete an employee*

user.

### 2.3.1.3 Get user

Existing user detail can be retrieved from this API as shown in [Figure 2.7](#).

- Request: `/api/ncheck/user?code=<employeeCode>`

- Method: *GET*
- Parameters

*Table 2.6 API parameters for get user*

Parameter	Type	Description	Availability
<b>employeeCode</b>	string	Unique identification code for a user	Required

- Response

[Table 2.7](#) Status codes in the API response for get user API is showing the status code of the API response. [Figure 2.7](#) JSON response for get user API is showing the Json object in the response.

*Table 2.7 Status codes in the API response for get user API*

Status code	Description
<b>USER_NOT_AVAILABLE</b>	User is not found
<b>ERROR</b>	System error (Need to check server logs for more details)
<b>SUCCESS</b>	Successful

```
{
  "statusCode": "SUCCESS",
  "statusDescription": "Successfully get the person",
  "returnValue": {
    "employeeCode": "13312",
    "firstName": "Peter",
    "lastName": "Hanzward",
    "email": "peter@hanzward.com",
    "status": 0,
    "loginName": "peter@hanzward.com",
    "password": null
  }
}
```

*Figure 2.7 JSON response for get user API*

Figure 2.8 Is showing the C# sample code to get the user.

```
WebRequest req = WebRequest.Create(server_url + "/api/ncheck/user?code=13312");
req.Method = "GET";
//add access token
req.Headers["Authorization"] = "Bearer " + access_token;
req.ContentType = "application/json; charset=utf-8";
HttpWebResponse resp = req.GetResponse() as HttpWebResponse;
if (resp.StatusCode == HttpStatusCode.OK)
{
    var responseString = new StreamReader(resp.GetResponseStream()).ReadToEnd();
    JObject jsonResponse = JObject.Parse(responseString);
}
```

Figure 2.8 C# code sample to get an employee detail

### 2.3.2 Biometric data

User/employee biometric data images as face, fingerprint, and iris images can be managed using following APIs.

1. *Update biometric*
2. *Delete biometric*

#### 2.3.2.1 Update biometric

User biometric(s) including face, finger and iris can be enrolled using this API.

- Request: `/api/ncheck/biometric?code=<employeeCode>/`
- Method: *POST*
- Parameters:

Table 2.8 API parameters for update biometric

Parameter	Type	Description	Availability
<b>employeeCode</b>	string	Unique identification code for a user	Required

- Body  
JSON Biometric Array with parameters as shown in [Table 2.8](#) API parameters for update biometric. Example JSON body has shown in [Figure 2.9](#) Biometrics Json array body for update biometric API.

Table 2.9 Parameters in the JSON body to update biometrics API

Parameter	Type	Description	Availability
<b>modality</b>	string	Biometric modality (face, finger, iris)	Required

<b>image</b>	byte[]	Biometric raw Image.	Optional
<b>template</b>	byte[]	Biometric template. Required is no image.	Optional

```
[
  {
    "modality": "face",
    "image": "/9j/4Sc5RXhpZ...",
    "template": null
  },
  {
    "modality": "face",
    "image": "/9j/4SUxRXhpZgAATU0AKgAA...",
    "template": null
  }
]
```

Figure 2.9 Biometrics Json array body for update biometric API

- Response

JSON response has shown in [Figure 2.10](#) Json response for add biometric API

```
{
  "statusCode": "SUCCESS",
  "statusDescription": "Successfully enrolled biometrics",
  "returnValue": [
    {
      "statusCode": "SUCCESS",
      "statusDescription": "Successfully enrolled biometrics",
      "returnValue": null
    },
    {
      "statusCode": "SUCCESS",
      "statusDescription": "Successfully enrolled biometrics",
      "returnValue": null
    }
  ]
}
```

Figure 2.10 Json response for add biometric API

- statusCode

*String* parameter which shows the status of the request. This is a required field. Status codes are showing in following table.

Table 2.10 Status codes in the API response for update biometrics API

Status code	Description
INVALID_PARAMETERS	No biometric data sent
USER_NOT_AVAILABLE	No user available for the given employee code
SUCCESS	Successful

- statusDescription  
Detail of the status as a *String*. This is an optional field.
- returnValue  
Individual biometric result array as a *Json object* for successful requests with the following information.
  - \* statusCode  
Status of the enrolled biometric. The status codes are showing in below.

Table 2.11 Status codes for individual biometric result in the API response for update biometrics API

Status code	Description
ENROLL_FAILED	The biometric has enrolled to different user
FAILED_TO_EXTRACT	Failed to extract the biometrics
SUCCESS	Successful

- \* statusDescription  
Description of the status as a *String*.
- \* ReturnVale  
This is always null.

Figure 2.11 is showing a sample C# code to update biometrics.

```
WebRequest req = WebRequest.Create(server_url + "/api/ncheck/biometric?code=13312");
req.Method = "POST";
//add access token
req.Headers["Authorization"] = "Bearer " + access_token;
req.ContentType = "application/json; charset=utf-8";

//biometrics list to add JSON biometric objects
List<JObject> biometrics = new List<JObject>();

//JSON object for face biometric
JObject jsonFace1 = new JObject();
jsonFace1.Add("modality", "face");
jsonFace1.Add("image", image1);
jsonFace1.Add("template", null);

//JSON object for face biometric
JObject jsonFace2 = new JObject();
jsonFace2.Add("modality", "face");
jsonFace2.Add("image", image2);
jsonFace2.Add("template", null);

biometrics.Add(jsonFace1);
biometrics.Add(jsonFace2);

var postData = JsonConvert.SerializeObject(biometrics);
var data = Encoding.UTF8.GetBytes(postData);
req.ContentLength = data.Length;
using (var stream = req.GetRequestStream())
{
    stream.Write(data, 0, data.Length);
}
HttpWebResponse resp = req.GetResponse() as HttpWebResponse;
if (resp.StatusCode == HttpStatusCode.OK)
{
    var responseString = new StreamReader(resp.GetResponseStream()).ReadToEnd();
    JObject jsonResponse = JObject.Parse(responseString);
}
```

Figure 2.11 C# code sample to update biometrics

#### 2.3.2.2 Delete biometric

All user biometric(s) can be deleted using this API.

- Request: `/api/ncheck/biometric/<employeeCode>/`
- Method: DELETE
- Parameters:

Table 2.12 parameters for delete biometric

Parameter	Type	Description	Availability
employeeCode	string	Unique identification code for a user	Required

- Response:

JSON response is shown in [Figure 2.12](#) JSON response for delete biometric API

```
{
  "statusCode": "SUCCESS",
  "statusDescription": "Successfully deleted the person",
  "returnValue": null
}
```

Figure 2.12 JSON response for delete biometric API

- statusCode

Request status in *String*. This is a required field. Available status codes are shown in below.

Table 2.13 Status codes in the API response for delete biometric

Status code	Description
USER_NOT_AVAILABLE	User is not found
ERROR	System error (Need to check server logs for more details)
SUCCESS	Successfull

- statusDescription

Detail of the status as *String*. This is a required field.

- returnValue: Null

[Figure 2.13](#) is showing the C# code sample for delete user biometric(s).

```
WebRequest req = WebRequest.Create(server_url + "/api/nccheck/biometric/13312");
SetSSLValidator();
req.Method = "DELETE";
//add access token
req.Headers["Authorization"] = "Bearer " + access_token;
req.ContentType = "application/json; charset=utf-8";
HttpWebResponse resp = req.GetResponse() as HttpWebResponse;
if (resp.StatusCode == HttpStatusCode.OK)
{
    var responseString = new StreamReader(resp.GetResponseStream()).ReadToEnd();
    JObject jsonResponse = JObject.Parse(responseString);
}
```

Figure 2.13 C# code sample to delete biometrics

### 2.3.3 Attendance events

User/employee attendance data can be managed using following APIs

1. Get attendance events
2. Add events

### 3. Delete events

#### 2.3.3.1 Get attendance events

Get attendance event details of selected user for a given time range.

- Request: `/api/ncheck/event?code=<employeeCode>&from=<fromDateTime>&to=<toDateTime>`
- Method: `GET`
- Parameters:

Table 2.14 parameters for get attendance events API

Parameter	Type	Description	Availability
<b>employeeCode</b>	string	Unique identification code for a user	Optional
<b>toDateTime</b>	string	Formatted date time string as yyyy-MMdd HH:mm:ss	Required
<b>toDateTime</b>	string	Formatted date time string as yyyy-MMdd HH:mm:ss	Required

- Response  
All attendance events as *Json array* with below parameters.

```
{
  "statusCode": "SUCCESS",
  "statusDescription": "Successfully get the event logs",
  "returnValue": [
    {
      "employeeCode": "abc",
      "inTime": "2020-09-25 13:23:22",
      "outTime": "2020-09-25 13:23:28",
      "shiftName": null,
      "tzOffset": 19800
    },
    {
      "employeeCode": "abc",
      "inTime": "2020-09-25 13:23:42",
      "outTime": "2020-09-25 13:26:18",
      "shiftName": null,
      "tzOffset": 19800
    },
    {
      "employeeCode": "abc",
      "inTime": "2020-09-25 14:10:33",
      "outTime": "2020-09-25 14:19:58",
      "shiftName": null,
      "tzOffset": 19800
    }
  ]
}
```

Figure 2.14 JSON response for get attendance events API



- **statusCode**  
Status of the requested event details as *String*. This is a required field.  
Available status codes are shown in below.

*Table 2.15 Status codes in the response for get attendance events API*

Status code	Description
<b>USER_NOT_AVAILABLE</b>	User is not found
<b>INVALID_TIME_FORMAT</b>	Date format is invalid
<b>ERROR</b>	System error (Need to check server logs for more details)
<b>SUCCESS</b>	Successful

- **statusDescription**  
Detail of the status as *String*. This is a required field
- **returnValue**  
Requested event details as *Json array* if the status code is *Success* with below information.

- \* **statusCode**  
Status of the event detail as *String*. This is a required field.  
Status codes are shown in below

*Table 2.16 Status codes for each individual attendance records in the get attendance events API*

Status code	Description
<b>USER_NOT_AVAILABLE</b>	User is not found
<b>INVALID_TIME_FORMAT</b>	Date format is invalid
<b>ERROR</b>	System error (Need to check server logs for more details)
<b>SUCCESS</b>	Successful

- \* **statusDescription**  
Detail of the status as *String*. This is a required field
- \* **returnValue**  
*Json object* containing event details if the status code is *Success*.
  - **employeeCode**  
Employee code of the user as *String*
  - **inTime**  
Checked-in time as a date time string, formatted in yyyy-MM-dd HH:mm:ss
  - **outTime**  
Checked-out time as a date time string, formatted in yyyy-MM-dd HH:mm:ss

- Shift  
Shift name as *String*. If empty, default shift is selected.
- tzOffset  
UTC timezone offset *int*. Default is 0.

*Figure 2.15* Is showing the C# sample for get attendance events for a user

```
DateTime dtFrom = DateTime.Parse("2020-09-24");
DateTime dtTo = DateTime.Parse("2020-09-26");
string dateRagne = string.Format("from={0}&to={1}", Uri.EscapeDataString(dtFrom.Date.ToString("yyyy-MM-dd HH:mm:ss")),
    Uri.EscapeDataString(dtTo.Date.ToString("yyyy-MM-dd HH:mm:ss")));
String param = "?code=abc&" + dateRagne;
WebRequest req = WebRequest.Create(server_url + "/api/ncheck/event"
    + param);
req.Method = "GET";
//add access token
req.Headers["Authorization"] = "Bearer " + access_token;
req.ContentType = "application/json; charset=utf-8";
HttpWebResponse resp = req.GetResponse() as HttpWebResponse;
if (resp.StatusCode == HttpStatusCode.OK)
{
    var responseString = new StreamReader(resp.GetResponseStream()).ReadToEnd();
    JObject jsonResponse = JObject.Parse(responseString);
}
```

*Figure 2.15 C# code sample for get attendance events*

### 2.3.3.2 Add events

This API can be used for adding attendance event for user(s).

- Request: */api/ncheck/event/*
- Method: POST
- Body  
JSON array object with adding event(s) details as shown in *Figure 2.16* JSON body for add events API.
  - employeeCode  
Employee code the user needs to delete the attendance events
  - inTime  
Checked-in time as a date time string, formatted in yyyy-MM-dd HH:mm:ss
  - outTime  
Checked-out time as a date time string, formatted in yyyy-MM-dd HH:mm:ss
  - Shift  
Shift name as a *String*. If empty, default shift is selected.

– tzOffset

UTC timezone offset as an *integer*. Default is 0.

```
[
  {
    "employeeCode": "e1Gchk",
    "inTime": "2020-09-25 08:04:12",
    "outTime": "2020-09-25 17:22:25",
    "shiftName": "Default",
    "tzOffset": 19800
  },
  {
    "employeeCode": "YbUydy",
    "inTime": "2020-10-03 08:22:40",
    "outTime": "2020-10-03 18:23:11",
    "shiftName": "Default",
    "tzOffset": 19800
  }
]
```

Figure 2.16 JSON body for add events API

- Response

Response JSON Object with the status of the individual attendance events as JSON objects as shown in [Figure 2.17](#) JSON response for add events API

```
{
  "statusCode": "SUCCESS",
  "statusDescription": "Successfully updated event logs",
  "returnValue": [
    {
      "statusCode": "SUCCESS",
      "statusDescription": "Successfully updated checkin checkout events",
      "returnValue": {
        "employeeCode": "e1Gchk",
        "inTime": "2020-09-25 08:04:12",
        "outTime": "2020-09-25 17:22:25",
        "shiftName": "Default",
        "tzOffset": 19800
      }
    },
    {
      "statusCode": "SUCCESS",
      "statusDescription": "Successfully updated checkin checkout events",
      "returnValue": {
        "employeeCode": "YbUydy",
        "inTime": "2020-10-03 08:22:40",
        "outTime": "2020-10-03 18:23:11",
        "shiftName": "Default",
        "tzOffset": 19800
      }
    }
  ]
}
```

Figure 2.17 JSON response for add events API

– statusCode

Request status in *String*. This is a required field. Available status codes are shown in below.

Table 2.17 Status codes in the API response for add events

Status code	Description
INVALID_PARAMETERS	Matching event cannot be found
USER_NOT_AVAILABLE	User is not found
ERROR	System error (Need to check server logs for more details)
SUCCESS	Successful

- statusDescription  
Status description as a *String*. This is a required field.
- returnValue  
Selected events details as *Json array* if the response status code is *Success*. The available details are
  - \* statusCode  
Status of the deleted event as a *String*. This is a required field.  
Status codes as follows.

Table 2.18 Status codes for individual attendance events

Status code	Description
INVALID_PARAMETERS	Matching event cannot be found
USER_NOT_AVAILABLE	User is not found
ERROR	System error (Need to check server logs for more details)
SUCCESS	Successful

- \* statusDescription  
Status description as a *String*. This is a required field
- \* returnValue  
Added event detail as a *JSON object*.

Figure 2.18 is showing a C# code sample for delete events

```

WebRequest req = WebRequest.Create(server_url + "/api/ncheck/event");

req.Method = "POST";
//add access token
req.Headers["Authorization"] = "Bearer " + access_token;
req.ContentType = "application/json; charset=utf-8";

//events list to add JSON event objects
List<JObject> events = new List<JObject>();

//JSON object for a event
JObject jsonEvent1 = new JObject();
jsonEvent1.Add("employeeCode", "e1Gchk");
DateTime dtEvent1InTime = DateTime.Parse("2020-09-25 08:04:12");
jsonEvent1.Add("inTime", dtEvent1InTime);
DateTime dtEvent1OutTime = DateTime.Parse("2020-09-25 17:22:25");
jsonEvent1.Add("outTime", dtEvent1OutTime);
jsonEvent1.Add("shiftName", "Default");
jsonEvent1.Add("tzOffset", 19800);

//JSON object for a event
JObject jsonEvent2 = new JObject();
jsonEvent2.Add("employeeCode", "YbUydY");
DateTime dtEvent2InTime = DateTime.Parse("2020-10-03 08:22:40");
jsonEvent2.Add("inTime", dtEvent2InTime);
DateTime dtEvent2OutTime = DateTime.Parse("2020-10-03 18:23:11");
jsonEvent2.Add("outTime", dtEvent2OutTime);
jsonEvent2.Add("shiftName", "Default");
jsonEvent2.Add("tzOffset", 19800);

events.Add(jsonEvent1);
events.Add(jsonEvent2);

var settings = new JsonSerializerSettings
{
    DateFormatString = "yyyy-MM-dd HH:mm:ss",
    DateTimeZoneHandling = DateTimeZoneHandling.Utc
};

var postData = JsonConvert.SerializeObject(events, settings);
var data = Encoding.UTF8.GetBytes(postData);
req.ContentLength = data.Length;
using (var stream = req.GetRequestStream())
{
    stream.Write(data, 0, data.Length);
}
HttpWebResponse resp = req.GetResponse() as HttpWebResponse;
if (resp.StatusCode == HttpStatusCode.OK)
{
    var responseString = new StreamReader(resp.GetResponseStream()).ReadToEnd();
    JObject jsonResponse = JObject.Parse(responseString);
}

```

Figure 2.18 C# code sample to add events

### 2.3.3.3 Delete events

This API can be used for deleting attendance event from a user.

- Request: `/api/ncheck/event/`
- Method: DELETE

- Body

JSON array object with deleting event(s) details as shown in [Figure 2.19](#).

- employeeCode

Employee code the user need to delete the attendance events

- inTime

Checked-in time as a date time string, formatted in yyyy-MM-dd HH:mm:ss

- outTime

Checked-out time as a date time string, formatted in yyyy-MM-dd HH:mm:ss

- Shift

Shift name as a *String*. If empty, default shift is selected.

- tzOffset

UTC timezone offset as an *integer*. Default is 0.

```
[
  {
    "employeeCode": "abc",
    "inTime": "2020-09-25 13:22:12",
    "outTime": "2020-09-25 13:22:25",
    "shiftName": "Default",
    "tzOffset": 19800
  },
  {
    "employeeCode": "abc",
    "inTime": "2020-09-25 13:22:40",
    "outTime": "2020-09-25 13:23:11",
    "shiftName": "Default",
    "tzOffset": 19800
  }
]
```

Figure 2.19 JSON body for delete events API

- Response

Attendance Event Response JSON Object with the status of the individual attendance events as JSON objects as shown in

```
{
  "statusCode": "SUCCESS",
  "statusDescription": "Deleting events successful",
  "returnValue": [
    {
      "statusCode": "SUCCESS",
      "statusDescription": "Successfully deleted the event log",
      "returnValue": {
        "employeeCode": "abc",
        "inTime": "2020-09-25 13:22:12",
        "outTime": "2020-09-25 13:22:25",
        "shiftName": "Default",
        "tzOffset": 19800
      }
    },
    {
      "statusCode": "SUCCESS",
      "statusDescription": "Successfully deleted the event log",
      "returnValue": {
        "employeeCode": "abc",
        "inTime": "2020-09-25 13:22:40",
        "outTime": "2020-09-25 13:23:11",
        "shiftName": "Default",
        "tzOffset": 19800
      }
    }
  ]
}
```

Figure 2.20 JSON response for delete events API

- statusCode

Request status in *String*. This is a required field. Available status codes are shown in below.

Table 2.19 Status codes in the API response for delete events

Status code	Description
INVALID_PARAMETERS	Matching event cannot be found
USER_NOT_AVAILABLE	User is not found
ERROR	System error (Need to check server logs for more details)
SUCCESS	Successful

- statusDescription

Status description as a *String*. This is a required field.

- returnValue

Selected events details as *Json array* if the response status code is *Success*. The available details are

- \* statusCode

Status of the deleted event as a *String*. This is a required field. Status codes as follows.

Table 2.20 Status codes for individual attendance events

Status code	Description
INVALID_PARAMETERS	Matching event cannot be found
USER_NOT_AVAILABLE	User is not found
ERROR	System error (Need to check server logs for more details)
SUCCESS	Successful

- \* statusDescription  
Status description as a *String*. This is a required field
- \* returnValue  
Deleted event detail as a *JSON object*.

Figure 2.21 Is showing a C# code sample for delete events



```

WebRequest req = WebRequest.Create(server_url + "/api/ncheck/event");
req.Method = "DELETE";
//add access token
req.Headers["Authorization"] = "Bearer " + access_token;
req.ContentType = "application/json; charset=utf-8";

//events list to add JSON event objects
List<JObject> events = new List<JObject>();
//JSON object for a event
JObject jsonEvent1 = new JObject();
jsonEvent1.Add("employeeCode", "abc");
DateTime dtEvent1InTime = DateTime.Parse("2020-09-25 13:22:12");
jsonEvent1.Add("inTime", dtEvent1InTime);
DateTime dtEvent1OutTime = DateTime.Parse("2020-09-25 13:22:25");
jsonEvent1.Add("outTime", dtEvent1OutTime);
jsonEvent1.Add("shiftName", "Default");
jsonEvent1.Add("tzOffset", 19800);

//JSON object for a event
JObject jsonEvent2 = new JObject();
jsonEvent2.Add("employeeCode", "abc");
DateTime dtEvent2InTime = DateTime.Parse("2020-09-25 13:22:40");
jsonEvent2.Add("inTime", dtEvent2InTime);
DateTime dtEvent2OutTime = DateTime.Parse("2020-09-25 13:23:11");
jsonEvent2.Add("outTime", dtEvent2OutTime);
jsonEvent2.Add("shiftName", "Default");
jsonEvent2.Add("tzOffset", 19800);

events.Add(jsonEvent1);
events.Add(jsonEvent2);

var settings = new JsonSerializerSettings
{
    DateFormatString = "yyyy-MM-dd HH:mm:ss",
    DateTimeZoneHandling = DateTimeZoneHandling.Utc
};

var postData = JsonConvert.SerializeObject(events, settings);
var data = Encoding.UTF8.GetBytes(postData);
req.ContentLength = data.Length;
using (var stream = req.GetRequestStream())
{
    stream.Write(data, 0, data.Length);
}
HttpWebResponse resp = req.GetResponse() as HttpWebResponse;
if (resp.StatusCode == HttpStatusCode.OK)
{
    var responseString = new StreamReader(resp.GetResponseStream()).ReadToEnd();
    JObject jsonResponse = JObject.Parse(responseString);
}

```

Figure 2.21 C# code sample to delete events

---

## 3 PERIPHERAL INTEGRATION API

---

Android device manufacturers add various data input and output devices such as fingerprint sensors, temperature sensor in to their devices to use in various industrial applications. By using such peripherals integrated or attached to Android devices can be used in attendance applications. For this kind of devices, manufacturers provide their own API to access those devices. Following image shows basic device with integrated sensors.



*Figure 3.1 Picture with example device with fingerprint scanner, Temperature sensor and Motion Sensor functionality*

1. Temperature sensor
2. Fingerprint sensor
3. NFC scanner
4. Motion sensor
5. Camera
6. Light

NCheck supports integrating third party fingerprint scanners by using NCheck SDK capture device service API. NCheck SDK capture device service API provides an API interface to develop a module and integrate with NCheck Android to control and capture fingerprint using third party

fingerprint scanners. This document describes the procedure to develop a fingerprint scanner module to integrate a fingerprint scanner with NCheck.

### 3.1 Third Party sensor Integration with Android Standard client

Scanner integration application is used to configure and control the third-party fingerprint scanner device.

#### 3.1.1 Components of the application

It should have following components.

1. Configuration activity  
NCheck Android uses this activity to configure the Android service and read configuration details.
2. Android service  
An Android service developed by using the CaptureDeviceInterface (See API documentation) abstract class. This service connects, control and capture from fingerprint device.

#### 3.1.2 Create Scanner integration application

1. Create a new Android application.
2. Add NCheck Peripheral API Library module
  1. Open the project structure by File> Project Structure
  2. Select “Modules” from the left side menu

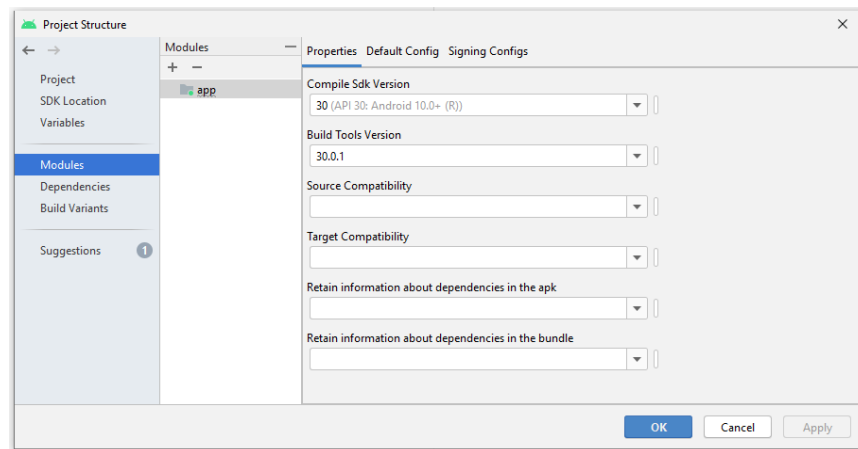


Figure 3.2 Project structure dialogue of Android Studio

3. Click the “+” button in the top left to add a new module

4. Select module type as “import JAR/AAR packages” from the Create new module dialog and select “Next” button

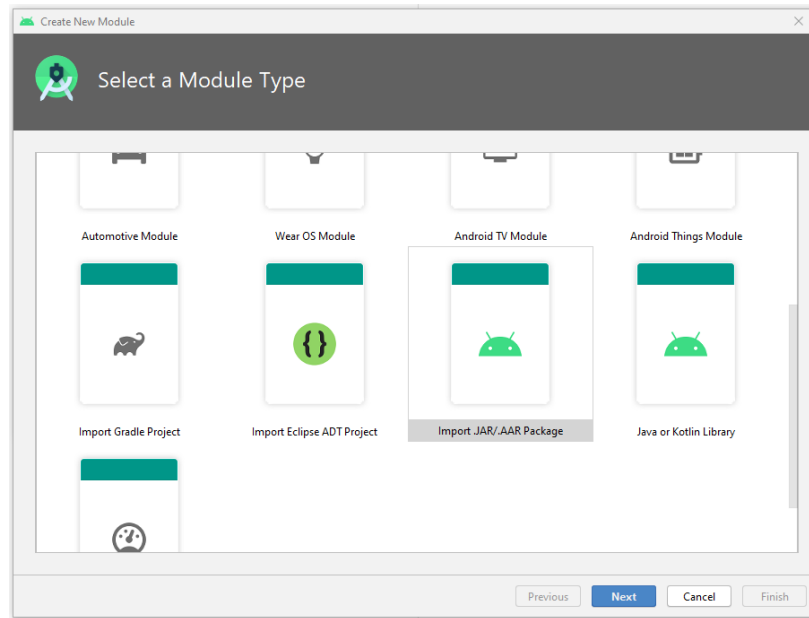


Figure 3.3 Select JAR/AAR packages from Create new module dialogue

5. Select and Import the peripheralSDK.aar module

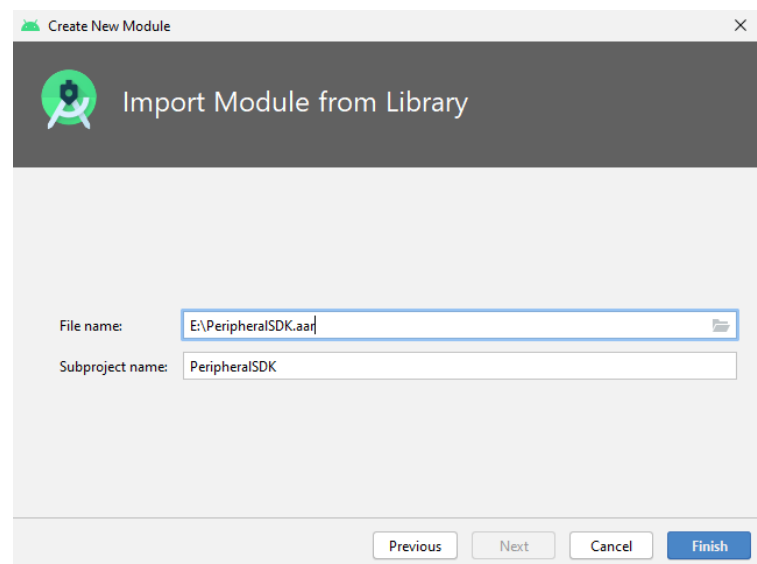


Figure 3.4 Import PeripheralSDK.aar from create new module dialogue

3. Add a configuration activity (MyScannerConfigurationActivity)  
Configuration activity should have all the peripheral service-related configuration
4. Configure Activity in the Android manifest file to access from NCheck External peripheral configuration as follows

```
<activity android:name="com.neurotec.ncheck.sdk. .,MyScannerConfigurationActivity ">
<intent-filter>
    <action android:name=" MyPeripheral" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
</activity>
```

Figure 3.5 Android manifest files changes to access from external peripheral configuration

As shown in [Figure 3.5](#), in this configuration, the action name (ex: MyPeripheral) is the unique identification for NCheck Android to find the configuration activity of peripheral integration application.

5. Add coding to send configuration information to NCheck android. Override onBackPressed method of the configuration activity as shown in [Listing 3.2](#). Then it will send the configuration information required to communicate with fingerprint scanner service.

```
@Override
public void onBackPressed() {
    CaptureDeviceService.setConfigurationResult(MyPeripheralService.class, configurationActivity: this);
    super.onBackPressed();
}
```

Figure 3.6 Code snippet to send the configuration information required to communicate with fingerprint scanner service

In this code MyPeripheralService is the class name of the peripheral application Android service.

6. Add new android service (MyPeripheralService) by extending CaptureDeviceService class from the API
7. Implement following abstract methods in the new service class
  - Public static CaptureDataType[] CaptureTypes  
Shadow method to replace supper class CaptureTypes static property
  - public abstract String GetName()  
Implement this method to provide the name of the peripheral device service.
  - public abstract CaptureDataType[] GetCaptureTypes()  
Implement this method to provide data type of the peripheral device. It should be CaptureDataType.FINGERPRINT or CaptureDataType.SENSOR
  - public abstract Point[] GetImageSizes()  
Implement this method to provide possible image resolutions in case of a fingerprint reader. It supports only one resolution. Multiple resolutions are for future use only.
  - public abstract void Start()  
Implement this method to start the communication status. Before starting, Status of the peripheral device service is DISCONNECTED. It should notify the status change by using SendStatus method of CaptureDeviceService class. When Start method is called, the peripheral device service should change the status to NOT\_READY. It should make attempt to establish the communication with the peripheral device in background. Once communication is established, the service status should be changed to READY and should be ready to capture data from the device.

- **public abstract void Stop()**  
Implement this method to stop communication with the hardware and return the status to DISCONNECTED. If capture request made before getting ready, the device should start capture immediately and change the status to CAPTURING. current capturing and
- **public abstract void ShowMessage(DisplayMessageType type, String msg)**  
Implement this method to display messages from NCheck bio attendance, if there is a display. Call to this method will send the type of the message and message text. Depend on the device capabilities, it can process the message. If the display, LED indicators or beepers are not available, it can do nothing. As an example, if there are three color LEDs, it can process only the type of the messages like green for INFO, yellow for WARNING and red for ERROR.
- **public abstract void ShowResults(String name, int inout, String time1, String time2)**  
Implement this method to display check in check out results in the peripheral device. This is useful on in the cases where display or user notification functionalities are available in the device. This method sends the full name of the user, event type , user check in time and user check out time.
- **public abstract boolean Capture(CaptureDataType dataType, Point resolution)**  
Implement this system to receive capture start. When it receive the call, it should start capture immediately if the peripheral service can communicate with the peripheral device. If it is in not ready to communicate with peripheral device, it should wait until establishing the communication with the hardware and start capture immediately after establishing the communication with the hardware. Capture method can specify the capture data type and image resolution When capture is enabled, Peripheral service can send captured data to NCheck Bio Attendance client by using SenCapturedData method. SendCaptureData can send raw image data as a Bitmap, textual data as a String or extracted template data as a byte array. For peripheral device service, the captured data should have only raw image data as a Bitmap
- **public abstract void CancelCapture(CaptureDataType dataType)**  
Implement this method to cancel current data capture operation. Data type parameter supports only FINGERPRINT.
- **public abstract Sense(int index)**  
Implement this method to start reading sensor data operation. . Parameter index is for future us and need to provide 0. Service should start reading sensor data and call sendSensorDataUpdate method to send sensor data to NCheck. Sensor data should include
  - 1.index  
Sensor index is reserved for future use.
  - 2.Range  
Possible range of data with min and max value.
  - 3.Precision  
Accuracy of data.
  - 4.Threshold  
Passing threshold of measuring data.

### 5.StrMessage

Textual message to display in NCheck Application

- public abstract void CancelSense(int index)  
Implement this method to cancel current reading sensor data operation. Parameter index is for future us and need to provide 0.

8. Configure Peripheral service in the Android manifest file as shown in [Figure 3.7](#).

```
<service  
    android:name=".sdk.samples.dummyscanner.MyPeripheralService"  
    android:enabled="true"  
    android:exported="true" />
```

Figure 3.7 Code snippet to configure Peripheral service in the Android manifest file

### 3.1.3 Testing scanner service

API provides a test activity called CaptureDeviceServiceTest. You can also launch this service activity by using LaunchTestActivity static method of the CaptureDeviceService class.

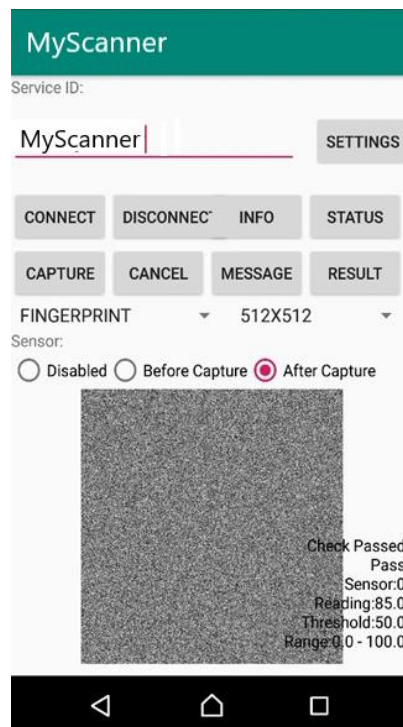


Figure 3.8 Scanner service testing view

---

# 4 ACCESS CONTROL INTEGRATION

---

NCheck Bio Attendance Standard clients can be integrated with access controls such as door, gate, etc.

NCheck Bio Attendance Standard client for Windows support either an external executable(.EXE) or web request provided by the third-party access controls. NCheck Bio Attendance Standard client for Android supports web request provided by the third-party access controls. NCheck Bio Attendance administrator allows to configure each client device to trigger access controls using provided APIs (.exe or REST API) in following client registration mode.

1. Standard clients registered in Cloud/On-premises mode  
NCheck Bio Attendance administrator can configure peripherals adding peripheral configuration from the NCheck Bio Attendance web control panel as mentioned in [manage peripheral configuration](#) section.
2. Standard clients registered in Standalone mode  
product administrator can configure peripherals adding peripheral configuration from the NCheck Bio Attendance Standard client control panel as mentioned in [client control panel](#) section.

Following diagram is showing how the NCheck Bio Attendance Standard clients should be connected with the access controls system.

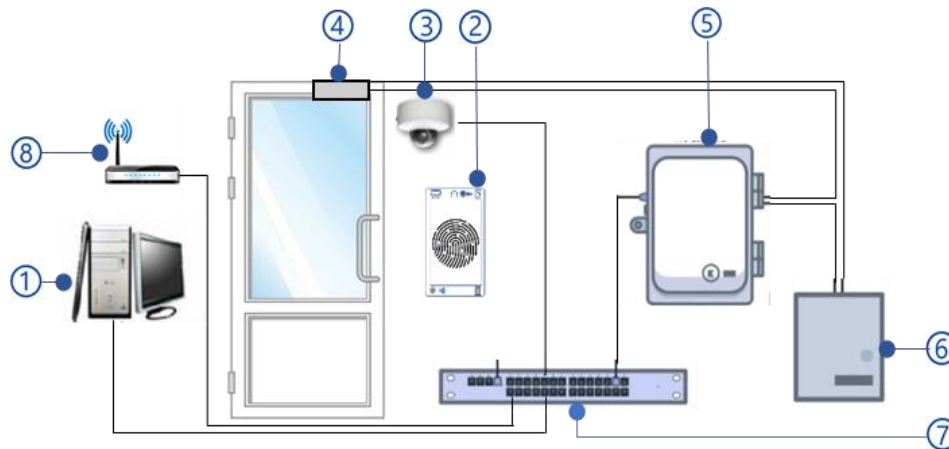


Figure 4.1 Access controlling with NCheck

1. NCheck Bio Attendance Standard client for Windows
  - If the client registration mode is cloud, NCheck Bio Attendance server installation is not required
  - If the client registration mode is standalone, NCheck Bio Attendance server is not required



2. Android device installed with NCheck Bio Attendance Standard client for Android Device has connected to the network using WIFI.
  3. Camera
  4. Door magnet
  5. Door control unit
  6. Power source for the magnet and door control unit
  7. Network switch
  8. Wifi router
- To connect the android device to the network

In successful check-in or check-out provided external executable or API by the access control manufacturer will be executed by the clients. Administrator allows to select the required parameter in order to pass to the provided API to trigger the access control. Please refer [Parameters](#) section for more details about available parameters.

#### 4.1 Parameters

Following parameter(s) can be passed in to the access control using the provided API.

*Table 4.1 Parameters for access controls*

Parameter name	parameter key	Parameter value
<b>Event</b>	EVENT	Event type as CHECKIN, CHECK-OUT or UNIDENTIFIED
<b>Sequence id</b>	SEQUENCE_ID	Id of the event
<b>Time stamp</b>	TIMESTAMP	Date and time of the event
<b>User reference</b>	USER_REF	Employee code of the user
<b>User name</b>	USER_NAME	First name and last name of the user event recorded
<b>Shift code</b>	SHIFT_CODE	Shift code of the NCheck Bio Attendance Standard clients. In the standalone mode shift code is DEFAULT
<b>Location</b>	LOCATION	Longitude and latitude of the event geo-location. Example: {Longitude, latitude}
<b>User status</b>	USER_STATUS	User is blocked or not. Pass 0 if the user has blocked otherwise 1
<b>Address</b>	ADDRESS	Address of the user
<b>Peripheral code</b>	PERIPHERAL_CODE	Unique identified number of the peripheral

#### 4.2 Trigger access control

Selected parameters will be passed with the provided API as follows.

Table 4.2 Format of the parameters passed into the API

API type	Format
External executable	<ul style="list-style-type: none"><li>• Value of the parameters will be separated with empty space as {.exe path} {parameter value 1} {parameter value 2} {parameter value 3}</li><li>• Example: doorControl.exe CHECK_IN</li></ul>
REST API	<ul style="list-style-type: none"><li>• URL parameters as {url}?{Paramter1}={value} &amp;{Paramter2}={value} &amp; {Paramter3}={value}</li><li>• Example: https://192.168.2.12:3215/doorControl? EVENT=CHECK-IN</li></ul>

# 5 CUSTOM FIELDS CONFIGURATION

NCheck Bio Attendance Customers may have requirements to customize forms with additional details. For an example customer may want to add *Generation* and *Team ID* fields to the *Add new employee* form as follows.

Refer [Add custom fields](#) section for more details about

The screenshot displays the 'ADD NEW EMPLOYEE' form. At the top, a teal header bar contains the title and a user icon. Below the header, a red note states: 'Required fields are marked in red \*'. The form is organized into two columns. The left column includes fields for 'Profile picture' (with 'BROWSE' and 'UNIDENTIFIED IMAGES' buttons), 'First name \*', 'Employee code \*' (with a 'GENERATE' button), 'Barcode', 'Administrator rights' (a dropdown menu), 'Address line 2', 'Country' (a dropdown menu), 'Zip code', and 'Primary user group' (a search field). The right column includes fields for 'Last name \*', 'Email', 'RFID', 'Address line 1', 'City', 'State', 'Telephone' (with a country dropdown and area code), and two custom fields, 'Generation' and 'Team Id', which are highlighted with a red rectangular box. At the bottom right, there are '+ ADD' and 'X CLOSE' buttons.

Figure 5.1 Add custom fields to the Add new employee form

## 5.1 Add custom fields

Once the customer decided which additional details need to be added in NCheck Bio Attendance, our developer will provide modal upgrade script. For more inquiry please contact our customer support from [support@ncheck.net](mailto:support@ncheck.net)

---

# 6 IMPORT EXPORT

---

NCheck Bio Attendance total work hour report can be exported in order to import in other applications such as payroll related software. This guide explains importing total work hour report in following payroll systems.

1. [Import total work hour report in Tally ERP system](#)
2. [Import total work hour report in Quickbooks](#)

## 6.1 Import total work hour report in Tally ERP system

[Tally](#) ERP uses Attendance Voucher to record employee's attendance data, based on Attendance/Production types (i.e., present or absent days, overtime hours and so on).

An Attendance Voucher allows you to record the attendance/production units for employees. Tally ERP gives you the flexibility to enter the attendance records through a single attendance voucher for a payroll period, or through multiple attendance vouchers as and when required within a payroll period. You also have the option of recording one attendance/production voucher per employee per day or collectively for a month or any other variation thereof for all the employees.

NCheck Bio Attendance can export employee's attendance records in daily, weekly or monthly using [Total work hour report](#). A Total Work Hours Report data can be imported to tally as a single Attendance Voucher. If the attendance data is imported in daily basic, it will create multiple attendance vouchers for a given pay period. If Attendance data is imported weekly, monthly or user selected period, it will create a single attendance voucher for a given pay period.

NCheck Bio Attendance can export Total Work Hours report into an Excel file. Tally provides a TDL extension for Importing Payroll Data which includes a User Manual. Total Work Hours exported to excel from NCheck Bio Attendance can be imported to Tally Payroll Vouchers using this Payroll Data Import TDL extension from Excel files.

Please use following procedure to export attendance data to Tally ERP system.

1. Make sure that Tally System attendance/Production Payroll units are defined in hours
2. Total work hour report can be generated as follows

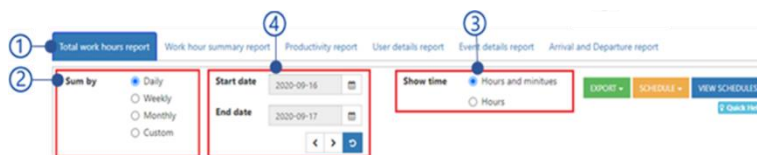


Figure 6.2 Export total work hour report

1. Select the total work hour report from reports

2. Select sum by daily, weekly, or monthly
3. Select the required date range and refresh the report
4. Select *Show time in Hours* to show time in decimals in exported csv file. Select refresh button
5. Select Reload button to reload the report
6. You can see the total work hour report as follows

From	To	First Name	Last Name	Employee Code	Productive Work Hours	Overtime Work Hours	Productive Hours	Work Hours
2020-10-07	2020-10-07	Akila	Thalgahagoda	2b12ar	9:23	-	9:23	9:23
2020-10-07	2020-10-07	Kuganathan	Rajeevan	bvP9RX	8:35	-	8:35	8:35
2020-10-07	2020-10-07	Mathura	Sivarajah	emp12	9:23	-	9:23	9:23
2020-10-07	2020-10-07	Nishara	Jayarathna	v8csAE	10:46	-	10:46	10:46
2020-10-07	2020-10-07	Pradeep	Athukorala	emp25	10:44	-	10:44	10:44
2020-10-07	2020-10-07	Sandeep	Mohanta	emp9	8:43	-	8:43	8:43
2020-10-07	2020-10-07	Saranga	Vitharana	he59BG	9:49	-	9:49	9:49
2020-10-07	2020-10-07	Sriyan	Fernando	emp10	9:12	-	9:12	9:12
2020-10-07	2020-10-07	Gumathra	Vickuranna	amnd	9:12	-	9:12	9:12

Figure 6.3 Total work hour report view

3. Hide all the columns from the total work hour report except First name and Work hour columns. To hide the columns, use the table option menu as follows

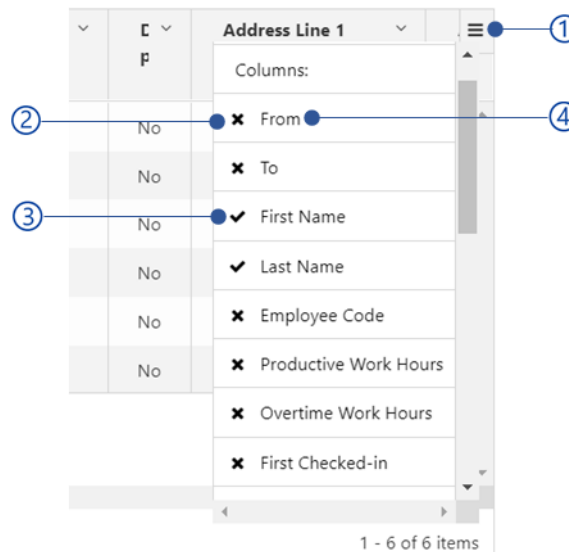


Figure 6.4 Table options menu

1. Table options menu button
2. Hided column
3. Visible column
4. Column name  
Select the on the column name to hide/visible the column in report

4. Rename column First Name as Employee Name and Work hours as Attendance Value. Use column options menu to rename the columns as follows (Optional)

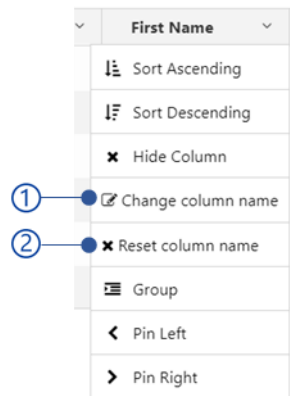


Figure 6.5 Column options menu

1. Change column name

Select this option to change the column name. The new name can be set in the change column name dialogue as follows.

A screenshot of a dialog box titled 'CHANGE COLUMN NAME'. It has a teal header bar with the title and a document icon. Below the header, there is a label 'Column name' followed by a text input field containing 'First Name'. At the bottom of the dialog, there are two buttons: a green button with a checkmark and the text 'CHANGE', and a white button with a close icon and the text 'CLOSE'.

Figure 6.6 Change column name dialogue

2. Reset column name

Reset the column name to its default

5. Select Export button and choose the CSV report option to export report in CSV format.
6. Open the report. Make sure that all employees are exported in the csv file

7. Add new *Attendance/Production Type* column after Employee Name column. Assign Attendance/Production Type name defined in the Tally System.  
Ex: Attendance

	A	B	C
1	Employee Name	Attendance Value	Attendance/Production type
2	Harvey	6.36	Attendance
3	Ryan	4.93	Attendance
4	Rahul	4.21	Attendance
5	Eva	3.49	Attendance
6	William	2.44	Attendance
7	Katherine	1.65	Attendance
8	Billy	1.64	Attendance
9			

Figure 6.7 Add Attendance/Production Type column

8. Import Payroll data as mentioned in [user manual – import payroll data](#)

## 6.2 Import total work hour report in Quickbooks

[QuickBooks](#) desktop allows to import timesheet data as IIF (Intuit Interchange Format) files. IIF format is a text-based tab-delimited format very similar to .tsv file format. This guide describes how you can migrate NCheck Bio Attendance total work hour report (Daily) data in IIF format. To create such IIF file, user need to perform these main steps.

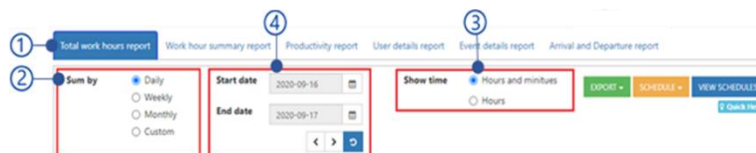


1. *Import work hour data (Daily) from NCheck Bio Attendance as a CSV file*
2. *Manipulate work hour report to format data and remove unnecessary data*
3. *Download sample IIF file*
4. *Update correct header information using Timer list IIF file*
5. *Merge manipulated report data*
6. *Import to QuickBooks*

Figure 6.8 Logo of Quickbooks

### 6.2.1 Import work hour data (Daily) from NCheck Bio Attendance as a CSV file

1. Login to NCheck Bio Attendance and go to reports tab
2. Export Total work hour report as follows



1. Select the total work hour report from reports
2. Select *Sum by* as *Daily*
3. *Show time* should be in *Hours and minutes*
4. Select the required date range and refresh the report

3. Hide all the columns except First Name, Last Name, To and Work hours columns. To hide the columns, use the table option menu as follows

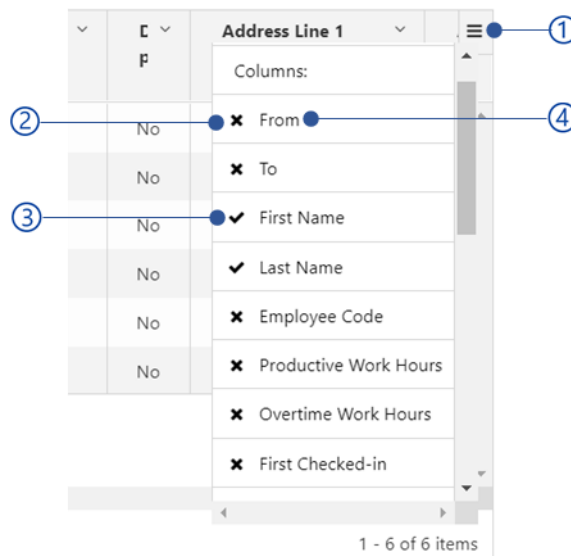


Figure 6.9 Table options menu

1. Table options menu button
2. Hided column
3. Visible column
4. Column name

Select the on the column name to hide/visible the column in report

4. Rename column name To as DATE and Workhour as DURATION. Use column option menu to rename column names as follows.

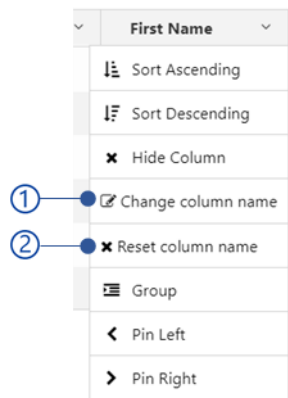
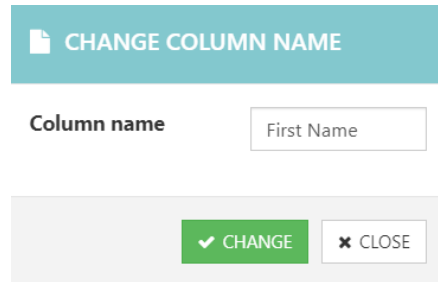


Figure 6.10 Column options menu



1. Change column name

Select this option to change the column name. The new name can be set in the change column name dialogue as follows.

A dialog box titled "CHANGE COLUMN NAME" with a teal header bar. Below the header, there is a label "Column name" followed by a text input field containing "First Name". At the bottom of the dialog, there are two buttons: a green button with a white checkmark and the text "CHANGE", and a white button with a black 'x' icon and the text "CLOSE".

CHANGE COLUMN NAME

Column name First Name

✓ CHANGE ✕ CLOSE

Figure 6.11 Change column name dialogue

2. Reset column name

Reset the column name to its default

5. Select *Export* button and select CSV to import report in CSV format

### 6.2.2 Manipulate work hour report to format data and remove unnecessary data

Open the csv file using a spreadsheet software, such as Microsoft Excel. Refer following guide to manipulate work hour report in Microsoft Excel.

1. Concatenate *First name* and *Last name* columns and create new column *EMP*  
Refer [this reference](#) for more details about combine columns.
2. Set *DATE* column date format to DD/MM/YY format  
Refer [this article](#) for more details about format column date format.

### 6.2.3 Download sample IIF file

You can download the sample *.IIF* file [here](#). You can re-name the file as you wish. Open it using a spreadsheet software like Microsoft Excel.

## 6.2.4 Update correct header information using *Timer list* IIF file

1. *Timer List* IIF file can be exported from QuickBooks using File>Utilities>Export>Timer Lists as follows

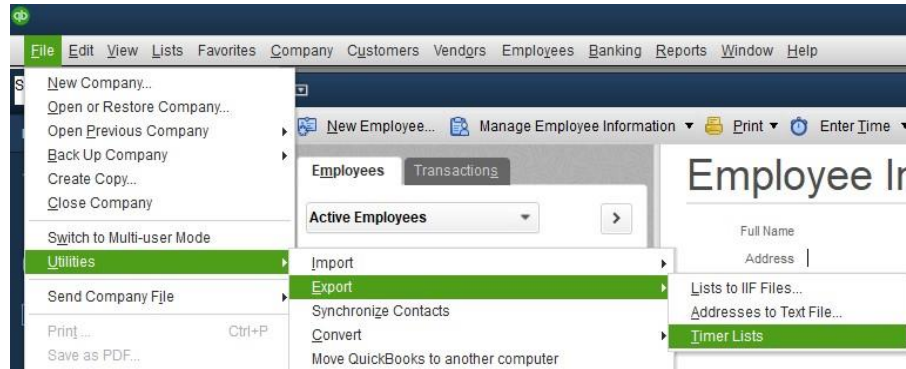


Figure 6.12 Export timer list

2. Exported Timer List IIF file header will have the Company details as follows.

	A	B	C	D	E	F	G	H	I	J	K
1	!TIMERHDR	VER	REL	COMPANYNAME	IMPORTEDBEFORE	FROMTIMER	COMPANYCREATETIME				
2	TIMERHDR		8	0 ABC	N	N	1600233938				
3	!HDR	PROD	VER	REL	IIFVER	DATE	TIME	ACCNTNT	ACCNTNTSPLITTIME		
4	HDR	QuickBooks Pro for Windows	Version 6.0D	Release R8P		1	2020-09-16	1600243068	N	0	
5	!CLASS	NAME	REFNUM	TIMESTAMP	HIDDEN	DELCOUNT					
6	!CUST	NAME	REFNUM	TIMESTAMP	BADDR1	BADDR2	BADDR3	BADDR4	BADDR5	SADDR1	SADD
7	!VEND	NAME	REFNUM	TIMESTAMP	PRINTAS	ADDR1	ADDR2	ADDR3	ADDR4	ADDR5	VTYP
8	!EMP	NAME	REFNUM	TIMESTAMP	INIT	FIRSTNAME	MIDINIT	LASTNAM	SALUTATION	HIDDEN	DELO
9	!OTHERNAME	NAME	REFNUM	TIMESTAMP	BADDR1	BADDR2	BADDR3	BADDR4	BADDR5	PHONE1	PHON
10											
11											
12											

Figure 6.13 Company details of timer list header

3. Copy the first 4 lines highlighted in yellow color. Replace the 1st 4 rows of the sample file with this content. Now sample file will look like as follows.

	A	B	C	D	E	F	G	H	I	J	K
1	!TIMERHDR	VER	REL	COMPANYNAME	IMPORTEDBEFORE	FROMTIMER	COMPANYCREATETIME				
2	TIMERHDR		8	0 ABC	N	N	1600233938				
3	!HDR	PROD	VER	REL	IIFVER	DATE	TIME	ACCNTNT	ACCNTNTSPLITTIME		
4	HDR	QuickBooks Pro for Windows	Version 6.0D	Release R8P		1	44090	1600243068	N	0	
5	!TIMEACT	DATE	JOB	EMP	ITEM	PITEM	DURATION	PROJ	NOTE	BILLINGSTATUS	
6											
7											

Figure 6.14 Edited timer list

## 6.2.5 Merge manipulated report data

Follow these steps to merge work hour data we just modified to this IIF file.

1. First column value should be TIMEACT
2. For DATE column, merge the contents from the work hour report you just modified
3. JOB column  
The name of the customer (or job). If you're entering the name of a job, enter the

customer's name followed by a colon followed by the name of the job. Both the customer and the job names must also be on your Customers & Jobs list (CUST).

4. EMP column  
merge the contents from the work hour report you just modified.
5. ITEM column  
The name of the service item. The service item must also be on your Item list (INVITEM).
6. IITEM column  
Payroll Item for Paychecks.
7. DURATION column  
Merge the contents from work hour report you just modified.
8. PROJ column  
The QuickBooks class assigned to the activity (classes give you a way to group activities in meaningful ways in time reports). The Class must also be on your Class list (CLASS).
9. NOTE column  
Description/Note for each time entry. You may enter up to 1000 characters.
10. BILLINGSTATUS column  
Indicates the billing status. Enter one of these values mentioned below
  - 0: not billable
  - 1: Not billed
  - 2: Billed

#### 6.2.6 Import to QuickBooks

Now your worksheet IIF is ready to import in to QuickBooks. Follow below steps to import the worksheet to QuickBooks

1. Go to File>Utilities>Import>Timer Activities to import the worksheet IIF file. Once data is imported, a Timer Import Data report will be shown the data which was extracted.

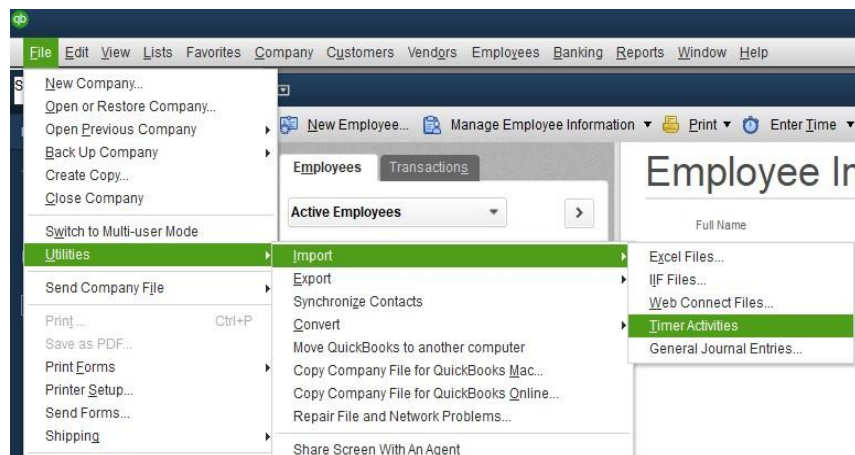
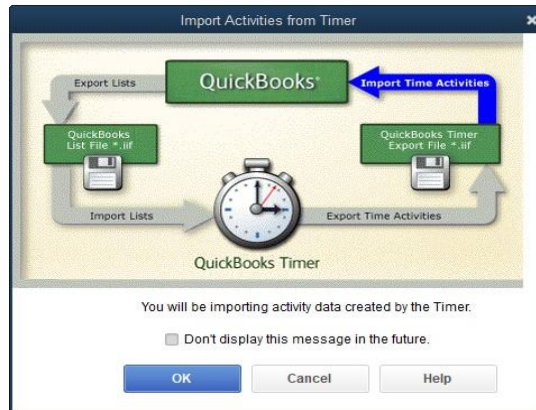


Figure 6.15 Import to Quickbooks

2. A confirmation dialogue will be shown as below. Click *OK* to select the IIF file we just created



*Figure 6.16 Import confirmation dialogue*

3. Done! Now your timer data will be imported to QuickBooks

---

## 7 CONTACT

---

You can contact NCheck Bio Attendance team regarding issues and questions you have. Please send an email to [support@ncheck.net](mailto:support@ncheck.net) with your query