WebAPI Integration Guide

English

Part number:
42-4004EN-02

Date of issue:
2022-01-07

IntelliSpace Corsium

Data management solution

PHILIPS

Manufacturer's address

Philips IntelliSpace Corsium is designed and manufactured by:

Remote Diagnostic Technologies Ltd
Ascent 1
Farnborough Aerospace Centre
Aerospace Boulevard
Farnborough
Hampshire
GU14 6XW
United Kingdom

Tel: +44 (0) 1256 362 400
Email: rdt_info@philips.com
www.philips.com

CE
0413

EC REP
Philips Medical Systems Nederland B.V.
Veenpluis 6
5684PC Best
The Netherlands

Australian TGA sponsor:
Philips Electronics Australia Ltd
65 Epping Road
North Ryde
NSW Australia 2113

Contents

1 Introduction

This integration guide is for the IntelliSpace Corsium WebAPI.

IntelliSpace Corsium (Corsium) is a secure web service, hosted by RDT, a Philips company, capable of receiving clinical data from Tempus Pro vital signs monitors and then storing, processing, and distributing the data in both clinical and non-clinical contexts.

Note	For instructions for use for IntelliSpace Corsium, refer to the *IntelliSpace Corsium Transport Administrator Manual* and *IntelliSpace Corsium Transport User Manual*

This document describes the IntelliSpace Corsium WebAPI web service for third party software developers to integrate into any solution that can support http client requests. It facilitates read-only access to the patient data stored in Corsium.

The purpose of this document, which is written for software engineers, is to describe how to pull from Corsium into ePCR software applications using the WebAPI.

In order to provision effective evaluation of the data sharing approach RDT provides a Corsium WebAPI SDK consisting of the following:

- Reference Corsium deployment containing typical Tempus Pro patient encounter data
- Reference software

1.1 Obtaining this manual

Consult the manual. This is provided in electronic format.

Note	Contact your local Philips service organization to obtain the latest electronic version of this manual.

1.2 Note icon

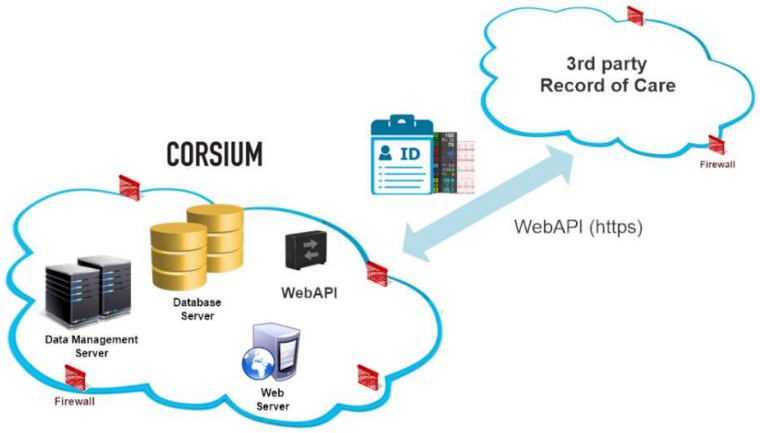
The note icon is used for important and helpful information:

Note	A point of particular interest or emphasis intended to provide more effective or convenient operation.

1.3 Background

RDT have created WebAPI Service for third party software developers to pull Tempus Pro data from Corsium. Corsium WebAPI has the following attributes:

- Has a simple API that can be called by the 3rd party ePCR software to pull Tempus Pro patient encounter data.
- Uses HTTPS to ensure data is transferred securely. If the data contains PI (patient identifiers) then it will be encrypted by default before being shared via HTTPS.
- Outputs the data in a JSON format to provision ease of integration by third party developers.



1.4 WebAPI Architecture

Corsium WebAPI service is built on a secure, load-balanced, highly available AWS (Amazon Web Services) infrastructure.

1.5 Data ownership and access

Access to data stored in Corsium is tightly controlled.

Corsium considers the stored data to be owned by the owner of the Tempus Pro device, typically the EMS Agency. Access to that data by others must be granted by the owner of the data.

1.6 Responsibilities

The Corsium WebAPI is intended to be used by the system integrator's engineers, in accordance with the instructions provided by RDT.

The system integrator is responsible for the verification and validation of the integrated system. RDT's technical support department will provide support if requested.

The system integrator is responsible for the finished system, in particular with applicable local requirements and regulations e.g. for privacy of patient data.

It is essential to ensure:

- The data supplied is integrated with the record for the correct patient.
- Data from a single incident must not be mixed with data for other patients.
- All data is received and maintained on secure systems appropriate to storage of medical data.
- The final integrated system is sufficiently proof against failure of operation, contains software controls for all possible error conditions and advises users to maintain appropriate protocols to ensure their equipment and systems remain operational.

1.7 Data access

The EMS Agency will email, via Corsium, the following WebAPI details to the 3rd party ePCR:

- Username
- Password
- URL
- Encryption information

This information will be AES256 encrypted. A second email will be sent with the key needed to decrypt the information. The 3rd party ePCR will need these details to authenticate with Corsium.

Corsium URLs are specific to a particular customer account.

Note	Emails will expire in 5 days.
Note	Please contact RDT for the decryption tool.
Note	If one ePCR provider is configured to view multiple transport accounts, then there will be separate login details for each transport account.

1.8 Test reference

RDT provides simple reference application for WebAPI testing and validation. The test application can be configured to pull data from IntelliSpace Corsium by specifying the URL and credentials for an ePCR user sent via email at the time of ePCR user creation or shown on Corisum screen when new ePCR is configured.

1.8.1 Email #1 with attachment : WebAPISettings-<yyyyMMdd'T'HHmm>.txt

URL:	<Customer specific, sent in email #1>
Username:	<Customer specific, sent in email #1, AES256 encrypted>
Password:	<Customer specific, sent in email #1, AES256 encrypted>
Double Encryption:	<Customer specific, details sent in email #1>

1.8.2 Email #2 with attachment : DecryptionKey-<yyyyMMdd'T'HHmm>.txt

Symmetric Key:	<Customer specific, sent in email #2, used to decrypt Username and Password>
Initialization Vector:	<Customer specific, sent in email #2, used to decrypt Username and Password>

Note	Check "7.7 Sample email attachments for ePCR accounts" for a sample of email attachments.

Test reference also contains source code of the application written in C# language and using MVC.net framework. This can be used as a reference to understand the mechanism of making requests to the WebAPI server and pull patient data from Corsium.

1.9 SDK accounts

RDT provides SDK accounts with pre-configured incident data on its demo deployment server for WebAPI https://corsium.info. SDK accounts have fixed login details and do not require configuration files sent by email. There are two SDK accounts configured on the WebAPI server:

1.9.1 SDK account without double encryption

URL:	https://corsium.info/WebAPI/sdk
Username:	<random GUID, e.g. 686d4517-bd7f-4b8d-9ece-7a6ce799ad6f>

Password: 3P(RTe$t1n9

Encryption: None

1.9.2 SDK account with double encryption

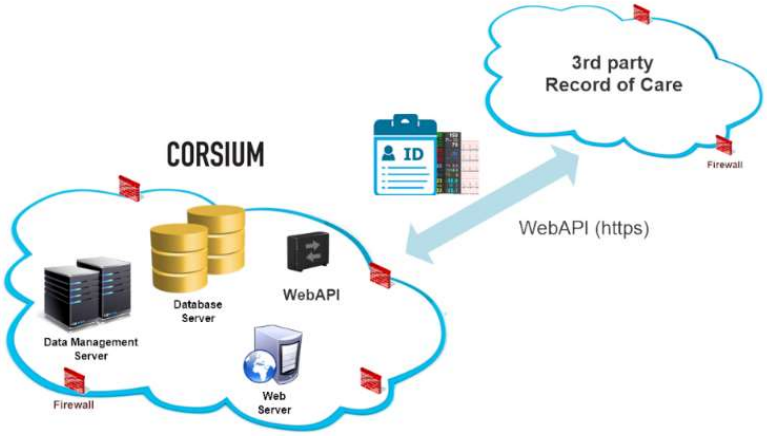
URL: https://corsium.info/WebAPI/sdkencr

Username: <random GUID, e.g. 686d4517-bd7f-4b8d-9ece-7a6ce799ad6f>

Password: 3P(RTe$t1n9

Encryption: AES

Full details of GUI and preconfigured patient data are shown in Sections 5 and 6.



www.corsium.info (demo URL populated with preconfigured data)

1.10 API Request

- All API requests (GET and POST) should include following header as part of the request
  - Accept application/json
- All POST API requests should include “Content-Type” header as part of the request. The API supports HTTP standard Content-Type header which includes charset information. Charset information can be omitted in which case “utf-8” charset is assumed. Following examples show the valid Content-Type headers supported by the API.
  - Content-Type application/json;charset=utf-8, **or**
  - Content-Type application/json
- All POST API requests should send API parameters in the request body as JSON object

  e.g. for APILogin request the JSON object might look like :

  {“Username”:”epcr@royalems.com”,”Password”:”3P(RTe$t1n9”,”DEM”:”None”,”Key”:””}
- All API requests for pulling patient data should include following authorization header
  - Authorization bearer <Token received from APILogin response>
- All identifiers returned in the response data (e.g. Organization ID, Incident GUIDs, Event GUIDs), which are used in subsequent API calls are case-insensitive. They should be used in subsequent API calls as returned in the response data.

Note	Authorisation header is not required for APILogin request.

Note	See “7.5 API access using Postman” for steps to login and access patient data using **Postman** client.
Note	See “7.8 Sample code for WebAPI requests in C#” for sample code in C# to make API calls and receive JSON response.

1.11 API response format

The response from a http request against a Corsium WebAPI resource consists of the following:

- HTTP Status code – Corsium WebAPI will return an appropriate HTTP status code.
- Response body – Each successfully processed response (HTTP Status code 20x) is returned in a standard format that will give the status of the resource request, the results, and if applicable, an Corsium WebAPI error message.

Response body format (Response):

```
{
    “ErrorCode”: “”,
    “IsSuccessful”: true,
    “DataType”: “real”,
    “Data”: <resource-specific JSON-formatted data as defined below>,
    “CRC”: “r6pq”
}
```

All resource-specific data in the Clinical API is returned in the JSON format. All dates/times are in Coordinated Universal Time (UTC) and are returned in the ISO-8601 format.

Attribute	Type	Description
ErrorCode	String	Returns error message code, if request was not successful
IsSuccessful	Boolean	True if request was successful
DataType	String	Type of the data. Can be one of the following: • Sandbox • Real
Data	String	Resource-specific JSON-formatted data as defined below
CRC	String	2 byte CRC value. If required, please contact RDT for further details of how the CRC is calculated.

Note	All data in the response is in English, except: • Incident PDF • ECG in JPEG format • Snapshot in JPEG format • Any event names or event details that were configured using Tempus Pro Configuration Utility • Events, where Tempus Pro user manually entered the text, e.g. general notes, event notes, medics, etc. The data mentioned above will be returned in the configured language.
Note	RDT recommends that WebAPI implementation is tested using the Sandbox customer account before using on a customer account with real data.
Note	Check “7.1 Response error messages” for the list of error messages.
Note	It is possible for a historic incident on Tempus Pro to become live again. It could be due to Tempus Pro reconnecting to Corsium after loss of connectivity for more than 20 minutes or existing patients being re-admitted on the device. After this if the incident becomes historic again, it will apear in the incidents list with an updated IncidentUpdateTime. In all such cases, it is expected that ePCR provider re-queries all APIs of this incident to get updated incident data. It is also possible for API calls to return error responses, while an incident transitions from historic to live. All such errors should be handled gracefully and all APIs should be re-queried for such incidents when they become historic again.

1.12 APIs

The following APIs are supported:

1.12.1 Authentication

Name	Type	Description	Reference
APILogin	POST	This API is used for authentication of client application	2.1
APIRefreshToken	POST	This API is used to refresh the access token once it becomes invalid.	2.2
APILogout	POST	This API is used to logout from service.	2.3

1.12.2 Customer account

Name	Type	Description	Reference
Organizations	GET	This API returns the name of organization owning the patient data.	3.1

1.12.3 Incident details (For each customer account)

Name	Type	Description	Reference

Incidents	GET	This API returns a list of historic patient incidents for the specific organization.	3.2
LiveIncidents	GET	This API returns a list of live patient incidents for the specific organization.	3.3

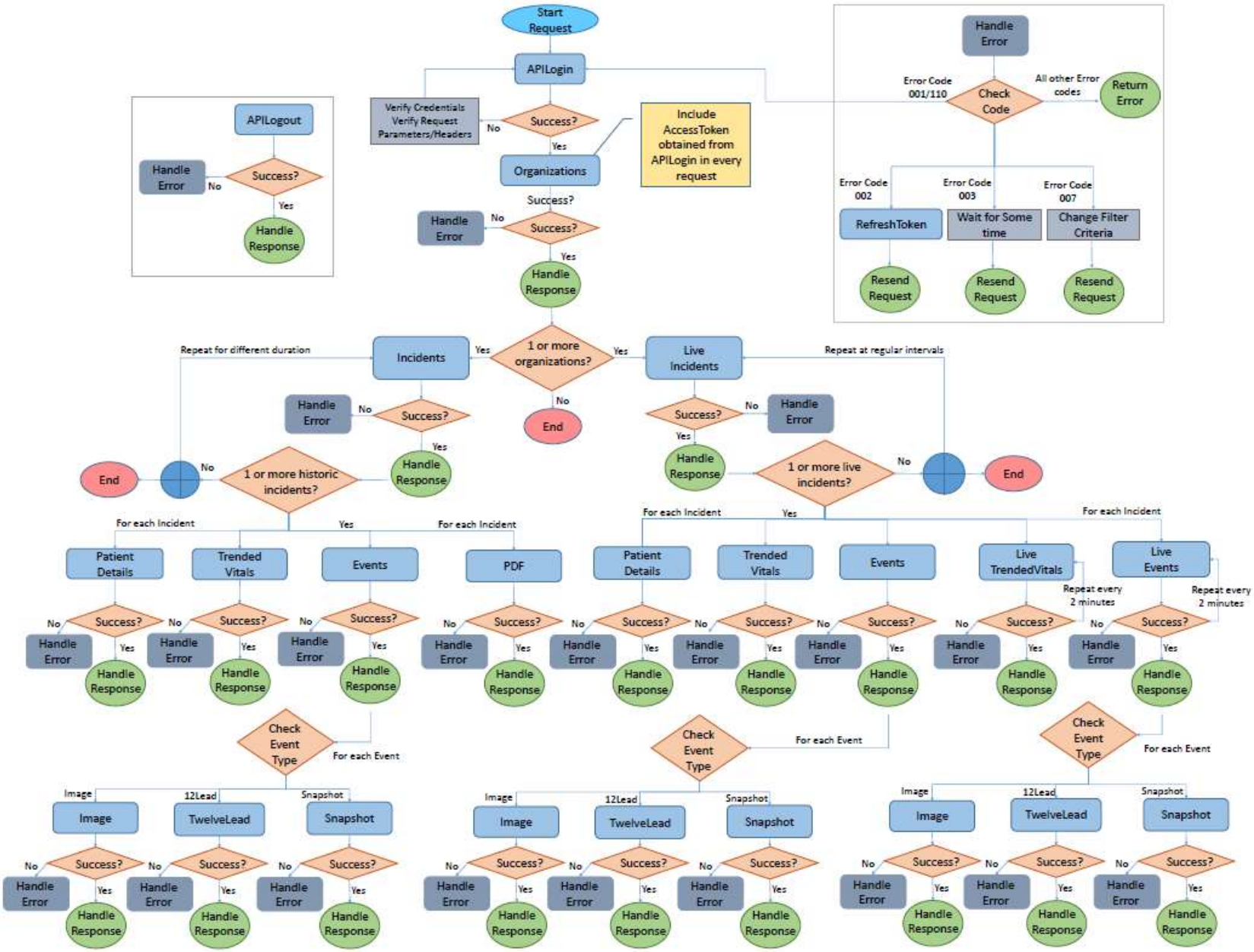
1.12.4 Patient data (For each incident)

Name	Type	Description	Reference
PatientDetails	GET	This API returns the patient details for the specified patient incident.	4.1
TrendedVitals	GET	This API returns the patient vitals for a specified historic patient incident.	4.2
LiveTrendedVitals	GET	This API returns the 2 minutes of patient trended vitals for a specified live patient incident.	4.3
Events	GET	This API returns a list of events for a specified historic patient incident.	4.4
LiveEvents	GET	This API returns a list of events recorded within 2 minutes for a specified live patient incident.	4.5
TwelveLead	GET	This API returns the 12 lead data in the requested format.	4.6
Snapshot	GET	This API returns the Snapshot data in JPEG format.	4.7
Image	GET	This API returns the Image data in JPEG format.	4.8
PDF	GET	This API returns a detailed PDF report for the specified historic incident.	4.9

Note	RDT reserves right to enhance WebAPI functionality in the future. RDT will endeavor to limit the impact on existing functionality.
Note	In TwelveLead API response lead specific detailed measurements are not available in any of the ECG formats i.e. JPEG, DICOM and aECG.
Note	Lead specific detailed measurements are available in Incident PDF via “PDF” API, if “Detailed Measurements (PDF)” option is selected in ECG display settings by Corsium transport account admin.

1.13 API flow chart

Following diagram shows the flow chart of all API calls.


Start Request
APILogin
Verify Credentials
Verify Request Parameters/Headers
No
Success?
Yes
Include AccessToken obtained from APILogin in every request
Organizations
Success?
No
Handle Error
Success?
Yes
Handle Response
APILogout
Handle Error
No
Success?
Yes
Handle Response
Handle Error
Error Code 001/110
Check Code
All other Error codes
Return Error
Error Code 002
Error Code 003
Error Code 007
RefreshToken
Wait for Some time
Change Filter Criteria
Resend Request
Resend Request
Resend Request
Repeat for different duration
Incidents
Yes
1 or more organizations?
Yes
Live Incidents
Repeat at regular intervals
No
End
Handle Error
No
Success?
Yes
Handle Response
Success?
No
Handle Error
Yes
Handle Response
End
No
1 or more historic incidents?
1 or more live incidents?
No
End
For each Incident
Yes
For each Incident
For each Incident
Yes
For each Incident
Patient Details
Trended Vitals
Events
PDF
Patient Details
Trended Vitals
Events
Live TrendedVitals
Live Events
Repeat every 2 minutes
Repeat every 2 minutes
Success?
No
Handle Error
Yes
Handle Response
Check Event Type
For each Event
Check Event Type
For each Event
Check Event Type
For each Event
Image
12Lead
Snapshot
Image
TwelveLead
Snapshot
Success?
No
Handle Error
Yes
Handle Response

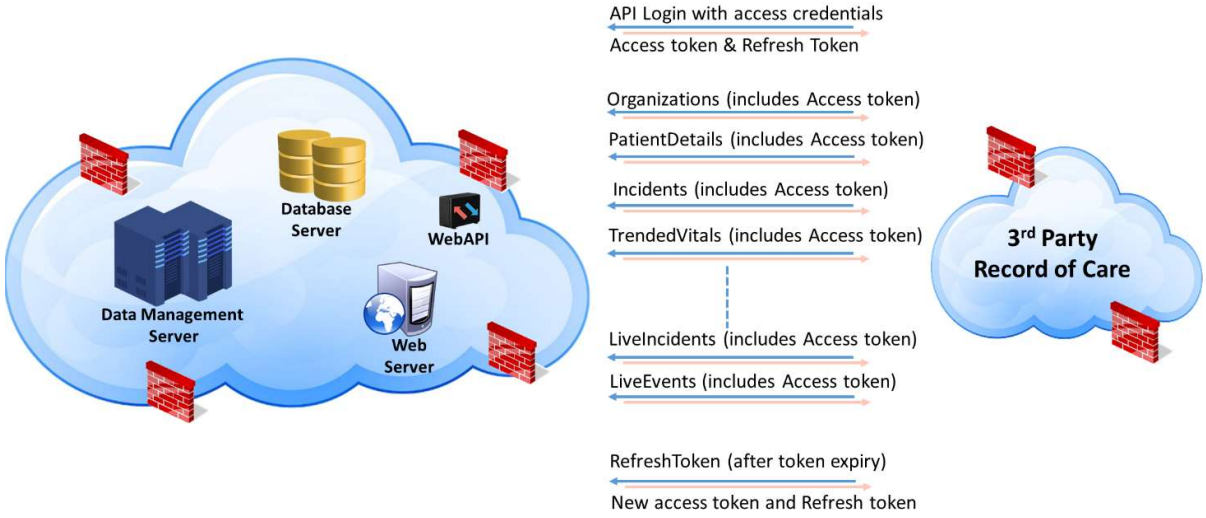
2 Account Login/Logout

2.1 API Login

Access to any of the resources in the Corsium WebAPI requires the use of an https client session that has been authenticated. Authentication is performed using the Corsium WebAPI resource, APILogin.

This resource requires login credentials for a valid Corsium user account associated with an organisation. The access rights of this user account control the data that is available through the Corsium WebAPI.

The authentication method used by the resource is token-based authentication. In this approach, the client application first sends a request to the Corsium with a valid credentials. Additional data security is achieved by requesting encrypted data sharing using a symmetric key or a public key. Corsium sends an Access token to the client as a response. This token contains enough data to identify the user and has an expiry time. The client application then uses the token to access the restricted resources in subsequent requests. If the access token expires, the client application should request a new access token by using refresh token API.



Note	When the error code 002 (Authorization key has expired) is returned, the ePCR user has a limited time to use Refresh token before the Access token expires. If WebAPI client doesn't use "Refresh token" within that window then both "Access token" and "Refresh token" will eventually expire. ePCR user will need to re-login in order to get access again.
Note	If transport provider makes any change to a particular ePCR user, error code 010 (ePCR details changed) will be returned in the next API response, if the ePCR user is logged in. The ePCR user will need to re-login, using the new login details, if applicable.
Note	For ePCR systems hosted on load-balanced infrastructure, it may be necessary to exclude WebAPI caller IP addresses from authentication tokens. Including caller IP addresses in authentication tokens is an optional, enhanced control that operates at the WebAPI request level. As an alternative or complementary security control, up to four CIDR ranges may be whitelisted for WebAPI caller IP addresses in order to restrict traffic to known, trusted networks. Only Corsium Administrators have privileges to change these security settings.

2.1.1 Resource URI

POST /Account/APILogin

Request example:

```
https://corsium.info/WebAPI/sdk/Account/APILogin
```

Request parameters should be passed in the body as a JSON object shown below:

```
{"Username":"25b8c015-7022-43f5-8b6b-8d0a55621ac0", "Password":"3P(RTe$t1n9",
"DEM":"AES", "Key":""}
```

Note	See "1.9 API Request" for details about API Request format.
Note	Refer to "7.5.1 API Login" for an example of APILogin request using Postman.

2.1.2 Parameters

Name	Type	Req/Opt	Description
Username	[string]	Req	Username used to authenticate data access.
Password	[string]	Req	Password used to authenticate data access.
DEM	[string]	Req	Data Encryption Method. Possible values are: • None • AES • PKI
Key	[string]	Optional	Base64 encoded public key. This is required when PKI is chosen as data encryption method.

2.1.3 Headers

Name	Value	Details
Accept	application/json	-
Content-Type	application/json, **or** application/json;charset=utf-8	Error code will be returned if any other charset is specified.

2.1.4 Response data

Option 1: DEM:None

If user is authenticated successfully the response data contain the generated access token and refresh token Access token is used for subsequent request in authorization header of the request.


```
{
    “ErrorCode”: null,
    “IsSuccessful”: true,
    “DataType”:”real”,
    “Data”: {
        “Access_token”:”eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9…
                        0ZS5jb20ifQ.b011rw8mhoq0pTXY-
                        lK3v_dPI4nbZpG2YrF7jOWDzc”,
        “Token_type”:”bearer”,
        “Expires_in”:1800,
        “Refresh_token”:”cbd9b3ba-350d-4bc9-b4b8-d038c694d44a”,
        “VersionWebAPI”: “2.0.0”,
        “VersionCorsium”: “1.1.4.0”
    },
    “CRC”: “57S0”
}
```


Option 2: DEM:AES

In the case that AES encryption method is selected, a randomly generated 256 bit AES Key and Initialization Vector is sent along with the token data to indicate how the data will be encrypted in further responses. The ePCR software will need to use the Key (K) and Initialization Vector (V) to decrypt the data prior to data processing.


```
{
    “ErrorCode”: null,
    “IsSuccessful”: true,
    “DataType”:”real”,
    “Data”: {
        “Access_token”:”eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9…
                        0ZS5jb20ifQ.b011rw8mhoq0pTXY-
                        lK3v_dPI4nbZpG2YrF7jOWDzc”,
        “Token_type”:”bearer”,
        “Expires_in”:1800,
        “Refresh_token”:”cbd9b3ba-350d-4bc9-b4b8-d038c694d44a”,
        “Algorithm”:”AES_256_CBC”,
        “K”:”B374A26A71490437AA024E4FADD5B497F…F12F6FB65AF2720B59CCF”
        “V”:”7E892875A52C59A3B588306B13C31FBD”,
        “VersionWebAPI”: “2.0.0”,
```

```
        "VersionCorsium": "1.1.4.0"
    },
    "CRC": "5tr9"
}
```

Note	"K" and "V" parameters are Base64 encoded binary data. Decode them to get the encryption Key and Initialization Vector. Below are some additional details of the AES encryption algorithm: • Block size = 128 • Key size = 256 • Initialization Vector = <V value in bytes> • Key = <K value in bytes> • Mode = CBC • Padding = PKCS7 Steps to decrypt response data using "K" and "V" parameters : 1. Base64 decode "K" and "V" parameters to get actual Key and Initialization Vector bytes. 2. Base64 decode response data to get response data bytes. 3. AES decrypt decoded response data using Key and Initialization Vector obtained in Step 1. Refer to "7.6.1 AES encrypted data" for an example of data decryption.
Note	Character encoding used during data encryption is UTF-16LE. Use this encoding to get JSON string back from decrypted byte array. Refer to Step 5 of "7.6.1 AES encrypted data" for an example.

Option 3: DEM:PKI, Key:<user-provided base 64 encoded public key>

In the case that PKI encryption method is selected, a randomly generated 256 bit AES Key and Initialization Vector are sent along with the token data to indicate how the data will be encrypted in further responses. The "K" and "V" parameters in the data field are encrypted with the public key provided as part of the request. Thereby the "K" and "V" parameters need to be decrypted using private key held by the customer.

```
{
    "ErrorCode": null,
    "IsSuccessful": true,
    "DataType":"real",
    "Data": {
        "Access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9…
                        0ZS5jb20ifQ.b011rw8mhoq0pTXY-
                        lK3v_dPI4nbZpG2YrF7jOWDzc",
        "Token_type":"bearer",
        "Expires_in":1800,
        "Refresh_token":"cbd9b3ba-350d-4bc9-b4b8-d038c694d44a",
        "Algorithm":"AES_256_CBC",
        "K":"B374A26A71490437…F1259CCF", //Encrypted with Public Key and Base64 encoded
        "V":"7E89287…B588306B13C31FFBD", //Encrypted with public key and Base64 encoded
```

```
        "VersionWebAPI": "2.0.0",
        "VersionCorsium": "1.1.4.0"
    },
    "CRC": "5tr9"
}
```


The following table provides details of the fields present in the JSON-formatted data response:

Attribute	Type	Description
Access_token	String	This token will need to be included in request header of all subsequent API requests. This will be verified on the server for a valid user session.
Token_type	String	It is always "bearer" for WebAPI request.
Expires_in	Integer	Number of seconds in which token will expire.
Refresh_token	String	This token will be used to re-request the access_token when it is about to expire.
Algorithm	String	Algorithm used for symmetric encryption of data. Currently it will always be "AES_256_CBC"
K	String	Dynamically generated key for symmetrically encrypted data.
V	String	Dynamically generated initialization vector of symmetrically encrypted data
VersionCorsium	String	Version of Corsium used by Transport provider
VersionWebAPI	String	Version of WebAPI on Corsium. Version format is X.Y.Z, where • X is major update to WebAPI interface, meaning it is no longer compatible with the previous version • Y is minor backwards-compatible interface update • Z is bug-fixes

Note 	"K" and "V" parameters are Base64 encoded binary data. Decode them to get the encryption Key and Initialization Vector which are encrypted using public key specified in the request. Decrypt these values using private key to get actual "K" and "V" values. Steps to decrypt response data using "K" and "V" parameters : 1. Base64 decode "K" and "V" parameters to get encrypted Key and Initialization Vector bytes. 2. Decrypt Key and Initialization Vectors using Private Key used for PKI to get actual Key and Initialization Vector bytes. 3. Base64 decode response data to get response data bytes. 4. AES decrypt decoded response data using Key and Initialization Vector obtained in Step 2. Refer to "7.6.2 PKI encrypted data" for an example
Note 	Character encoding used during data encryption is UTF-16LE. Use this encoding to get JSON string back from decrypted byte array. Refer to Step 5 of "7.6.1 AES encrypted data" for an example.

2.2 API Refresh Token

This API is used to refresh the access token in the case that it has expired.

2.2.1 Resource URI

POST /Account/APIRefreshToken

Request example:

```
https://corsium.info/WebAPI/sdk/Account/APIRefreshToken
```

Request parameters will be passed in the body as a JSON object shown below:

```
{"refreshToken":"cbd9b3ba-350d-4bc9-b4b8-d038c694d44a"}
```

Note	Refer to "7.5.3 Refresh Token" for an example of APIRefreshToken request using Postman.

2.2.2 Parameters

Name	Type	Req/Opt	Description
RefreshToken	[string]	Req	Refresh token obtained from previous API login request

2.2.3 Headers

Name	Value	Details
Accept	application/json	-
Authorization	bearer <Access token>	<Access token> is obtained from APILogin response

Note	See "1.9 API Request" for details on API Request format

2.2.4 Response data

Response data is same as APILogin response with a new access token and a refresh token.

2.3 API Logout

This API is used to logout from the Corsium WebAPI service.

2.3.1 Resource URI

POST /Account/APILogout

Request example:

```
https://corsium.info/WebAPI/sdk/Account/APILogout
```

2.3.2 Parameters

None

2.3.3 Headers

Name	Value	Details
Accept	application/json	-
Authorization	bearer <Access token>	<Access token> is obtained from APILogin response

2.3.4 Response data

Response data is empty.

If “IsSuccessful” field is true, it means that the user has been logged out successfully. Both access and refresh tokens are made invalid on the server.

If “IsSuccessful” field is false, appropriate error message is sent in the “ErrorCode” field.

3 Corsium Clinical API Resources

Patient “Incidents” are the key resource for accessing the clinical information in Corsium. Vitals, events, snapshots, 12 lead ECGs, images are all sub-resources of the incident resource.

Incidents are ‘owned’ by an organization, specifically by the organization that sent the device data to Corsium.

All of the Clinical API resources are read-only. Only the HTTP GET verb is supported.

3.1 Organizations

This API returns the name of organization owning the patient data.

3.1.1 Resource URI

GET /Api/Organizations

Request example:

```
https://corsium.info/WebAPI/sdk/Api/Organizations
```

3.1.2 Parameters

None

3.1.3 Headers

Name	Value	Details
Accept	application/json	-
Authorization	bearer <Access token>	<Access token> is obtained from APILogin response

3.1.4 Response data

```
[
    {
        “OrganizationId”: “8b1caff4-4778-4cdf-bb54-8ec37f33b89f”,
        “Name”: “Major Metro EMS”
    }
]
```

The following table provides details of the fields present in the JSON-formatted data response:

Attribute	Type	Description
OrganizationId	String	ID of the organization
Name	String	Name of the organization

Note	
	Each ePCR user account is associated with one customer account. Therefore, there will be only one record in the organization list.

Note	The ePCR provider shall have separate login for each Transport provider.

3.2 Incidents

A request for the incidents resource returns a list of historic patient incidents for the specific EMS Agency.

The list can optionally be filtered by a device serial number.

Note	The organizationId is a required parameter. Only historic incidents belonging to that organization are returned in the incident list.
Note	The Incident list includes only historic incidents, any live incidents are excluded from the list. To get the list of the live incidents LiveIncidents API should be called, see “3.3 LiveIncidents”.

3.2.1 Resource URI

GET /Api/Clinical/Incidents

Request example:

```
https://corsium.info/WebAPI/sdk/Api/Clinical/Incidents?organizationId=966B3E60-4749-420C-A2DA-D3FC416E82AD&fromDate=2020-07-26T00:00:00&toDate=2020-07-26T01:00:00
```

3.2.2 Parameters

Name	Type	Req/Opt	Description
organizationId	[string]	Req	ID of the organization for desired Incident list.
tempusSerialNumber	[string]	Opt	Filter Incidents by Tempus Pro device Serial Number. No wildcards are accepted. The Device number has to be an exact match.
useIncidentStartTime	[boolean]	Opt	If set to True, then incidents are returned on the basis of StartTime, not IncidentUpdateTime.
search	[string]	Opt	Incidents returned based on a keyword. Following information will be considered for search in the patient incident data : • Patient ID • Patient Name • Device Name • Device Group Specified search keyword is searched anywhere in the value of the fields mentioned above. Search value is case-insensitive and does not support wildcards.

Name	Type	Req/Opt	Description
fromDate	[DateTime]	Req	Filter incidents that were updated only after this date, inclusive. Format: yyyy-MM-ddTHH:mm:ss
toDate	[DateTime]	Req	Return incidents that were updated only before this date, inclusive. Format: yyyy-MM-ddTHH:mm:ss

Note	Data access time is controlled by the EMS Agency. The default data access period is 72 hours. Contact the EMS Agency to change the access period. If API requests data outside allowed time period, then an error code is returned, see “7.1 Response error messages”.
Note	By default the maximum date range for the incident search is 24 hours. If tempusSerialNumber is specified, then the maximum date range for the incident search is 6 months.
Note	The maximum number of incidents returned for Incidents API call is 500. If there are more than 500 incidents in the configured timeframe, then an error code is returned, see “7.1 Response error messages”. If the timeframe is set to 1 hour or less, then all incidents are returned, regardless the number.
Note	If useIncidentStartTime is not used or set to False, the incidents are filtered by the IncidentUpdateTime, not the StartTime. This means that if the incident was readmitted and updated at the later date, it will re-appear in the incident list based on the IncidentUpdateTime. Incidents are sorted in the chronological order.
Note	If useIncidentStartTime is set to True, then incidents are returned on the basis of StartTime, not IncidentUpdateTime.

3.2.3 Headers

Name	Value	Details
Accept	application/json	
Authorization	bearer <Access token>	<Access token> is obtained from APILogin response

3.2.4 Response data

```
[
    {
        “IncidentId”: “D275FE64-B9A3-067D-5CBA-C61872832EF0”,
        “TempusSerialNumber”: “40007”,
        “SendOrganizationName”: “HEMS”,
        “StartTime”: “2019-03-22T13:32:00”,
        “PatientId”: “HR1246”,
        “IncidentUpdateTime”: “2019-03-22T14:23:56”,
```

```
        "DemoData": true
    },
    {
        "IncidentId": "5FE6S274-B9A3-93HD-PTG4-32ETC618728F",
        "TempusSerialNumber": "40007",
        "SendOrganizationName": "HEMS",
        "StartTime": "2019-03-22T13:55:00",
        "PatientId": "LM5619",
        "IncidentUpdateTime": "2019-03-22T14:23:56",
        "DemoData": true
    }
]
```


The following table provides details of the fields present in the JSON-formatted data response:

Attribute	Type	Description
IncidentId	String	Unique identifier for an incident.
TempusSerialNumber	String	Tempus Pro device serial number.
SendOrganizationName	String	Corsium customer name (Transport account organisation name)
StartTime	String	Incident start time in UTC format.
IncidentUpdateTime	String	Time when the incident was last updated in database in UTC format.
PatientId	String	ID of the patient. This field will be blank if sharing patient identification information is restricted by Corsium administrator.
DemoData	Boolean	This field is set to True if incident contains demo data.

3.3 LiveIncidents

A request for the incidents resource returns a list of live patient incidents for a given EMS Agency.

Note	The organizationId is a required parameter. Only live incidents belonging to that organization are returned in the incident list.
Note	Complete list of currently live incidents is returned when LiveIncidents API is called.
Note	The incidents are sorted in chronological order.

3.3.1 Resource URI

GET /Api/Clinical/LiveIncidents

Request example:

```
https://corsium.info/WebAPI/sdk/Api/Clinical/LiveIncidents?organizationId=966B3E60-4749-420C-A2DA-D3FC416E82AD
```

3.3.2 Parameters

Name	Type	Req/Opt	Description
organizationId	[string]	Req	ID of the organization for desired live Incident list.
tempusSerialNumber	[string]	Opt	Filter Incidents by Tempus Pro device Serial Number. No wildcards are accepted. The Device number has to be an exact match.

3.3.3 Headers

Name	Value	Details
Accept	application/json	
Authorization	bearer <Access token>	<Access token> is obtained from APILogin response

3.3.4 Response data


```
[
    {
        "IncidentId": "D275FE64-B9A3-067D-5CBA-C61872832EF0",
        "TempusSerialNumber": "40007",
        "SendOrganizationName": "HEMS",
        "StartTime": "2019-03-22T13:32:00",
        "PatientId": "HR1246",
        "PatientDetailsUpdateTime": "2019-03-22T16:23:56.864",
        "DemoData": false
    },
    {
        "IncidentId": "5FE6S274-B9A3-93HD-PTG4-32ETC618728F",
        "TempusSerialNumber": "40007",
        "SendOrganizationName": "HEMS",
        "StartTime": "2019-03-22T13:55:00",
        "PatientId": "LM5619",
        "PatientDetailsUpdateTime": "2019-03-23T14:23:56",
        "DemoData": true
```

```
    }
]
```

The following table provides details of the fields present in the JSON-formatted data response:

Attribute	Type	Description
IncidentId	String	Unique identifier for an incident.
TempusSerialNumber	String	Tempus Pro device serial number.
SendOrganizationName	String	Corsium customer name (Transport account organisation name).
StartTime	String	Incident start time in UTC format.
PatientId	String	ID of the patient. This field will be blank if sharing patient identification information is restricted by the Corsium administrator.
DemoData	Boolean	This field is set to True if incident contains demo data.
PatientDetailsUpdateTime	String	Time when the patient details were last updated in database in UTC format. For PatientDetailsUpdateTime only: the time shall be returned with milliseconds, i.e. 2016-03-25T11:15:58.056

4 Incidents Sub-resources

This section describes APIs that can be called for a specific incident.

Note	There are limits on how often APIs can be called. In the case that limits are exceeded, an error code is returned, see Appendix 7.1 for details. We recommend that you wait for 1 minute before calling the next API.
Note	Please contact RDT if the throttling limit is hit on a regular basis.

4.1 Patient Details

A request for the PatientDetails resource returns patient details for a specified patient incident. Use one of the persistent IncidentIds returned from the incident resource as the IncidentID in the Patient Details URI when making this request.

4.1.1 Resource URI

GET /Api/Clinical/Incidents/{IncidentId}/PatientDetails

Request example:

```
https://corsium.info/WebAPI/sdk/Api/Clinical/Incidents/FA68370A-BF24-4BD1-99BE-DF946C8ECDDE/PatientDetails
```

4.1.2 Parameters

None

4.1.3 Headers

Name	Value	Details
Accept	application/json	-
Authorization	bearer <Access token>	<Access token> is obtained from APILogin response

4.1.4 Response data

```
[
	{
		“IncidentId”: “D275FE64-B9A3-067D-5CBA-C61872832EF0”,
		“StartTime”: “2019-03-22T13:32:00”,
		“PatientDetailsUpdateTime”: “2019-03-22T10:23:56.658”,
		“IncidentUpdateTime”: “2019-03-22T14:23:56”,
		“SendOrganizationName”: “HEMS”,
		“PatientAge”: “35 years”,
		“PatientType”: “Adult”,
```

```
            "PatientGender": "Male",
            "FirstName: "John",
            "LastName": "Smith",
            "TempusSerialNumber": "40007",
            "TempusName": "MEDIC-21",
            "TempusOrganization": "HEMS 2",
            "SupportCentre": "GOSH",
            "DeviceGroup": "Bike Team",
            "TempusSoftware": "v04.16.083",
            "PatientId": "HR1246",
            "Height": "173",
            "HeightUnits": "cm",
            "WeightUnits": "kg",
            "Weight": "75",
            "ReportType": "Standard",
            "Allergies":[
              {
                    "Name": "Aspirin",
                    "Code": ""
              },
              {
                    "Name": "Latex",
                    "Code": ""
              }]
      }
]
```


The following table provides details of the fields present in the JSON-formatted data response:

Attribute	Type	Description
IncidentId	String	Unique identifier for an incident.
StartTime	String	Tempus Pro incident start time in UTC format.
PatientDetailsUpdateTime	String	Time when the patient details were last updated in database in UTC format. For PatientDetailsUpdateTime only: the time shall be returned with milliseconds, i.e. 2016-03-25T11:15:58.056
IncidentUpdateTime	String	Time when the incident was last updated in database in UTC format. If incident is live, this field is blank.
SendOrganizationName	String	Corsium customer name (Transport account Organisation name)
TempusSerialNumber	String	Tempus Pro device serial number.
TempusName	String	Tempus Pro device name
TempusOrganization	String	Organization name configured on Tempus

Attribute	Type	Description
DeviceGroup	String	Device group configured on Tempus Pro
SupportCentre	String	Support centre configured on Tempus Pro
TempusSoftware	String	Tempus Pro software version.
PatientAge	String	Age of the patient in the following format: <value> years Blank if age was not set.
PatientType	String	This field will contain one of the following: • Adult • Neonate • Paediatric
PatientGender	String	Gender of the patient which will be set to one of the following: • Male • Female • Unknown • Other
ReportType	String	This field will contain report type and can be one of the following: • Standard • TC3
LastName	String	Last name of the patient. This field will be blank if transport provider is configured not to share PHI.
FirstName	String	First name of the patient. This field will be blank if transport provider is configured not to share PHI.
PatientId	String	Patient ID in the case of Standard report type. Last 4 / Patient ID in the case of TC3 report type. This field will be blank if sharing patient identification information is restricted by the Corsium administrator.
Weight	String	The weight of the patient, this field will be blank in the case that the weight is not set.
WeightUnits	String	Weight measurement unit which will be set to one of the following: • kg • lbs
Height	String	Height value of the patient, this field will be blank in the case that the height is not set.

Attribute	Type	Description
HeightUnits	String	Height measurement unit which will be set to one of the following: • cm • inch
Service	String	*In TC3 report type only* Service
Unit	String	*In TC3 report type only* Unit
Evac	String	*In TC3 report type only* This parameter will be set to one of the following: • Urgent • Priority • Routine • *Blank* (In the case that evac is not set)
BattleRoster	String	*In TC3 report type only* Battle Roster
Allergies	Array of Allergy objects	Please refer to the section below.

Definition of Allergies

Attribute	Type	Description
Name	String	Name of the allergy
Code	String	Identification code of Allergy stored in Tempus Pro. This field is currently blank, it may be used in the future.

4.2 Trended Vitals

A request for the TrendedVitals resource returns a list of patient vitals for a specified historic patient incident. Use one of the persistent IncidentIds returned from the Incidents resource as the IncidentId in the Vitals Resource URI when making this request.

4.2.1 Resource URI

GET /Api/Clinical/Incidents/{IncidentId}/TrendedVitals

Request example:

```
https://corsium.info/WebAPI/sdk/Api/Clinical/Incidents/FA68370A-BF24-4BD1-99BE-
DF946C8ECDDE/TrendedVitals
```

4.2.2 Parameters

None

4.2.3 Headers

Name	Value	Details
Accept	application/json	-
Authorization	bearer <Access token>	<Access token> is obtained from APILogin response

4.2.4 Response data


```
[
        "IncidentId": "f2e0e11f-721f-4d32-8a84-267a60b5c829",
        "VitalDetailsInfo": [{
               "DataCaptureTimeUtc": "2016-03-25T11:14:00",
               "Vitals":[
                    {
                        "Name": "HR",
                        "Value": "80",
                        "Measurement": "bpm",
                        "Code": "8867-4",
                        "Source": "ECG",
                        "Alert": false
                    },
                    {
                        "Name": "SysPAP",
                        "Value": "120",
                        "Measurement": "mmHg",
                        "Code": "8867-4",
                        "Source": "P1:PAP",
                        "Alert": false
                    }]
              },
              {
                "DataCaptureTimeUtc": "2016-03-25T11:15:00",
                "Vitals":[
                    {
                        "Name": "HR",
                        "Value": "80",
                        "Measurement": "bpm",
                        "Code": "8867-4",
                        "Source": "ECG",
                        "Alert": false
                    },
                    {
                        "Name": "SysPAP",
```

```
                    "Value": "120",
                    "Measurement": "mmHg",
                    "Code": "8867-4",
                    "Source": "P1:PAP",
                    "Alert": false
                }]
        }]
]
```

The following table provides details of the fields present in the JSON-formatted data response:

Attribute	Type	Description
IncidentId	String	Unique identifier for an incident
VitalDetailsInfo	Array of VitalDetailsInfo objects	Please refer to the section below

Definition of VitalDetailsInfo

Attribute	Type	Description
DataCaptureTimeUtc	String	Time when vitals were recorded in UTC format
Vitals	Array of Vital objects	Please refer to the section below.

Definition of Vitals

Attribute	Type	Description
Name	String	Name of the vital, see Appendix 7.2 for details
Value	String	Value of the vital or in the case that the recorded value is outside of the valid range then the following text will be provided: “Out of range”
Measurement	String	Unit of measurement, see Appendix 7.2 for details
Code	String	Code of the vital, e.g. LOINC code. This can be configured using Tempus Pro Configuration Utility.
Source	String	Source information for the vital signs, see Appendix 7.2 for details
Alert	Boolean	Indicates if the value is inside or outside the limits configured on the device at the time the vital was recorded. Options are as follows: • True – outside limits; • False – within limits.

Note	All trended vitals for the historic incident are returned. Vitals are in chronological order.
Note	Vitals are trended at 1 minute intervals, except NIBP where all discrete readings shall be recorded.

4.3 Live Trended Vitals

A request for the LiveTrendedVitals resource returns a list of vitals recorded within 2 minutes before the specified time.

Use one of the persistent IncidentIds returned from the Incidents resource as the IncidentId in the Vitals Resource URI when making this request.

4.3.1 Resource URI

GET /Api/Clinical/Incidents/{IncidentId}/LiveTrendedVitals

Request example:

```
https://corsium.info/WebAPI/sdk/Api/Clinical/Incidents/FA68370A-BF24-4BD1-99BE-DF946C8ECDDE/LiveTrendedVitals?toDate=2016-03-25T11:15:00
```

4.3.2 Parameters

Name	Type	Req/Opt	Description
toDate	[DateTime]	Req	Return vitals that were updated within 2 minutes before this date, inclusive. Format: yyyy-MM-ddTHH:mm:ss

4.3.3 Headers

Name	Value	Details
Accept	application/json	-
Authorization	bearer <Access token>	<Access token> is obtained from APILogin response

4.3.4 Response data

```
[
       “IncidentId”: “f2e0e11f-721f-4d32-8a84-267a60b5c829”,
       "VitalDetailsInfo": [{
              “DataCaptureTimeUtc”: “2016-03-25T11:14:00”,
              “Vitals”:[
                  {
                      “Name”: “HR”,
                      “Value”: “80”,
                      “Measurement”: “bpm”,
```

```
                    "Code": "8867-4",
                    "Source": "ECG",
                    "Alert": false
                },
                {
                    "Name": "SysPAP",
                    "Value": "120",
                    "Measurement": "mmHg",
                    "Code": "8867-4",
                    "Source": "P1:PAP",
                    "Alert": false
                }]
            },
            {
              "DataCaptureTimeUtc": "2016-03-25T11:15:00",
              "Vitals":[
                {
                    "Name": "HR",
                    "Value": "80",
                    "Measurement": "bpm",
                    "Code": "8867-4",
                    "Source": "ECG",
                    "Alert": false
                },
                {
                    "Name": "SysPAP",
                    "Value": "120",
                    "Measurement": "mmHg",
                    "Code": "8867-4",
                    "Source": "P1:PAP",
                    "Alert": false
                }]
        }]
]
```


Note	Incident vitals are returned in the chronological order.
Note	If there was loss in connectivity, or the incident was monitored for some time without being connected to Corsium, then the returned vitals will include the all vitals captured in that period.
Note	To ensure data integrity RDT recommends to pull the incident as historic, once it is no longer live.

The following table provides details of the fields present in the JSON-formatted data response:

Attribute	Type	Description
IncidentId	String	Unique identifier for an incident
VitalDetailsInfo	Array of VitalDetailsInfo objects	Please refer to the section below

Definition of VitalDetailsInfo

Attribute	Type	Description
DataCaptureTimeUtc	String	Time when vitals were recorded in UTC format
Vitals	Array of Vital objects	Please refer to the section below.

Definition of Vitals

Attribute	Type	Description
Name	String	Name of the vital, see Appendix 7.2 for details
Value	String	Value of the vital or in the case that the recorded value is outside of the valid range then the following text will be provided: “Out of range”
Measurement	String	Unit of measurement, see Appendix 7.2 for details
Code	String	Code of the vital, e.g. LOINC code. This can be configured using Tempus Pro Configuration Utility.
Source	String	Source information for the vital signs, see Appendix 7.2 for details
Alert	Boolean	Indicates if the value is inside or outside the limits configured on the device at the time the vital was recorded. Options are as follows: • True – outside limits; • False – within limits.

4.4 Events

A request for the Events resource returns a list of events for a specified historic patient incident. Use one of the persistent IncidentIds returned from the Incidents resource as the IncidentId in the Events Resource URI when making this request.

No file data (for ECG, snapshots or images) is included in the response. The data in the returned list includes EventIds for each event that can then be used in the Resource URI’s for separate requests to retrieve individual events with files embedded in each response, see example.

4.4.1 Resource URI

GET /Api/Clinical/Incidents/{IncidentId}/Events

Request example:

```
https://corsium.info/WebAPI/sdk/Api/Clinical/Incidents/FA68370A-BF24-4BD1-99BE-DF946C8ECDDE/Events
```

4.4.2 Parameters

Name	Type	Req/Opt	Description
eventType	[string]	Opt	Only events of specified event type are returned in the response. If a blank value is specified for eventType, all events will be returned in the response. “eventType” value is case-insensitive and supports only one event type at a time. See Section 4.4.4 for list of supported Event Types

4.4.3 Headers

Name	Value	Details
Accept	application/json	-
Authorization	bearer <Access token>	<Access token> is obtained from APILogin response

4.4.4 Response data


```
[
       “IncidentId”: “06608acb-0f57-4908-9a08-43041b6afed8”,
       “EventsInfo”: [
          {
               “EventId”: “8de9a652-46f8-42e9-aae8-7bb487297147”,
               “DataCaptureTimeUtc”: “2016-03-25T11:15:58”,
               “EventTimeUtc”: “2016-03-25T11:15:58”,
               “HasVitals”: true,
               “EventType”: ”DRUG”,
               “EventLabel”: “Adenosine”,
               “EventDetails”: [{
                      “Code”: “”,
                      “Dose”: “as per protocol”,
                      “Route”: “IV”
                      }],
               “EventNotes”: ”Patient responded well”,
               “EventDataStatus”: ”Not required”,
               “EventStatus”:”New”,
               “Vitals”: [
                   {
                        “Name”: “HR”,
                        “Value”: “80”,
                        “Measurement”: “bpm”,
                        “Code”: “8867-4”,
```

```
                    "Source": "ECG",
                    "Alert": false
                }
        }]
]
```

The following table provides details of the fields present in the JSON-formatted data response:

Note	All incident events are returned in the chronological order.
Note	Only current events are included in the historic event data response. Deleted or superseded events are not part of the response.

Attribute	Type	Description
IncidentId	String	Unique identifier for an incident.
EventsInfo	Array of EventsInfo objects	Please refer to the section below.

Definition of EventsInfo

Attribute	Type	Description
DataCaptureTimeUtc	String	Time when event was first recorded or edited or deleted in UTC format
EventTimeUtc	String	Time of the event in UTC format. This may be different to the DataCaptureTimeUtc, as event time can be edited on Tempus. The event time is the time that is displayed in Summary tab on the Tempus/Corsium. For Tempus LS events only: the event time shall be returned with milliseconds, i.e. 2016-03-25T11:15:58.056
EventId	String	Unique ID of the event.
HasVitals	Boolean	Set to true, when event has associated vitals.
EventType	String	Type of the recorded event. See table below for possible event types.
EventLabel	String	Name of the recorded event. For Drug, Fluid, Assessment and Intervention events the name can be edited using Tempus Pro Configuration Utility. See the table below for event label details.
EventDetails	Array of EventDetails	Data as defined below for different Event types

Attribute	Type	Description
EventNotes	String	Notes entered during the encounter for the specific data item, this field can be blank. For the General Note event this field will contain the entered note.
EventDataStatus	String	Status of the event data. Possible values: • Not ready – the event has come but the file is not available • Ready – both event and associated file data have been received • Not Available - file data is not available • Not required – file data is not required
EventStatus	String	Indicated whether the event is new or modified ("Current" status is returned). • New • Current
Vitals	Array of Vitals	See Section 4.2 for Vitals details. Event vitals include only the following: • HR / Pulse • SpO2 • PI • PVI • Resp • EtCO2 • NIBP • T1 / T2 All other vitals shall not be included in the response.

Event Type	Event Label
DRUG_MARKER	Drug Marker
FLUID_MARKER	Fluid Marker
INTERVENTION_MARKER	Intervention Marker
ASSESSMENT_MARKER	Assessment Marker
DRUG	*<Name of the Drug>*
FLUID	*<Name of the Fluid>*
INTERVENTION	*<Name of the Intervention>*
ASSESSMENT	*<Name of the Assessment>*

Event Type	Event Label
CAMERA_IMAGE	Camera Image *Use "Image" API to retrieve file data, see Section 4.8.*
LARYNGOSCOPE_IMAGE	Laryngoscope Image *Use "Image" API to retrieve file data, see Section 4.8*
GENERAL_NOTE	Note
TWELVE_LEADS	12 lead ECG *Use "TwelveLead" API to retrieve file data, see Section 4.6*
WAVEFORM	Waveforms *Use "Snapshot" API to retrieve file data, see Section 4.7*
ARRHYTHMIA	*<Arrhythmia name>* Possible names: • Extreme Brady • Extreme Tachy • Vent Fib • Asystole • Vent Tachy *Use "Snapshot" API to retrieve file data, see Section 4.7*
PATIENT_ALARM	Alarm Waveforms *Use "Snapshot" API to retrieve file data, see Section 4.7*
GCS	GCS
INJURY_MAP	Injury Map
DRESSING_MAP	Dressing Map
BURNS_MAP	Burns Map
TQ_MAP	TQ Map
MEDICS	Medics *Medic events will be omitted from the response, if transport provider is configured not to share PHI.*
AVPUPAIN	AVPU and Pain
AIRWAY_NOTE	Airway Type
BREATHING_NOTE	Breathing Type
CIRC_TQ_NOTE	Circulation Type
CIRC_DRESSING_NOTE	Dressing Type
OTHER_TREATMENT_NOTE	Other Type
TC3_NOTE	TC3 Note
DEFIB_SHOCK	Shock *#X* *where X is the shock count, e.g. Shock #1, Shock #2, etc.* *Use "Snapshot" API to retrieve file data, see Section 4.7*

Event Type	Event Label
DEFIB_ANALYSIS_REQUESTED	Analysis Requested
DEFIB_SHOCK_ADVISED	Shock Advised *Use "Snapshot" API to retrieve file data, see Section 4.7*
DEFIB_NO_SHOCK_ADVISED	No Shock Advised *Use "Snapshot" API to retrieve file data, see Section 4.7*
DEFIB_PACING	Pacing *Use "Snapshot" API to retrieve file data, see Section 4.7*
DEFIB_PACING_STOPPED	Pacing Stopped
DEFIB_CPR_START	CPR Start
DEFIB_CPR_END	CPR End
DEFIB_INTERVENTION_START	Defib Incident Start
DEFIB_MANUAL_MODE	Manual Mode
DEFIB_AED_MODE	AED Mode
DEFIB_PACER_MODE	Pacer Mode
DEFIB_MONITOR_MODE	Monitor Mode
DEFIB_PADS_ATTACHED	Pads Attached
DEFIB_PADS_DETACHED	Pads Detached
DEFIB_SNAPSHOT_WAVEFORMS	Waveforms *Manually captured waveforms by Tempus LS.* *Use "Snapshot" API to retrieve file data, see Section 4.7*
DEFIB_SECURITY_DISCHARGE	Security Discharge

Definition of EventDetails

Event Type	Attribute	Type	Description
ASSESSMENT	Code	String	Identification code of the assessment, this value can be edited using the Tempus Pro Configuration Utility.
INTERVENTION	Code	String	Identification code of the intervention, this value can be edited using the Tempus Pro Configuration Utility.
DRUG	Code	String	Identification code of the drug, this value can be edited using the Tempus Pro Configuration Utility.
	Dose	String	Amount of drug administered along with units, this field can be blank.

Event Type	Attribute	Type	Description
	Route	String	Route through which drug was administered, this field can be blank.
FLUID	Code	String	Identification code of the fluid, this value can be edited using the Tempus Pro Configuration Utility.
	Volume	String	Volume of fluid given along with units, this field can be blank.
GCS	Eye	String	The eye score can be one of the following: 1,2,3,4
	Verbal	String	The verbal score can be one of the following: 1,2,3,4,5 or set to T when the patient is intubated.
	Motor	String	The Eye score can one of the following: 1,2,3,4,5,6
	Score	String	GCS total score.
MEDICS	FirstName	String	First name of the medic
	LastName	String	Last name of medic
	Role	String	*In Standard report type only* Role of the medic
	Last4	String	*In TC3 report type only* Last4 of the medic
INJURY_MAP	BodyMaps	Array of BodyMaps	Please refer to the section below.
DRESSING_MAP	BodyMaps	Array of BodyMaps	Please refer to the section below.
BURNS_MAP	Percent	String	Percentage of the body that is burnt. Note that this field is not present in injury and dressing maps.
	BodyMaps	Array of BodyMaps	Please refer to the section below.
TQ_MAP	TQMaps	Array of TQMaps	Please refer to the section below.
TWELVE_LEADS	ReviewDetails	Array of ReviewDetails	Please refer to the section below.
AVPUPAIN	AVPU	String	AVPU value recorded
	Pain	String	Pain value recorded
DEFIB_SHOCK	Count	Integer	Shock count.
	EnergySel	Integer	Selected energy in Joules.

Event Type	Attribute	Type	Description
	EnergyDel	Integer	Delivered energy in Joules.
	Impedance	Integer	Impedance in Ohms.
	Sync	Boolean	True, if Sync shock delivered
	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
DEFIB_ANALYSIS_REQUESTED	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
DEFIB_SHOCK_ADVISED	Rhythm	String	Details of analysis. Currently always: • Shockable rhythm
	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
DEFIB_NO_SHOCK_ADVISED	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
DEFIB_PACING	Mode	String	Pacing mode, can be one of the following: • Fix • Demand
	Current	Integer	Pacing current in mA.
	Frequency	Integer	Pacing frequency /min.
	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
DEFIB_PACING_STOPPED	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number

Event Type	Attribute	Type	Description
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
DEFIB_CPR_START	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
DEFIB_CPR_END	Rate	Integer	CPR rate /min
	Quality	Integer	CPR quality
	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
DEFIB_INTERVENTION_START	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
	Offset	Integer	Offset time in milliseconds from start of intervention.
DEFIB_MANUAL_MODE	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
DEFIB_AED_MODE	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
DEFIB_PACER_MODE	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
DEFIB_MONITOR_MODE	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version

Event Type	Attribute	Type	Description
DEFIB_PADS_ATTACHED	Connector	String	Type of connected pads. Can be one of the following: • Adult Pads • Paediatric Pads
	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
DEFIB_PADS_DETACHED	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
DEFIB_SECURITY_DISCHARGE	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
DEFIB_SNAPSHOT_WAVEFORMS	Offset	Integer	Offset time in milliseconds from start of intervention.
	LSSerial	String	Tempus LS serial number
	LSName	String	Tempus LS name
	LSSoftware	String	Tempus LS software version
WAVEFORM	n/a	n/a	*No EventDetails available*
ARRHYTHMIA	n/a	n/a	*No EventDetails available*
PATIENT_ALARM	n/a	n/a	*No EventDetails available*
CAMERA_IMAGE	n/a	n/a	*No EventDetails available*
LARYNGOSCOPE_IMAGE	n/a	n/a	*No EventDetails available*
GENERAL_NOTE	n/a	n/a	*No EventDetails available*
AIRWAY_NOTE	n/a	n/a	*No EventDetails available*
BREATHING_NOTE	n/a	n/a	*No EventDetails available*
CIRC_TQ_NOTE	n/a	n/a	*No EventDetails available*
CIRC_DRESSING_NOTE	n/a	n/a	*No EventDetails available*
OTHER_TREATMENT_NOTE	n/a	n/a	*No EventDetails available*
TC3_NOTE	n/a	n/a	*No EventDetails available*
ASSESSMENT_MARKER	n/a	n/a	*No EventDetails available*
INTERVENTION_MARKER	n/a	n/a	*No EventDetails available*
DRUG_MARKER	n/a	n/a	*No EventDetails available*

Event Type	Attribute	Type	Description
FLUID_MARKER	n/a	n/a	*No EventDetails available*

Definition of BodyMaps

Attribute	Type	Description
Name	String	Name of the injury, burn type or dressing. Note that the injury, burn and dressing types can be modified using the Tempus Pro Configuration Utility.
Code	String	Identification code of the body may entry, this value can be edited using the Tempus Pro Configuration Utility. Note that the code field will always be blank for the 'Other 1' and 'Other 2' fields.
Location	String	Comma separated list of locations recorded for the injury burn or dressing, see Appendix 7.3 for details. This field will be blank in the case that there is no injury, burn or dressing of the specific type.

Definition of TQMaps

Attribute	Type	Description
Name	String	The name of the TQ region will be set to one of the following: • R Arm (Right arm) • L Arm (Left arm) • R Leg (Right leg) • L Leg (Left leg)
Location	String	Comma separated list of locations recorded for the TQ, see Appendix 7.4. This field will be blank in the case that no TQ is recorded for the specific region/type.
Start	String	The recorded TQ start time for the applicable TQ region. This field will be blank in the case that no TQ is recorded for the specific region/type.
Type	String	The recorded TQ type can be one of the following: • Junctional • Extremity

Definition of ReviewDetails

Attribute	Type	Description
ReviewStatus	String	ECG review status. Can be one of the following: • No review requested • Review requested • Reviewed • Acknowledged
ReviewRequestSentTo	String	Name of the Support centre, where the ECG was sent for review
ReviewRequestSentTimeUTC	String	Time when the ECG was sent for review
ReviewedBy	String	Name of medic who reviewed ECG
ReviewTimeUtc	String	Time in UTC format when ECG was reviewed.
ReviewDeliveryTimeUtc	String	Time in UTC when review was delivered to Tempus Pro.
Diagnosis	String	Entered diagnosis, if applicable.
Instruction	String	Entered instruction, if applicable.
Notes	String	Entered notes, if applicable.
AckStatus	String	Acknowledgement status. Can be one of the following: • *Blank* • No instruction to acknowledge • Not acknowledged • Received and understood • Received and declined *Blank status shall be returned, if ECG review was not requested, or review was not yet performed.*
AckByTempusName	String	Tempus Pro name that has acknowledged ECG review, if applicable.
AckByTempusNumber	String	Tempus Pro number that has acknowledged ECG review, if applicable.
AckTimeUtc	String	Time in UTC when review was acknowledged, if applicable.
AckETA	String	ETA time in acknowledgement, if applicable.

4.5 LiveEvents

A request for the LiveEvents resource returns a list of events updated within 2 minutes before the specified time.

Use one of the persistent IncidentIds returned from the Incidents resource as the IncidentId in the Events Resource URI when making this request.

No waveform images are included in the response. The data in the returned list includes EventIDs for each event that can then be used in the Resource URIs for separate requests to retrieve individual events with files embedded in each response.

Note	If event has been updated, the original entry will be marked as “Superseded”. Subsequent events shall have “Current” or “Deleted” status. “Deleted” status means that the event has been deleted from the SRoC on Tempus. “Current” status means that this is the latest version of the event on the Tempus.

4.5.1 Resource URI

GET /Api/Clinical/Incidents/{IncidentId}/LiveEvents

Request example:

```
https://corsium.info/WebAPI/sdk/Api/Clinical/Incidents/FA68370A-BF24-4BD1-99BE-DF946C8ECDDE/LiveEvents?toDate=2016-03-25T11:15:00
```

4.5.2 Parameters

Name	Type	Req/Opt	Description
toDate	[DateTime]	Req	Return events that were updated within 2 minutes before this date, inclusive. Format: yyyy-MM-ddTHH:mm:ss

4.5.3 Headers

Name	Value	Details
Accept	application/json	-
Authorization	bearer <Access token>	<Access token> is obtained from APILogin response

4.5.4 Response data

```
[
       “IncidentId”: “06608acb-0f57-4908-9a08-43041b6afed8”,
       “EventsInfo”: [
          {
              “EventId”: “8de9a652-46f8-42e9-aae8-7bb487297147”,
              “DataCaptureTimeUtc”: “2016-03-25T11:15:58”,
              “EventTimeUtc”: “2016-03-25T11:15:58”,
              “HasVitals”: true,
              “EventType”: ”DRUG”,
              “EventLabel”: “Adenosine”,
              “EventDetails”: [{
                     “Code”: “”,
                     “Dose”: “as per protocol”,
                     “Route”: “IV”
                     }],
```

```
            "EventNotes": "Patient responded well",
            "EventDataStatus": "Not required",
            "EventStatus":"New",
            "Vitals": [
                {
                    "Name": "HR",
                    "Value": "80",
                    "Measurement": "bpm",
                    "Code": "8867-4",
                    "Source": "ECG",
                    "Alert": false
                },
                . . .
                {
                    "Name": "SysPAP",
                    "Value": "120",
                    "Measurement": "mmHg",
                    "Code": "8867-4",
                    "Source": "P1:PAP",
                    "Alert": false
                }],
        }]
]
```


Note	Incident events are returned in the chronological order.
Note	If there was loss in connectivity, or the incident was monitored for some time without being connected to Corsium, then the returned events list will include the all events captured in that period.
Note	Returned events include new, modified, deleted and superseded events. It is the ePCR providers responsibility to work out the current events list. RDT recommends to pull the incident as historic, once it is no longer live.

The following table provides details of the fields present in the JSON-formatted data response:

Attribute	Type	Description
IncidentId	String	Unique identifier for an incident.
EventsInfo	Array of EventsInfo objects	Please refer to the section below.

Definition of EventsInfo

Attribute	Type	Description
DataCaptureTimeUtc	String	Time when event was first recorded or edited or deleted in UTC format

Attribute	Type	Description
EventTimeUtc	String	Time of the event in UTC format. This may be different to the DataCaptureTimeUtc, as event time can be edited on Tempus. The event time is the time that is displayed in Summary tab on the Tempus/Corsium. For Tempus LS events only: the event time shall be returned with milliseconds, i.e. 2016-03-25T11:15:58.056
EventId	String	Unique ID of the event.
HasVitals	Boolean	Set to true, when event has associated vitals.
EventType	String	Type of the recorded event. See table below for possible event types.
EventLabel	String	Name of the recorded event. For Drug, Fluid, Assessment and Intervention events the name can be edited using Tempus Pro Configuration Utility. See the table below for event label details.
EventDetails	Array of EventDetails	Data as defined in Section 4.4 for different Event types
EventNotes	String	Notes entered during the encounter for the specific data item, this field can be blank. For General note event this field will contain the entered note.
EventDataStatus	String	Status of the event data. Possible values: • Not ready – the event has come but the file is not available • Ready – both event and associated file data have been received • Not available - file data is not available, even though it is a file event • Not required – file data is not required
EventStatus	String	Indicated whether the event is created new, modified, current or deleted. • Deleted • Current • Superseded • New
Vitals	Array of Vitals	See Section 4.1 for Vitals details.

4.6 TwelveLead

Retrieving 12 lead ECG images from an Incident is a two-step process.

The first step is to request a list of all events for a specific Incident (see “4.4 Events” or “4.5 LiveEvents”). This list will include an EventId for each 12 lead ECG.

The second step is to use the EventId's returned in the events response to request a specific individual 12 lead ECG resource with the image data embedded in the response or, optionally, the raw XML file embedded in the response.

This allows a caller to reduce the impact of retrieving large amounts of raw image data in a single request.

When issuing the request for the TwelveLead resource, use one of the persistent IncidentIds returned from the Incidents resource in the TwelveLeads Resource URI.

Note	ECG in JPEG format shall have date/time details in Corsium configured time zone, not UTC.

4.6.1 Resource URI

GET /Api/Clinical/Incidents/{IncidentId}/Events/TwelveLead/{EventId}

Request example:

```
https://corsium.info/WebAPI/sdk/Api/Clinical/Incidents/77640879-819D-4212-9B7F-
5567803CEC15/Events/TwelveLead/506F0767-D508-457D-95DB-32F695F3D43E?Format=JPEG
```

4.6.2 Parameters

Name	Type	Req/Opt	Description
Format	String	Opt	Provides 12 lead data in specified format. The default format is JPEG. Possible values are : • JPEG • aECG • DICOM In the case of JPEG, the PHI shall not be included if data is not encrypted or transport provider is configured not to share PHI.

Note	The Annotated ECG (aECG) HL7 is a standard medical record data format for storing and retrieving electrocardiogram data for a patient. DICOM® (Digital Imaging and Communications in Medicine) is the international standard to transmit, store, retrieve, print, process, and display medical imaging information. Please contact RDT for the copy of the standards.
Note	In TwelveLead API response, lead specific detailed measurements are not available in any of the ECG formats i.e. JPEG, DICOM and aECG. However, they are available in Incident PDF data from "PDF" API, if "Detailed Measurements (PDF)" option is selected in ECG display settings by Corsium transport account admin.

4.6.3 Headers

Name	Value	Details
Accept	application/json	-
Authorization	bearer <Access token>	<Access token> is obtained from APILogin response

4.6.4 Response data

```
[
        {
                "IncidentId": "06608acb-0f57-4908-9a08-43041b6afed8",
                "EventId": "8de9a652-46f8-42e9-aae8-7bb487297147",
                "Format": "JPEG",
                "TwelveLeadData": "GG/YU6TU/WHjqygweu ... jdtY/Sx76320y7nmDuPyuw"
        }
]
```

The following table provides details of the fields present in the JSON-formatted data response:

Attribute	Type	Description
IncidentId	String	Unique identifier for an incident.
EventId	String	Unique ID of the event.
Format	String	Format in which the data is sent. Possible values: • JPEG • aECG • DICOM
TwelveLeadData	String	ECG data in selected format with base64 encoding.

4.7 Snapshot

Retrieving snapshots from an incident is a two-step process.

The first step is to request a list of all events for a specific incident (see "4.4 Events" or "4.5 LiveEvents"). This list will include an EventId for each event with associated snapshot.

The second step is to use one of EventIds returned in the events response to request a specific individual snapshot resource with the image data embedded in the response.

This allows a caller to reduce the impact of retrieving large amounts of raw image data in a single request.

When issuing the request for the snapshot resource, use one of the persistent IncidentIds returned from the incidents resource in the Snapshot Resource URI.

Snapshot data is returned in JPEG format, PHI shall not be included if data is not encrypted or transport provider is configured not to share PHI.

Note	Snapshot shall have date/time details in Corsium configured time zone, not UTC.

4.7.1 Resource URI

GET /Api/Clinical/Incidents/{IncidentId}/Events/Snapshot/{EventId}

Request example:

```
https://corsium.info/WebAPI/sdk/Api/Clinical/Incidents/77640879-819D-4212-9B7F-
5567803CEC15/Events/Snapshot/1B4D8868-68CB-492F-95D3-B9A4EDDB9B7F
```

4.7.2 Parameters

None

4.7.3 Headers

Name	Value	Details
Accept	application/json	-
Authorization	bearer <Access token>	<Access token> is obtained from APILogin response

4.7.4 Response data

```
[
        {
                "IncidentId": "06608acb-0f57-4908-9a08-43041b6afed8",
                "EventId": "8de9a652-46f8-42e9-aae8-7bb487297147",
                "SnapshotData": "GG/YU6TU/WHjqygweu ... jdtY/Sx76320y7nmDuPyuw"
        }
]
```

The following table provides details of the fields present in the JSON-formatted data response:

Attribute	Type	Description
IncidentId	String	Unique identifier for an incident.
EventId	String	Unique ID of the event.
SnapshotData	String	Snapshot data is in a JPEG format and base64 encoding

4.8 Image

Retrieving images from an incident is a two-step process.

The first step is to request a list of all events for a specific incident (see “4.4 Events” or “4.5 LiveEvents”). This list will include an EventId for each event with associated image.

The second step is to use one of EventIds returned in the events response to request a specific individual Image resource with the image data embedded in the response.

This allows a caller to reduce the impact of retrieving large amounts of raw image data in a single request.

When issuing the request for the Image resource, use one of the persistent IncidentIds returned from the incidents resource in the Image Resource URI.

Images are returned in JPEG format.

Note	If PHI are removed from the incident, API call for Images shall return the error code, see Appendix 7.1.

4.8.1 Resource URI

GET /Api/Clinical/Incidents/{IncidentId}/Events/Image/{EventId}

Request example:

```
https://corsium.info/WebAPI/sdk/Api/Clinical/Incidents/77640879-819D-4212-9B7F-
5567803CEC15/Events/Image/5918A3E5-292B-4531-B849-864914ED3BD2
```

4.8.2 Parameters

None

4.8.3 Headers

Name	Value	Details
Accept	application/json	-
Authorization	bearer <Access token>	<Access token> is obtained from APILogin response

4.8.4 Response data

```
[
        {
                "IncidentId": "06608acb-0f57-4908-9a08-43041b6afed8",
                "EventId": "8de9a652-46f8-42e9-aae8-7bb487297147",
                "ImageData": "GG/YU6TU/WHjqygweu ... jdtY/Sx76320y7nmDuPyuw"
        }
]
```

The following table provides details of the fields present in the JSON-formatted data response:

Attribute	Type	Description
IncidentId	String	Unique identifier for an incident.
EventId	String	Unique ID of the event.
ImageData	String	Image data in JPEG format with base64 encoding

4.9 PDF

A request for an Incident PDF will return detailed PDF report for the selected incident. If transport provider is configured not to share PHI, then returned report shall not contain the following patient identifiers: patient name, patient ID, medics details or images.

PDF shall be constrained by the following limits:

- Last 72 hours of vitals data
- Latest 10 ECGs
- Latest 10 images

- Latest 20 snapshots

Note	PDF can be requested only for historic incidents. If PDF is requested for the incident that is currently live, the error code is returned, see “7.1 Response error messages” for details.
Note	PDF shall contain the following sections, if applicable: • Patient Details • Vitals Summary • Events • Medics • ECGs • Waveforms • Trend Graph • Images • Body Maps • Trend Table
Note	All data in the PDF shall be in Corsium configured time zone, not UTC.

4.9.1 Resource URI

GET /Api/Clinical/Incidents/{IncidentId}/PDF

Request example:

```
https://corsium.info/WebAPI/sdk/Api/Clinical/Incidents/FA68370A-BF24-4BD1-99BE-DF946C8ECDDE/PDF
```

4.9.2 Parameters

None

4.9.3 Headers

Name	Value	Details
Accept	application/json	-
Authorization	bearer <Access token>	<Access token> is obtained from APILogin response

4.9.4 Response data

```
[
	{
		“IncidentId”: “06608acb-0f57-4908-9a08-43041b6afed8”,
		“Name”: “Tempus Report 2015-02-26 14-17-34 SN-00600012.pdf”,
		“Data”: “GG/YU6TU/WHjqygweu ... jdtY/Sx76320y7nmDuPyuw”
```

```
        }
]
```

The following table provides details of the fields present in the JSON-formatted data response:

Attribute	Type	Description
IncidentId	String	Unique identifier for an incident.
Name	String	Name of PDF report. Filename of the report shall be following: Tempus Report <creation date> <creation time> SN-<device serial number>.pdf Date format: YYYY-MM-DD Time format: HH-MM-SS *Example:* Tempus Report 2015-02-26 14-17-34 SN-00600012.pdf
Data	String	Base64 encoded PDF binary data

5 Test Reference

RDT provides simple reference application for WebAPI testing and validation. This test reference application can be configured to pull data with any instance of IntelliSpace Corsium by specifying the URL and credentials for an ePCR user sent via email at the time of ePCR user creation.

The test reference application comes with source code written in C# language and using MVC.net framework. This can be used as a reference to understand the mechanism of making requests to the WebAPI server and pull patient data from Corsium.

Warning

The Test Reference Application relies on IISExpress, which is a webserver. Intended use is for installation on a developer's desktop machine ONLY. Do NOT install the Test Reference Application on a public internet facing server, or on any computer that is in a publicly accessible network. Installing the Test Reference Application on publicly accessible computer systems is a data security risk.

Warning

You must seek approval from your information security officer before installing the Test Reference Application. Explain the data flow security and data privacy mechanisms provided by IntelliSpace Corsium so that any additional or compensating security controls may be implemented in your own network according to your organisational policies. Refer to IFU 42-4002EN IntelliSpace Corsium Security Summary.

Compensating security controls typically include application whitelisting, device access control, domain whitelisting, intrusion detection and prevention systems, antimalware tools and hard disk encryption of data at rest.

5.1 Test Reference Installation

The test application is a web application which can be run using an embedded web server. The test reference application distribution folder contains files as shown below.

Follow the steps below to install and run the test reference from your local PC:

1. Install IISExpress 10 using the installer file (iisexpress_amd64_en-US.msi)
2. Copy SDKWebAPI folder on local machine
3. Navigate to SDKWebAPI folder
4. Run the StartTestReference.bat file to start the Test reference application

Note	Ensure that user has read and write permission on the folder containing the Test Reference application.

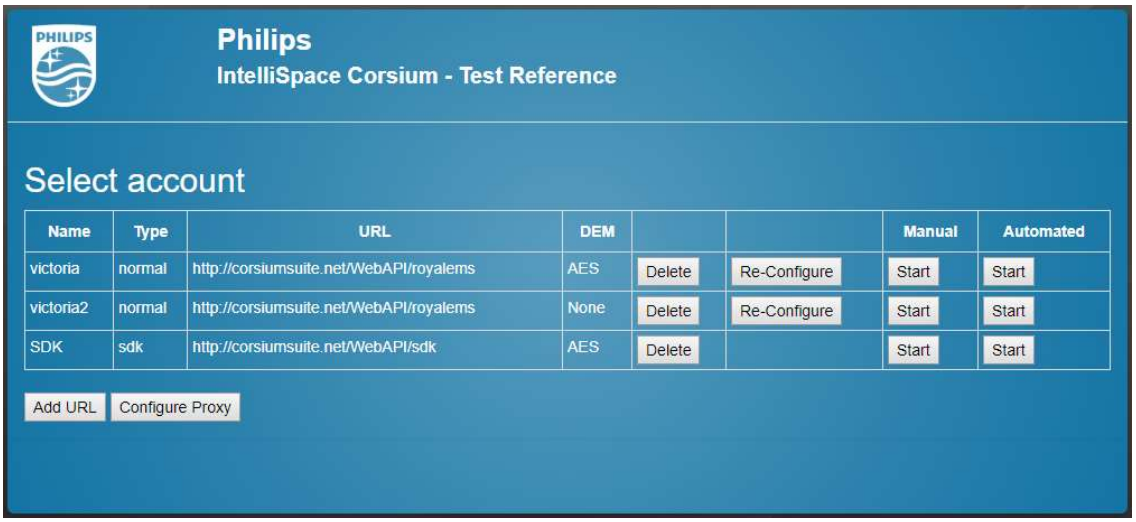
Note	Test Reference application should be stored in the location in such way that there are no spaces in the path, e.g. “C:\Users\DefaultUser\Desktop\SDKWebAPI”

5.2 Test Reference Configuration

5.2.1 Corsium URL List

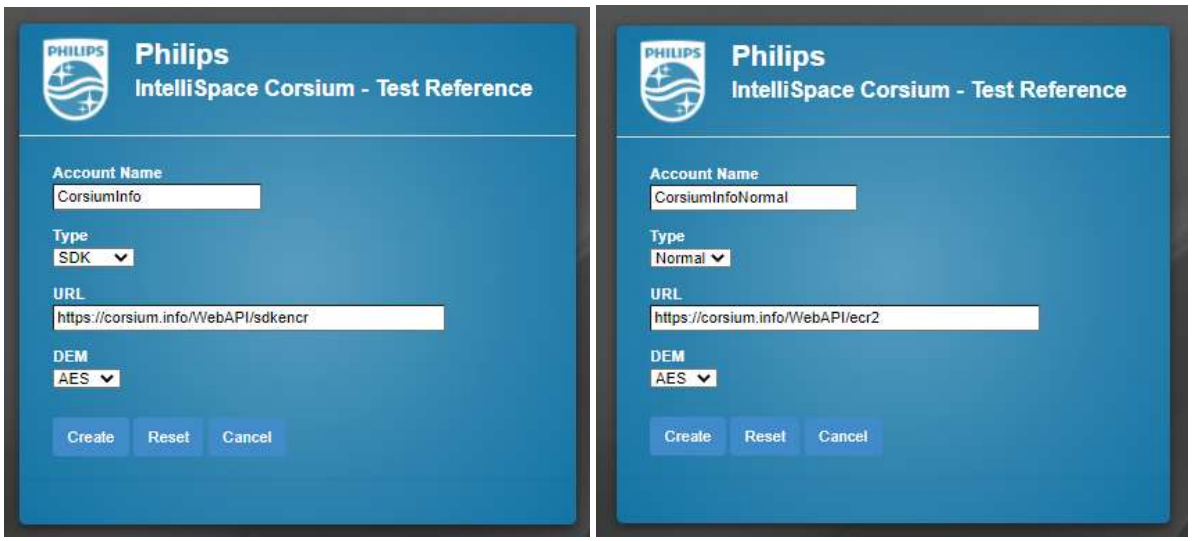
Starting Test reference application displays the main page of the application which contains list of Corsium URLs configured for WebAPI access.

A typical screen looks like as follows:



5.2.2 Add URL

A new WebAPI access URL can be added to the test reference application by clicking the “Add URL” button, which brings up the following screen:



Following table describes the fields and their meaning

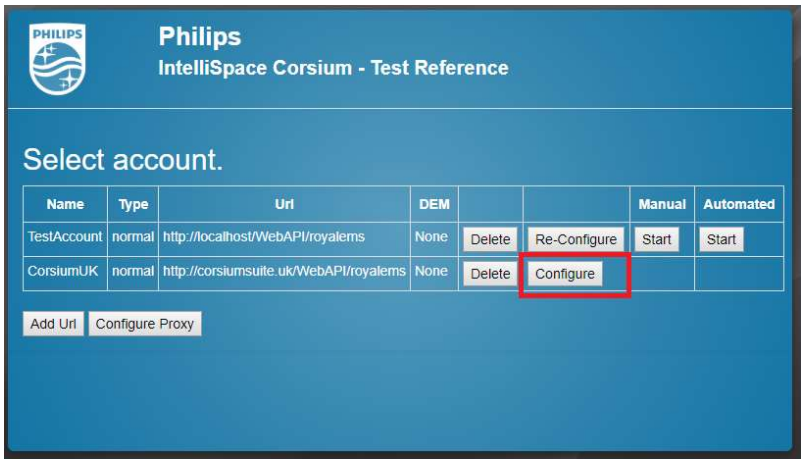
Field	Description
Account Name	Unique identifier for the Corsium WebAPI connection. This can be any meaningful name without spaces.
Type	There are two types of account. • SDK – SDK account contains pre-configured demo patient data and doesn't require user credentials to access the data. This can be used to view test data in order to facilitate an understanding of the format of the WebAPI data. • Normal – Normal accounts are actual transport accounts configured on Corsium. Only specific ePCR users configured for these customer accounts will be able to view patient data from these WebAPI Urls.
URL	WebAPI URL for a transport account. This will be typically of the format: *https://<corsiumsite>/WebAPI/<account postfix>* Normal account: URL is included in the email received by the ePCR provider, when Transport account administrator configured the access to the data. SDK account (Without Encryption): https://corsium.info/WebAPI/sdk SDK account (With Encryption): https://corsium.info/WebAPI/sdkencr
DEM	There are three types of DEMs (Data Encryption Method) • None • AES • PKI This has already been described in the API access details, see "2.1 API Login". Normal account: encryption details are sent via email. If double encryption is disabled, then DEM setting is "None". If double encryption is enabled, then DEM setting is "AES" or "PKI". SDK account (Without Encryption, https://corsium.info/WebAPI/sdk): None SDK account (With Encryption, https://corsium.info/WebAPI/sdkencr): AES
Public Key	This field is displayed if PKI has been selected as DEM method, to specify the public key for encryption.
Private Key	This field is displayed if PKI has been selected as DEM method, to specify the private key for encryption.

5.2.3 Configure account

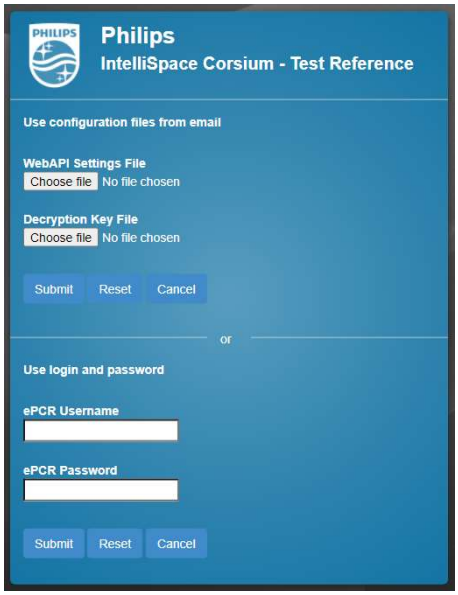
Once a new URL is added successfully, the user is redirected back to the main page.

Note	If account type is SDK there is no need to configure access credentials, the hard-coded values will be used.

For the Normal account a "Configure" button is displayed for the newly-added account.



Click on the Configure button to specify user credentials for WebAPI access.



Following table describes the fields and their meaning:

Field	Description
WebAPI Settings File	If ePCR Credentials are delivered via email, then two emails will be sent to the configured address when an ePCR user is created. The attached WebAPI settings file should be saved locally and then imported using the “Choose File” button.
Decryption Key File	If ePCR Credentials are delivered via email, then two emails will be sent to the configured address when an ePCR user is created. The attached Decryption Key file should be saved locally and then imported using the “Choose File” button.
OR	
ePCR Username	If ePCR Credentials are configured directly in the Corsium Administration console, enter the ePCR username in this field.
ePCR Password	If ePCR Credentials are configured directly in the Corsium Administration console, enter the ePCR password in this field.

Once the normal account is configured, the buttons for manual and automated tests are displayed against the configured account.

For “SDK” type of account the “Start” buttons appear immediately after account configuration.

5.2.4 Re-Configure Account

If credentials are changed during the course of testing, the user details can be re-configured using the “Re-configure” button which appears for the accounts which are already configured.

5.2.5 Delete URL

If an URL is no longer required, it can be deleted using the “Delete” button against the URL.

5.2.6 Configure proxy

If users are behind a firewall, they can configure a proxy to access WebAPI, which is used within their organization. If there is no proxy this URL will be blank.

5.2.7 Start Manual Test

Manual testing can be started using the “Start” button in the Manual column of the configured URL. Manual test uses credentials for the configured URL and logs user in to display the patient data. All features of Manual testing have been described in section 5.4 of this document.

5.2.8 Start Automated Test

Automated testing allows user to observe the automatic data pull from Corsium. The user can check the calls made and the responses received.

5.3 Demo incidents

SDK test account contains number of demo incidents available for data pull, see Section 6 for the details of incidents. Each demo incident will be re-created every 24 hours.

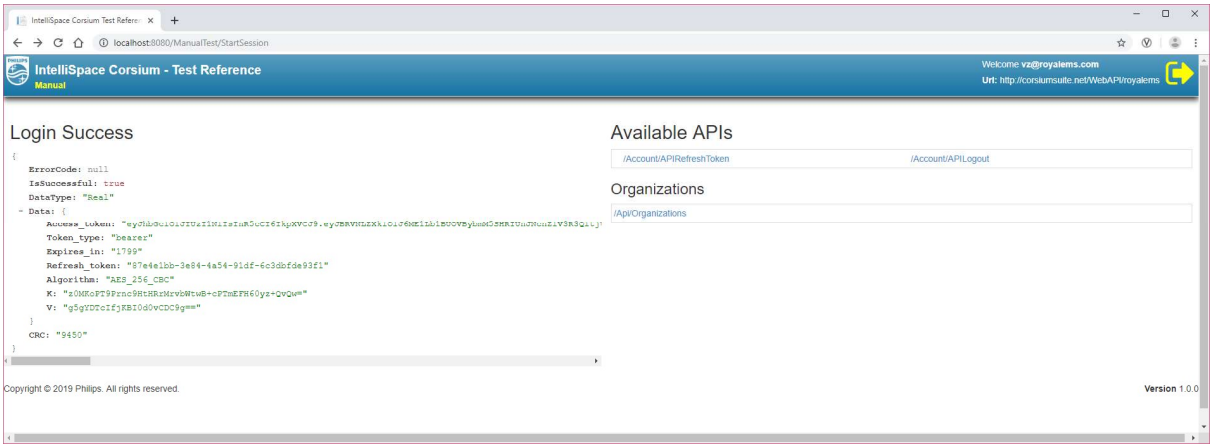
When incidents are re-generated new Incident IDs are assigned.

5.4 Manual Testing

The following screenshots provide details of the Test application which can be used to validate APIs on pre-configured or actual patient data. The sample screens are for demonstration purposes only and do not contain real data.

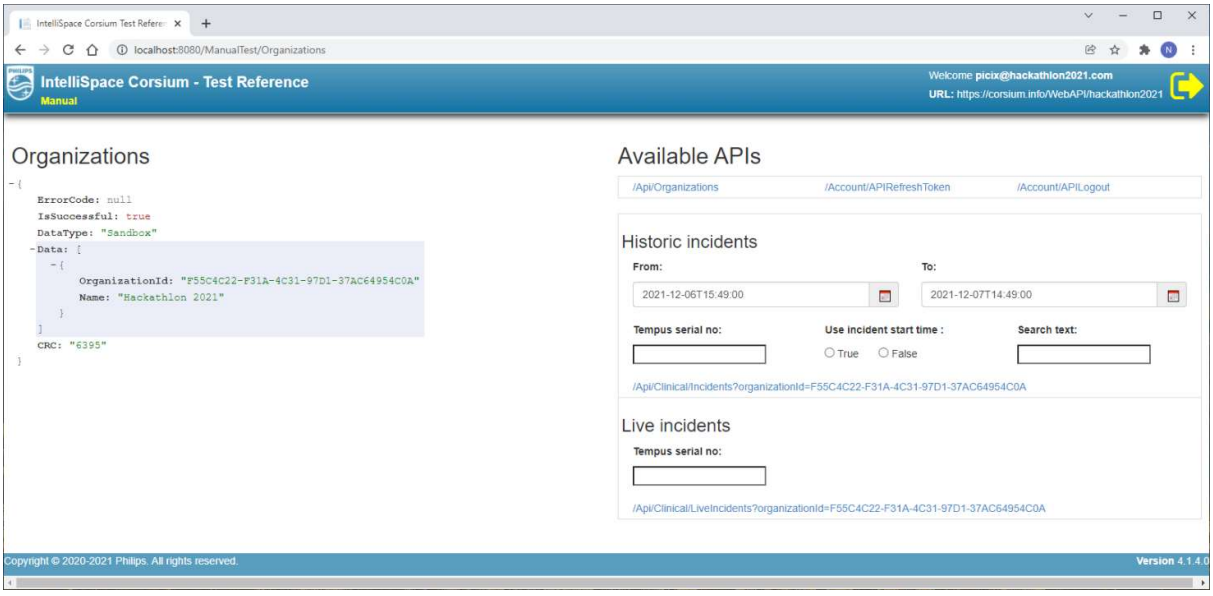
5.4.1 Successful Login

Clicking “Start” button for manual testing validates user credentials and if authentication is successful, user is redirected to test reference dashboard where JSON response from the API is shown on the left side of the screen and further available APIs on the right.



5.4.2 Organizations

Displays a list of organizations assigned to this ePCR user. There will always be one organization in the response.



5.4.3 Incidents

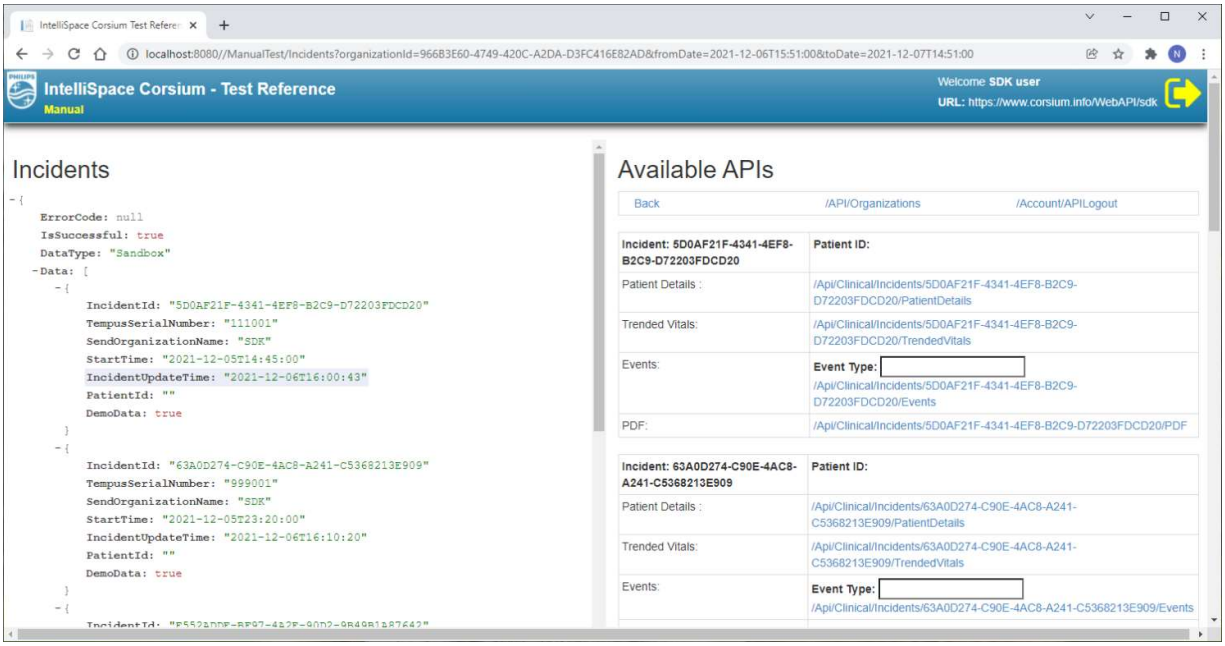
This screen shows list of historical incidents available for the organization.

Users need to specify From and To date to get incident data within specified period for historic incidents (from the Organizations page). Time period specified cannot be longer than 24 hours. Optionally, incidents can be searched by the Tempus serial number, in which case the maximum dare range is 6 months.

Optionally, the incident list can be returned based on the Incident Start time, not by the Incident Update Time.

Each historic incident has the following APIs available:

- Patient Details
- Trended Vitals
- Events
- PDF report

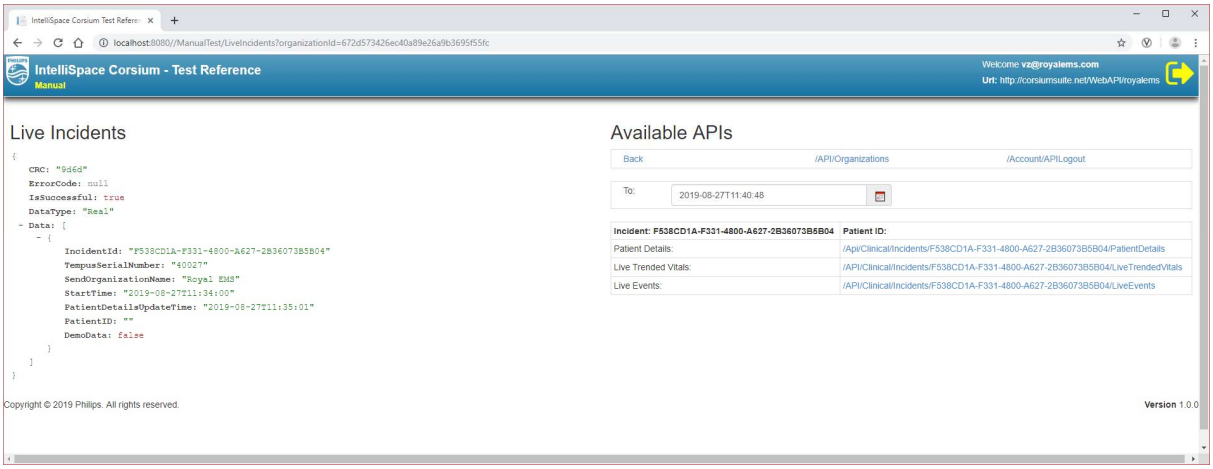


5.4.4 Live Incidents

This screen shows list of currently live incidents available for the organization. All currently Live incidents are returned in the response.

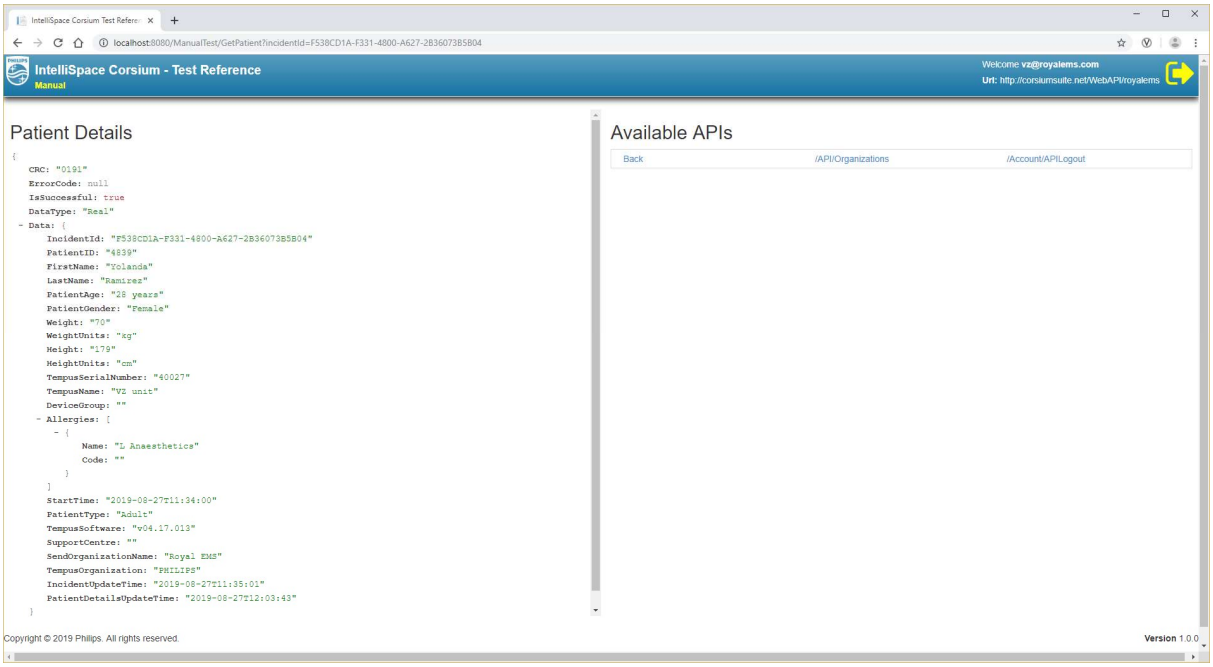
Each Live incident has the following APIs available:

- Patient Details
- Live Trended Vitals (“To” time is required)
- Live Events (“To” time is required)



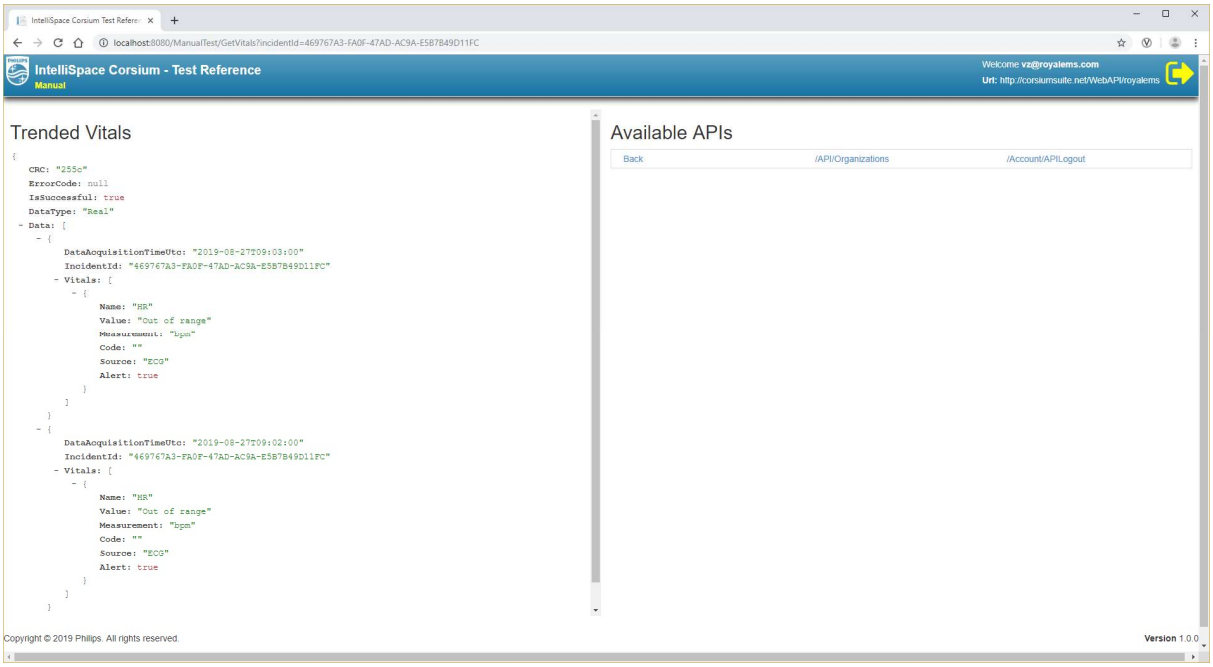
5.4.5 Patient Details

This screen displays patient details for an incident (both historic or live):



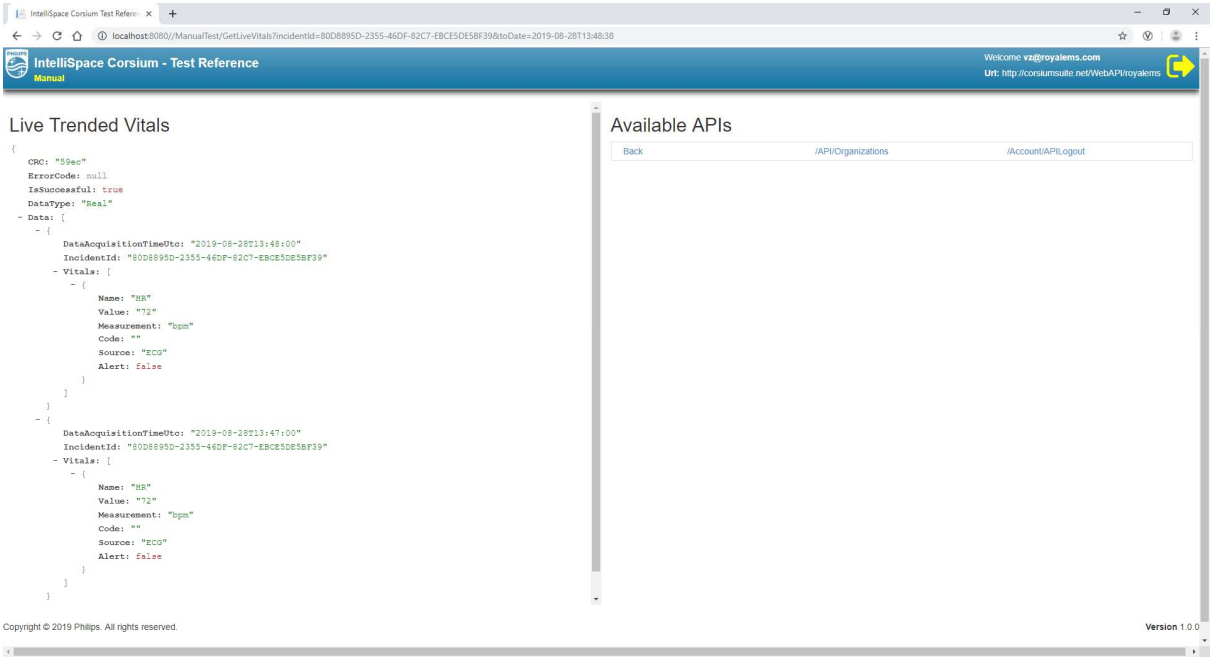
5.4.6 Trended Vitals

This screen displays complete list of trended vitals available for an incident:



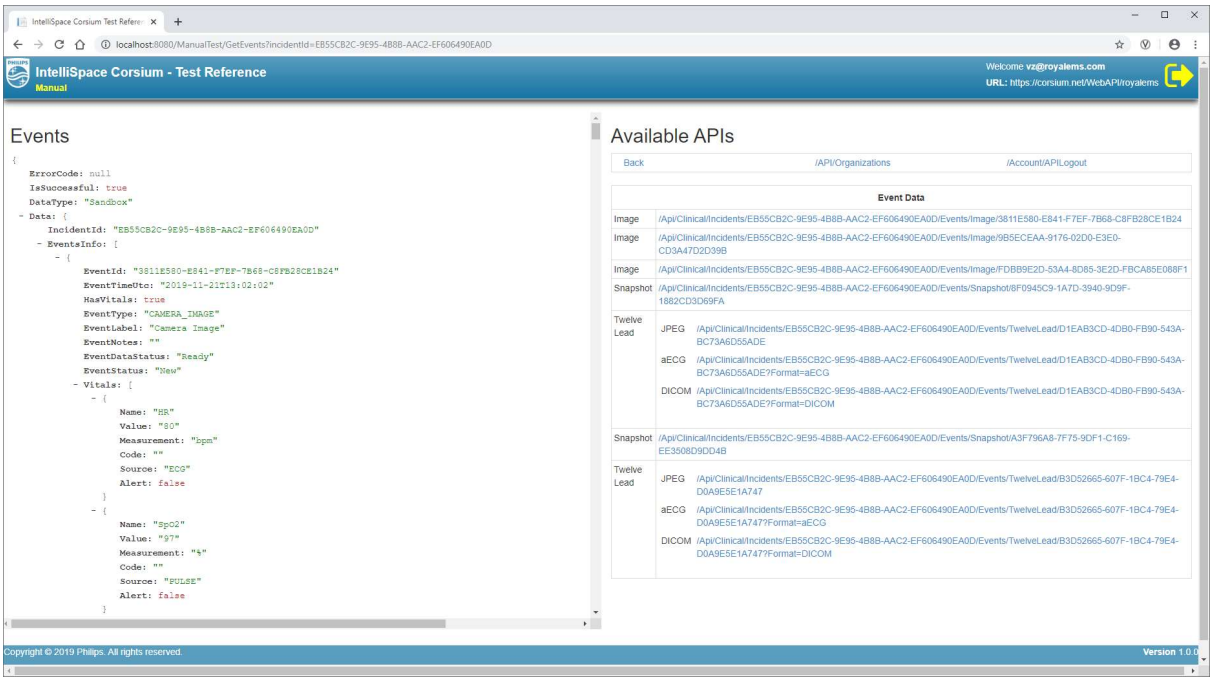
5.4.7 Live Trended Vitals

This screen displays 2 minutes of trended vitals available for a live incident. The “To” time must be specified in the Live Incidents screen. In the Test reference application by default the “To” time is populated with the current time. Live Trended vitals up to 2 minutes before the specified ”To” time are fetched and displayed.



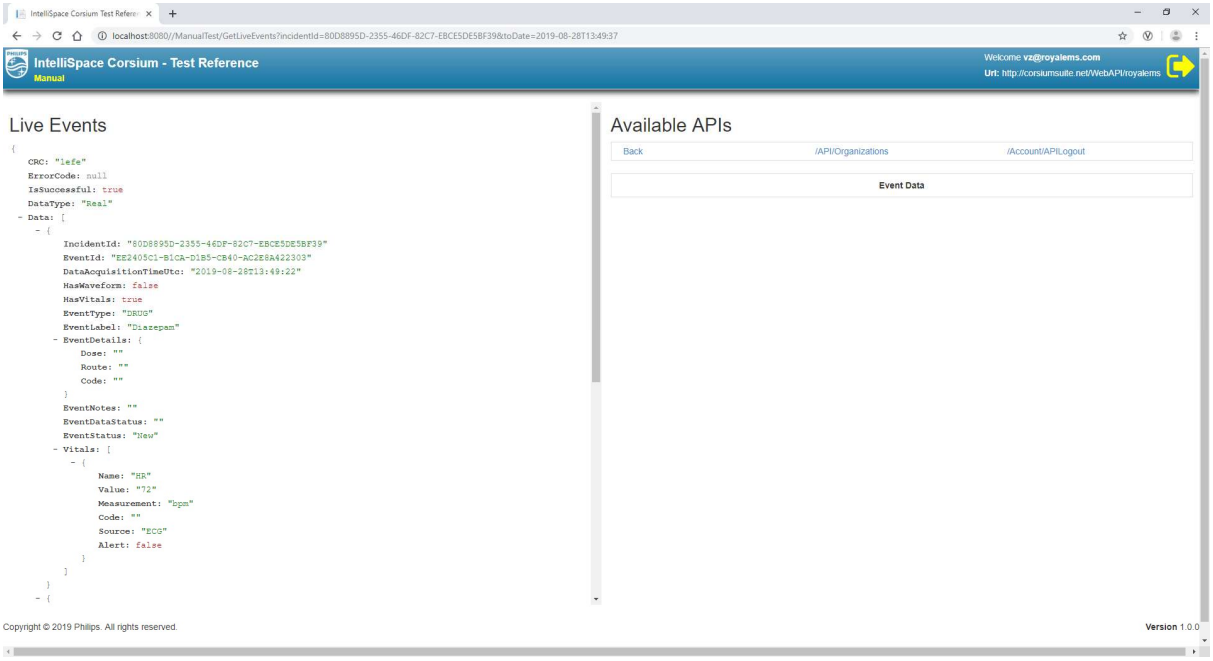
5.4.8 Events

This screen displays complete list of events (SRoC) available for a historic incident. If there are events with associated file data, then relevant APIs will be displayed in the right pane:



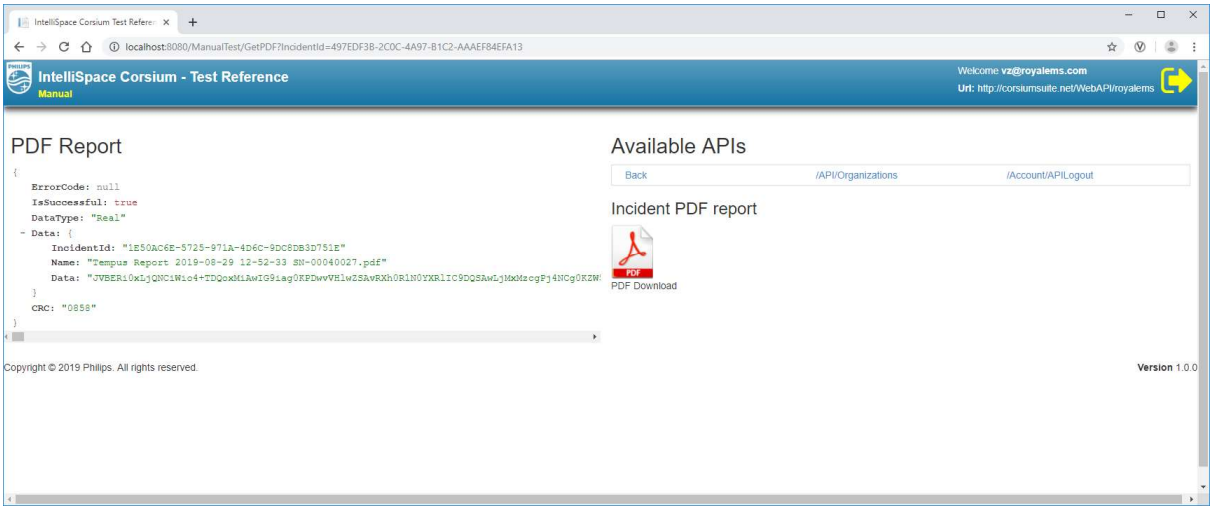
5.4.9 Live Events

This screen displays events recorded within 2 minutes of the “To” time. “To” time must be specified in the Live Incidents screen. In the Test reference application by default the “To” time is populated with the current time. SROC events captured up to 2 minutes before the specified ”To” time are fetched and displayed.



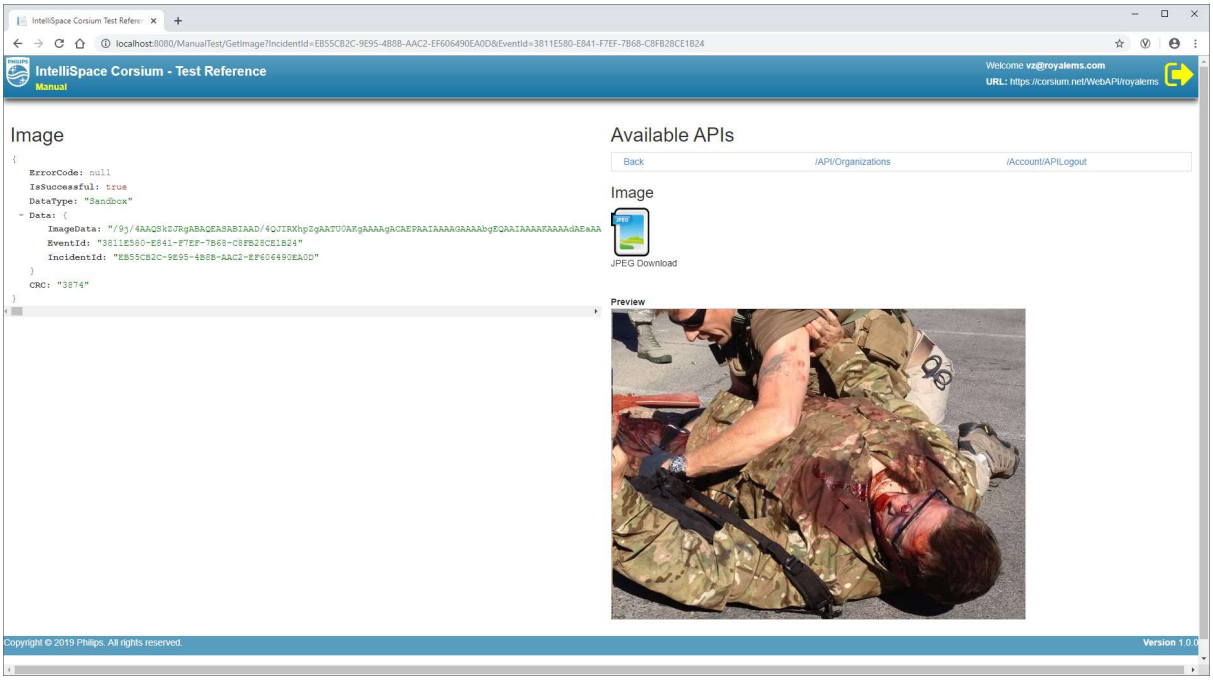
5.4.10 PDF

This screen allows to download the incident PDF. Pressing on the PDF icon will trigger download.



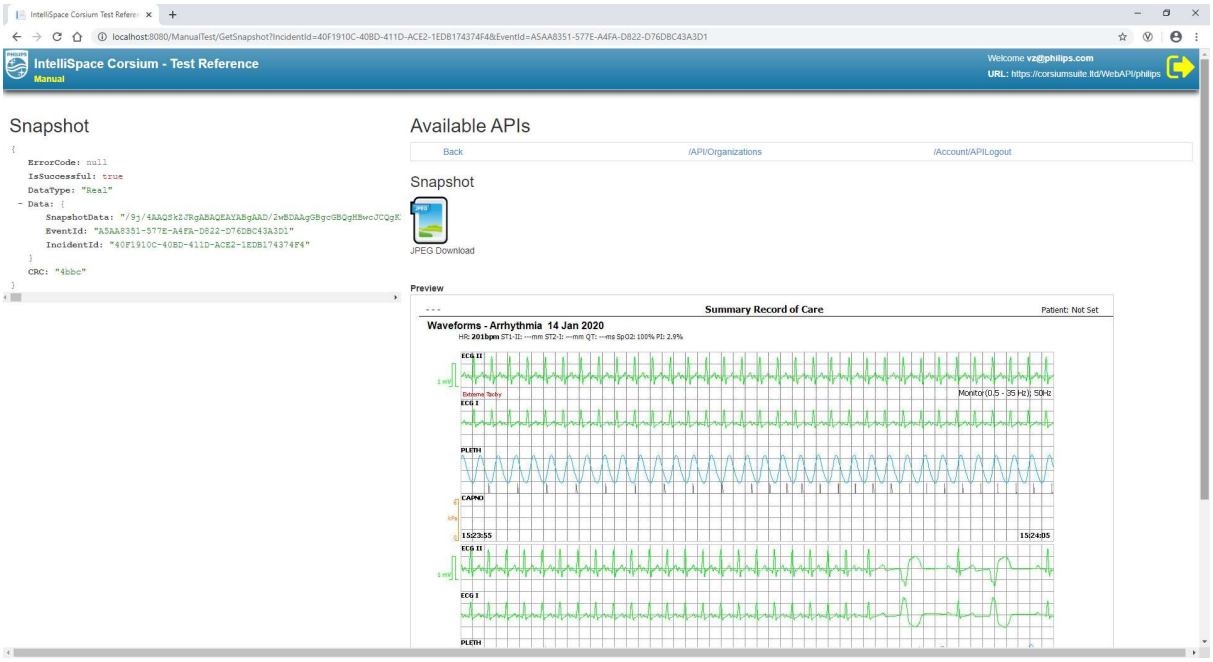
5.4.11 Images

This screen allows view the captured image in the right pane.



5.4.12 Snapshots

This screen allows to download Snapshot in JPEG format. Pressing on the JPEG icon will trigger download. The preview of the image is also available.

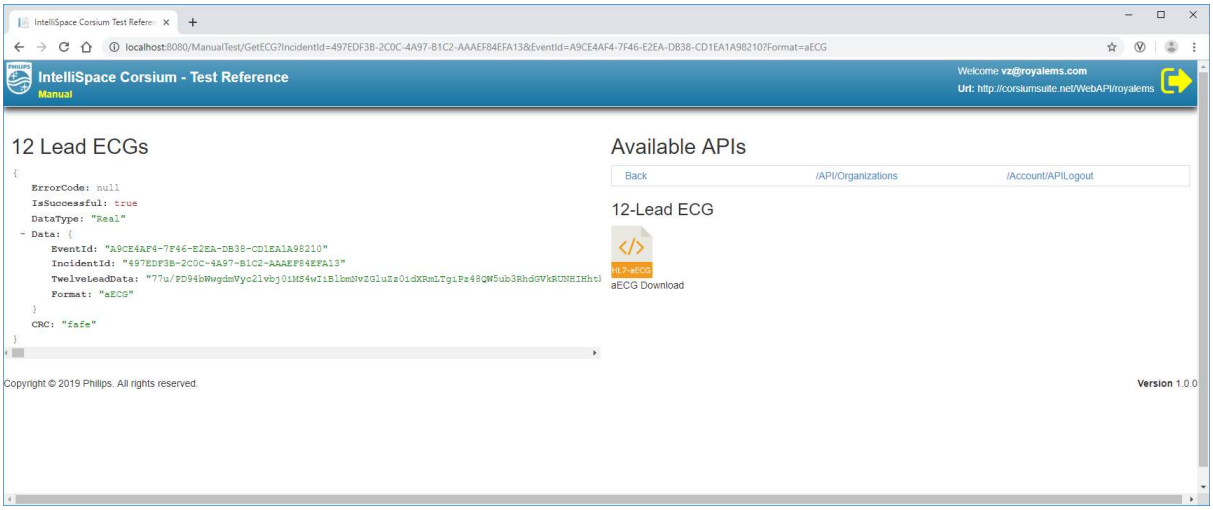


5.4.13 ECGs

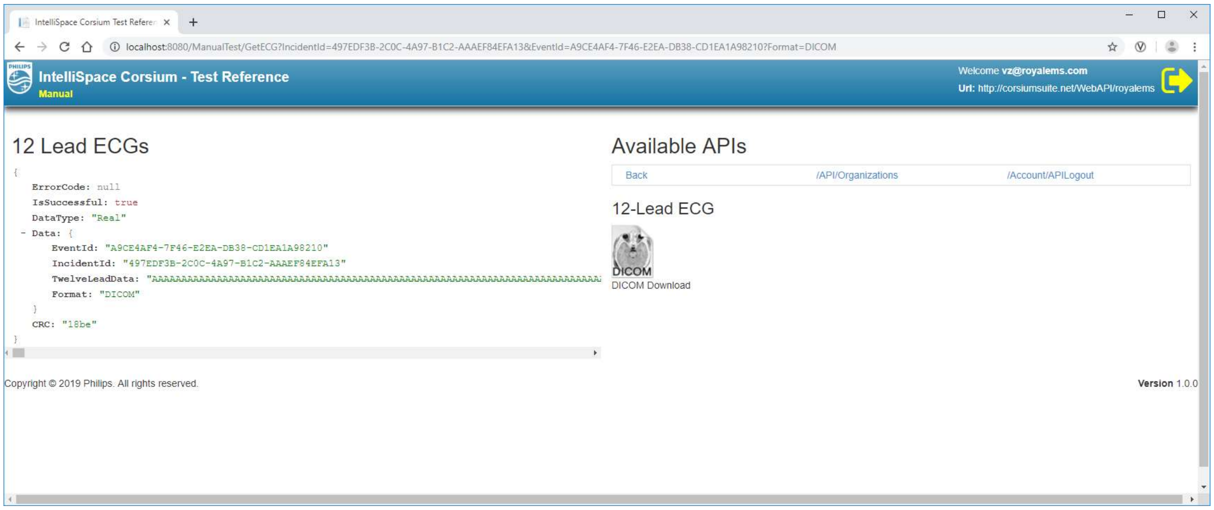
This screen allows to download ECG file in the requested format. Pressing on the icon will trigger download.

ECG in JPEG format:

ECG in aECG format:



ECG in DICOM format:

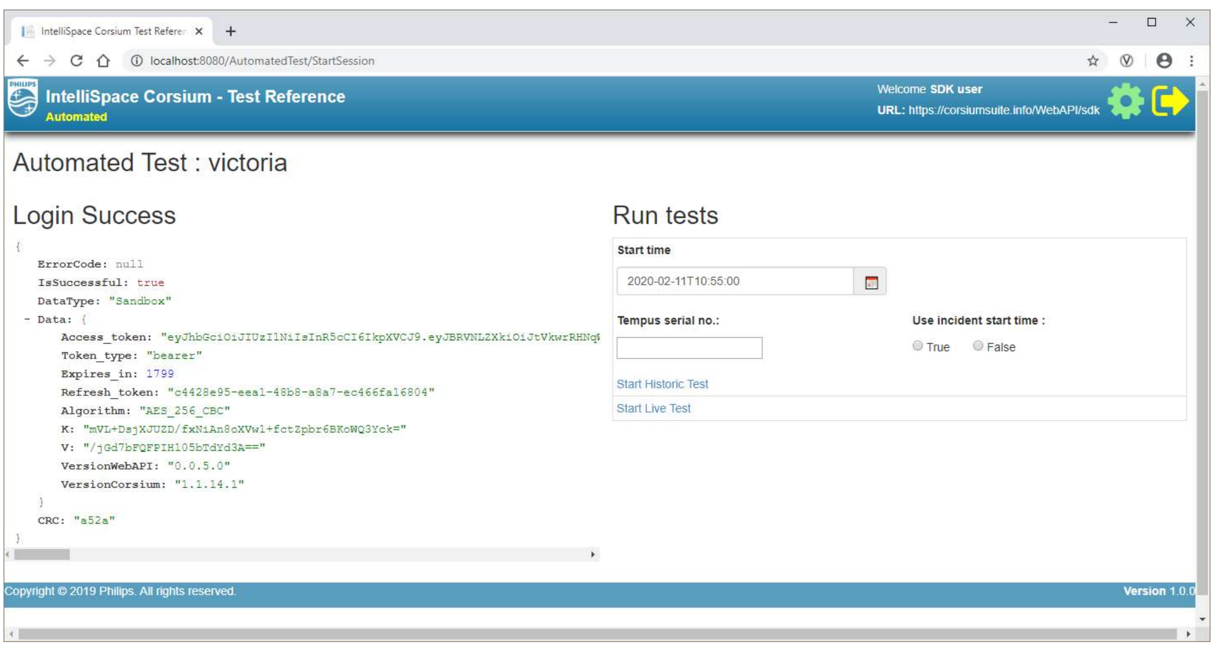


5.5 Automated Testing

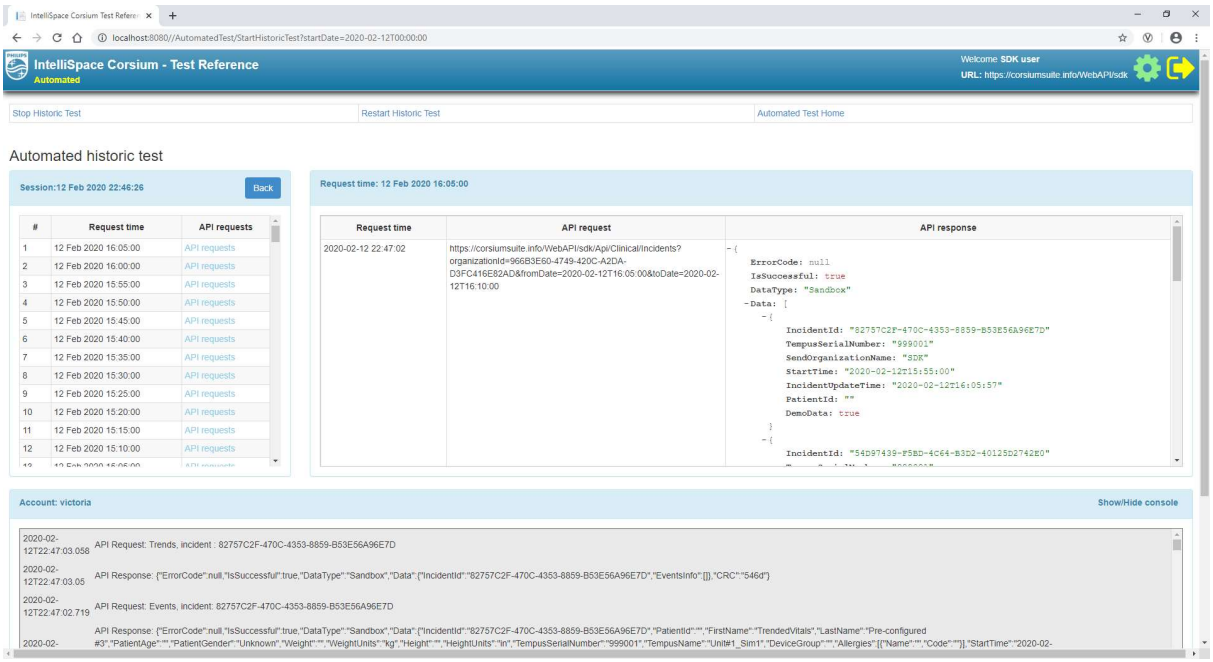
Following sections provide details of the automated testing available in Test Reference application which can be used as a reference for API calls and the responses. The sample screens are for demonstration purposes only and do not contain real data.

5.5.1 Automated Testing home page

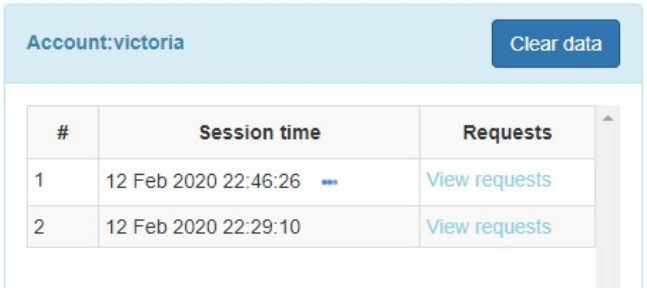
Clicking “Start” button for automated testing validates user credentials and if authentication is successful, user is redirected to test reference dashboard where JSON response from the API is shown on the left side of the screen and further available tests on the right.



5.5.2 API request and response console



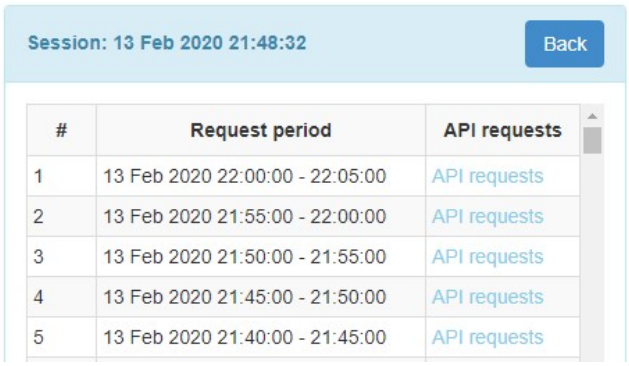
Once the automated test started, the user console is displayed. On the left side the list of all sessions run. The session in progress displayed with moving symbol:



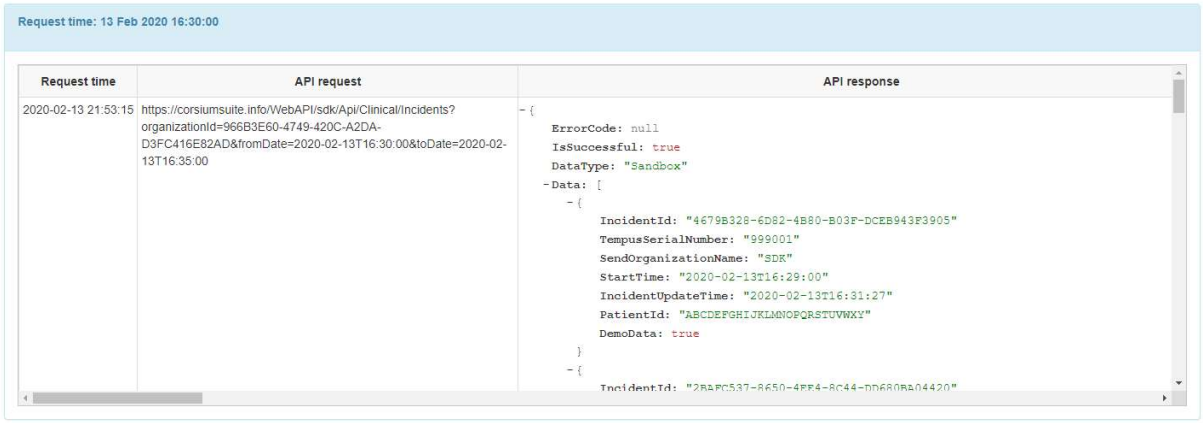
Pressing on the “View requests” option displayed API calls made within that session. Pressing on the “Clear data” buttons stops the automated test (if in progress), and clears all session logs.

5.5.3 Viewing requests

API calls for the session are grouped by data request periods:



Selecting “API requests” option displays full list of API calls made within that period:



The user can see time the response was made, the exact API request and the API response.

Note that the file data string is truncated for the display purposes.

5.5.4 Historic Data Pull

Historic data pull is configured to pull the data at 5 minute intervals starting from the configured date/time. Once the historic data is caught up, the incident list is pulled every 5 minutes. If new incident data is available, then all incident data is fetched.

Historic data pull can be optionally filtered by the Tempus number, or the incidents can be returned by the Start date.

5.5.5 Live Data Pull

Live data pull is configured to pull the data at 5 minute intervals. Once the new incident is connected, the data is caught up using the historic APIs for Events and Vitals, then live event and vitals data is pulled every 2 minutes. The update time for Patient details is checked regularly, and re-pulled, if update is detected.

6 Validation

The verification and validation of the data sharing is the responsibility of the 3rd Party ePCR organisation utilising the WebAPI. It is essential to ensure:

- The data supplied is integrated with the record for the correct patient.
- Data from a single incident must not be mixed with data for other patients.
- All data is received and maintained on secure systems appropriate to storage of medical data.

The ePCR integrator should use the Corsium WebAPI to pull the pre-configured patients detailed below from the reference instance of Corsium and then verify that the expected data is received correctly by validating their implementation using the data tables shown below. The integrator may ask RDT to review the validation results.

6.1 Pre-configured Patient #1

Data Field	Details	Validation
Report Type	Standard	
Incident Start Time	Current time – 70 minutes	
Trended Vitals	All vitals are present and contain a range of typical values.	
Last name	Pre-configured #1	
First name	SRoC Male	
Sex	Male	
Age	27 years	
ID	MIL 30 - 123456789	
Weight	78 kg	
Height	70 inch	
Allergy	Bee Venom, Morphine, Cephalosporins	

Time Offset (relative to incident start)	Event	Validation Result
3 mins	Photo 1	
4 mins	Photo 2	
5 mins	GCS 6 2,2,2	
6 mins	Hextend 100ml	
7 mins	Oxygen	
9 mins	Laryngoscope Image	
10 mins	Medic Craig, Davis	
12 mins	Patient has a shoulder and chest injuries. Burns to right leg are second degree.	
12 mins	Injury map: **Open wound**: R Mid torso, L Mid torso	

	Crush: RUE Upper arm, R Shoulder, Posterior RUE Upper arm, Posterior R Shoulder	
14 mins	Burn: **2nd degree:** RLE Foot, R Hip, RLE Knee, RLE Thigh, RLE Lower Leg, Posterior RLE Foot, R Buttock, Posterior RLE Knee, Posterior RLE Thigh, Posterior RLE Lower Leg **Burn area**: 23%	
21 mins	Waveform Snapshot	
30 mins	12 lead ECG	
31 mins	Extreme Tachy	
32 mins	Amiodarone	
40 mins	GCS:11 3,4,4	
43 mins	12 lead ECG	
55 mins	Morphine	

6.2 Pre-configured Patient #2

Data Field	Details	Validation
Incident Start Time	Current time – 87 minutes	
Trended Vitals	87 minutes of trended vitals. All vitals are present and contain a range of typical values.	
Report Type	TC3	
Last name	Pre-configured#2	
First name	TC3 Female	
Sex	Female	
Age	Adult	
ID	ABCDEFGHIJKLMNOPQRSTUVWXY	
Weight	172 lbs	
Height	70 inch	
Allergy	NKDA	
Unit	ABCDEFGHIJKLMNOPQRSTUVWXY	
Service	012345678901234567 89ABCDE	
Evac	Urgent	
BattleRoster	TPVWXY	

Time Offset (relative to incident start)	Event	Validation Result
3 mins	Photo 1	
4 mins	Photo 2	

Time Offset (relative to incident start)	Event	Validation Result
5 mins	AVPU | Pain P, 8	
6 mins	Hextend 100ml	
7 mins	Oxygen	
9 mins	Laryngoscope	
10 mins	Medic Craig, Davis	
11 mins	Patient has a shoulder and chest injuries. Burns to right leg are second degree.	
17 mins	AVPU P, 7	
18 mins	Injury map: **Artillery:** RUE Upper arm, R Mid torso, R Shoulder, Posterior RUE Upper arm, Posterior R Shoulder **Burn:** RLE Foot, RLE Knee, RLE Lower Leg, Posterior RLE Foot, R Buttock, Posterior RLE Knee, Posterior RLE Lower Leg	
18 mins	TQ: RL	
18 mins	Airway intact	
18 mins	Chest seal	
19 mins	Splint	
19 mins	Pressure Dressing	
19 mins	Right eye-shield	
19 mins	Burns map: **2nd degree:** RLE Foot, RLE Knee, RLE Lower Leg, Posterior RLE Foot, R Buttock, Posterior RLE Knee, Posterior RLE Lower Leg **Burn area:** 15%	
25 mins	Snapshot	
31 mins	12 lead ECG	
31 mins	Extreme Tachy	
32 mins	Amiodarone	
48 mins	12 lead ECG	
48 mins	AVPU | Pain V, 7	
66 mins	AVPU | Pain A, 7	
67 mins	GCS:13 4,4,5	
67 mins	Morphine	
75 mins	Pain: 9	
76 mins	AVPU: P	

6.3 Pre-configured Patient #3

Data Field	Details	Validation
Incident Start Time	Current time – 1000 minutes	
Trended Vitals	1000 minutes of trended vitals All vitals are present. Vitals will start at the minimum value then increase every by the step value until the Maximum value is reached. When the maximum is reached the value will then be decremented by the One Minute Step value until the minimum value is reached.	
Report type	Standard	
Last name	Pre-configured#3	
First name	TrendedVitals	
Sex	Unknown	
Age	Neonate	
ID	Not set	
Weight	Not set	
Height	Not set	
Allergy	Not set	
Events	There are no events	

Vital	Unit	Min	Max	Step	Alarm Min	Alarm Max
HR	bpm	0	300	1	50	120
ST1	mm	-10.0	+10.0	1	-1.0	+1.0
ST2	mm	-10.0	+11.0	3	-10.0	+11.0
QT	ms	100	600	2	OFF	OFF
NIBP (Sys)	mmHg	40	240	1	90	160
NIBP (Dia)	mmHg	20	200	1	50	90
NIBP (Mean)	mmHg	26	220	1	60	110
Resp	rpm	1	149	1	5	50
ETCO2	mmHg	0	150	1	25	60
SpO2	%	0 *0 is "out of range"*	100	1	90	100
PI	%	0	10.0	0.1	OFF	OFF
PVI	%	0	100	1	OFF	OFF
SpOC	ml/dl	0	35	1	OFF	OFF
SpHb	g/dl	1	25	1	7.0	17.0
SpMet	%	0.1	100.1	1	OFF	3.0
SpCO	%	1	98	1	OFF	10
T1	C	20	45	1	35.0	37.8
T2	C	20	45	1	35.0	37.8
Arterial + Unlabelled						
IBP (Sys)	mmHg	-99	310	1	90	160

Vital	Unit	Min	Max	Step	Alarm Min	Alarm Max
IBP (Mean)	mmHg	-19	290	3	50	90
IBP (Dia)	mmHg	-59	301	2	60	110
Venous						
IBP (Sys)	mmHg	-99	310	1	6	14
IBP (Mean)	mmHg	-19	290	3	-4	6
IBP (Dia)	mmHg	-59	301	2	0	10
PAP						
IBP (Sys)	mmHg	-21	289	4	OFF	OFF
IBP (Mean)	mmHg	-22	290	4	0	16
IBP (Dia)	mmHg	-59	301	2	OFF	300
ICP						
IBP (Sys)	mmHg	-99	310	1	OFF	OFF
IBP (Mean)	mmHg	-22	293	5	0	10
IBP (Dia)	mmHg	-20	291	5	OFF	OFF

6.4 Pre-configured Patient #4

Data Field	Details	Validation
Incident Start Time	Current time – 30 minutes	
Trended Vitals	30 minutes of trended vitals All vitals are blank apart from HR.	
Patient type	Standard	
Last name	Pre-configured #4	
First name	FullBodyMaps	
Sex	Other	
Age	Paediatric	
ID	Not set	
Weight	35 kg	
Height	130 cm	
Allergy	Other: Pollen	

Time Offset (relative to incident start)	Event	Validation Result
5 mins	Injury map: **Injury 1**: Head, RUE Hand, Posterior LUE Upper arm, Posterior LLE Foot, Posterior LLE Thigh **Injury 2**: R Hip, Posterior Head, Posterior L Mid torso, Posterior RLE Foot, Posterior RLE Thigh **Injury 3**: Groin, RLE Knee, R Shoulder, Posterior R Mid torso **Injury 4**: Upper torso, L Hip, LLE Knee, Posterior RUE Upper arm **Injury 5**: RUE Forearm, LUE Hand, L Shoulder, Posterior LLE Knee	

Time Offset (relative to incident start)	Event	Validation Result
	Injury 6: R Lower torso, Posterior LUE Hand, Posterior L Shoulder, Posterior RLE Knee **Injury 7**: RLE Lower Leg, L Lower torso, Posterior Upper torso, L Buttock **Injury 8**: LUE Forearm, LLE Lower Leg, Groin, Posterior R Shoulder **Injury 9**: RUE Upper arm, Posterior LUE Forearm, Posterior LLE Lower Leg, R Buttock **Injury 10**: R Mid torso, Posterior L Lower torso, Posterior RUE Hand, Posterior RLE Lower Leg **Injury 11**: RLE Foot, RLE Thigh, L Mid torso, Posterior R Lower torso **Injury 12**: LUE Upper arm, LLE Foot, LLE Thigh, Posterior RUE Forearm	
10 mins	**R Arm:** R Shoulder: Junctional RUE Upper arm, RUE Forearm, RUE Hand: Extremity **L Arm:** L Shoulder: Junctional LUE Upper arm, LUE Forearm, LUE Hand: Extremity **R Leg:** R Hip: Junctional RLE Thigh, RLE Knee, RLE Lower Leg, RLE Foot: Extremity **L Leg:** L Hip: Junctional LLE Thigh, LLE Knee, LLE Lower Leg, LLE Foot: Extremity	
15 mins	Burns map: **1st degree**: Head, Upper torso, R Mid torso, RUE Hand, RLE Knee, LLE Foot, L Hip, LLE Thigh, L Lower torso, Posterior Upper torso, Posterior LUE Upper arm, Posterior LUE Hand, Posterior LLE Lower Leg, Posterior L Lower torso, Posterior RUE Upper arm, R Buttock, Posterior RLE Knee **2nd degree**: RUE Forearm, R Hip, RLE Lower Leg, LUE Forearm, L Mid torso, LUE Hand, LLE Knee, L Shoulder, Posterior Head, Posterior L Mid torso, Posterior LLE Foot, L Buttock, Posterior LLE Thigh, Posterior RUE Hand, Posterior RLE Lower Leg, Posterior R Shoulder, Posterior R Lower torso **3rd degree**: Groin, RUE Upper arm, RLE Foot, RLE Thigh, R Shoulder, R Lower torso, LUE Upper arm, LLE Lower Leg, Groin, Posterior LUE Forearm, Posterior LLE Knee, Posterior L Shoulder, Posterior RUE Forearm, Posterior R Mid torso, Posterior RLE Foot, Posterior RLE Thigh Burn area: 25%	

Time Offset (relative to incident start)	Event	Validation Result
20 mins	Dressing map: **Dressing 1**: RUE Upper arm, Posterior LUE Forearm, Posterior LLE Lower Leg, R Buttock **Dressing 2**: R Mid torso, Posterior L Lower torso, Posterior RUE Hand, Posterior RLE Lower Leg **Dressing 3**: RLE Foot, RLE Thigh, L Mid torso, Posterior R Lower torso **Dressing 4**: LUE Upper arm, LLE Foot, LLE Thigh, Posterior RUE Forearm **Dressing 5**: Head, RUE Hand, Posterior LUE Upper arm, Posterior LLE Foot, Posterior LLE Thigh **Dressing 6**: R Hip, Posterior Head, Posterior L Mid torso, Posterior RLE Foot, Posterior RLE Thigh **Dressing 7**: Groin, RLE Knee, R Shoulder, Posterior R Mid torso **Dressing 8**: Upper torso, L Hip, LLE Knee, Posterior RUE Upper arm **Dressing 9**: RUE Forearm, LUE Hand, L Shoulder, Posterior LLE Knee **Dressing 10**: R Lower torso, Posterior LUE Hand, Posterior L Shoulder, Posterior RLE Knee **Dressing 11**: RLE Lower Leg, L Lower torso, Posterior Upper torso, L Buttock **Dressing 12**: LUE Forearm, LLE Lower Leg, Groin, Posterior R Shoulder	

6.5 Pre-configured Patient #5

Data Field	Details	Validation
Incident Start Time	Current time – 90 minutes	
Trended Vitals	90 minutes of trended vitals ST1, ST2 change the lead to cover all leads. RESP changes between CAPNO and ECG. HR (PULSE) changes between ECG and PULSE. IBP change channels, include not set. Where available, include “Out of Range” values. Includes “No breathes” for RESP.	
Report type	Standard	
Last name	Pre-configured #5	
First name	UnusualValues	
Sex	Male	
Age	35 years	
ID	12345678	
Weight	100 kg	
Height	182 cm	

Data Field	Details	Validation
Allergy	NKDA	

Time Offset (relative to incident start)	Event	Validation Result
3 mins	Photo 1	
4 mins	Photo 2	
5 mins	GCS 6 2,2,2	
6 mins	Hextend 100ml	
7 mins	Oxygen	
9 mins	Laryngoscope Image	
10 mins	Medic Craig, Davis	
12 mins	Patient has a shoulder and chest injuries. Burns to right leg are second degree.	
12 mins	Injury map: **Open wound:** R Mid torso, L Mid torso **Crush**: RUE Upper arm, R Shoulder, Posterior RUE Upper arm, Posterior R Shoulder	
14 mins	Burn Map: **2nd degree:** RLE Foot, R Hip, RLE Knee, RLE Thigh, RLE Lower Leg, Posterior RLE Foot, R Buttock, Posterior RLE Knee, Posterior RLE Thigh, Posterior RLE Lower Leg **Burn area:** 23%	
21 mins	Snapshot	
30 mins	12 lead ECG	
31 mins	Extreme Tachy	
32 mins	Amiodarone	
40 mins	GCS:11 3,4,4	
43 mins	12 lead ECG	
55 mins	Morphine	

6.6 Pre-configured Patient #6

Data Field	Details	Validation
Incident Start Time	Current time – 90 minutes	
Trended Vitals	90 minutes of trended vitals. All vitals are present and contain a range of typical values.	
Report type	TC3	
Last name	Pre-configured #6	

Data Field	Details	Validation
First name	TC3Full	
Sex	Unknown	
Age	Adult	
ID	ABCD1234	
Weight	172 lbs	
Height	70 inch	
Allergy	NKDA, Latex, Insulin, Sulfa, Penicillin	
Unit	UVWXY	
Service	789ABC	
EVAC	Priority	
BattleRoster	TP1234	
Notes	Add event notes to all events as well.	

Time Offset (relative to incident start)	Event	Validation Result
3 mins	Photo 1	
4 mins	Photo 2	
5 mins	AVPU | Pain P, 8	
6 mins	Hextend 100ml	
7 mins	Oxygen	
9 mins	Laryngoscope	
10 mins	Medic Craig, Davis	
11 mins	Patient has a shoulder and chest injuries. Burns to right leg are second degree.	
17 mins	AVPU P, 7	
18 mins	Injury map: **Artillery**: RUE Upper arm, R Mid torso, R Shoulder, Posterior RUE Upper arm, Posterior R Shoulder **Burn**: RLE Foot,RLE Knee,RLE Lower Leg,Posterior RLE Foot,R Buttock,Posterior RLE Knee,Posterior RLE Lower Leg	
18 mins	TQ: RL	
18 mins	Airway intact	
18 mins	Chest seal	
19 mins	Splint	
19 mins	Pressure Dressing	
19 mins	Right eye-shield	
19 mins	Burns map: **2nd degree**: RLE Foot, RLE Knee, RLE Lower Leg, Posterior RLE Foot, R	

Time Offset (relative to incident start)	Event	Validation Result
	Buttock, Posterior RLE Knee, Posterior RLE Lower Leg **Burn area**: 15%	
25 mins	Snapshot	
31 mins	12 lead ECG	
31 mins	Extreme Tachy	
32 mins	Amiodarone	
48 mins	12 lead ECG	
48 mins	AVPU | Pain V, 7	
66 mins	AVPU | Pain A, 7	
67 mins	GCS:13 4,4,5	
67 mins	Morphine	
76 mins	Pain: 9	
76 mins	AVPU: P Note: Patient in pain	
80 mins	Airway Note: Intubation needed	
81 mins	Breathing Note: Oxygen given	
83 mins	Circulation Note: TQs applied	
85 mins	Dressing Note: Dressings on the burns	
86 mins	Other Note: Patient stable	
90 mins	TQ Truncal, Note: TQ applied	
90 mins	TC3 Note: Patient critical	

6.7 Pre-configured Patient #7

Data Field	Details	Validation
Incident Start Time	Current time – 22 minutes	
Trended Vitals	22 minutes of trended vitals. Only HR captured by LS present	
Report type	Standard	
Last name	Pre-configured#7	
First name	LS Events	
Sex	Female	
Age	Adult	
ID	ABCDEFGHIJKLMNOPQRSTUVWXY	
Weight	172 lbs	
Height	70 inch	
Allergy	NKDA	

LS Offset	Event	Validation Result
(+00:00:00)	Defib Incident Start: Serial # 7021.000172	
(+00:00:05)	Pads Detached	
(+00:00:15)	Pads Attached: Adult Pads	
(+00:01:31)	CPR Start	
(+00:02:39)	CPR End: 99/min, 91%	
(+00:03:12)	Pads Detached	
(+00:03:17)	Pads Attached: Adult Pads	
(+00:03:29)	CPR Start	
(+00:04:33)	CPR End: 100/min	
(+00:05:03)	Pads Detached	
(+00:05:09)	Pads Attached: Adult Pads	
(+00:05:29)	Shock #1: 200J (199J), 49Ω	
(+00:06:58)	Shock #2: 200J (197J), 49Ω, Sync	
(+00:07:42)	AED Mode	
(+00:07:56)	Security Discharge	
(+00:07:56)	No Shock Advised	
(+00:10:25)	Shock Advised: Shockable rhythm	
(+00:10:28)	Shock #3: 150J (149J), 49Ω	
(+00:11:40)	Analysis Requested	
(+00:11:50)	Shock Advised: Shockable rhythm	
(+00:11:53)	Shock #4: 150J (149J), 49Ω	
(+00:11:56)	Security Discharge	
(+00:11:56)	Manual Mode	
(+00:12:31)	Waveforms	
(+00:13:25)	Pacer mode	
(+00:13:28)	Pacing: Fix, 10mA, 60/min	
(+00:14:53)	Pacing: Demand, 10mA, 60/min	
(+00:16:34)	Pacing stopped	
(+00:16:35)	Manual Mode	
(+00:17:04)	Monitor Mode	
(+00:18:49)	Manual Mode	
(+00:19:20)	Pads Detached	
(+00:19:30)	Pads Attached: Paediatric Pads	
(+00:21:11)	Pads Detached	

7 Appendix

7.1 Response error messages

Response Code	Error Message
001	Invalid credentials
002	Authorization key has expired
003	Request limit exceeded
004	Data access denied
005	Invalid API
006	Invalid API request parameters
007	Too many records found. Narrow your search time.
008	Internal error occurred
009	PHI removed from the incident
010	ePCR details changed
011	*not used*
012	PDF cannot be created for the live incident
013	API request parameter values are missing
014	Invalid API request date format
015	Invalid API date range
016	Invalid API request headers
017	Invalid API content parameters
018	Invalid API data encryption method

7.2 Vitals details

Vital sign	Vital name	Vital source	Measurement	Range
Heart Rate (ECG)	HR	ECG	bpm	30 to 300
Pulse Rate (Pulse Oximetry)	PULSE	PULSE	bpm	25 to 239
Heart Rate (Tempus LS)	HR	TEMPUS LS	bpm	16 to 300
Pulse Oximetry SpO2	SPO2	PULSE	%	1 to 100
Non-invasive Systolic Arterial Pressure	SysNIBP	NIBP	mmHg	Adult 40 to 260 Paediatric 40 to 230 Neonate 40 to 130
Non-invasive Diastolic Arterial Pressure	DiaNIBP	NIBP	mmHg	Adult 20 to 200 Paediatric 20 to 160 Neonate 20 to 100

Vital sign	Vital name	Vital source	Measurement	Range
Non-invasive Mean Arterial Pressure	MeanNIBP	NIBP	mmHg	Adult 26 to 220 Paediatric 26 to 183 Neonate 26 to 110
Temperature	T1 or T2	T1 or T2	C	20 to 45
Respiration Rate (Capno)	Resp	CAPNO	rpm	1 to 149
Respiration Rate (Impedance)	Resp	ECG	rpm	30 to 150
End Tidal CO2	ETCO2	CAPNO	mmHg	0 to 150
ST Elevation	ST1 or ST2	ECG-*lead* Where *lead* is one of the following: I,II,III,aVR,aVL,aVF, V1,V2,V3,V4,V5	mm	+/- 15
QT	QT	ECG	ms	0 to 1000
Carboxyhaemoglobin (SpCO)	SpCO	PULSE	%	0 to 99.9
Total Haemoglobin (SpHb)	SpHb	PULSE	g/dl	0 to 25
Methaemoglobin (SpMet)	SpMet	PULSE	%	1 to 99.5
Oxygen Content (SpOC)	SpOC	PULSE	ml/dl	0 to 35
Pulse Oximetry Perfusion Index	PI	PULSE	%	1 to 20 Values less than 1.0 are provided with a precision of two decimal places
Pulse Oximetry Variability Index	PVI	PULSE	%	0 to 100
Bladder Systolic Invasive Pressure	SysBDR	P1:BDR or P2:BDR or P3:BDR or P4:BDR	mmHg	-99 to 310
Bladder Diastolic Invasive Pressure	DiaBDR	P1:BDR or P2:BDR or P3:BDR or P4:BDR	mmHg	-99 to 310
Bladder Mean Invasive Pressure	MeanBDR	P1:BDR or P2:BDR or P3:BDR or P4:BDR	mmHg	-99 to 310
Intracranial Systolic Invasive Pressure	SysICP	P1:ICP or P2:ICP or P3:ICP or P4:ICP	mmHg	-99 to 310
Intracranial Diastolic Invasive Pressure	DiaICP	P1:ICP or P2:ICP or P3:ICP or P4:ICP	mmHg	-99 to 310

Vital sign	Vital name	Vital source	Measurement	Range
Intracranial Mean Invasive Pressure	MeanICP	P1:ICP or P2:ICP or P3:ICP or P4:ICP	mmHg	-99 to 310
Central Venous Systolic Invasive Pressure	SysCVP	P1:CVP or P2:CVP or P3:CVP or P4:CVP	mmHg	-99 to 310
Central Venous Diastolic Invasive Pressure	DiaCVP	P1:CVP or P2:CVP or P3:CVP or P4:CVP	mmHg	-99 to 310
Central Venous Mean Invasive Pressure	MeanCVP	P1:ICP or P2:ICP or P3:ICP or P4:ICP	mmHg	-99 to 310
Pulmonary Artery Systolic Invasive Pressure	SysPAP	P1:PAP or P2:PAP or P3:PAP or P4:PAP	mmHg	-99 to 310
Pulmonary Artery Diastolic Invasive Pressure	DiaPAP	P1:PAP or P2:PAP or P3:PAP or P4:PAP	mmHg	-99 to 310
Pulmonary Artery Mean Invasive Pressure	MeanPAP	P1:PAP or P2:PAP or P3:PAP or P4:PAP	mmHg	-99 to 310
Umbilical Venous Systolic Invasive Pressure	SysUVP	P1:UVP or P2:UVP or P3:UVP or P4:UVP	mmHg	-99 to 310
Umbilical Venous Diastolic Invasive Pressure	DiaUVP	P1:UVP or P2:UVP or P3:UVP or P4:UVP	mmHg	-99 to 310
Umbilical Venous Mean Invasive Pressure	MeanUVP	P1:UVP or P2:UVP or P3:UVP or P4:UVP	mmHg	-99 to 310

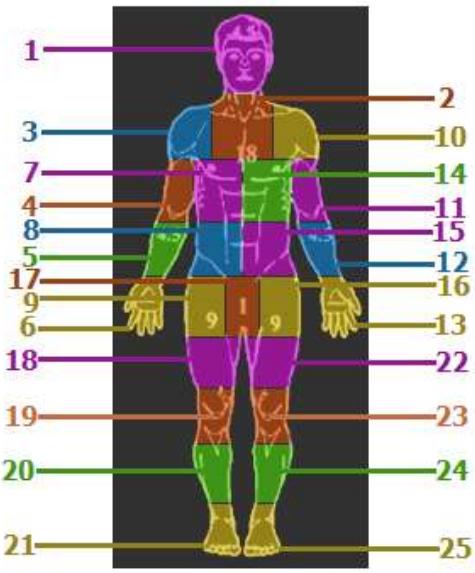
Vital sign	Vital name	Vital source	Measurement	Range
Arterial Systolic Invasive Pressure	SysART	P1:ART or P2:ART or P3:ART or P4:ART	mmHg	-99 to 310
Arterial Diastolic Invasive Pressure	DiaART	P1:ART or P2:ART or P3:ART or P4:ART	mmHg	-99 to 310
Arterial Mean Invasive Pressure	MeanART	P1:ART or P2:ART or P3:ART or P4:ART	mmHg	-99 to 310
Aortic Systolic Invasive Pressure	SysAO	P1:AO or P2:AO or P3:AO or P4:AO	mmHg	-99 to 310
Aortic Diastolic Invasive Pressure	DiaAO	P1:AO or P2:AO or P3:AO or P4:AO	mmHg	-99 to 310
Aortic Mean Invasive Pressure	MeanAO	P1:AO or P2:AO or P3:AO or P4:AO	mmHg	-99 to 310
Abdominal Aortic Sys Invasive Pressure	SysABP	P1:ABP or P2:ABP or P3:ABP or P4:ABP	mmHg	-99 to 310
Abdominal Aortic Dia Invasive Pressure	DiaABP	P1:ABP or P2:ABP or P3:ABP or P4:ABP	mmHg	-99 to 310
Abdominal Aortic Mean Invasive Pressure	MeanABP	P1:ABP or P2:ABP or P3:ABP or P4:ABP	mmHg	-99 to 310
Brachial Aortic Systolic Invasive Pressure	SysBAP	P1:BAP or P2:BAP or P3:BAP or P4:BAP	mmHg	-99 to 310

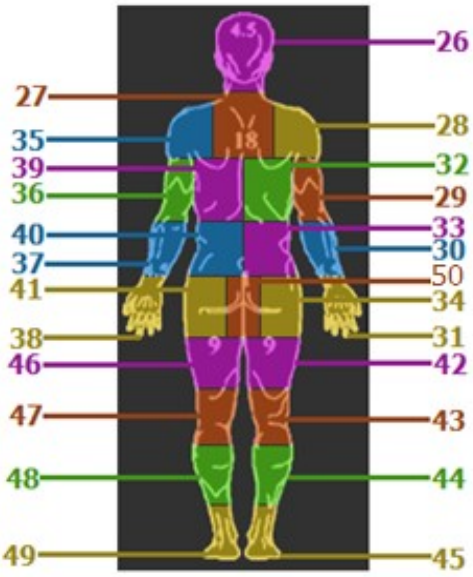
Vital sign	Vital name	Vital source	Measurement	Range
Brachial Aortic Diastolic Invasive Pressure	DiaBAP	P1:BAP or P2:BAP or P3:BAP or P4:BAP	mmHg	-99 to 310
Brachial Aortic Mean Invasive Pressure	MeanBAP	P1:BAP or P2:BAP or P3:BAP or P4:BAP	mmHg	-99 to 310
Femoral Artery Systolic Invasive Pressure	SysFAP	P1:FAP or P2:FAP or P3:FAP or P4:FAP	mmHg	-99 to 310
Femoral Artery Diastolic Pressure	DiaFAP	P1:FAP or P2:FAP or P3:FAP or P4:FAP	mmHg	-99 to 310
Femoral Artery Mean Pressure	MeanFAP	P1:FAP or P2:FAP or P3:FAP or P4:FAP	mmHg	-99 to 310
Labial Artery Systolic Invasive Pressure	SysLAP	P1:LAP or P2:LAP or P3:LAP or P4:LAP	mmHg	-99 to 310
Labial Artery Diastolic Invasive Pressure	DiaLAP	P1:LAP or P2:LAP or P3:LAP or P4:LAP	mmHg	-99 to 310
Labial Artery Mean Invasive Pressure	MeanLAP	P1:LAP or P2:LAP or P3:LAP or P4:LAP	mmHg	-99 to 310
Radial Artery Systolic Invasive Pressure	SysRAP	P1:RAP or P2:RAP or P3:RAP or P4:RAP	mmHg	-99 to 310
Radial Artery Diastolic Invasive Pressure	DiaRAP	P1:RAP or P2:RAP or P3:RAP or P4:RAP	mmHg	-99 to 310

Vital sign	Vital name	Vital source	Measurement	Range
Radial Artery Mean Invasive Pressure	MeanRAP	P1:RAP or P2:RAP or P3:RAP or P4:RAP	mmHg	-99 to 310
Umbilical Artery Sys Invasive Pressure	SysUAP	P1:UAP or P2:UAP or P3:UAP or P4:UAP	mmHg	-99 to 310
Umbilical Artery Diastolic Invasive Pressure	DiaUAP	P1:UAP or P2:UAP or P3:UAP or P4:UAP	mmHg	-99 to 310
Umbilical Artery Mean Invasive Pressure	MeanUAP	P1:UAP or P2:UAP or P3:UAP or P4:UAP	mmHg	-99 to 310
Unlabelled Systolic Invasive Pressure	Sys	P1 or P2 or P3 or P4	mmHg	-99 to 310
Unlabelled Diastolic Invasive Pressure	Dia	P1 or P2 or P3 or P4	mmHg	-99 to 310
Unlabelled Mean Invasive Pressure	Mean	P1 or P2 or P3 or P4	mmHg	-99 to 310

Note	The NIBP is not measured in regular intervals of 1 minute but intermittently. The capture time shall not be rounded off to the nearest minute.
Note	The PI percentage is obtained by dividing the value by 1000.
Note	The pulse rate measured using pulse Oximetry will be provided in the case that ECG heart rate measurements are not available.
Note	The ECG impedance respiration rate will be provided in the case that capnometry respiration rate measurements are not available.

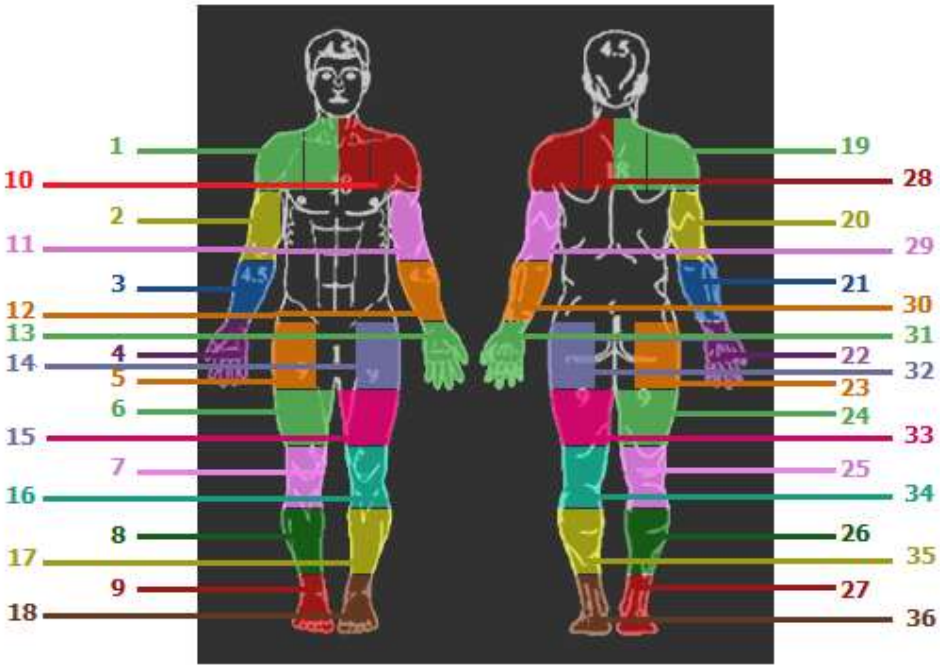
7.3 Body maps locations





Location	Description	Location	Description
1	Head	26	Posterior Head
2	Upper torso	27	Posterior Upper torso
3	R Shoulder	28	Posterior R Shoulder
4	RUE Upper arm	29	Posterior RUE Upper arm
5	RUE Forearm	30	Posterior RUE Forearm
6	RUE Hand	31	Posterior RUE Hand
7	R Mid torso	32	Posterior R Mid torso
8	R Lower torso	33	Posterior R Lower torso
9	R Hip	34	R Buttock
10	L Shoulder	35	Posterior L Shoulder
11	LUE Upper arm	36	Posterior LUE Upper arm
12	LUE Forearm	37	Posterior LUE Forearm
13	LUE Hand	38	Posterior LUE Hand
14	L Mid torso	39	Posterior L Mid torso
15	L Lower torso	40	Posterior L Lower torso
16	L Hip	41	L Buttock
17	Groin	42	Posterior RLE Thigh
18	RLE Thigh	43	Posterior RLE Knee
19	RLE Knee	44	Posterior RLE Lower leg
20	RLE Lower leg	45	Posterior RLE Foot
21	RLE Foot	46	Posterior LLE Thigh
22	LLE Thigh	47	Posterior LLE Knee
23	LLE Knee	48	Posterior LLE Lower leg
24	LLE Lower leg	49	Posterior LLE Foot
25	LLE Foot	50	Posterior Groin

7.4 TQ map locations



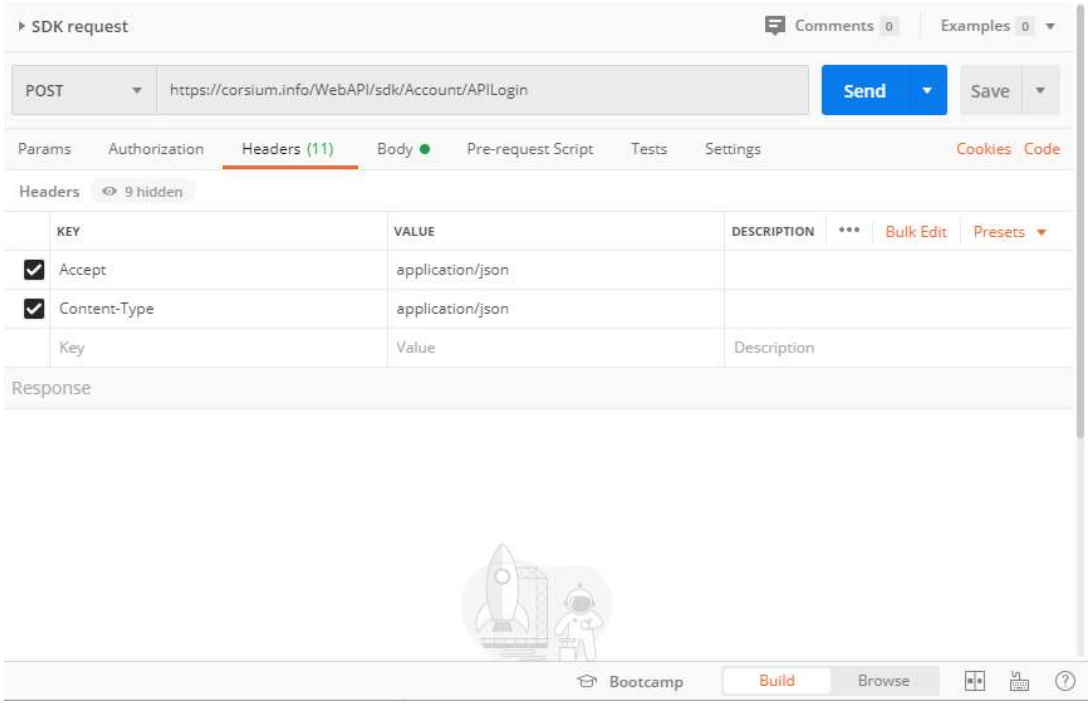
Location	Description	Location	Description
1	R Shoulder	19	R Shoulder
2	RUE Upper Arm	20	RUE Upper Arm
3	RUE Forearm	21	RUE Forearm
4	RUE Hand	22	RUE Hand
5	R Hip	23	R Hip
6	RLE Thigh	24	RLE Thigh
7	RLE Knee	25	RLE Knee
8	RLE Lower Leg	26	RLE Lower Leg
9	RLE Foot	27	RLE Foot
10	L Shoulder	28	L Shoulder
11	LUE Upper Arm	29	LUE Upper Arm
12	LUE Forearm	30	LUE Forearm
13	LUE Hand	31	LUE Hand
14	L Hip	32	L Hip
15	LLE Thigh	33	LLE Thigh
16	LLE Knee	34	LLE Knee
17	LLE Lower Leg	35	LLE Lower Leg
18	LLE Foot	36	LLE Foot

7.5 API access using Postman

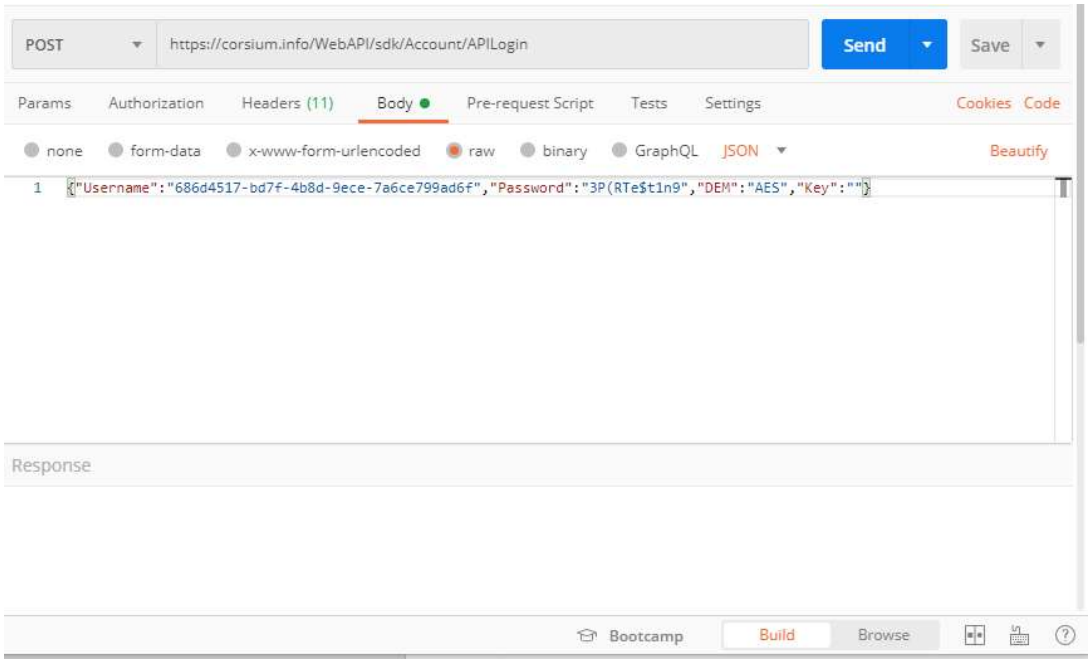
This section describes steps to login to Corsium WebAPI service and pull patient data using Postman client. Similar configuration can be used to access data using other clients such as Talend.

7.5.1 API Login

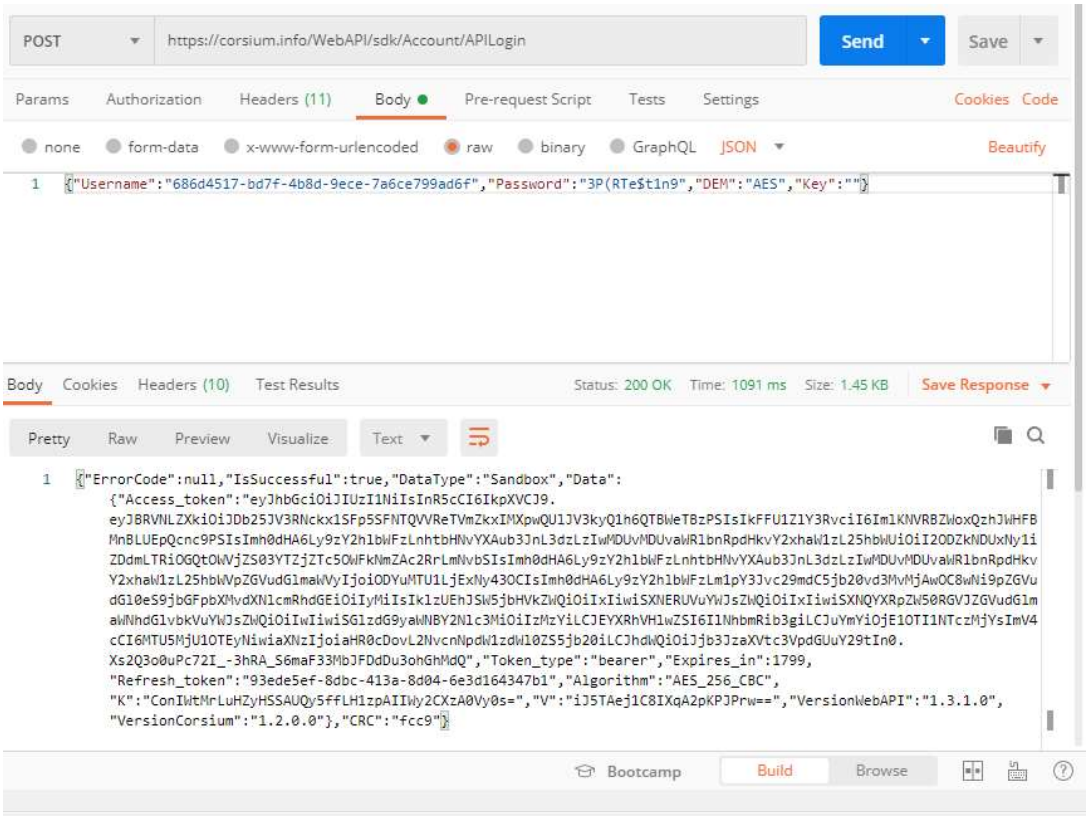
Step 1: Set the appropriate request headers.



Step 2: Specify the login details in the request body as JSON message.



Step 3: Click on Send button to generate the response.

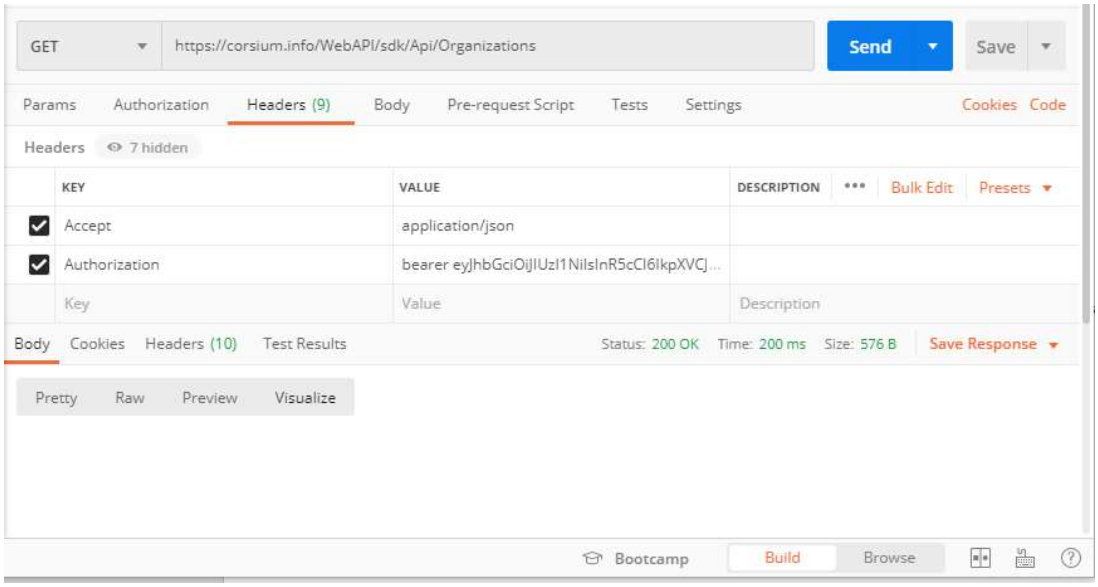


Step 4: Note down “Access_token”, “K” and “V” values from the response.

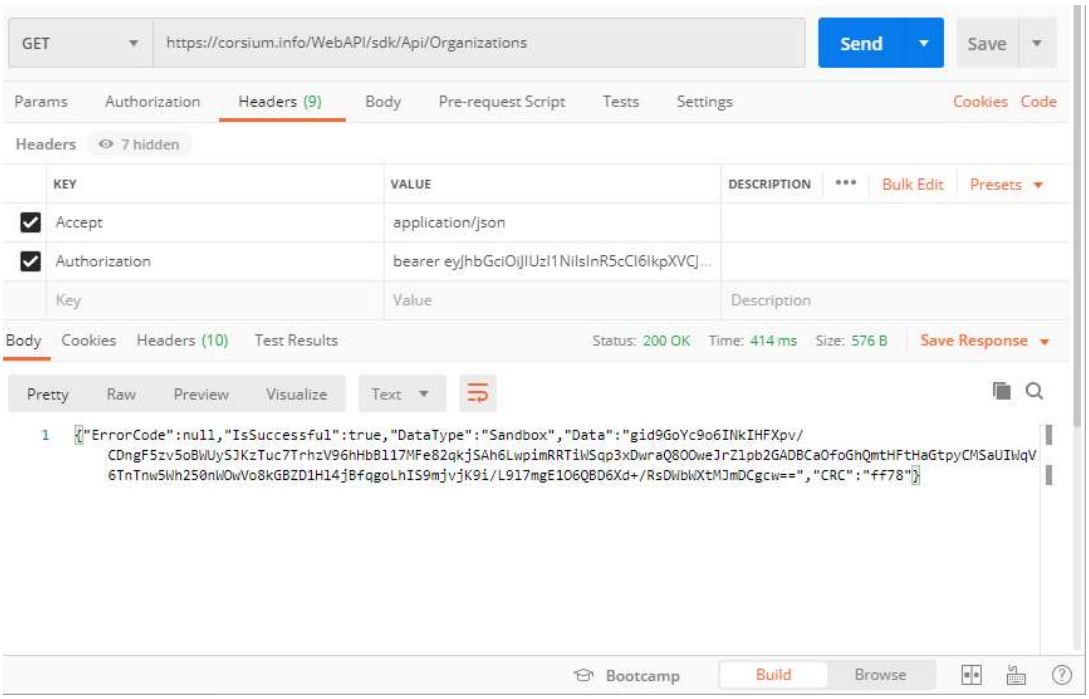
7.5.2 Patient data access (e. g. Organizations API)

Step 1: Set the appropriate request header. For the GET request, only “Accept” needs to be set to “application/json”.

Step 2: Copy the “Access_token” from the APILogin response and include it in the “Authorization” header as shown below.



Step 3: Click on the Send button to retrieve the data. Note that the “Data” section is encrypted since “DEM” parameter is set to AES.



Step 4: Decrypt data using the “K” and “V” parameter obtained from the APILogin response. See “7.6.1 AES encrypted data”

7.5.3 Patient data access (e. g. Incidents API)

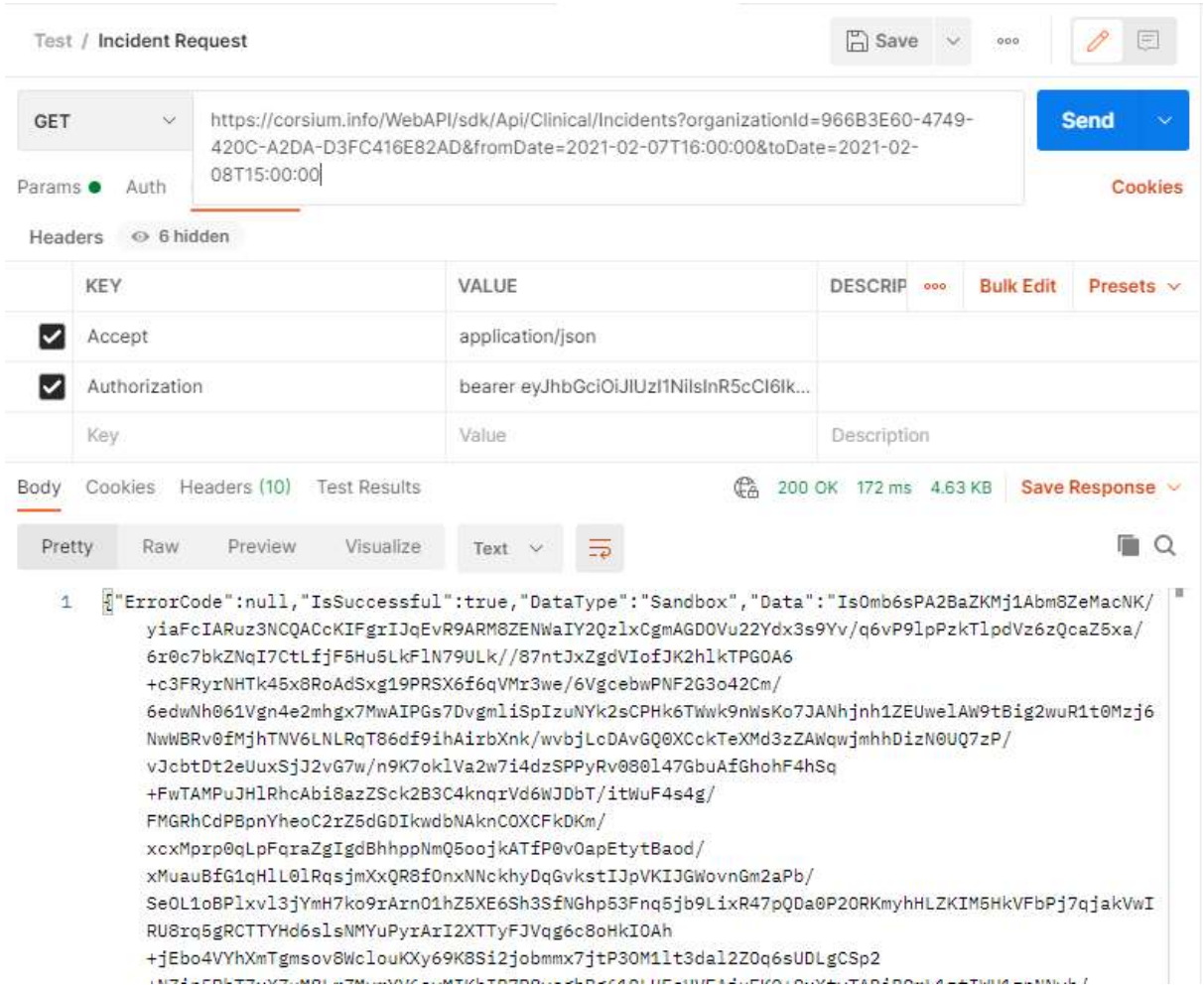
Step 1: Set the appropriate request header. For the GET request, only “Accept” needs to be set to “application/json”.

Step 2: Obtain Organization ID from the “Organizations” API response data to use it in “Incidents” API request.

Step 3: Specify fromDate and toDate parameters as well in the format shown in the diagram below.

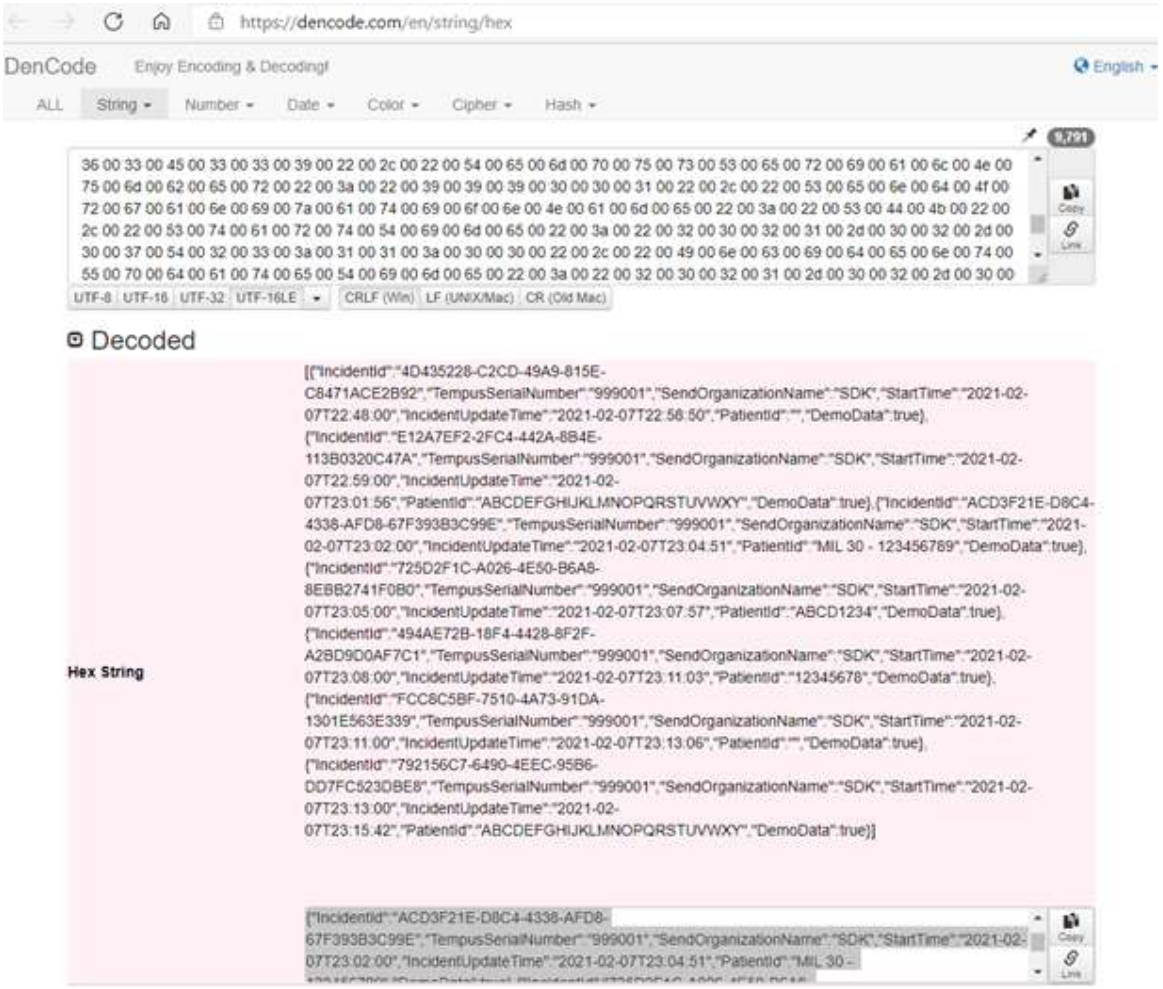
Step 4: Copy the “Access_token” from the APILogin response and include it in the “Authorization” header as shown below.

Step 5: Click on the Send button to retrieve the data. Note that the “Data” section is encrypted since “DEM” parameter is set to AES.



Step 6: Decrypt data using the “K” and “V” parameter obtained from the APILogin response. See “7.6.1 AES encrypted data”

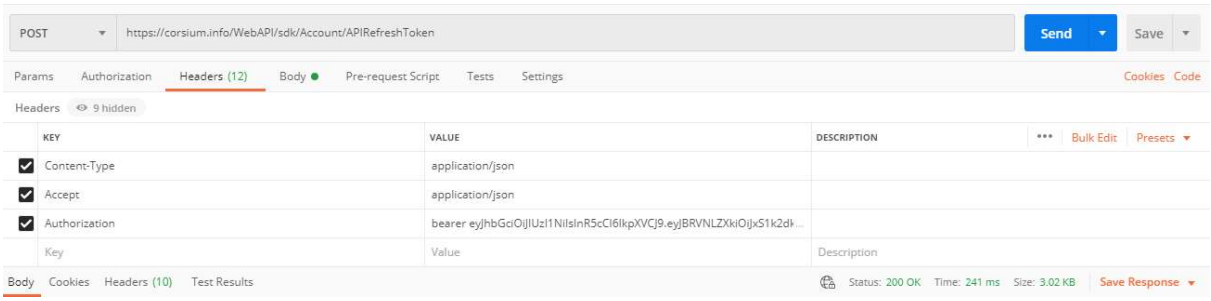
Step 7: The decrypted data looks like as follows:



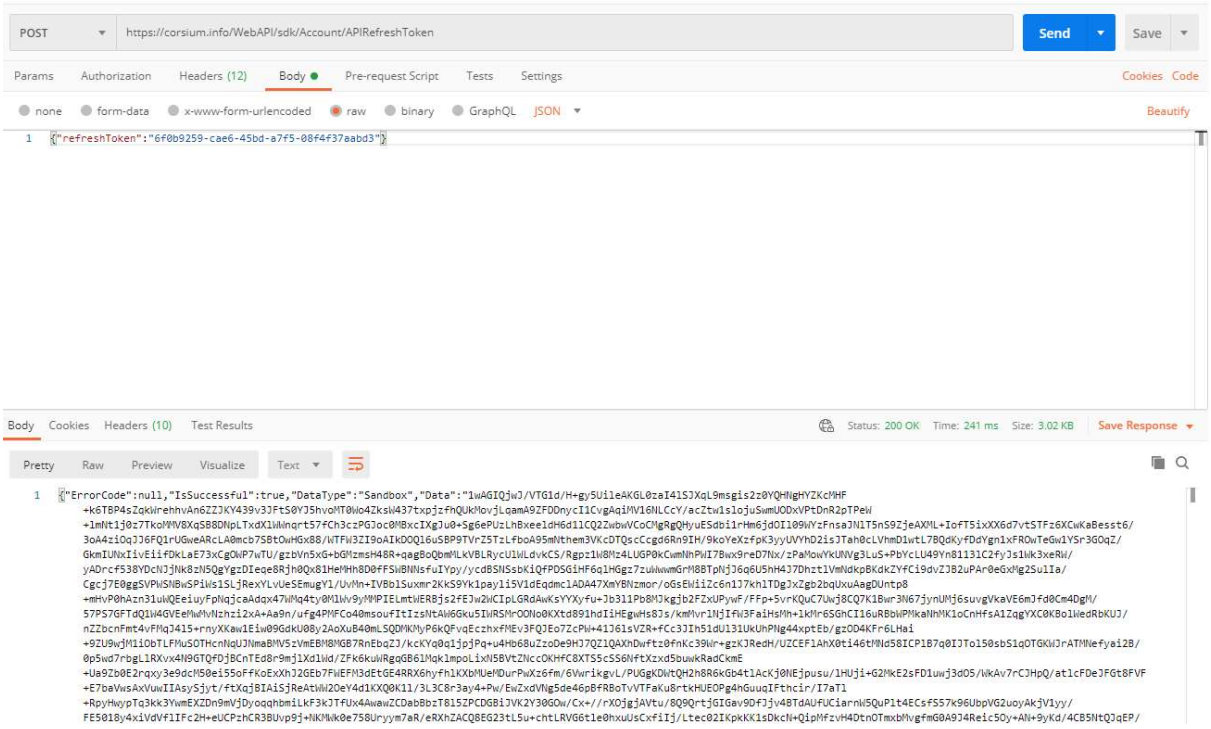
7.5.4 Refresh Token

Step 1: Set the appropriate request header. For the GET request, only “Accept” needs to be set to “application/json”

Step 2: Copy the “Access_token” from the APILogin response and include it in the “Authorization” header as shown below:



Step 3: Specify the refresh token related details in the request body as JSON message.



7.6 Data Decryption

If double encryption is enabled for the customer account by Corsium administrator, data is encrypted before it is sent in the response. Depending on DEM parameter, data is encrypted either using AES or public key provided by the customer. This data needs to be decrypted using steps described below. Following sections also provide an example of how to do it using online tools.

7.6.1 AES encrypted data

Below are the steps to decrypt data when encryption mode is AES.

Step 1: Note down the “K” and “V” parameters from the Login response. A sample response looks like as follows:

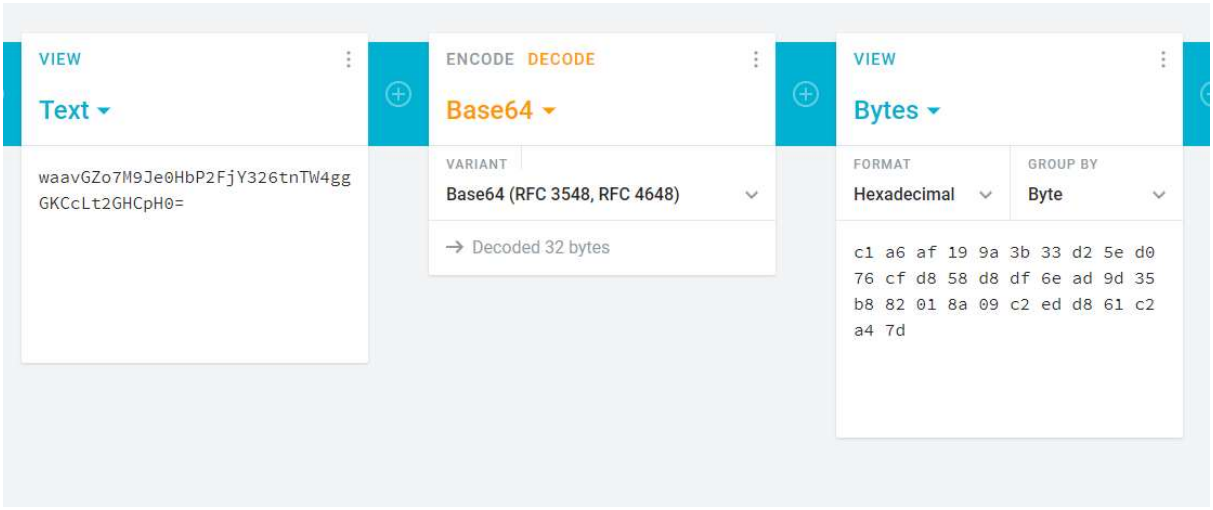
```
{
  “ErrorCode”: null,
  “IsSuccessful”: true,
  “DataType”:”real”,
  “Data”: {
    “Access_token”:”eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9…
                   0ZS5jb20ifQ.b011rw8mhoq0pTXY-
                   lK3v_dPI4nbZpG2YrF7jOWDzc”,
    “Token_type”:”bearer”,
    “Expires_in”:1800,
    “Refresh_token”:”cbd9b3ba-350d-4bc9-b4b8-d038c694d44a”,
    “Algorithm”:”AES_256_CBC”,
    “K”:”waavGZo7M9Je0HbP2FjY326tnTW4ggGKCcLt2GHCpH0=”
    “V”:”l0BCVwqQ+4QpzCXSRCU5Gw==”,
```

```
    "VersionWebAPI": "2.0.0",
    "VersionCorsium": "1.1.4.0"
  },
  "CRC": "5tr9"
}
```

Note that the "K" and "V" parameters are Base64 encoded. Decode them to get actual Key and Initialization vector bytes.

Online tools can be used to get actual Key and Initialization Vector bytes in hexadecimal format. For example:

https://cryptii.com/pipes/base64-to-hex



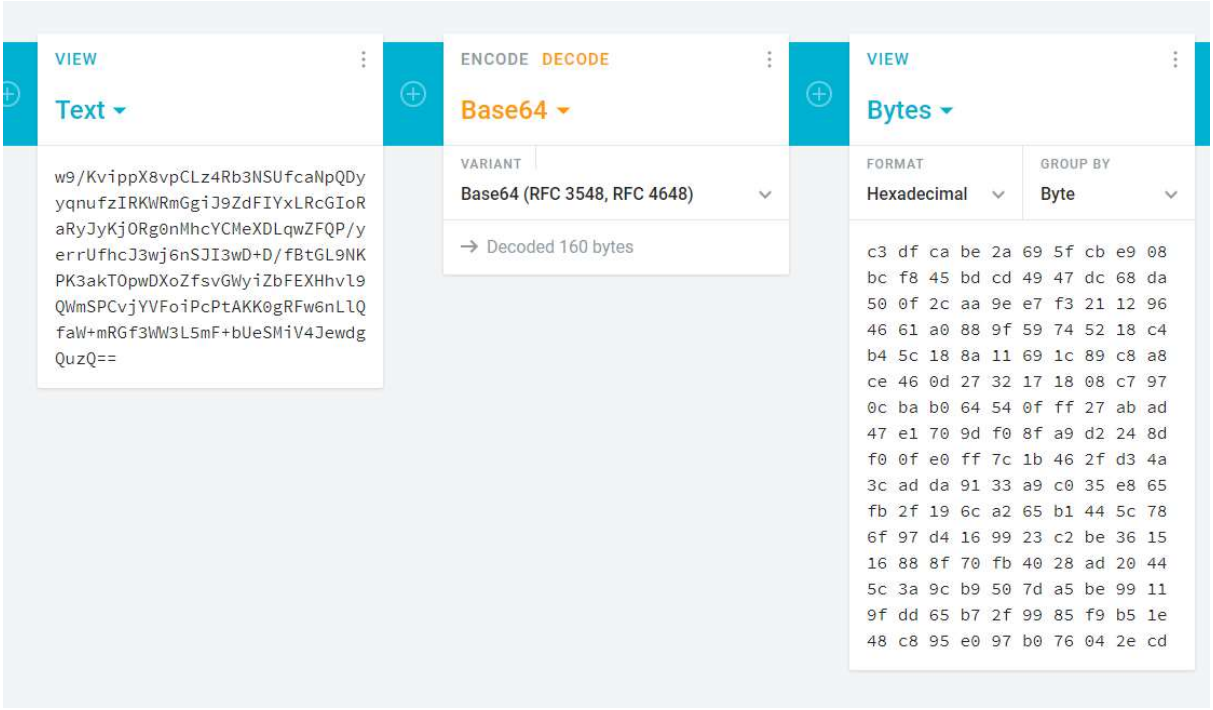
Step 2: Call other APIs to get actual data. Here is the sample response from "Organizations" API when data is encrypted:

```
{
  "ErrorCode": null,
  "IsSuccessful": true,
  "DataType":"real",
  "Data":"
w9/KvippX8vpCLz4Rb3NSUfcaNpQDyyqnufzIRKWRmGgiJ9ZdFIYxLRcGIoRaRyJyKjORg0nMhcYCMeXDLqwZFQP/
yerrUfhcJ3wj6nSJI3wD+D/fBtGL9NKPK3akTOpwDXoZfsvGWyiZbFEXHhvl9QWmSPCvjYVFoiPcPtAKK0gRFw6nL
lQfaW+mRGf3WW3L5mF+bUeSMiV4JewdgQuzQ==" // encrypted using AES and Base64 encoded
  "CRC": "895A"
}
```

Step 3: Perform a Base64 decode to get encrypted data bytes.

Online tools can be used to get data bytes in hexadecimal format. For example:

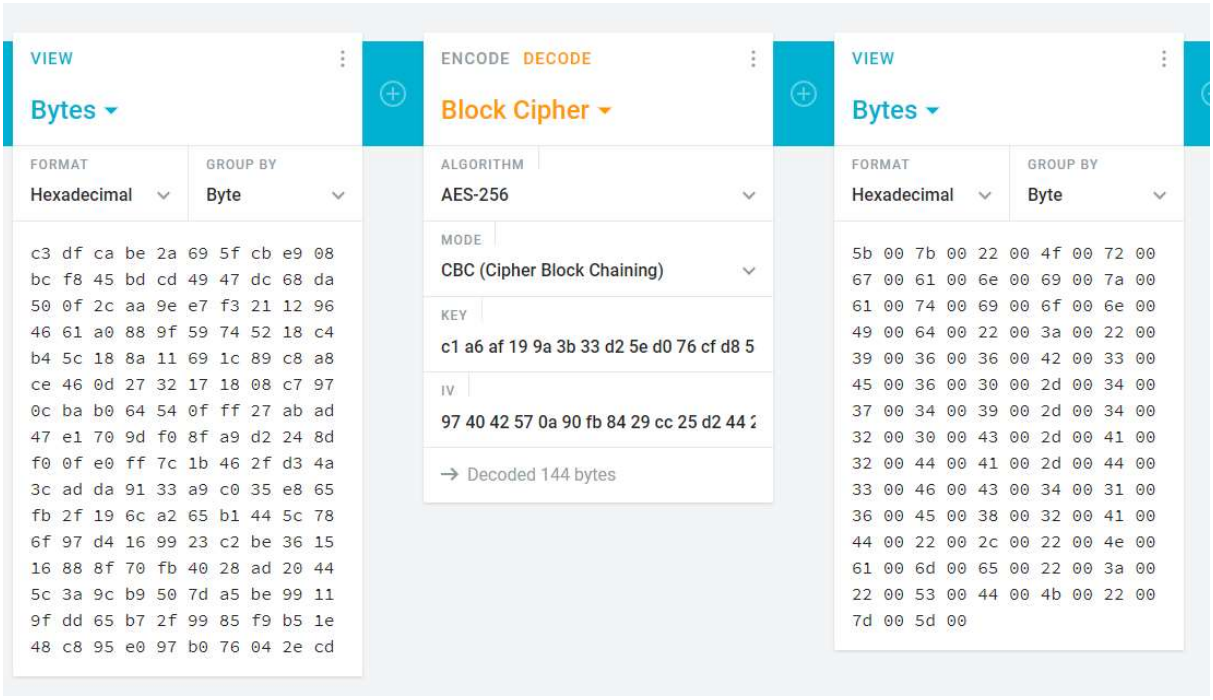
https://cryptii.com/pipes/base64-to-hex



Step 4: Decrypt data bytes obtained in Step 3 using AES_256_CBC algorithm. Use the “K” and “V” values from Step 1 for decryption.

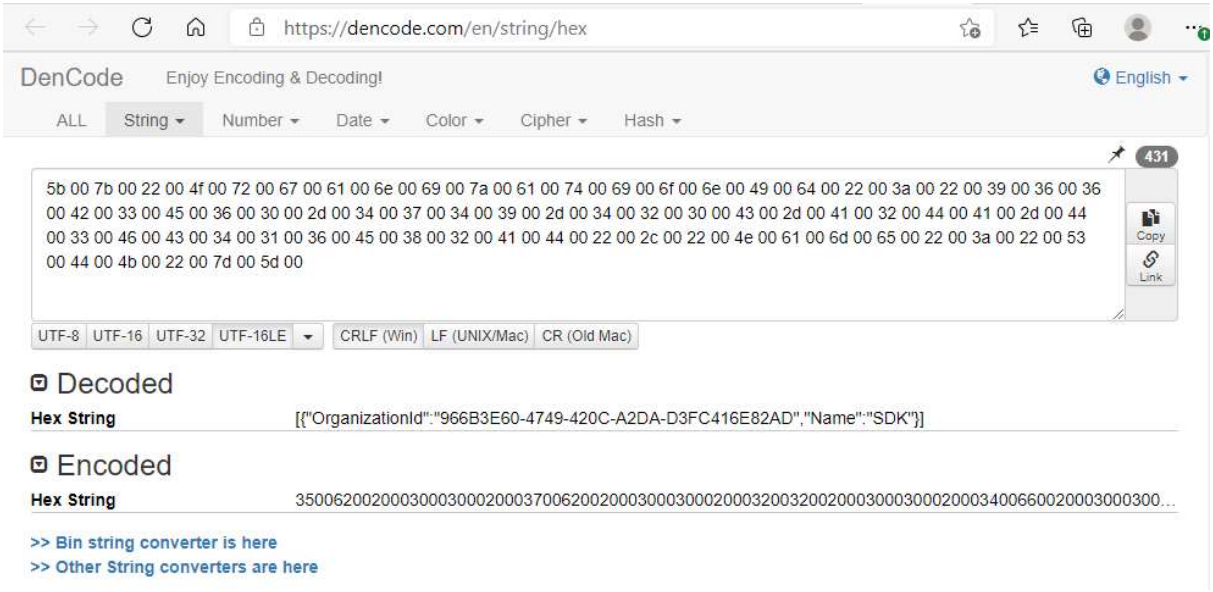
Online tools can be used to view decrypted data from the hexadecimal bytes obtained from the previous step. Example: https://cryptii.com/pipes/aes-encryption

Note that the Key and IV parameters are specified in the tool as hexadecimal bytes.



Step 5: Convert binary data obtained after decryption in Step 4 into a character string. The encoding used is UTF-16LE.

Online tools can be used to decode hexadecimal bytes obtained in Step 4 to get character string. Example: https://dencode.com/en/string/hex



7.6.2 PKI encrypted data

In PKI mode data is encrypted using public key provided by the customer. It is expected that corresponding private key is available with the customer. For the example below, consider the following private-public key pair.

Public key

```
-----BEGIN PUBLIC KEY-----
MIGeMA0GCSqGSIb3DQEBAQUAA4GMADCBiAKBgGJ5wG5sEg6qvo4XRZGgXe9OqixF
D04zbq3cbOLY/PLF5dJKEA6f9tJvzgVRNWzDOzQmcmossItJKCeptFO4XQZDkxk7
SHcqUuVlAptdnGzExRiAFBGzh2EYqxWQiYRYqNi3JZV0ESKP3JaMgzJr/VhhPjYj
PeJYmuZcUAW5HibvAgMBAAE=
-----END PUBLIC KEY-----
```

Private key

```
-----BEGIN RSA PRIVATE KEY-----
MIICWgIBAAKBgGJ5wG5sEg6qvo4XRZGgXe9OqixFD04zbq3cbOLY/PLF5dJKEA6f
9tJvzgVRNWzDOzQmcmossItJKCeptFO4XQZDkxk7SHcqUuVlAptdnGzExRiAFBGz
h2EYqxWQiYRYqNi3JZV0ESKP3JaMgzJr/VhhPjYjPeJYmuZcUAW5HibvAgMBAAEC
gYBHnX2ZkFSNgBT2puv8HSFTQOU3Lr4TxAmwvQYe72j9cXhSt3P3/tN0dMTiTw3I
XC75HSDGIFv/36MpPRycag6T8Y0CAUuecjb//y4PFJiTtEKoWAvTpshJANKqwPkZ
iuoYWTdUWgLlY78O+1NMbm2VUspzoN2guB2s3IQyNd7PkQJBAKQz1Hasq0IhOoDs
Gu1Vp8Ql4sCeKN7OvD8TJl+3J1+aMrg9+/1sz+Xm85HEDxx3m3ua/7tpU3sPvMw2
JKsD46cCQQCZh0l1+wktjmE+mq0tf0+vHyk6o5KAKsd3m3MNSUUANrS3eVMEdqjb
```

```
yq8fMc4Y0ON80YbWr8BBblL6zm+N8at5AkAAk4X9XKHFO6Se6zU0/AX4tEFVd7ig
/4y9k+2gjF4BAeOHrgACZa1dGzBjpJKBARJ0npijKb13QxjghpgeWYclAkBy/qJH
fICjXsH34mNUcMd+pBh9k4dn+KLvM8A5c2f7gP5bbsR/02EnP+lrtCJq9AtwsoAM
zhdy6u+8bGOiLiIBAkBN9LGi5K6WrFAdDxLY3q4cbnm3DDI6KxOpTvqrYdLdMzHv
3FgnkG6iGa6TtUC+vRpNh/OfooLhoXn78D4KlLZU
-----END RSA PRIVATE KEY-----
```

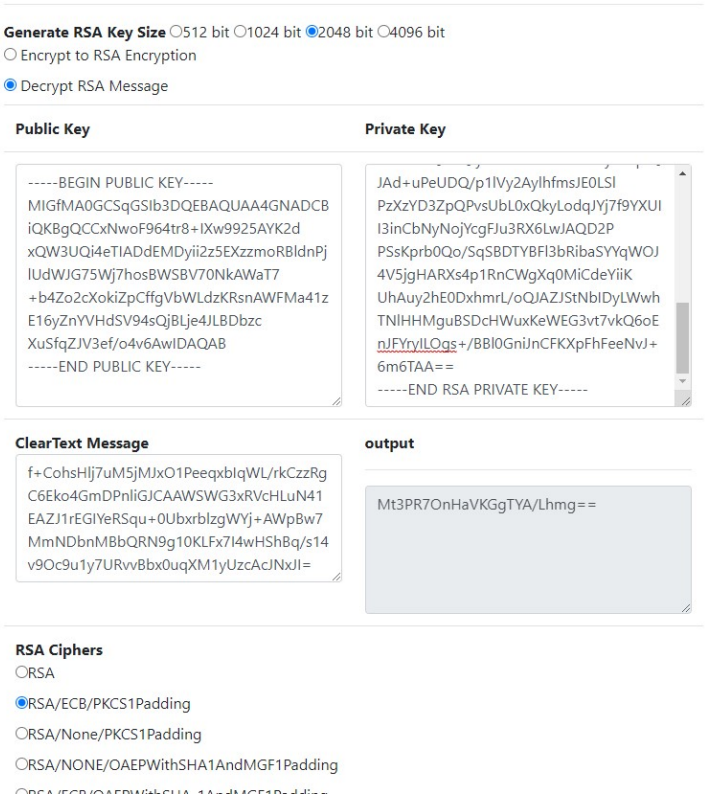
Note that in case of PKI encryption only “K” and “V” parameters in the data field of APILogin response are encrypted using PKI. All other data is AES encrypted, which can be decrypted using Key (K) and Initialization Vector (V) obtained from the Login response. Below are the steps to decrypt data when encryption mode is PKI.

Step 1: Decrypt “K” and “V” parameters contained in the “Data” field from APILogin response using the private key. Sample encrypted data from the APILogin call will look like as follows:

```
{
  “ErrorCode”: null,
  “IsSuccessful”: true,
  “DataType”:”real”,
  “Data”: {
    “Access_token”:”eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9…
                   0ZS5jb20ifQ.b011rw8mhoq0pTXY-
                   lK3v_dPI4nbZpG2YrF7jOWDzc”,
    “Token_type”:”bearer”,
    “Expires_in”:1800,
    “Refresh_token”:”cbd9b3ba-350d-4bc9-b4b8-d038c694d44a”,
    “Algorithm”:”AES_256_CBC”,
    “K”:”B97F…F12F6FB65F2720B59CCF”, //Encrypted with Public Key and Base64 encoded
    “V”:”7E8…C59A3B588306B13C31FBD”, //Encrypted with public key and Base64 encoded
    “VersionWebAPI”: “2.0.0”,
    “VersionCorsium”: “1.1.4.0”
  },
  “CRC”: “5tr9”
}
```

Following online tool can be used to decrypt data using RSA keys. The picture below shows decryption of (V) data using specified public and private keys.

https://8gwifi.org/RSAFunctionality?keysize=2048



Step 2: Once “K” and “V” values are obtained, rest of the messages can be decrypted using the steps described in “7.6.1 AES encrypted data”

7.7 Sample email attachments for ePCR accounts

7.7.1 Email #1 with attachment: WebAPISettings-<yyyyMMdd’T’HHmm>.txt

This file contains information as follows :

User Name : ohJl2WibbP+ph+qhxHCz3qeTeny19b55DTzt2lpNqoXuEo3Y3+Yy4cu1dTLl2sDe

Password : C0Jq/GRp8ag+MywNNz3DuOz2c5lOtpjD2RakusUp16M=

URL : https://corsium.info/WebAPI/webapitest

Double Encryption : Disabled

7.7.2 Email #2 with attachment: DecryptionKey-<yyyyMMdd’T’HHmm>.txt

This file contains information as follows:

Symmetric Key: PMZJvNa3WA8jkPj3cGZfD0TCkiG3oJS4aTCiYBHK2zc=

Initialization Vector: Tp8wHI1ZEIyb25F2LF5f5A==

7.8 Sample code for WebAPI requests in C#

For the purpose of this sample application following setup has been considered:

URL: https://corsium.info/WebAPI/sdkencr

Username: <random GUID, e.g. 686d4517-bd7f-4b8d-9ece-7a6ce799ad6f>

Password: 3P(RTe$t1n9

Encryption: AES

7.8.1 Main Program

`Program.cs` - This is the main program which handles all API requests in step-by-step manner.

```
using ISCAPIDataModel.Model;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.Collections.Generic;
using System.Text;

namespace APITester
{
    /// <summary>
    /// This is the main program to display complete working of WebAPI requests and responses
    /// </summary>
    class Program
    {
        static WebAPIParams WebAPIParams = new WebAPIParams();
        static void Main(string[] args)
        {
            //Initialize log file
            ApplicationLog.Initialize();

            //URL for the Test reference site
            WebAPIParams.WebAPIUrl = "https://corsium.info/WebAPI/sdkencr";
            WebAPIParams.Username = Guid.NewGuid().ToString();
            WebAPIParams.Password = "3P(RTe$t1n9";
            WebAPIParams.DEM = "AES";
            WebAPIParams.PrivateKey = null;
            WebAPIParams.PublicKey = null;

            //The following parameters can be used to test API call in PKI mode.
            //Uncomment the block of code below to call API with PKI mode of encryption
            /*
            WebAPIParams.DEM = "PKI";
            WebAPIParams.PublicKey = @"-----BEGIN PUBLIC KEY-----…wIDAQAB-----END PUBLIC KEY-----";
            WebAPIParams.PrivateKey = @"-----BEGIN RSA PRIVATE KEY-----eeN..6TAA==------END RSA
PRIVATE KEY-----";
            */

            //The following parameters can be used to test API call without encryption.
            //Uncomment the block of code below to call API without encryption
            /*
            WebAPIParams.WebAPIUrl = "https://corsium.info/WebAPI/sdk";
            WebAPIParams.DEM = "None";
            */

            WebAPIParams.AccessToken = "";
            WebAPIParams.RefreshToken = "";
            WebAPIParams.AesKey = "";
            WebAPIParams.AesVector = "";

            //Start Login - call API Login
            string responseData = WebAPIUtility.Login(WebAPIParams.WebAPIUrl, WebAPIParams.Username,
WebAPIParams.Password, WebAPIParams.DEM, WebAPIParams.PublicKey);
            dynamic responseObject = JObject.Parse(responseData);
            if (responseObject.IsSuccessful == true)
            {
                //Obtain token data and encryption key and vector from response
                var tokenData = responseObject["Data"];
                WebAPIParams.AccessToken = tokenData["Access_token"].ToString();
                WebAPIParams.RefreshToken = tokenData["Refresh_token"].ToString();
                WebAPIParams.AesKey = "";
                WebAPIParams.AesVector = "";
                if ("AES".Equals(WebAPIParams.DEM))
```

```
                {
                    WebAPIParams.AesKey = tokenData["K"].ToString();
                    WebAPIParams.AesVector = tokenData["V"].ToString();
                }
                // In case of PKI encryption mode Key and Vector parameters are encoded using public
                // key passed during APILogin request
                if ("PKI".Equals(WebAPIParams.DEM))
                {
                    WebAPIParams.AesKey = WebAPIUtility.DecryptPKI(WebAPIParams.PrivateKey,
tokenData["K"].ToString());
                    WebAPIParams.AesVector = WebAPIUtility.DecryptPKI(WebAPIParams.PrivateKey,
tokenData["V"].ToString());
                }
                //Token data is available.  Process Organization Data
                //Starting point of processing WebAPI requests
                ProcessOrganizationData();
            }
            else
            {
                //Something wrong with login request
                Console.Write("Error in response :" + responseObject["Code"]);
            }
        }

        /// <summary>
        /// Method to process customer account data
        /// </summary>
        private static void ProcessOrganizationData()
        {
            string responseData = WebAPIUtility.ListOrganizations(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken);
            responseData = WebAPIUtility.Decrypt(responseData, WebAPIParams.AesKey,
WebAPIParams.AesVector);
            Response<Organization[]> responseObject =
JsonConvert.DeserializeObject<Response<Organization[]>>(responseData);
            if (responseObject.IsSuccessful)
            {
                Organization[] allOrganizations = responseObject.Data;
                foreach (var organization in allOrganizations)
                {
                    //Process historic incidents
                    ProcessIncidents(organization);

                    //Process live incidents
                    ProcessLiveIncidents(organization);
                }
            }
            else
            {
                HandleErrorResponse(responseObject.ErrorCode);
            }
        }

        /// <summary>
        /// Method to process all incidents for an organization
        /// </summary>
        private static void ProcessIncidents(Organization organization)
        {
            //Till current time
            string toDate = DateTime.Now.ToString("yyyy-MM-ddTHH:mm:ss");
            //Get Incident data of last 23 hours
            string fromDate = DateTime.Now.AddHours(-23).ToString("yyyy-MM-ddTHH:mm:ss");

            //List incidents based on default parameters
            string responseData1 = WebAPIUtility.ListIncidents(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, organization.OrganizationId, fromDate, toDate, null, false, null);
            responseData1 = WebAPIUtility.Decrypt(responseData1, WebAPIParams.AesKey,
WebAPIParams.AesVector);
            Response<Incident[]> incidentData1 =
JsonConvert.DeserializeObject<Response<Incident[]>>(responseData1);
            ProcessIncidentResponse(incidentData1);

            //List incidents based using tempus serial number
            string responseData2 = WebAPIUtility.ListIncidents(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, organization.OrganizationId, fromDate, toDate, "888003", false, null);
            responseData2 = WebAPIUtility.Decrypt(responseData2, WebAPIParams.AesKey,
WebAPIParams.AesVector);
            Response<Incident[]> incidentData2 =
JsonConvert.DeserializeObject<Response<Incident[]>>(responseData2);
```

```
            ProcessIncidentResponse(incidentData2);

            //List incidents based on incident start time flag
            string responseData3 = WebAPIUtility.ListIncidents(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, organization.OrganizationId, fromDate, toDate, null, true, null);
            responseData3 = WebAPIUtility.Decrypt(responseData3, WebAPIParams.AesKey,
WebAPIParams.AesVector);
            Response<Incident[]> incidentData3 =
JsonConvert.DeserializeObject<Response<Incident[]>>(responseData3);
            ProcessIncidentResponse(incidentData3);

            //List incidents based on search parameter "SROC"
            string responseData4 = WebAPIUtility.ListIncidents(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, organization.OrganizationId, fromDate, toDate, null, false, "SROC");
            responseData4 = WebAPIUtility.Decrypt(responseData4, WebAPIParams.AesKey,
WebAPIParams.AesVector);
            Response<Incident[]> incidentData4 =
JsonConvert.DeserializeObject<Response<Incident[]>>(responseData4);
            ProcessIncidentResponse(incidentData4);
        }

        /// <summary>
        /// Method to process all live incidents for an organization
        /// </summary>
        private static void ProcessLiveIncidents(Organization organization)
        {
            //List incidents based on default parameters
            string responseData1 = WebAPIUtility.ListLiveIncidents(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, organization.OrganizationId, null);
            responseData1 = WebAPIUtility.Decrypt(responseData1, WebAPIParams.AesKey,
WebAPIParams.AesVector);
            Response<Incident[]> incidentData1 =
JsonConvert.DeserializeObject<Response<Incident[]>>(responseData1);
            ProcessLiveIncidentResponse(incidentData1);

            //List incidents based using tempus serial number
            string responseData2 = WebAPIUtility.ListLiveIncidents(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, organization.OrganizationId, "444001");
            responseData2 = WebAPIUtility.Decrypt(responseData2, WebAPIParams.AesKey,
WebAPIParams.AesVector);
            Response<Incident[]> incidentData2 =
JsonConvert.DeserializeObject<Response<Incident[]>>(responseData2);
            ProcessLiveIncidentResponse(incidentData2);
        }

        /// <summary>
        /// Method to process specific incident
        /// </summary>
        private static void ProcessIncidentResponse(Response<Incident[]> incidentData)
        {
            if (incidentData.IsSuccessful)
            {
                foreach (var incident in incidentData.Data)
                {
                    //Process Patient Data
                    ProcessPatientDetails(incident);
                    //Process Events
                    ProcessEvents(incident);
                    //Fetch Trended Vitals
                    ProcessTrends(incident);
                    //Fetch PDF
                    ProcessPDFData(incident);
                }
            }
            else
            {
                HandleErrorResponse(incidentData.ErrorCode);
            }
        }

        /// <summary>
        /// Method to process live incident
        /// </summary>
        private static void ProcessLiveIncidentResponse(Response<Incident[]> incidentData)
        {
            if (incidentData.IsSuccessful)
            {
                foreach (var incident in incidentData.Data)
                {
```

```
                    //Process Patient Data
                    ProcessPatientDetails(incident);
                    //Process Events
                    ProcessLiveEvents(incident);
                    //Fetch Trended Vitals
                    ProcessLiveTrends(incident);
                }
            }
            else
            {
                HandleErrorResponse(incidentData.ErrorCode);
            }
        }

        /// <summary>
        /// Method to process all events for an incident. Events can also be pulled based on event
type
        /// </summary>
        private static void ProcessEvents(Incident incident)
        {
            //Pull all events
            string responseData1 = WebAPIUtility.ListEvents(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, incident.IncidentId);
            responseData1 = WebAPIUtility.Decrypt(responseData1, WebAPIParams.AesKey,
WebAPIParams.AesVector);
            Response<EventWrapper> eventData1 =
JsonConvert.DeserializeObject<Response<EventWrapper>>(responseData1);
            ProcessEventResponse(incident, eventData1);

            //Pull only 12 Lead ECG Events
            string responseData2 = WebAPIUtility.ListEvents(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, incident.IncidentId, "TWELVE_LEAD");
            responseData2 = WebAPIUtility.Decrypt(responseData2, WebAPIParams.AesKey,
WebAPIParams.AesVector);
            Response<EventWrapper> eventData2 =
JsonConvert.DeserializeObject<Response<EventWrapper>>(responseData2);
            ProcessEventResponse(incident, eventData2);
        }

        /// <summary>
        /// Method to make a call to live events API for the current time
        /// </summary>
        /// <param name="incident">Live incident for which API is called</param>
        private static void ProcessLiveEvents(Incident incident)
        {
            string toDate = DateTime.Now.ToString("yyyy-MM-ddTHH:mm:ss");
            //Pull all events
            string responseData1 = WebAPIUtility.ListLiveEvents(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, incident.IncidentId, toDate);
            responseData1 = WebAPIUtility.Decrypt(responseData1, WebAPIParams.AesKey,
WebAPIParams.AesVector);
            Response<EventWrapper> eventData1 =
JsonConvert.DeserializeObject<Response<EventWrapper>>(responseData1);
            ProcessEventResponse(incident, eventData1);
        }
        /// <summary>
        /// Method to make a call to live vitals API for the current time
        /// </summary>
        /// <param name="incident">Live incident for which API is called</param>
        private static void ProcessLiveTrends(Incident incident)
        {
            string toDate = DateTime.Now.ToString("yyyy-MM-ddTHH:mm:ss");
            //Pull all trended vitals
            string responseData = WebAPIUtility.ListLiveVitals(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, incident.IncidentId, toDate);
            responseData = WebAPIUtility.Decrypt(responseData, WebAPIParams.AesKey,
WebAPIParams.AesVector);
            Response<VitalDetailsWrapper> responseObject =
JsonConvert.DeserializeObject<Response<VitalDetailsWrapper>>(responseData);
            ProcessTrendResponse(responseObject);
        }

        /// <summary>
        /// Method to process event response object
        /// </summary>
        private static void ProcessEventResponse(Incident incident, Response<EventWrapper>
eventData)
        {
            if (eventData.IsSuccessful)
```

```
            {
                foreach (var evt in eventData.Data.EventsInfo)
                {
                    ProcessEventData(incident, evt);
                }
            }
            else
            {
                HandleErrorResponse(eventData.ErrorCode);
            }
        }

        /// <summary>
        /// Method to handle all types of events
        /// </summary>
        private static void ProcessEventData(Incident incident, Event evt)
        {
            //Display only few details of the event such as event name and Date
            ApplicationLog.Write("Event Details-> Name: " + evt.EventLabel + ", Time: " +
evt.EventTimeUtc);
            if (evt.EventDataStatus == "Not Available") return;
            //Pull Event Data List
            if (evt.EventType == "CAMERA_IMAGE" || evt.EventType == "LARYNGOSCOPE_IMAGE")
            {
                string responseData = WebAPIUtility.GetImage(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, incident.IncidentId, evt.EventId);
                responseData = WebAPIUtility.Decrypt(responseData, WebAPIParams.AesKey,
WebAPIParams.AesVector);
                Response<Image> responseObject =
JsonConvert.DeserializeObject<Response<Image>>(responseData);
                if (responseObject.IsSuccessful)
                {
                    string fileName = "image-" + evt.EventId + ".jpg";
                    FileUtil.WriteFile(incident.IncidentId, fileName,
responseObject.Data.ImageData);
                }
                else
                {
                    HandleErrorResponse(responseObject.ErrorCode);
                }
            }
            if (evt.EventType == "TWELVE_LEADS")
            {
                //Get 12 lead data in JPEG format
                string responseData = WebAPIUtility.GetEcg(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, incident.IncidentId, evt.EventId, "JPEG");
                responseData = WebAPIUtility.Decrypt(responseData, WebAPIParams.AesKey,
WebAPIParams.AesVector);
                Response<TwelveLead> responseObject =
JsonConvert.DeserializeObject<Response<TwelveLead>>(responseData);
                if (responseObject.IsSuccessful)
                {
                    string fileName = "ecg-" + evt.EventId + ".jpg";
                    byte[] ecgData = Convert.FromBase64String(responseObject.Data.TwelveLeadData);
                    FileUtil.WriteFile(incident.IncidentId, fileName, ecgData);
                }
                else
                {
                    HandleErrorResponse(responseObject.ErrorCode);
                }
            }
            if (evt.EventType == "WAVEFORM"
                    || evt.EventType == "ARRHYTHMIA"
                    || evt.EventType == "PATIENT_ALARM"
                    || evt.EventType == "DEFIB_SHOCK"
                    || evt.EventType == "DEFIB_SNAPSHOT_WAVEFORMS"
                    || evt.EventType == "DEFIB_SHOCK_ADVISED"
                    || evt.EventType == "DEFIB_NO_SHOCK_ADVISED"
                    || evt.EventType == "DEFIB_PACING")
            {
                string responseData = WebAPIUtility.GetSnapshot(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, incident.IncidentId, evt.EventId);
                responseData = WebAPIUtility.Decrypt(responseData, WebAPIParams.AesKey,
WebAPIParams.AesVector);
                Response<Snapshot> responseObject =
JsonConvert.DeserializeObject<Response<Snapshot>>(responseData);
                if (responseObject.IsSuccessful)
                {
                    //Process Snapshot data;
```

```
                    string fileName = "snapshot-" + evt.EventId + ".jpg";
                    byte[] snapshotData =
Convert.FromBase64String(responseObject.Data.SnapshotData);
                    FileUtil.WriteFile(incident.IncidentId, fileName, snapshotData);
                }
                else
                {
                    HandleErrorResponse(responseObject.ErrorCode);
                }
            }
        }

        /// <summary>
        /// Method to handle PDF response
        /// </summary>
        private static void ProcessPDFData(Incident incident)
        {
            string responseData = WebAPIUtility.GetPdf(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, incident.IncidentId);
            responseData = WebAPIUtility.Decrypt(responseData, WebAPIParams.AesKey,
WebAPIParams.AesVector);
            Response<PDFResponse> responseObject =
JsonConvert.DeserializeObject<Response<PDFResponse>>(responseData);
            if (responseObject.IsSuccessful)
            {
                //Process PDF data;
                string fileName = incident.IncidentId + ".pdf";
                byte[] fileData = Convert.FromBase64String(responseObject.Data.Data);
                FileUtil.WriteFile(incident.IncidentId, fileName, fileData);
            }
            else
            {
                HandleErrorResponse(responseObject.ErrorCode);
            }
        }

        /// <summary>
        /// Method to process Trended vitals
        /// </summary>
        private static void ProcessTrends(Incident incident)
        {
            string responseData = WebAPIUtility.ListTrendedVitals(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, incident.IncidentId);
            responseData = WebAPIUtility.Decrypt(responseData, WebAPIParams.AesKey,
WebAPIParams.AesVector);
            Response<VitalDetailsWrapper> responseObject =
JsonConvert.DeserializeObject<Response<VitalDetailsWrapper>>(responseData);
            ProcessTrendResponse(responseObject);
        }
        /// <summary>
        /// Method to handle trended vital response
        /// </summary>
        /// <param name="responseObject"></param>
        private static void ProcessTrendResponse(Response<VitalDetailsWrapper> responseObject)
        {
            if (responseObject.IsSuccessful)
            {
                //Process Vitals data. Display one of the vital responses
                VitalDetailsWrapper vitalObject = responseObject.Data;
                if (vitalObject != null)
                {
                    List<VitalDetails> vitalDetails = vitalObject.VitalDetailsInfo;
                    if (vitalDetails.Count > 0)
                    {
                        VitalDetails details = vitalDetails[0];
                        StringBuilder buffer = new StringBuilder("First Vitals-> Time: " +
details.DataCaptureTimeUtc + ", ");
                        foreach (Vital vt in details.Vitals)
                        {
                            buffer.Append(vt.Name + ":" + vt.Value + ", ");
                        }
                        ApplicationLog.Write(buffer.ToString());
                    }
                }
            }
            else
            {
                HandleErrorResponse(responseObject.ErrorCode);
            }
```

```
        }
        //Method to process PatientDetails
        private static void ProcessPatientDetails(Incident incident)
        {
            string responseData = WebAPIUtility.GetPatientDetails(WebAPIParams.WebAPIUrl,
WebAPIParams.AccessToken, incident.IncidentId);
            responseData = WebAPIUtility.Decrypt(responseData, WebAPIParams.AesKey,
WebAPIParams.AesVector);
            Response<PatientDetails> responseObject =
JsonConvert.DeserializeObject<Response<PatientDetails>>(responseData);
            if (responseObject.IsSuccessful)
            {
                //Process Patient Details. This program simply writes one of the attributes to log
file
                PatientDetails details = responseObject.Data;
                ApplicationLog.Write("Patient Name :" + details.FirstName + " " + details.LastName);
            }
            else
            {
                HandleErrorResponse(responseObject.ErrorCode);
            }
        }

        /// <summary>
        /// Method to handle all types of errors
        /// </summary>
        private static void HandleErrorResponse(string errorCode)
        {
            if (errorCode == "002")
            {
                //authorization key is expired...request for another key
                //Refresh Token
            }

            if (errorCode == "010" || errorCode == "001")
            {
                //Re-authenticate to WebAPI
            }
            if (errorCode == "003")
            {
                //Sleep for some time and resend the request
            }
            if (errorCode == "008")
            {
                //Contact RDT support
            }
            if (errorCode == "005" || errorCode == "006" || errorCode == "007" || errorCode == "006"
                || errorCode == "013" || errorCode == "014" || errorCode == "015" || errorCode ==
"016"
                || errorCode == "017" || errorCode == "018")
            {
                //Correct request parameters and re-send the request
            }
            if (errorCode == "004" || errorCode == "009" || errorCode == "012")
            {
                //Data has been restricted and can't be obtained via WebAPI
            }
        }
    }
}
```

7.8.2 WebAPIUtility

`WebAPIUtility.cs` – This class provides methods to make different types of API calls.

```
using System;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Text;
using System.Web;
using ISCAPIDataModel.Model;
using Newtonsoft.Json;

namespace APITester
{
    /// <summary>
    /// This is a utility class to provide methods for different types of WebAPI calls
    /// </summary>
```

```
    public class WebAPIUtility
    {
        /// <summary>
        /// This method calls organization API and returns JSON Response
        /// </summary>
        /// <param name="WebAPIUrl">URL to call</param>
        /// <param name="TokenData">Token data obtained from login request</param>
        /// <returns>List of organizations in JSON format</returns>
        public static string ListOrganizations(string WebAPIUrl, string TokenData)
        {
            string urlToCall = WebAPIUrl + "/Api/Organizations";
            string responseData = RequestPoster.GetContent(urlToCall, TokenData);
            return responseData;
        }

        /// <summary>
        /// This method calls Incidents API for  specific Organization.
        /// This call also uses various parameters of the API
        /// </summary>
        /// <param name="WebAPIUrl">URL to call</param>
        /// <param name="TokenData">Token data obtained from login request</param>
        /// <param name="organizationId">Obtained from Organizations API call</param>
        /// <param name="fromDate">Start timestamp of the date range. Format 'yyyy-MM-
ddTHH:mm:ss'</param>
        /// <param name="toDate">End timestamp of the date range Format 'yyyy-MM-
ddTHH:mm:ss'</param>
        /// <param name="tempusSerialNumber">Serial no. of the Tempus device</param>
        /// <param name="useIncidentStartDate">Optional flag to indicate, if incidents should be
seached based on Incident start time</param>
        /// <param name="search">Optional parameter to search incidents based on keyword</param>
        /// <returns>List of incidents in JSON format</returns>
        public static string ListIncidents(string WebAPIUrl, string TokenData, string
organizationId, string fromDate, string toDate, string tempusSerialNumber=null, bool
useIncidentStartDate=false, string search=null)
        {
            string urlToCall = WebAPIUrl + "/Api/Clinical/Incidents?organizationId=" +
organizationId + "&fromDate=" + fromDate + "&toDate=" + toDate;
            if (tempusSerialNumber != null) urlToCall += "&tempusSerialNumber=" +
tempusSerialNumber;
            if (useIncidentStartDate) urlToCall += "&useIncidentStartDate=" + useIncidentStartDate;
            if (search != null) urlToCall += "&search=" + search;
            string responseData = RequestPoster.GetContent(urlToCall, TokenData);
            return responseData;
        }

        /// <summary>
        /// This method calls PatientDetails API for each Incident
        /// </summary>
        /// <param name="WebAPIUrl">URL to call</param>
        /// <param name="TokenData">Token data obtained from login request</param>
        /// <param name="IncidentId">Id of the incident for which API is called</param>
        /// <returns>Patient details in JSON format</returns>
        public static string GetPatientDetails(string WebAPIUrl, string TokenData, string
IncidentId)
        {
            //string encodedIncident = HttpUtility.UrlEncode(IncidentId);
            string urlToCall = WebAPIUrl + "/Api/Clinical/Incidents/" + IncidentId +
"/PatientDetails";
            string responseData = RequestPoster.GetContent(urlToCall, TokenData);
            return responseData;
        }

        /// <summary>
        /// This method calls TrendedVitalsAPI for each Incident
        /// </summary>
        /// <param name="WebAPIUrl">URL to call</param>
        /// <param name="TokenData">Token data obtained from login request</param>
        /// <param name="IncidentId">Id of the incident for which API is called</param>
        /// <returns>List of trended vitals in JSON format</returns>
        public static string ListTrendedVitals(string WebAPIUrl, string TokenData, string
IncidentId)
        {
            //string encodedIncident = HttpUtility.UrlEncode(IncidentId);
            string urlToCall = WebAPIUrl + "/Api/Clinical/Incidents/" + IncidentId +
                               "/TrendedVitals";
            string responseData = RequestPoster.GetContent(urlToCall, TokenData);
            return responseData;
        }
```

```
        /// <summary>
        /// This method calls Events API for each Incident
        /// </summary>
        /// <param name="WebAPIUrl">URL to call</param>
        /// <param name="TokenData">Token data obtained from login request</param>
        /// <param name="IncidentId">Id of the incident for which API is called</param>
        /// <param name="eventType">Optional parameter to get events based on their
eventType</param>
        /// <returns>List of events in JSON format</returns>
        public static string ListEvents(string WebAPIUrl, string TokenData, string IncidentId,
string eventType=null)
        {
            //string encodedIncident = HttpUtility.UrlEncode(IncidentId);
            string urlToCall = WebAPIUrl + "/Api/Clinical/Incidents/" + IncidentId +
                               "/Events";
            if (eventType != null) urlToCall += "?search=" + eventType;
            string responseData = RequestPoster.GetContent(urlToCall, TokenData);
            return responseData;
        }

        /// <summary>
        /// This method calls PDF API for each Incident
        /// </summary>
        /// <param name="WebAPIUrl">URL to call</param>
        /// <param name="TokenData">Token data obtained from login request</param>
        /// <param name="IncidentId">Id of the incident for which API is called</param>
        /// <returns>PDF response data in JSON format</returns>
        public static string GetPdf(string WebAPIUrl, string TokenData, string IncidentId)
        {
            string encodedIncident = HttpUtility.UrlEncode(IncidentId);
            string urlToCall = WebAPIUrl + "/Api/Clinical/Incidents/" + IncidentId + "/PDF";
            string responseData = RequestPoster.GetContent(urlToCall, TokenData);
            return responseData;
        }

        /// <summary>
        /// This method calls Image API for each Image event
        /// </summary>
        /// <param name="WebAPIUrl">URL to call</param>
        /// <param name="TokenData">Token data obtained from login request</param>
        /// <param name="IncidentId">Id of the incident for which API is called</param>
        /// <param name="EventId">Id of the Image event obtained from event list</param>
        /// <returns>Image response in JSON format</returns>
        public static string GetImage(string WebAPIUrl, string TokenData, string IncidentId, string
EventId)
        {
            string urlToCall = WebAPIUrl + "/Api/Clinical/Incidents/" + IncidentId +
"/Events/Image/" + EventId;
            string responseData = RequestPoster.GetContent(urlToCall, TokenData);
            return responseData;
        }

        /// <summary>
        /// This method calls Snapshot API call for each Waveform event
        /// </summary>
        /// <param name="WebAPIUrl">URL to call</param>
        /// <param name="TokenData">Token data obtained from login request</param>
        /// <param name="IncidentId">Id of the incident for which API is called</param>
        /// <param name="EventId">Id of the Snapshot event obtained from event list</param>
        /// <returns>Snapshot response in JSON format</returns>
        public static string GetSnapshot(string WebAPIUrl, string TokenData, string IncidentId,
string EventId)
        {
            string urlToCall = WebAPIUrl + "/Api/Clinical/Incidents/" + IncidentId +
"/Events/Snapshot/" + EventId;
            string responseData = RequestPoster.GetContent(urlToCall, TokenData);
            return responseData;
        }

        /// <summary>
        /// This method calls TweleveLead API for each 12Lead Event
        /// </summary>
        /// <param name="WebAPIUrl">URL to call</param>
        /// <param name="TokenData">Token data obtained from login request</param>
        /// <param name="IncidentId">Id of the incident for which API is called</param>
        /// <param name="EventId">Id of the Snapshot event obtained from event list</param>
        /// <param name="format">Format of the ECG data (JPEG, aECG, DICOM)</param>
        /// <returns>TwelveLead respnse in JSON format</returns>
```

```
        public static string GetEcg(string WebAPIUrl, string TokenData, string IncidentId, string
EventId, string format)
        {
            string urlToCall = WebAPIUrl + "/Api/Clinical/Incidents/" + IncidentId +
"/Events/TwelveLead/" +
                               EventId + "?Format=" + format;
            string responseData = RequestPoster.GetContent(urlToCall, TokenData);
            return responseData;
        }

        /// <summary>
        /// This mentod calls LiveIncidents api to get a list of live incidents
        /// </summary>
        /// <param name="WebAPIUrl">URL to call</param>
        /// <param name="TokenData">Token data obtained from login request</param>
        /// <param name="organizationId">Obtained from Organizations API call</param>
        /// <param name="tempusSerialNumber">Serial no. of the Tempus device</param>
        /// <returns></returns>
        public static string ListLiveIncidents(string WebAPIUrl, string TokenData, string orgId,
string tempusSerialNumber=null)
        {
            string urlToCall = WebAPIUrl + "/Api/Clinical/LiveIncidents?organizationId=" + orgId;
            if (tempusSerialNumber != null) urlToCall += "&tempusSerialNumber=" +
tempusSerialNumber;
            string responseData = RequestPoster.GetContent(urlToCall, TokenData);
            return responseData;
        }

        /// <summary>
        /// This method calls LiveVitals API to get live trended vitals in the last 2 minutes of
specified toDate
        /// </summary>
        /// <param name="WebAPIUrl">URL to call</param>
        /// <param name="TokenData">Token data obtained from login request</param>
        /// <param name="IncidentId">Id of the live incident obtained from the incident list</param>
        /// <param name="toDate">Live vitals to be queried up to this timestamp</param>
        /// <returns></returns>
        public static string ListLiveVitals(string WebAPIUrl, string TokenData, string IncidentId,
string toDate)
        {

            //string encodedIncident = HttpUtility.UrlEncode(IncidentId);
            string urlToCall = WebAPIUrl + "/Api/Clinical/Incidents/" + IncidentId +
"/LiveTrendedVitals?toDate=" + toDate;
            string responseData = RequestPoster.GetContent(urlToCall, TokenData);
            return responseData;
        }

        /// <summary>
        /// This method calls LiveEvents API to get live SROC events in the last 2 minutes of
specified toDate
        /// </summary>
        /// <param name="WebAPIUrl">URL to call</param>
        /// <param name="TokenData">Token data obtained from login request</param>
        /// <param name="IncidentId">Id of the live incident obtained from the incident list</param>
        /// <param name="toDate">Live vitals to be queried up to this timestamp</param>
        /// <returns></returns>
        public static string ListLiveEvents(string WebAPIUrl, string TokenData, string IncidentId,
string toDate)
        {
            //string encodedIncident = HttpUtility.UrlEncode(IncidentId);
            string urlToCall = WebAPIUrl + "/Api/Clinical/Incidents/" + IncidentId +
"/LiveEvents?toDate=" + toDate;
            string responseData = RequestPoster.GetContent(urlToCall, TokenData);
            return responseData;
        }

        /// <summary>
        /// This method is called to refresh token once it gets expired
        /// </summary>
        /// <param name="WebAPIUrl">URL to call</param>
        /// <param name="TokenData">Token data obtained from login request</param>
        /// <param name="CurrentRefreshToken">Refresh token obtained from the login response</param>
        /// <returns></returns>
        public static string RefreshToken(string WebAPIUrl, string TokenData, string
CurrentRefreshToken)
        {

            try
```

```
            {
                string urlToCall = WebAPIUrl + "/Account/APIRefreshToken";
                var reqParam = "{refreshToken:" + CurrentRefreshToken + "}";
                string responseData = RequestPoster.PostContent(urlToCall, reqParam, TokenData);
                return responseData;
            }
            catch (Exception ex)
            {
                Console.Write($"SDK-RefreshToken-exception : {ex}");
                return null;
            }
        }

        /// <summary>
        /// This method calls APILogin to authenticate WebAPI request and get authorization
        /// token along with encryption key
        /// </summary>
        /// <param name="webApiUrl">URL to call</param>
        /// <param name="Username">Username</param>
        /// <param name="Password">Password</param>
        /// <param name="dem"> Data Encryption Method ("None", "AES" or "PKI")</param>
        /// <param name="publicKey">Public key in RSA format if "PKI" is used in DEM</param>
        /// <returns></returns>
        public static string Login(string webApiUrl, string Username, string Password, string dem,
string publicKey)
        {

            try
            {
                var userRequest = "{\"Username\":\"" + Username + "\", \"Password\":\"" + Password +
"\", \"DEM\":\"" + dem + "\", \"Key\":\"" + publicKey + "\"}";
                string responseData = RequestPoster.PostContent(webApiUrl + "/Account/APILogin",
userRequest, null);
                return responseData;
            }
            catch (Exception ex)
            {
                Console.Write($"Login Exception: {ex}");
                throw;
            }
        }
        /// <summary>
        /// Decrypts the data part of the response object
        /// </summary>
        /// <param name="responseData">Response data in AES encrpted format</param>
        /// <param name="AESKey">Base64 encoded key obtained from response</param>
        /// <param name="AESVector">Base64 encoded initialization vector obtained from
response</param>
        /// <returns></returns>
        public static string Decrypt(String responseData, string AESKey, string AESVector)
        {
            //AESKey and AESVector are obtained after making APILogin request
            if (!string.IsNullOrEmpty(AESKey) && !string.IsNullOrEmpty(AESVector))
            {
                var response = JsonConvert.DeserializeObject<ResponseWithCRC<object>>(responseData);
                var data = response.Data.ToString();
                if (response.IsSuccessful)
                {
                    AesCryptoServiceProvider aes256 = new AesCryptoServiceProvider()
                    {
                        BlockSize = 128,
                        KeySize = 256,
                        IV = Convert.FromBase64String(AESVector),
                        Key = Convert.FromBase64String(AESKey),
                        Mode = CipherMode.CBC,
                        Padding = PaddingMode.PKCS7,
                    };
                    ICryptoTransform transform = aes256.CreateDecryptor();
                    var src = Convert.FromBase64String(data);
                    var dest = transform.TransformFinalBlock(src, 0, src.Length);
                    string decryptedData = Encoding.Unicode.GetString(dest);
                    var jsonSerializerSettings = new JsonSerializerSettings { DateParseHandling =
DateParseHandling.None };
                    var dataWithinResponse = JsonConvert.DeserializeObject<object>(decryptedData,
jsonSerializerSettings);
                    response.Data = dataWithinResponse;
                }
                return JsonConvert.SerializeObject(response);
            }
```

```
            return responseData;
        }

        /// <summary>
        /// Method to decrypt data using private key which was encrypted using public key
        /// </summary>
        public static string DecryptPKI(string pemPrivateKey, string data)
        {
            return PKIUtil.Decrypt(pemPrivateKey, data);
        }
    }
}
```

7.8.3 RequestPoster

This is a utility class to make Http Get and Post Requests to WebAPI URL.

RequestPoster.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;

namespace APITester
{
    /// <summary>
    /// This is a utility class to make get and post request to WebAPI url
    /// </summary>
    public class RequestPoster
    {
        /// <summary>
        /// This method sends GET request to fetch data from RESTful WebAPI.
        /// </summary>
        /// <param name="url">URL to call</param>
        /// <param name="tokenDataString">Token data for authentication and authorization</param>
        /// <returns>Response data</returns>
        public static string GetContent(string url, string tokenDataString)
        {
            ApplicationLog.Write("Request: " + url);
            //tokenDataString is obtained after making APILogin request
            WebClient client = new WebClient();
            client.Headers.Add("Authorization", "bearer " + tokenDataString);
            client.Headers.Add("Accept", "application/json");

            var response = client.DownloadData(url);
            string responseData = Encoding.UTF8.GetString(response);
            ApplicationLog.Write("Response: " + responseData);
            return responseData;
        }

        /// <summary>
        /// This method sends POST request to the WebAPI
        /// </summary>
        /// <param name="url">URL to call</param>
        /// <param name="postData">Data to be sent to the API</param>
        /// <param name="tokenDataString">Token data for authentication and authorization</param>
        /// <returns>Response data</returns>
        public static string PostContent(string url, string postData, string tokenDataString)
        {
            ApplicationLog.Write("Request: " + url+", Data: "+postData);
            var http = (HttpWebRequest)WebRequest.Create(new Uri(url));
            http.Accept = "application/json";
            http.ContentType = "application/json";
            // This complete format can also be used for Content-Type header. Uncomment the line
below to use it
            //http.ContentType = "application/json;charset=utf-8";
            http.Method = "POST";
            if (!String.IsNullOrEmpty(tokenDataString))
            {
                http.Headers.Add("Authorization", "bearer " + tokenDataString);
            }
            if (!String.IsNullOrEmpty(postData))
            {
```

```
                UTF8Encoding encoding = new UTF8Encoding();
                Byte[] bytes = encoding.GetBytes(postData);

                Stream newStream = http.GetRequestStream();
                newStream.Write(bytes, 0, bytes.Length);
                newStream.Close();
            }
            else
            {
                http.ContentLength = 0;
            }

            var response = http.GetResponse();
            var content = string.Empty;
            var stream = response.GetResponseStream();
            if (stream != null)
            {
                var sr = new StreamReader(stream);
                content = sr.ReadToEnd();
            }
            ApplicationLog.Write("Response: " + content);
            return content;
        }
    }
}
```

7.8.4 WebAPI Params

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace APITester
{
    /// <summary>
    /// Class to hold various WebAPI connecton attributes
    /// </summary>
    public class WebAPIParams
    {
        public string WebAPIUrl { get; set; }
        public string Username { get; set; }
        public string Password { get; set; }
        public string DEM { get; set; }
        public string PrivateKey { get; set; }
        public string PublicKey { get; set; }

        public string AccessToken { get; set; }
        public string RefreshToken { get; set; }
        public string AesKey { get; set; }
        public string AesVector { get; set; }
    }
}
```

7.8.5 Model Objects

These objects represent JSON data which is sent in the response

Allergy.cs

```
namespace ISCAPIDataModel.Model
{
    public class Allergy
    {
        public string Name { get; set; }
        public string Code { get; set; }
    }
}
```

Event.cs

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
```

```
namespace ISCAPIDataModel.Model
{
    /// <summary>
    /// Event Model class
    /// </summary>
    public class Event
    {
        public string EventId { get; set; }
        [JsonProperty(NullValueHandling = NullValueHandling.Ignore)]
        public DateTime? DataCaptureTimeUtc { get; set; }

        public string EventTimeUtc { get; set; }

        public bool HasVitals { get; set; }
        public string EventType { get; set; }
        public string EventLabel { get; set; }
        [JsonProperty(NullValueHandling = NullValueHandling.Ignore)]
        public Dictionary<string, object> EventDetails { get; set; }
        [JsonProperty(NullValueHandling = NullValueHandling.Ignore)]
        public string EventNotes { get; set; }
        public string EventDataStatus { get; set; }
        public string EventStatus { get; set; }
        [JsonProperty(NullValueHandling = NullValueHandling.Ignore)]
        public Vital[] Vitals { get; set; }
        [JsonProperty(NullValueHandling = NullValueHandling.Ignore)]
        public ReviewDetails ReviewDetails { get; set; }
    }
    /// <summary>
    /// Wrapper of Event
    /// </summary>
    public class EventWrapper
    {
        public string IncidentId { get; set; }
        public List<Event> EventsInfo { get; set; }
    }

}
```

Image.cs

```
namespace ISCAPIDataModel.Model
{
    /// <summary>
    ///
    /// </summary>
    public class Image
    {
        public byte[] ImageData { get; set; }
        public string EventId { get; set; }
        public string IncidentId { get; set; }
    }
}
```

Incident.cs

```
using System;
using Newtonsoft.Json;
namespace ISCAPIDataModel.Model
{
    /// <summary>
    /// Incident Model
    /// </summary>
    public class Incident
    {
        public string IncidentId { get; set; }
        public string TempusSerialNumber { get; set; }
        public string SendOrganizationName { get; set; }
        public DateTime StartTime { get; set; }
        [JsonProperty(NullValueHandling = NullValueHandling.Ignore)]
        public string PatientDetailsUpdateTime { get; set; }
        [JsonProperty(NullValueHandling = NullValueHandling.Ignore)]
        public string IncidentUpdateTime { get; set; }
        public string PatientId { get; set; }
        public bool DemoData { get; set; }
    }
}
```

Organization.cs

```
namespace ISCAPIDataModel.Model
{
    /// <summary>
    /// Organization model class
    /// </summary>
    public class Organization
    {
        public string OrganizationId { get; set; }
        public string Name { get; set; }
    }
}
```

PatientDetails.cs

```
using System.Collections.Generic;
using Newtonsoft.Json;

namespace ISCAPIDataModel.Model
{
    public class PatientDetails
    {

        public string IncidentId{ get; set; }
        public string PatientId { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string PatientAge { get; set; }
        public string PatientGender { get; set; }
        public string Weight { get; set; }
        public string WeightUnits { get; set; }
        public string Height { get; set; }
        public string HeightUnits { get; set; }
        public string TempusSerialNumber { get; set; }
        /// <summary>
        /// Gets or sets the Tempus Name.
        /// </summary>
        public string TempusName { get; set; }
        public string DeviceGroup { get; set; }
        public List<Allergy> Allergies { get; set; }
        public string StartTime { get; set; }
        public string PatientType { get; set; }
        public string TempusSoftware { get; set; }
        public string SupportCentre { get; set; }
        public string SendOrganizationName { get; set; }
        public string TempusOrganization { get; set; }
        public string IncidentUpdateTime { get; set; }
        public string PatientDetailsUpdateTime { get; set; }
        public string ReportType { get; set; }
        [JsonProperty(NullValueHandling = NullValueHandling.Ignore)]
        public string Service { get; set; }
        [JsonProperty(NullValueHandling = NullValueHandling.Ignore)]
        public string Unit { get; set; }
        [JsonProperty(NullValueHandling = NullValueHandling.Ignore)]
        public string Evac { get; set; }
        [JsonProperty(NullValueHandling = NullValueHandling.Ignore)]
        public string BattleRoster { get; set; }
    }
}
```

PDFResponse.cs

```
namespace ISCAPIDataModel.Model
{
    public class PDFResponse
    {
        public string IncidentId { get; set; }
        public string Name { get; set; }
        public string Data { get; set; }
    }
}
```

Response.cs

```
namespace ISCAPIDataModel.Model
{
    public class Response<T>
    {
```

```
        public string ErrorCode {get; set;}
        public bool IsSuccessful { get; set; }
        public string DataType { get; set; }
        public T Data { get; set; }
    }

    public class ResponseWithCRC<T>: Response<T>
    {
        [Newtonsoft.Json.JsonProperty(Order = 10)]
        public string CRC { get; set; }
    }

}
```

Snapshot.cs

```
namespace ISCAPIDataModel.Model
{
    public class Snapshot
    {
        public string SnapshotData { get; set; }
        public string EventId { get; set; }
        public string IncidentId { get; set; }
    }
}
```

TwelveLead.cs

```
namespace ISCAPIDataModel.Model
{
    public class TwelveLead
    {
        public string EventId { get; set; }
        public string IncidentId { get; set; }
        public string TwelveLeadData { get; set; }
        public string Format { get; set; }
    }
}
```

ReviewDetails.cs

```
namespace ISCAPIDataModel.Model
{
    public class ReviewDetails
    {
        public string ReviewStatus { get; set; }
        public string ReviewRequestSentTo { get; set; }
        public string ReviewRequestSentTimeUTC { get; set; }
        public string ReviewedBy { get; set; }
        public string ReviewTimeUtc { get; set; }
        public string ReviewDeliveryTimeUtc { get; set; }
        public string Diagnosis { get; set; }
        public string Instruction { get; set; }
        public string Notes { get; set; }
        public string AckStatus { get; set; }
        public string AckByTempusName { get; set; }
        public string AckByTempusNumber { get; set; }
        public string AckTimeUtc { get; set; }
        public string AckETA { get; set; }
    }
}
```

Vital.cs

```
namespace ISCAPIDataModel.Model
{
    public class Vital
    {
        public string Name { get; set; }
        public string Value { get; set; }
        public string Measurement { get; set; }
        public string Code { get; set; }
        public string Source { get; set; }
        public bool Alert { get; set; }
    }
}
```

VitalDetails.cs

```
using System.Collections.Generic;

namespace ISCAPIDataModel.Model
{
    public class VitalDetails
    {
        public string DataCaptureTimeUtc { get; set; }
        public List<Vital> Vitals { get; set; }
    }
    public class VitalDetailsWrapper
    {
        public string IncidentId { get; set; }
        public List<VitalDetails> VitalDetailsInfo { get; set; }
    }
}
```

7.8.6 Utility Classes

FileUtil.cs – This is a utility class to write files in specific format

```
using System;
using System.IO;

namespace APITester
{
    /// <summary>
    /// Utility class to write file in their specific format such as images, ecgs, snapshots and pdf
    /// </summary>
    class FileUtil
    {
        /// <summary>
        /// Method to write file data for an event
        /// </summary>
        /// <param name="incidentId">Incident for the file data event</param>
        /// <param name="fileName">Name of the file</param>
        /// <param name="data">File data</param>
        public static void WriteFile(string incidentId, string fileName, byte[] data)
        {
            try
            {
                string exePath =
Path.GetDirectoryName(System.Diagnostics.Process.GetCurrentProcess().MainModule.FileName);
                string fileDir = exePath + "\\logs\\filedata\\"+incidentId;
                if (!Directory.Exists(fileDir))
                {
                    Directory.CreateDirectory(fileDir);
                }
                File.WriteAllBytes(Path.Combine(fileDir, fileName), data);
            }
            catch(Exception ex)
            {
                ApplicationLog.Write("Unable to write data to file: " + ex.Message);
            }
        }
    }
}
```

ApplicationLog.cs – This is a utility class to log messages into a file.

```
using System;
using System.IO;

namespace APITester
{
    /// <summary>
    /// Utility to log messages to files
    /// </summary>
    public class ApplicationLog
    {
        private static string LogFile=null;
```

```
        /// <summary>
        /// Initialization of the log class
        /// </summary>
        public static void Initialize()
        {
            string exePath =
Path.GetDirectoryName(System.Diagnostics.Process.GetCurrentProcess().MainModule.FileName);
            string LogDir = exePath + "\\logs";
            if (!Directory.Exists(LogDir))
            {
                Directory.CreateDirectory(LogDir);
            }
            LogFile = Path.Combine(LogDir, "WebAPI-" + DateTime.Now.ToString("yyyyMMddHHmmss") +
".log");
        }

        /// <summary>
        /// Write message to the log file
        /// </summary>
        /// <param name="message">Message to be written</param>
        public static void Write(string message)
        {
            using (StreamWriter writer = new StreamWriter(LogFile, true))
            {
                writer.WriteLine(DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss.fff") + " " + message);
            }
        }
    }
}
```

`PKIUtil.cs` – This class provides methods to decrypt and encrypt data using public key infrastructure

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;

namespace APITester
{
    class PKIUtil
    {
        /// <summary>
        /// Encrypt data based on the provided public key
        /// </summary>
        /// <param name="pemPublicKey"></param>
        /// <param name="data"></param>
        /// <returns>encrypted data</returns>
        public static string Encrypt(string pemPublicKey, string data)
        {
            var rsa = GetRSACryptoServiceProviderFromPEMPublicKey(pemPublicKey);
            var dataToEncrypt = Encoding.UTF8.GetBytes(data);
            var encryptedBytes = rsa.Encrypt(dataToEncrypt, false);
            return Convert.ToBase64String(encryptedBytes);
        }

        /// <summary>
        /// Decrypt the data based on the supplied private key
        /// </summary>
        /// <param name="pemPrivateKey"></param>
        /// <param name="data"></param>
        /// <returns>decrypted data</returns>
        public static string Decrypt(string pemPrivateKey, string data)
        {
            var rsa = GetRSACryptoServiceProviderFromPEMPrivateKey(pemPrivateKey);
            var dataBytes = Convert.FromBase64String(data);
            var decryptedBytes = rsa.Decrypt(dataBytes, false);
            return Encoding.UTF8.GetString(decryptedBytes);
        }

        internal static RSACryptoServiceProvider GetRSACryptoServiceProviderFromPEMPublicKey(string
pemPublicKey)
        {
            byte[] seqOid = { 0x30, 0x0D, 0x06, 0x09, 0x2A, 0x86, 0x48, 0x86, 0xF7, 0x0D, 0x01,
0x01, 0x01, 0x05, 0x00 };
            var publicKeyBytes = GetBytesFromPEM(pemPublicKey, "PUBLIC KEY");
            using (var mem = new MemoryStream(publicKeyBytes))
            {
```

```
using (var binr = new BinaryReader(mem))
{
    try
    {
        var twobytes = binr.ReadUInt16();
        switch (twobytes)
        {
            case 0x8130:
                binr.ReadByte();
                break;
            case 0x8230:
                binr.ReadInt16();
                break;
            default:
                return null;
        }

        var seq = binr.ReadBytes(15);
        if (!CompareBytearrays(seq, seqOid))
            return null;

        twobytes = binr.ReadUInt16();
        if (twobytes == 0x8103)
            binr.ReadByte();
        else if (twobytes == 0x8203)
            binr.ReadInt16();
        else
            return null;

        var bt = binr.ReadByte();
        if (bt != 0x00)
            return null;

        twobytes = binr.ReadUInt16();
        if (twobytes == 0x8130)
            binr.ReadByte();
        else if (twobytes == 0x8230)
            binr.ReadInt16();
        else
            return null;

        twobytes = binr.ReadUInt16();
        byte lowbyte;
        byte highbyte = 0x00;

        if (twobytes == 0x8102)
            lowbyte = binr.ReadByte();
        else if (twobytes == 0x8202)
        {
            highbyte = binr.ReadByte();
            lowbyte = binr.ReadByte();
        }
        else
            return null;
        byte[] modint = { lowbyte, highbyte, 0x00, 0x00 };
        int modsize = BitConverter.ToInt32(modint, 0);

        byte firstbyte = binr.ReadByte();
        binr.BaseStream.Seek(-1, SeekOrigin.Current);

        if (firstbyte == 0x00)
        {
            binr.ReadByte();
            modsize -= 1;
        }

        byte[] modulus = binr.ReadBytes(modsize);

        if (binr.ReadByte() != 0x02)
            return null;
        int expbytes = binr.ReadByte();
        byte[] exponent = binr.ReadBytes(expbytes);

        RSACryptoServiceProvider rsa = new RSACryptoServiceProvider();
        RSAParameters rsaKeyInfo = new RSAParameters
        {
            Modulus = modulus,
            Exponent = exponent
        };
```

```
                    rsa.ImportParameters(rsaKeyInfo);
                    return rsa;
                }
                catch (Exception)
                {
                    return null;
                }
            }
        }
        public static RSACryptoServiceProvider GetRSACryptoServiceProviderFromPEMPrivateKey(string
pemPrivateKey)
        {
            var privateKeyBytes = GetBytesFromPEM(pemPrivateKey, "RSA PRIVATE KEY");
            MemoryStream ms = new MemoryStream(privateKeyBytes);
            BinaryReader rd = new BinaryReader(ms);
            try
            {
                var shortValue = rd.ReadUInt16();
                switch (shortValue)
                {
                    case 0x8130:
                        rd.ReadByte();
                        break;
                    case 0x8230:
                        rd.ReadInt16();
                        break;
                    default:
                        return null;
                }

                shortValue = rd.ReadUInt16();
                if (shortValue != 0x0102)
                {
                    return null;
                }

                var byteValue = rd.ReadByte();
                if (byteValue != 0x00)
                {
                    return null;
                }
                CspParameters parms = new CspParameters();
                parms.Flags = CspProviderFlags.UseMachineKeyStore;

                RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(parms);
                RSAParameters rsAparams = new RSAParameters();

                rsAparams.Modulus = rd.ReadBytes(DecodeIntegerSize(rd));

                RSAParameterTraits traits = new RSAParameterTraits(rsAparams.Modulus.Length * 8);

                rsAparams.Modulus = AlignBytes(rsAparams.Modulus, traits.size_Mod);
                rsAparams.Exponent = AlignBytes(rd.ReadBytes(DecodeIntegerSize(rd)),
traits.size_Exp);
                rsAparams.D = AlignBytes(rd.ReadBytes(DecodeIntegerSize(rd)), traits.size_D);
                rsAparams.P = AlignBytes(rd.ReadBytes(DecodeIntegerSize(rd)), traits.size_P);
                rsAparams.Q = AlignBytes(rd.ReadBytes(DecodeIntegerSize(rd)), traits.size_Q);
                rsAparams.DP = AlignBytes(rd.ReadBytes(DecodeIntegerSize(rd)), traits.size_DP);
                rsAparams.DQ = AlignBytes(rd.ReadBytes(DecodeIntegerSize(rd)), traits.size_DQ);
                rsAparams.InverseQ = AlignBytes(rd.ReadBytes(DecodeIntegerSize(rd)),
traits.size_InvQ);

                rsa.ImportParameters(rsAparams);
                return rsa;
            }
            catch (Exception ex)
            {
                ApplicationLog.Write(ex.Message);
                throw;
            }
            finally
            {
                rd.Close();
            }
        }

        private static bool CompareBytearrays(byte[] a, byte[] b)
        {
```

```
        if (a.Length != b.Length)
            return false;
        int i = 0;
        foreach (byte c in a)
        {
            if (c != b[i])
                return false;
            i++;
        }
        return true;
    }

    private static byte[] GetBytesFromPEM(string pemString, string section)
    {
        var header = $"-----BEGIN {section}-----";
        var footer = $"-----END {section}-----";
        var start = pemString.IndexOf(header, StringComparison.Ordinal);
        if (start >= 0)
        {
            start += header.Length;
        }
        else
        {
            start = 0;
        }
        var end = pemString.Length - footer.Length;
        if (end < 0)
            end = pemString.Length;
        pemString = pemString.Substring(start, end - start).Replace(" ", string.Empty);
        return Convert.FromBase64String(pemString);
    }

    private static byte[] AlignBytes(byte[] inputBytes, int alignSize)
    {
        int inputBytesSize = inputBytes.Length;

        if ((alignSize != -1) && (inputBytesSize < alignSize))
        {
            byte[] buf = new byte[alignSize];
            for (int i = 0; i < inputBytesSize; ++i)
            {
                buf[i + (alignSize - inputBytesSize)] = inputBytes[i];
            }
            return buf;
        }
        else
        {
            return inputBytes;
        }
    }
    private static int DecodeIntegerSize(BinaryReader rd)
    {
        byte byteValue;
        int count;

        byteValue = rd.ReadByte();
        if (byteValue != 0x02)
            return 0;

        byteValue = rd.ReadByte();
        if (byteValue == 0x81)
        {
            count = rd.ReadByte();
        }
        else if (byteValue == 0x82)
        {
            byte hi = rd.ReadByte();
            byte lo = rd.ReadByte();
            count = BitConverter.ToUInt16(new[] { lo, hi }, 0);
        }
        else
        {
            count = byteValue;
        }
        while (rd.ReadByte() == 0x00)
        {
            count -= 1;
        }
        rd.BaseStream.Seek(-1, SeekOrigin.Current);
```

```
            return count;
        }

    }

    public class RSAParameterTraits
    {
        public RSAParameterTraits(int modulusLengthInBits)
        {
            int assumedLength;
            double logbase = Math.Log(modulusLengthInBits, 2);
            if (logbase == (int)logbase)
            {
                assumedLength = modulusLengthInBits;
            }
            else
            {
                assumedLength = (int)(logbase + 1.0);
                assumedLength = (int)(Math.Pow(2, assumedLength));
                System.Diagnostics.Debug.Assert(false);
            }
            switch (assumedLength)
            {
                case 1024:
                    size_Mod = 0x80;
                    size_Exp = -1;
                    size_D = 0x80;
                    size_P = 0x40;
                    size_Q = 0x40;
                    size_DP = 0x40;
                    size_DQ = 0x40;
                    size_InvQ = 0x40;
                    break;
                case 2048:
                    size_Mod = 0x100;
                    size_Exp = -1;
                    size_D = 0x100;
                    size_P = 0x80;
                    size_Q = 0x80;
                    size_DP = 0x80;
                    size_DQ = 0x80;
                    size_InvQ = 0x80;
                    break;
                case 4096:
                    size_Mod = 0x200;
                    size_Exp = -1;
                    size_D = 0x200;
                    size_P = 0x100;
                    size_Q = 0x100;
                    size_DP = 0x100;
                    size_DQ = 0x100;
                    size_InvQ = 0x100;
                    break;
                default:
                    System.Diagnostics.Debug.Assert(false);
                    break;
            }
        }
        public int size_Mod = -1;
        public int size_Exp = -1;
        public int size_D = -1;
        public int size_P = -1;
        public int size_Q = -1;
        public int size_DP = -1;
        public int size_DQ = -1;
        public int size_InvQ = -1;
    }
}
```

Remote Diagnostic Technologies Ltd
Ascent 1
Farnborough Aerospace Centre
Aerospace Boulevard
Farnborough
Hampshire
GU14 6XW
UK

Tel: +44 (0) 1256 362 400
www.philips.com

Part number: 42-4004EN-02