MEDVANTAGE

PATIENT SELECTION / DATA CATEGORY REQUEST / ALL DATA REQUEST 170.315(G)(7), 170.315(G)(8), 170.315(G)(9)

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:

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 and Optional 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 For example, [{"Key":"1","Selected":true}, {"Key":"2","Selected":false}]	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}

Similarly, following is a list of remaining key values:

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 variables and their types/structures: 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	Mandatory
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)

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
        });
    }
}
```

PATIENT DATA EXPOT 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, string ExceptionMessage) = 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]
                        });
                    }
                    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
                });
            }
        }
```