MEDVANTAGE

API – TERMS

FOR

PATIENT SELECTION / DATA CATEGORY REQUEST / ALL DATA REQUEST

170.315(G)(7), 170.315(G)(8), 170.315(G)(9)

Table of Content

Overview

This API document provides the information on how the clients can retrieve details for specific patients once they discontinue the service from MedVanatge.

When the clients use EHR application other than MedVanatge, they may want to view patient details that are recorded in the MedVanatge application for further treatment. In this case, MedVanatge provides the API documentation along with URL and credentials. The users can access the URL and enter the MedVanatge ID, Username, and Password to retrieve patient details.

When extracting patient data, the system will search for parameters such as First Name, Last Name, DOB, and Sex to identify unique patient details.

The users can,

- Extract Patient data for specific date or for a specific date range.
- Extract patient's basic and clinical details such as Race, Ethnicity, Preferred Language, Vitals, Medications, Allergies, Lab Tests, Immunizations, and so forth.
- Download C-CDA file while extracting the data which can be used to display the patient details in another application as per their needs.

API Details

This guide provides information about the following three Web APIs (Application Program Interfaces) required for QPP (Quality Payment Program) Measure “Provide Patient Access”.

- Patient Selection
- Data Category Request
- All Data Request

A third-party application can use these APIs to retrieve a unique patient identifier based on patient identification. Patient identification can be First Name, Last Name, Gender, and Date of Birth. This identifier can be further used to get CCDS (Common Clinical Data Set) for a specific patient. You can filter the CCDS content by Date and/or Data Category.

Quality Payment Program (QPP) APIs are web APIs which are used to retrieve the patient information.

Regulation Text Citation	Certification Criterion
§ 170.315(g)(7)	Application access – patient selection
§ 170.315(g)(8)	Application access – data category request
§ 170.315(g)(9)	Application access – all data request

Key Information To Share

Before starting the any development on using MedVantage API, MedVantage will share credentials which are used to communicate with MedVantage.

MedVantage will share two different credentials that are

- Staging (Used to start development on implementing the API)
- Production (Once implementation completed MedVantage will share another credential for production use)

Credentials are as below.

- ClientID
- Username
- Password

These credentials are used when user start connecting to MedVantage for to fetch patient information.

API Syntax and Function Name

Patient Search

API Syntax

Ex: https://{YourSiteName.com}/fhir/get_fhir_Patient

Function Name:

SearchPatient

Required parameters and their data types:

**Only one parameter is required*

Parameter Name	Parameter Value	Data Type
id (optional)	Parameter	string
identifier (optional)	Parameter	string
name (optional)	Parameter	string
birthdate (optional)	Parameter	string
gender (optional)	Parameter	string
address (optional)	Parameter	string
addresscity (optional)	Parameter	string
addresspostalcode (optional)	Parameter	string
addressstate (optional)	Parameter	string
email (optional)	Parameter	string
family (optional)	Parameter	string
given (optional)	Parameter	string
Phone (optional)	Parameter	string
Telecom (optional)	Parameter	string

Return variables and their types/structures:

Sample data table and structure:

Return Variable Name	Return Variable Value	Data Type
resourceType	Parameter	Collection(FHIR-bundle)

Exceptions and exception handling methods and their returns:

Sample data table and structure:

Exception Type	Exception Value	Data Type	Output
Bad Request	400	String	Invalid Argument
Unauthorized	401	String	Authorization denied

Encounter Details

Get Encounter Details by Patient UUID

API Syntax: https://{YourSiteName.com}/get_api_patient__puuid__encounter

Function Name:

GetEncounterDetailsByPatientUuid

Required parameters and their data types:

Parameter Name	Parameter Value	Data Type
patientUuid	Parameter	String

Return variables and their types/structures:

Return Variable Name	Return Variable Value	Data Type
validationErrors	Parameter	Array
Error_description	Parameter	Array
data	Parameter	Array (any type of data)

API Header Authentication:

Return Variable Name	Return Variable Value	Data Type
Content-Type	application/json	String

Exceptions and exception handling methods and their returns:

Exception Type	Exception Value	Data Type	Output
Null Exception Handling	-	String	Json format
Error Exception Handling	-	String	Json format

View File XML/JSON

API Syntax: https://{YourSiteName.com}/get_fhir_Patient__export

Function Name:

ExportPatientData

Required parameters and their data types:

Parameter Name	Parameter Value	Data Type	Mandatory / Optional	
UserId	Parameter	Integer	Mandatory	
Token	Parameter	String	Mandatory	
isXML	Parameter	Boolean	Mandatory	
isJSON	Parameter	Boolean	Mandatory	
uhId	Parameter	Integer	Mandatory	
SelectedElement	Parameter	Selection List	Mandatory	Refer to the Section Elements below the table for key values
fromDate	Parameter	String	Optional	
toDate	Parameter	String	Optional	

Selection List Elements (Key Value)

If you want to view patient name, then, key value for patient name i.e. 1 should be true, for example,

{"Key":"1","Selected":true}

Clinical Elements	Key	Clinical Elements	Key
Patient Name	1	Laboratory Tests	11
Sex	2	Laboratory Values(s)/Result(s)	12
Date of Birth	3	Vital Signs	13
Race	4	Procedures	14
Ethnicity	5	Care Team Member(s)	15
Preferred Language	6	Immunizations	16
Smoking Status	7	Unique Device Identifier(s) for a Patient's Implantable Device(s)	17
Problems	8	Assessment and Plan of Treatment	18
Medications	9	Goals	19
Medication Allergies	10	Health Concerns	20

API Header Authentication:

Return Variable Name	Return Variable Value	Data Type
Content-Type	application/json	String
token	Parameter	String
uhid	Parameter	Integer

Exceptions and exception handling methods and their returns:

Exception Type	Exception Value	Data Type	Output
Null Exception Handling	-	String	Json format
Error Exception Handling	-	String	Json format

Return variables and their types/structures: Xml File / Json File

USCDI Data Export

API Syntax: https://{YourSiteName.com}/GetExportpatientUSCDIData

Parameter Name	Parameter Value	Data Type	Mandatory / Optional
UHID	Parameter	Integer	Optional
fromDate	Parameter	Integer	Optional
toDate	Parameter	String	Optional
otp	Parameter	Integer	Optional
mobileNo	Parameter	Integer	Optional

Function Name:

ExportPatientData

Return variable and their types/structure:

Return Variable Name	Return Variable Value	Data Type
uhId	Parameter	string
patientName	Parameter	string
sex	Parameter	string
dob	Parameter	string
raceType	Parameter	string
languageName	Parameter	string
ethinicityName	Parameter	string
problems	Parameter	string
medications	Parameter	string
procedures	Parameter	string
teamMember	Parameter	string
immunization	Parameter	string
healthConcerns	Parameter	string
vitals	Parameter	string
smokingStatus	Parameter	string
implantableDevice	Parameter	string array
goals	Parameter	string
planOfTreatment	Parameter	string
medicineAllergy	Parameter	string
labTest	Parameter	string
labResult	Parameter	string

Exception Type	Exception Value	Data Type	Output
Null Exception Handling	-	String	Json format
Error Exception Handling	-	String	Json format

Return variable and their types/structure: XML File/JSON File

C-CDA Data Export

API Syntax: https://{YourSiteName.com}/GetExportpatientCCDAData

Parameter Name	Parameter Value	Data Type	Mandatory / Optional
UHID	Parameter	Integer	Optional
fromDate	Parameter	Integer	Optional
toDate	Parameter	String	Optional
otp	Parameter	Integer	Optional
mobileNo	Parameter	Integer	Optional

Return variable and their types/structure:

Return Variable Name	Return Variable Value	Data Type
uhId	Parameter	string
patientName	Parameter	string
sex	Parameter	string
dob	Parameter	string
allergies	Parameter	string
chiefComplaint	Parameter	string
familyHistory	Parameter	string
immunization	Parameter	string
medications	Parameter	string
problems	Parameter	string
referralReason	Parameter	string
vitalSign	Parameter	string
socialHistory	Parameter	string
result	Parameter	string
procedures	Parameter	string
planofCare	Parameter	string
instruction	Parameter	string
planOfTreatment	Parameter	string
functonalAndCognitiveStatus	Parameter	string
advacnedDirectives	Parameter	string
payers	Parameter	string

medicalEquipment	Parameter	string
encounters	Parameter	string
assessment	Parameter	string
historyOfPresentIllness	Parameter	string
physicalExam	Parameter	string
generalStatus	Parameter	string
historyOfPastIllness	Parameter	string
reviewOfStatus	Parameter	string
dICOMObjectCatalog	Parameter	string
findings	Parameter	string
hospitalCourse	Parameter	string
hospitalDischargeDiagnosis	Parameter	string
hospitalDischargeMedications	Parameter	string
anesthesia	Parameter	string
raceType	Parameter	string
languageName	Parameter	string
ethincityName	Parameter	string
complications	Parameter	string
postoperativeDiagnosis	Parameter	string
preoperativeDiagnosis	Parameter	string
procedureDisposition	Parameter	string
procedureEstimatedBloodLoss	Parameter	string
procedureFindings	Parameter	string
procedureIndications	Parameter	string
procedureSpecimensTaken	Parameter	string
postprocedureDiagnosis	Parameter	string
medicationsAdministered	Parameter	string
socialHistoryNew	Parameter	string

Exception Type	Exception Value	Data Type	Output
Null Exception Handling	-	String	Json format
Error Exception Handling	-	String	Json format

SOFTWARE COMPONENTS AND CONFIGURATIONS

This section includes the software components and configurations that would be necessary for an application to implement to be able to successfully interact with the API and process its response(s).

PATIENT SEARCH (EXAMPLE OF C# CODE)

```
[HttpGet(nameof(GetPatientDetailsByUHID))]
 public async Task<IActionResult> GetPatientDetailsByUHID([FromQuery]
int? Pmid, string? UHID)
 {
    try
    {
        (var result, bool IsException, string ExceptionMessage) = await
_patientPersonalDashboardService.GetPatientDetailsByUHID(Pmid, UHID);

        if (!IsException)
        {
            if (result is not null && result.Tables.Count > 0)
            {
                return Ok(new
                {
                    status = 1,
                    message = "success",
                    responseValue = result.Tables[0]
                });
            }
```

```
 else
 {
 return BadRequest(new
 {
 status = 0,
 message = "failure",
 responseValue = "No record found"
 });
 }}
else
        {
            return StatusCode(StatusCodes.Status500InternalServerError,
new
            {
                status = 0,
                message = "failure",
                responseValue = ExceptionMessage
            });
        }

    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message, nameof(Controllers));
        return StatusCode(StatusCodes.Status500InternalServerError, new
        {
            status = 0,
            message = "failure",
            responseValue = ex.InnerException?.Message ?? ex.Message
        });
    }
 }
```

Get Encounter Details by Patient UUID (EXAMPLE OF C# CODE)

```
[Route("api/[controller]")]
 [Http Post]
 public class EncounterController : ControllerBase
 {
   private readonly IEncounterService encounterService;

   public EncounterController(IEncounterService encounterService)
   {
       this.encounterService = encounterService;
   }

   [HttpGet]
   [Route("getAll")]
   public IActionResult GetAll(string puuid)
   {
       var processingResult = encounterService.Search(new List<string>
(), true, puuid);

       if (!processingResult.HasErrors && processingResult.Data.Count ==
0)
       {
           return NotFound();
       }

       return Ok(processingResult);
   }
 }
```

PATIENT DATA EXPORT USCDI (EXAMPLE OF C# CODE)

```
[HttpGet(nameof(GetExportPatientUSCDIData))]
        public async Task<IActionResult>
GetExportPatientUSCDIData([FromQuery] ExportPatientData pobj)
        {
            try
            {
                //(var result, bool IsException, string ExceptionMessage)
= await _ExportPatientDataService.GetExportPatientData(pobj);
                var getExportPatientDataTask =
_ExportPatientDataService.GetExportPatientData(pobj);
                string ApiUrlUser = UserServiceAPIFactory.BaseUrl +
UserServiceAPIFactory.GetUserList;
                // string ApiUrlLab = LabServicesAPIFactory.BaseUrl +
LabServicesAPIFactory.GetTestResultListByUhid;
                //var UserResponse = await
APIExecuter.HttpGetAsync(ApiUrlUser ?? string.Empty);
                var UserResponseTask =
APIExecuter.HttpGetAsync(ApiUrlUser ?? string.Empty);
                // var LabResponseTask =
APIExecuter.HttpGetAsync(ApiUrlLab ?? string.Empty);
                await Task.WhenAll(getExportPatientDataTask,
UserResponseTask);
                (var result, bool IsException, string ExceptionMessage) =
await getExportPatientDataTask;
                var UserResponse = await UserResponseTask;
                var UserResult =
JsonConvert.DeserializeObject<MedvantageUserResponse?>(UserResponse);
                var UserData = UserResult?.ResponseValue;
```

```
 if (!IsException)
  {
 if (result is not null && result.Tables.Count > 0 && result.Tables[0]
is not null && result.Tables[0].Rows.Count > 0)
 {
 //string ApiUrlUser = UserServiceAPIFactory.BaseUrl +
UserServiceAPIFactory.GetUserList;
 //var UserResponse = await APIExecuter.HttpGetAsync(ApiUrlUser ??
string.Empty);
 //var UserResult =
JsonConvert.DeserializeObject<MedvantageUserResponse?>(UserResponse);
 //var UserData = UserResult?.ResponseValue;

 var Export =
result.Tables[0].ToJsonListObject<ExportPatientDataResponse>();

 var ExportData = from ExportPatientData in Export
 let teamMemberList = UserData?.Where(xyz =>
JsonConvert.DeserializeObject<List<teamMemberList>?>
(ExportPatientData?.teamMember).Any(x => xyz.Id == x.userId)).Select(s
=> new { userId = s.Id, name = s.Name }).ToList()
 select new
 {
 uhId = ExportPatientData.uhId,
 patientName = ExportPatientData.patientName,
 sex = ExportPatientData.sex,
 dob = ExportPatientData.dob
 raceType = ExportPatientData.raceType,
 languageName = ExportPatientData.languageName,
 ethinicityName = ExportPatientData.ethinicityName,
 problems = ExportPatientData.problems,
 medications = ExportPatientData.medications,
```

```
procedures = ExportPatientData.procedures,
teamMember = JsonConvert.SerializeObject(teamMemberList),
immunization = ExportPatientData.immunization,
healthConcerns = ExportPatientData.healthConcerns,
vitals = ExportPatientData.vitals,
smokingStatus = ExportPatientData.smokingStatus,
inplantableDevice = ExportPatientData.inplantableDevice,
goals = ExportPatientData.goals,
planOfTreatment = ExportPatientData.planOfTreatment,
medicineAllergy = ExportPatientData.medicineAllergy,
labTest = ExportPatientData.labTest,
labResult = ExportPatientData.labTestResult

                                    };
                        return Ok(new
                        {
                            status = 1,
                            message = "success",
                            responseValue = ExportData
                        });
                    }
                    else
                    {
                        return BadRequest(new
                        {
                            status = 0,
                            message = "failure",
                            responseValue = "No record found"
                        });
                    }
                }
```

```
else
{
return StatusCode(StatusCodes.Status500InternalServerError, new
{
status = 0,
message = "failure",
responseValue = ExceptionMessage
});
}
}
catch (Exception ex)
{
_logger.LogError(ex, ex.Message, nameof(Controllers));
return StatusCode(StatusCodes.Status500InternalServerError, new
{
status = 0,
message = "failure",
responseValue = ex.InnerException?.Message ?? ex.Message
});
}
}
```

PATIENT DATA EXPORT C-CDA (EXAMPLE OF C# CODE)

```
HttpGet(nameof(GetExportPatientCCDAData))]
public async Task<IActionResult> GetExportPatientCCDAData([FromQuery]
ExportPatientData pobj)
{
try
{
var getExportPatientDataTask
=_ExportPatientDataService.GetExportPatientCCDAData(pobj);

(var result, bool IsException, stringExceptionMessage) = await
getExportPatientDataTask;
if (!IsException)
{
 if (result is not null && result.Tables.Count > 0 && result.Tables[0]
is not null && result.Tables[0].Rows.Count > 0)
{
return Ok(new
{
status = 1,
message = "success",
responseValue = result.Tables[0]
});
}

status = 1,
message = "success", responseValue = result.Tables[0]
```

```
else
{
return BadRequest(new
{
status = 0,
message = "failure",
responseValue = "No recordfound“

});
}
}

else
{
return StatusCode(StatusCodes.Status500InternalServerError, new
{
status = 0,
message = "failure",
responseValue = ExceptionMessage

});
}
}
catch (Exceptionex)
{
_logger.LogError(ex, ex.Message, nameof(Controllers));
return StatusCode(StatusCodes.Status500InternalServerError, new
{

status = 0, message = "failure",
 responseValue = ex.InnerException?.Message ?? ex.Message});
}
}
```

Terms Of Use

Greetings and welcome to our network! Below, you'll find our terms and conditions governing the use of this network, accessible through various means. Whenever we mention the “Medvantage Site” in these terms and conditions, we are referring to the healthcare information network meticulously operated by Medvantage, irrespective of the manner in which you access it.

Registration

Access and Account Responsibility on Medvantage Site

To access specific areas of the Medvantage Site, registration is required. By becoming a registered member and creating an account with us, you agree to maintain the confidentiality of your chosen passwords or other account identifiers. You are solely responsible for all activities occurring under your account.

By registering on the Medvantage Site, you affirm that:

- Your account and password are personal and must not be shared or used by anyone else to gain access to the Medvantage Site.
- You will not engage in any activity that might aid unauthorized users in accessing any registration area of the Medvantage Site.
- You will not create registration accounts with the intention of misusing the site's functionality or misleading other users. Impersonation of another user is strictly prohibited.

Termination of Registration

To terminate your Medvantage account, email info@criteriontechnologies.com. Ceasing to accept these terms halts site use. We may revoke access if terms are violated, and we reserve the right to terminate accounts via email notification. Your compliance ensures Medvantage's seamless operation.