Open Dental Software

API Specification

Version 1

November 16, 2021

Table of Contents

About Open Dental API

This document describes the Open Dental RESTful API service. This is different from the FHIR API, which is targeted toward interfacing with medical systems. The API and the FHIR API share some of the same codebase, especially security, but the target audience is different. This API is generally recommended for most interfaces, unless you are connecting to an existing FHIR interface.

The following http method types are supported:
GET: Retrieves either one or multiple items.
POST: Creates a new item. Typically returns a "location" header and the object.
PUT: Updates an item. Returns 200 OK.
DELETE
These methods act on Resources, such as Patient or Appointment, which generally have a 1:1 relationship with Open Dental database tables. These methods frequently correspond to CRUD commands that get sent to the database such as Select, Insert, or Update. But it's more complicated than that, of course. Many methods will result in multiple database commands. A failure should return an explanation of why it failed. If a field is not included in a PUT (update), then it will not change the original field in the database.

Style

Resources in the URL are lowercase plural. Example: /patients

Resource identifiers are the database primary keys and are specified in the URL instead of as a parameter for individual resources. Example: /patients/421

When creating a resource (PUT) or retrieving a list, no identifier is used. Example: /patients

All fields and parameters use Pascal casing, where the first letter is capitalized, and optional other letters are capitalized. This is different than the style of most other REST implementations, but it is far more consistent with our existing codebase and db schema.

Fields can start with lower case whenever they are not standard database fields. See details below, in the Data Types section.

Parameters are listed in this documentation in order of importance instead of db order. Fields returned in JSON files are generally in the same order as in the database.

Some methods do not fall under the normal CRUD paradigm. Here are some examples:
/claimprocs/InsAdjust
/appointments/123/Confirm
/accountmodule/GetLedger (involves multiple db tables, so resource is not single db table)
/queries

Pagination

For any method that returns a list of items, you will need to use pagination to get chunks of up to 100 items at a time. Use Limit and Offset. Limit is usually not specified, and it defaults to the hard limit of 100 items for any request. With no Limit or Offset specified, results will include items 0 through 99. Here's a typical example:
GET /items?Offset=400
This will return items 400 through 499. The risk of page drift, where another record is added to the database in the middle of a series of requests, is miniscule.

URL

The current version of the API has the following endpoint:

https://api.opendental.com/api/v1

Example usage:
GET https://api.opendental.com/api/v1/patients

Our intent is to never change this URL, but if there is an important breaking change in the future, then we may create v2, etc. These version numbers do not correspond to versions of Open Dental. This API is intended to work smoothly with multiple versions of Open Dental. But the version of the customer database must be at least as high as the version that the method was added.

URL Parameter Encoding

Use percent encoding for all reserved characters in the URL query parameters. Quotes are not used in URL parameters. For example, De Angelo would be encoded as De%20Angelo.

Data Types

This API documentation does not generally repeat information which can be found in
https://www.opendental.com/OpenDentalDocumentation21-1.xml
So we frequently just list out the fields available for a resource without further explanation.

Enumerations: These are stored in the database as numbers, but this API uses the text equivalent. Example: Male instead of 0. Fields are still capitalized, just like the original.

Booleans: Fields that are boolean in the database are string in the API, either "true" or "false".

Dates: String with this format: yyyy-MM-dd

DateTime: String with this format: yyyy-MM-dd HH:mm:ss, which is a 24hr format.

Foreign Keys: These are stored in the database as numbers, but we frequently use a string version for parameters and fields. The string version field will start with a lower case letter to indicate that it's not a standard database field. The lower case letter versions are usually read-only and are not used to set values.
Examples:
ClinicNum=23
clinicAbbr="Monroe"
appointment.Confirmed=145 (FK to definition.DefNum)
appointment.confirmed="Called"

Strings: The JSON serializer currently outputs carriage returns (\r\n) as spaces. There may be other issues with complex text as well.

Behavior

The behavior of actions in the API are similar to when using the UI in Open Dental, but not identical. There tends to be a bit less automation. There are also no prompts for back and forth logic with user interaction. But the advantage of the API is that it will not allow you to make changes that would corrupt the database, so it's safe. If you were to use similar direct queries to try to change the database, you would quickly corrupt it. The API also makes the appropriate security log entries and performs some housekeeping duties such as archival, synchronization, and hashing in certain tables. In contrast, direct queries that change the database are strictly forbidden and we are actively building enforcement to block such queries.

Authorization

Each API request must include an Authorization header in this format:

Authorization: ODFHIR {DeveloperKey}/{CustomerKey}

For testing, the following credentials can be used:
Authorization: ODFHIR NFF6i0KrXrxDkZHt/VzkmZEaUWOjnQX2z

Security Permissions

The security permissions for the API are organized differently from those in Open Dental.

Permission	Methods	Details
ReadAll	Allergies GET AllergyDefs GET Appointments GET (single/multiple) Appointments GET SlotsWebSched Appointments GET Slots Appointments GET ASAP Appointments GET WebSched ApptFields GET Clinics GET Commlogs GET Definitions GET DiscountPlanSubs GET Diseases GET DiseaseDefs GET Documents GET Medications GET MedicationPats GET Operatories GET PatFields GET PatientNotes GET Patients GET (single/multiple) Patients GET Simple Preferences GET ProcedureLogs GET Providers GET Referrals GET Signalods GET Userods GET	All GET methods except queries. These are throttled at different rates for different users.
AllOthers	Allergies POST AllergyDefs POST DiscountPlanSubs POST Diseases POST DiseaseDefs POST Documents POST DownloadMount Documents POST DownloadSftp Documents POST SetByUrl Documents POST Thumbnails Documents POST UploadSftp	Any method not listed under a different permission

	Medications POST MedicationPats POST PatFields PUT PatientNotes PUT RefAttaches POST	
Comm	Appointments PUT Break Appointments PUT Note Appointments PUT Confirm ApptFields PUT Commlogs POST Popups POST Recalls PUT Status	
Documents	Documents POST Upload (not any other Documents methods)	Because each upload consumes bandwidth
Queries	Queries POST Queries PUT ShortQuery	
Appointments	Appointments PUT Appointments POST (create) Appointments POST WebSched	
Insurance	ClaimProcs PUT InsAdjust Claims PUT Status InsSubs POST InsSubs PUT InsSubs DELETE PatPlans POST PatPlans DELETE InsVerifies PUT	These are complex and rarely used
Patients	Patients PUT Patients POST	
Payments	Payments POST	Added in version 21.3
PayPlans	PayPlans POST	Added in version 21.3

The normal Open Dental permissions are used for logging API actions, but they do not restrict the API from making changes. For example, date restrictions do not apply.

Testing

We use Advanced REST Client (ARC) in Chrome for testing requests, but there are other options.

Open Dental hosts a test database for developers to test against. The base URL is the same endpoint listed above.

See the Authorization section above for the test credentials.

Setting up the API

The API web service is hosted at Open Dental headquarters. All requests are routed through this address to the appropriate office. Offices do not host their own API service. The API Key specified in the Authorization header

is linked to a specific office.

Steps to Enable the API

In order to use the API, the office must have an eConnector running. See https://www.opendental.com/manual/econnector.html for installation instructions.

Launch the Open Dental program. Enable the API by going to Setup -> Advanced Setup -> FHIR and checking the Enabled checkbox.

API Keys

When requesting data from the API server, an API key must be present in the request header. API keys are created by the 3rd-party developer from Open Dental's developer portal: https://api.opendental.com/portal/gwt/fhirportal.html
The API keys created from the developer portal can be entered into the Open Dental program to assign that API key to a customer. Requests to our API using these keys must then include the developer API key and the customer API key. To obtain a developer API key, contact vendor.relations@opendental.com. Please include the details below.

Developer name:
Company name:
Email address:
A list of the API Resources you need access to, the level of access such as read/write/update:
A description of the application you are intending to build (type and purpose):

To assign an API key to an Open Dental customer, go to Setup -> Advanced Setup -> FHIR. Click the Add Key in the lower left. Here is where you paste a key generated from the developer portal. The customer has the ability to enable or disable a key. The customer can view permissions granted to that key, but they cannot change those permissions. The interface in this window displays information from our HQ server, not the customer database.

When using the window described above, a copy of the keys is placed in a table in the customer's local database called "apikey". This table will then contain the customer's apikeys and each corresponding developer name exactly as entered in the developer portal. Occasionally, you may want to retrieve this information via a query. This will be rare, but can be accomplished with the following example:

```
SELECT CustApiKey FROM apikey WHERE DevName='YourName';
```

Allergies

Allergies GET

Version Added: 21.1

PatNum is required. Gets a list of all allergies that that are assigned to the patient.

Example Request
GET /allergies?PatNum=15

Example response:

```
[
  {
  "AllergyNum": 1,
```

```
  "AllergyDefNum": 4,
  "PatNum": 15,
  "defDescription": "Penicillin",
  "defSnomedType": "None",
  "Reaction": "Rash",
  "StatusIsActive": "true",
  "DateAdverseReaction": "2020-03-18"
},
etc...
]
```

Allergies POST

Version Added: 21.3

Attaches an allergyDef to a patient.

AllergyDefNum: Rarely used. Just use **defDescription** instead, which handles insertion of AllergyDef automatically.
PatNum: Required.
defDescription: Required unless you choose to use **AllergyDefNum**.
Reaction: Optional. String describing the adverse reaction.
StatusIsActive: Optional. Either "true" or "false". Default "true".
DateAdverseReaction: Optional. String in "yyyy-MM-dd" format. Default "0001-01-01".

Example requests:
POST /allergies

```
{
  "PatNum": 12,
  "defDescription": "Propranolol"
}
```

```
{
  "PatNum": 12,
  "defDescription": "Penicillin"
  "Reaction": "Hives",
  "DateAdverseReaction": "2019-09-23"
}
```

Example responses:
201 Created
400 BadRequest (Missing or Invalid fields)
404 NotFound "Patient not found" or "AllergyDef not found"

AllergyDefs

AllergyDefs GET

Version Added: 21.3

Gets a list of all allergies that can be assigned to patients.

Rarely used. Usually just use Allergies GET and POST.

Example requests:
GET /allergyDefs
GET /allergyDefs?Offset=200

Example response:

```
[
  {
  "AllergyDefNum": 44,
  "Description ": "Allergy - Phentermine",
  "IsHidden": "False",
  "DateTStamp": "2020-07-17 02:45:38",
  "SnomedType": "",
  "MedicationNum": 0,
  "UniiCode": "",
},
  {
  "AllergyDefNum": 45,
  "Description ": "Allergy - Sudogest",
  "IsHidden": "False",
  "DateTStamp": "2015-12-10 05:40:32",
  "SnomedType": "",
  "MedicationNum": 0,
  "UniiCode": "",
},
etc...
]
```

AllergyDefs POST

Version Added: 21.3

Inserts a single AllergyDef using a unique **Description**.

Rarely used. Usually just use Allergies GET and POST.

Example request:
POST /allergyDefs

```
{
   "Description": "Tylenol"
}
```

Example responses:
201 Created
400 BadRequest "Description is required" or "An AllergyDef with that name already exists".

Appointments

Appointments GET (single)

Version Added: 21.1

Example Request

GET /appointments/18

Example response:

```
{
"AptNum": 18,
"PatNum": 17,
"AptStatus": "Scheduled",
"Pattern": "//XXXX//",
"Confirmed": 19,
"confirmed": "Not Called",
"Op": 3,
"Note": "",
"ProvNum": 1,
"provAbbr": "DOC1",
"ProvHyg": 0,
"AptDateTime": "2020-07-31 08:30:00",
"ProcDescript": "Seal, Seal",
"ClinicNum": 0,
"IsHygiene": "false",
"DateTStamp":"2021-05-03 08:30:12",
"serverDateTime":"2021-05-04 09:32:45"
}
```


Appointments GET (multiple)

Version Added: 21.1

Parameters:

date: For a single day.
dateStart, dateEnd: For a date range, inclusive of both dates.
DateTStamp: Only include appointments with a DateTStamp altered after the specified date and time. This provides a good way for you to keep a synchronized copy of appointments. Store serverDateTime (added in v21.2) that gets returned, and use it to run the next GET.
ClinicNum: Not specifying ClinicNum will get for all clinics because it will not filter on clinics. Use ClinicNum 0 for appointments not attached to clinics.

Example Requests:
GET /appointments?Offset=400
GET /appointments?date=2020-07-30&Offset=200
GET /appointments?dateStart=2020-07-30&dateEnd=2020-08-02&DateTStamp=2020-07-30&ClinicNum=3

Example response:


```
[
{
"AptNum": 4,
"PatNum": 20,
"AptStatus": "Scheduled",
"Pattern": "//XXXX//",
"Confirmed": 19,
"confirmed": "Not Called",
"Op": 2,
"Note": "",
"ProvNum": 3,
"provAbbr": "DOC2",
"ProvHyg": 0,
"AptDateTime": "2020-07-31 09:00:00",
"ProcDescript": "Ex, PA",
```

```
"ClinicNum": 0,
"IsHygiene": "false",
"DateTStamp":"2021-05-03 08:30:12",
"serverDateTime":"2021-05-04 09:32:45"
},
  {
"AptNum": 3,
"PatNum": 21,
"AptStatus": "Complete",
"Pattern": "//XXXXXX//",
"Confirmed": 19,
"confirmed": "Not Called",
"Op": 6,
"Note": "",
"ProvNum": 3,
"provAbbr": "DOC2",
"ProvHyg": 4,
"AptDateTime": "2020-07-31 09:00:00",
"ProcDescript": "Ex",
"ClinicNum": 0,
"IsHygiene": "true",
"DateTStamp":"2021-05-03 08:30:12",
"serverDateTime":"2021-05-04 09:32:45"
}
]
```


Appointments GET ASAP

Version Added: 21.1

Gets the ASAP list. ProvNum is optional. ClinicNum is required if using Clinics.

ClinicNum: Required only if Clinics are enabled.
ProvNum: Optional.

Example Requests:
GET /appointments/ASAP
GET /appointments/ASAP?Offset=200

Example response:


```
[
{
"AptNum": 4,
"PatNum": 20,
"AptStatus": "Scheduled",
"Pattern": "//XXXX//",
"Confirmed": 19,
"confirmed": "Not Called",
"Op": 2,
"Note": "",
"ProvNum": 3,
"provAbbr": "DOC2",
"ProvHyg": 0,
"AptDateTime": "2020-07-31 09:00:00",
"ProcDescript": "Ex, PA",
"ClinicNum": 0,
```

```
"IsHygiene": "false",
"DateTStamp":"2021-05-03 08:30:12",
"serverDateTime":"2021-05-04 09:32:45"
},
  {
"AptNum": 3,
"PatNum": 21,
"AptStatus": "Scheduled",
"Pattern": "//XXXXXX//",
"Confirmed": 19,
"confirmed": "Not Called",
"Op": 6,
"Note": "",
"ProvNum": 3,
"provAbbr": "DOC2",
"ProvHyg": 4,
"AptDateTime": "2020-07-31 09:00:00",
"ProcDescript": "Ex",
"ClinicNum": 0,
"IsHygiene": "true",
"DateTStamp":"2021-05-03 08:30:12",
"serverDateTime":"2021-05-04 09:32:45"
}
]
```


Appointments GET SlotsWebSched

Version Added: 21.1

Rarely used.

This gets slots according to the existing logic used for WebSched existing patient or new patient. Customer must first have gone to Setup WebSched (Existing Patient or New Patient) and done all the setup, including creating Appointment Types and linking Appointment Types to WebSched. This all requires a fair amount of setup on the customer's end, but that restriction is usually preferred when patients are making their own appointments.

After calling this method, use Appointments POST WebSched to create a new WebSched appointment.

Parameters:

date: For a single day.
dateStart, dateEnd: For a date range, inclusive of both dates. If no dates at all are supplied, the default is the next two weeks.
ClinicNum: Required if clinics enabled.
defNumApptType: definition.DefNum where definition.Category=42 (NewPat) or 51 (ExistingPat). There must also be an AppointmentType assigned to that DefNum and it must be set to be used in WebSched.
isNewPatient: Optional. Defaults to false. Just make this match the kind of defNumApptType that you use.

Example Request:
GET /appointments/SlotsWebSched?dateStart=2021-02-15&dateEnd=2021-02-15&defNumApptType=326

Example response:


```
[
  {
"DateTimeStart": "2021-02-15 08:00:00",
```

```
"DateTimeEnd": "2021-02-15 08:30:00",
"ProvNum": 1,
"OpNum": 1
},
  {
"DateTimeStart": "2021-02-15 08:30:00",
"DateTimeEnd": "2021-02-15 09:00:00",
"ProvNum": 1,
"OpNum": 1
},
  {
"DateTimeStart": "2021-02-15 09:00:00",
"DateTimeEnd": "2021-02-15 09:30:00",
"ProvNum": 1,
"OpNum": 1
},
etc
]
```

Appointments GET Slots

Version Added: 21.1

This is closer to how search behaves from within Open Dental instead of WebSched. This requires no advanced setup. It looks at open schedule times, whether the schedules are attached to a provider or an operatory.

There are, however, some differences between this and the Search in Open Dental. This returns entire open slots instead of a series of suggested appointment times or the first appointment time for each day. It also currently searches by entire appointment length instead of the XXX provider times on appointments. For single operatories, the results will be the same as what a person would see if looking at the Appointments module. If a provider has existing appointments in multiple operatories, it considers all of them as a whole and only returns slots that are available in all operatories for that provider simultaneously. Only considers primary providers on appointments.

Parameters:

date: For a single day.
dateStart, dateEnd: For a date range, inclusive of both dates.
lengthMinutes: Optional. This allows ignoring slots that are too small.
ProvNum: Optional.

Example request:
GET /appointments/Slots?date=2021-02-15&ProvNum=1

Example response:

```
[
  {
"DateTimeStart": "2021-02-15 08:00:00",
"DateTimeEnd": "2021-02-15 10:30:00",
"ProvNum": 1,
"OpNum": 1
},
  {
"DateTimeStart": "2021-02-15 13:00:00",
```

```
"DateTimeEnd": "2021-02-15 15:00:00",
"ProvNum": 1,
"OpNum": 1
},
etc
]
```

Appointments GET WebSched

Version Added: 21.3

Rarely used. Probably just use Appointments GET (multiple) instead.

Gets a list of all appointments, indicating which were made through the WebSched service. This is displayed in the **eServiceLogType** field as either "Recall", "NewPat", "ExistingPat", or "ASAP". Appointments not made through WebSched will have an eServiceLogType of "None".

date: Optional. Search for a single day in "yyyy-MM-dd" format.

dateStart, dateEnd: Optional. Search for a date range, inclusive of both dates, in "yyyy-MM-dd" format.
DateTStamp: Optional. String in "yyyy-MM-dd HH:mm:ss" format.
ClinicNum: Optional.

Example requests:
GET /appointments/WebSched
GET /appointments/WebSched?date=2021-02-15
GET /appointments/WebSched?date=2021-02-15&Offset=200
Example responses:

```
[
{
  "AptNum": 3,
  "PatNum": 21,
  "AptStatus": "Scheduled",
  "Pattern": "//XXXXXX//",
  "Confirmed": 19,
  "confirmed": "Not Called",
  "Op": 6,
  "Note": "",
  "ProvNum": 3,
  "provAbbr": "DOC2",
  "ProvHyg": 4,
  "AptDateTime": "2021-07-31 09:00:00",
  "ProcDescript": "Ex",
  "ClinicNum": 0,
  "IsHygiene": "true",
  "DateTStamp":"2021-08-03 08:30:12",
  "serverDateTime":"2021-08-04 09:32:45" ,
  "eServiceLogType": "NewPat"
},
 {
  "AptNum": 35,
  "PatNum": 37,
  "AptStatus": "Scheduled",
  "Pattern": "/XXXX/",
  "Confirmed": 19,
```

```
  "confirmed": "Not Called",
  "Op": 5,
  "Note": "",
  "ProvNum": 3,
  "provAbbr": "DOC2",
  "ProvHyg": 7,
  "AptDateTime": "2021-10-18 13:30:00",
  "ProcDescript": "",
  "ClinicNum": 2,
  "IsHygiene": "false",
  "DateTStamp": "2021-08-23 10:52:22",
  "serverDateTime": "2021-09-13 09:09:37",
  "eServiceLogType": "None"
}
```


200 OK
400 BadRequest (Missing or Invalid fields)

Appointments POST (create)

Version Added: 21.1

The following fields cannot be set as part of a POST: AptNum, provAbbr, and ProcDescript. Attempts to set them will be silently ignored.

PatNum: Required.
Op: Required.
AptDateTime: Required. String in "yyyy-MM-dd HH:mm:ss". Use Appointments GET Slots to find available times.

AptStatus: Optional. Either "Scheduled", "Complete", "UnschedList", "ASAP", "Planned", or (rarely used) "PtNote", "PtNoteCompleted". Default "Scheduled".
Pattern: Optional. Time pattern in 5 minute increments. A string consisting of 'X' and '/' characters only. Default "/XX/" (20 minutes).
Confirmed: Optional. Definition.DefNum where definition.Category=2. Default is the first definition in that Category.
Note: Optional. Default blank.
ProvNum: Optional. Defaults to the PriProv of the patient, if set, or the dental office's default provider.
ProvHyg: Optional. Default 0.
ClinicNum: Optional. Default 0.
IsHygiene: Optional. Default "false".
IsNewPatient (added in 21.3)**:** Optional. Either "true" or "false". Default "false".

Example Request:
POST /appointments


```
{
"PatNum": 21,
"Op": 6,
"AptDateTime": "2020-07-31 09:00:00"
}
```


Example Response:
201 Created

Header "location": https://api.opendental.com/api/v1/appointments/3

```
{
  "AptNum": 3,
  "PatNum": 21,
  "AptStatus": "Scheduled",
  "Pattern": "/XX/",
  "Confirmed": 19,
  "confirmed": "Not Called",
  "Op": 6,
  "Note": "",
  "ProvNum": 3,
  "provAbbr": "DOC2",
  "ProvHyg": 0,
  "AptDateTime": "2020-07-31 09:00:00",
  "ProcDescript": "",
  "ClinicNum": 0,
  "IsHygiene": "false"
}
```

Appointments POST WebSched

Version Added: 21.3

See Appointments GET WebSched
See Appointments GET SlotsWebSched

Rarely used. Probably just use Appointments POST (create) instead.

Creates a WebSched appointment. This appointment is similar to appointments made through Open Dental's WebSched eServices, which the dental office must have set up.

Prior to running this method, use Appointments GET SlotsWebSched to get the specific timeslots available for WebSched appointments. The **ProvNum**, **OpNum**, and time stamps returned from that method are used below.

PatNum: Required.
DateTimeStart: Required. The start time of the appointment timeslot.
DateTimeEnd: Required. The end time of the appointment timeslot.
ProvNum: Required. The ProvNum of the appointment timeslot.
OpNum: Required. The OpNum of the appointment timeslot.
defNumApptType: Required. This must the be the same **defNumApptType** used to find the timeslot.

Example Request:
POST /appointments/WebSched

```
{
  "PatNum": 21,
  "dateTimeStart": "2021-11-19 09:00:00",
  "dateTimeEnd":"2021-11-19 09:30:00"
  "ProvNum": 5,
  "OpNum": 3,
  "defNumApptType": 326
}
```

Example Response:

```
{
```

```
  "AptNum": 121,
  "PatNum": 21,
  "AptStatus": "Scheduled",
  "Pattern": "//XX",
  "Confirmed": 395,
  "confirmed": "Created from Web Sched",
  "Op": 3,
  "Note": "Appointment Reason: New Patient Exam",
  "ProvNum": 5,
  "provAbbr": "SMITH",
  "ProvHyg": 0,
  "AptDateTime": "2021-11-19 09:00:00",
  "ProcDescript": "LimEx, 2BW",
  "ClinicNum": 1,
  "IsHygiene": "false",
  "DateTStamp": "0001-01-01 00:00:00",
  "serverDateTime": "0001-01-01 00:00:00"
}
```


201 Created
400 BadRequest (Missing or Invalid fields)
404 NotFound "Patient not found" or "Timeslot is no longer available"

Appointments PUT (update)

The following fields cannot be set as part of a PUT: AptNum, PatNum, provAbbr, and ProcDescript. Attempts to set them will be silently ignored. All remaining fields are optional and it is common to only set one or two fields. Note will overwrite any existing note. Use appointments PUT Note to instead append a note.

It is rare to use this method. There would be complex issues involved. It is recommended to instead use Appointments PUT Note, PUT Confirm, and PUT Break.

AptNum: Required in the URL.
AptStatus: Either "Scheduled", "Complete", "UnschedList", "Broken", "Planned", "PtNote", or "PtNoteCompleted".
Pattern: Time pattern in 5 minute increments. A string consisting of 'X' and '/' characters only.
Confirmed: Definition.DefNum where definition.Category=2.
Op: OperatoryNum for a single operatory.
Note: String.
ProvNum: ProvNum for a provider.
ProvHyg: ProvNum for a hygeine provider.
AptDateTime: Start time for the appointment. String in "yyyy-MM-dd HH:mm:ss" format.
ClinicNum: ClinicNum of a clinic.
IsHygiene: Either "true" or "false". True if a hygiene appointment.
IsNewPatient (added in 21.3)**:** Either "true" or "false". Default "false".

Example Request:
PUT /appointments/34


```
{
  "Note":"Note on appointment."
}
```


Example responses:
200 OK

400 BadRequest (Invalid fields)
404 NotFound "Appointment not found"

Appointments PUT Break

Version Added: 21.3

Breaks an appointment. Only appointments with an AptStatus of Scheduled can be broken. Creates a CommLog entry if the office has that preference turned on.

AptNum: Required in the URL.
sendToUnscheduledList: Required. Either "True" or "False". Usually use "True" to send the broken appointment to the Unscheduled List. This will allow the patient or the office to see that this appointment is ready to be scheduled when they are ready. The only reason you would ever use "False" is if you were setting breakType to "Missed" or "Cancelled". "False" would leave the appointment in place on the appointment book.
breakType: Optional. Rarely used. Only used if you want a procedure to be added, which is usually associated with a fee. Use "Missed" to add a procedure with code D9986, indicating that they missed their appointment without notice. Use "Cancelled" to add procedure with code D9987, indicating less than 24 hrs notice. Normally, if more than 24 hrs notice is given, you would not specify a breakType. These options cannot be used unless the offices has gone to Setup-Appointments-Appts Preferences, and changed the broken appointment procedure type to no longer be "None".

Example Request:

PUT /appointments/5/Break

```
{
  "sendToUnscheduledList":"True"
}
```

Example responses:
200 OK
400 Bad Request (with explanation)
404 Not Found

Appointments PUT Note

Version Added: 21.1

Only one field is allowed in the JSON object: "Note". The note passed in does not damage any existing note. If a note already exists, a CR is included before adding the new note.

Example Request:

PUT /appointments/34/Note

```
{
  "Note":"Requests reschedule"
}
```

Example responses:
200 OK
400 Bad Request (with explanation)
404 Not Found

Appointments PUT Confirm

Version Added: 21.1

Only one field is allowed in the JSON object, either **confirmVal** or **defNum**. A confirmVal of "None" corresponds to the default status for all new appointments, which is based on the first item listed in Definitions, Appt Confirmed. The other four confirmVal options are managed by the office in eServices Setup, Automated Messaging. Use the defNum field to set the status to any ApptConfirmed type.

confirmVal: Either "None", "Sent", "Confirmed", "Not Accepted", or "Failed".
defNum (added in version 21.2): Definition.DefNum where definition.Category=2.

Example Requests:

PUT /appointments/34/Confirm

```
{
  "confirmVal":"Confirmed"
}
```

```
{
  "defNum" : 209
}
```

Example responses:
200 OK
400 Bad Request (with explanation)
404 Not Found

ApptFields

An ApptField is a highly customizable field that shows on appointments. For example, the office might have an ApptField called "Ins Verified", and you might use the API to set it to "Yes" for a specific appointment. There is some prep work that you must do with the office ahead of time. From within the Open Dental UI at the office, you must first add the field under Setup, Appointments, Appointment Field Defs. There would be additional work to make it also show in the Appt Views and/or hover Bubble.

ApptFields GET

Version Added: 21.1

~~AptNum and FieldName are required parameters.~~ If an ApptField exists for the appointment, it gets the value. If an ApptField does not exist, it returns an empty string.

AptNum: Required.
FieldName: Required.

Example request:
GET /apptfields?AptNum=101&FieldName=Ins%20Verified

Example response:

```
{
  "FieldName": "Ins Verified",
  "AptNum":101,
  "FieldValue":"Yes"
}
```

ApptFields PUT

Version Added: 21.1

If an ApptField already exists for the appointment, it gets set to the new value, overwriting the old value. If an ApptField does not yet exist for the appointment, then an ApptField gets inserted into the database.

AptNum: Required.
FieldName: Required.
FieldValue: Required.

Example request:
PUT /apptfields

```
{
  "FieldName": "Ins Verified",
  "AptNum":101,
  "FieldValue":"Yes"
}
```

Example responses:
200 OK
400 Bad Request (with explanation)

ClaimProcs

ClaimProcs PUT InsAdjust

Version Added: 21.1

This adds or changes a claimproc that is acting as an insurance adjustment. **PatPlanNum** is required. There's no way to obtain the PatPlanNum from the API; you would instead use a query. "**date**" is optional and defaults to today. It should be a date within the benefit year that you are interested in. Any adjustment that is created will also use that date. Either **insUsed** or **deductibleUsed** is Required. Pass in the total insurance and/or deductible used. The logic will take into consideration existing paid claims. For example, if payments of $200 are already entered into Open Dental, and you pass in insUsed of $300, then it will result in a $100 adjustment so that it will properly show the $300. If the insUsed passed in exactly equals payments already in Open Dental, then any existing adjustment will be deleted. The calculations do not distinguish family or lifetime benefits.

PatPlanNum: Required.
insUsed: This or **deductibleUsed** is required.
deductibleUsed: This or **insUsed** is required.
date: Optional. String in "yyyy-MM-dd" format. Default today's date.

Example request:
PUT /claimprocs/InsAdjust

```
{
  "PatPlanNum": 123,
  "date": "2020-01-01",
  "insUsed":"300",
  "deductibleUsed":"25"
}
```

Example response:

200 OK (Regardless of how the math worked out. ClaimProcs could have been added or deleted.)

Claims

Claims PUT Status

Version Added: 21.3

Sets the ClaimStatus of a claim to "Sent" and automatically creates an Etrans entry. Use either Queries POST or Queries PUT ShortQuery to obtain the **ClaimNum** of the desired claim.

ClaimNum: Required in the URL.
DateSent: Required. Date the claim was most recently sent. String in "yyyy-MM-dd" format."
DateSentOrig: Optional. String in "yyyy-MM-dd" format. Defaults to **DateSent.** Will be ignored for claims that have been marked as "Sent" previously.

Example requests:
PUT /claims/26/Status

```
{
 "DateSent": "2021-09-13"
}
```

```
{
 "DateSent": "2021-09-13",
 "DateSentOrig": "2021-09-01"
}
```

Example responses:
200 OK
400 BadRequest (Missing or invalid fields)
404 NotFound "Claim not found"

Clinics

Clinics GET

Version Added: 21.1

Gets all non-hidden clinics, in order.

Example request:
GET /clinics

Example response:

```
[
{
"ClinicNum": 1,
"Description": "Clinic1",
"Address": "333 Blackwood St",
"Address2": "",
"City": "Salem",
"State": "Oregon",
"Zip": "97301",
"BillingAddress": "333 Blackwood St ",
```

```
"BillingAddress2": "",
"BillingCity": " Salem ",
"BillingState": " Oregon ",
"BillingZip": "97301",
"PayToAddress": "",
"PayToAddress2": "",
"PayToCity": "",
"PayToState": "",
"PayToZip": "",
"Phone": "5105552005",
"Abbr": "Clinic1"
}
]
```


Commlogs

Commlogs GET

Version Added: 21.1

Get all commlogs for a patient.

PatNum: Required.

Example request:
GET /commlogs?PatNum=15

Example response:


```
[
{
"CommlogNum": 2,
"PatNum": 15,
"CommDateTime": "2021-02-07 03:25:29",
"CommType": "239",
"commType": "Misc",
"Note": "Left msg on answering machine",
"Mode_": "Phone",
"SentOrReceived": "Sent"
},
  {
"CommlogNum": 1,
"PatNum": 15,
"CommDateTime": "2021-02-07 03:09:00",
"CommType": "236",
"commType": "Insurance",
"Note": "Note for John Smith",
"Mode_": "Mail",
"SentOrReceived": "Received"
}
]
```


Commlogs POST (create)

Version Added: 21.1

Creates a commlog for the patient.

PatNum: Required.
Note: Required.

CommDateTime: Optional. String in "yyyy-mm-dd HH:mm:ss" format. Default now.
CommType: Optional. definition.DefNum where definition.Category=27. Default Miscellaneous.
commType: Optional. definition.ItemName where definition.Category=27. Will be used over **CommType** if both are specified.
Mode_: Either "None", "Email", "Mail", "Phone", "In Person", "Text", "Email and Text", or "Phone and Text". Default "Phone".
SentOrReceived: Either "Neither", "Sent", or "Received". Default "Sent".

Specify the commlog type by using either the CommType or commType field, not both. If both are present in the JSON, the CommType field will be used.

Example requests:
POST /commlogs

```
{
  "PatNum": 15,
  "Note": "Left msg on answering machine"
}
```

```
{
  "PatNum": 30,
  "commType": "ApptRelated",
  "CommDateTime": "2021-01-01 11:19:00",
  "Mode_": "Text",
  "SentOrReceived": "Received",
  "Note": "Appointment confirmed for 9:15am."
}
```

Example response:
201 Created
(no "location" Header or object because we don't support GET single commlogs)

DiscountPlanSubs

DiscountPlanSubs GET

Version Added: 21.3

Gets the DiscountPlan that is attached to the patient.

PatNum: Required.

Example request:
GET /discountPlanSubs?PatNum=56

Example response:

```
{
  "DiscountSubNum": 32,
  "DiscountPlanNum": 8,
  "PatNum": 56,
  "DateEffective": "2021-01-01",
  "DateTerm": "2022-01-01",
```

```
  "SubNote": ""
}
```
200 OK
400 BadRequest "PatNum is Required"
404 NotFound "Patient not found"

DiscountPlanSubs POST

Version Added: 21.3

Subscribes a patient to an existing DiscountPlan.

DiscountPlanNum: Required.
PatNum: Required.
DateEffective: Optional. String in "yyyy-MM-dd" format. The date the plan starts impacting procedure fees. Default "0001-01-01" to indicate the beginning of the current calendar year.
DateTerm: Optional. String in "yyyy-MM-dd" format. The date the plan no longer impacts procedure fees. Default "0001-01-01" to indicate the end of the current calendar year.
SubNote: Optional.

Example request:
POST /discountPlanSubs

```
{
  "DiscountPlanNum": 6,
  "PatNum": 67,
  "DateEffective": "2021-01-01",
  "DateTerm": "2022-01-01"
}
```

Example response:
201 Created
400 BadRequest (Missing or invalid fields)
404 NotFound "Patient not found" or "DiscountPlan not found"

Definitions

Defintions GET

Version Added: 21.1
includeHidden is an optional parameter that defaults to false.

Example requests:
GET /definitions
GET /definitions?Offset=200
GET /definitions?Category=1
GET /definitions?Category=1&includeHidden=true

Example response:

```
[
  {
"DefNum": 293,
"ItemName": "Sales Tax",
"ItemValue": "+",
```

```
"Category": 1,
"category": "AdjTypes",
"isHidden": "false"
},
  {
"DefNum": 8,
"ItemName": "Professional Discount",
"ItemValue": "-",
"Category": 1,
"category": "AdjTypes",
"isHidden": "false"
},
  {
"DefNum": 9,
"ItemName": "Cash Discount",
"ItemValue": "-",
"Category": 1,
"category": "AdjTypes",
"isHidden": "false"
},
etc...
]
```


Diseases

Diseases GET

Version Added: 21.3

PatNum is required. Gets a list of all diseases (Problems) that that are assigned to the patient.

Example request:
GET /diseases?PatNum=41

Example responses:


```
[
{
  "DiseaseNum": 2,
  "PatNum": 41,
  "DiseaseDefNum": 44,
  "diseaseDefName": "COPD",
  "PatNote": "",
  "ProbStatus": "Active",
  "DateStart": "0001-01-01 00:00:00",
  "DateStop": "0001-01-01 00:00:00"
},
  {
  "DiseaseNum": 8,
  "PatNum": 41,
  "DiseaseDefNum": 58,
  "diseaseDefName": "Severe Back Pain",
  "PatNote": "",
  "ProbStatus": "Inactive",
  "DateStart": "0001-01-01 00:00:00",
  "DateStop": "0001-01-01 00:00:00"
```

```
},
etc…
]
```
200 OK
400 BadRequest "PatNum is required"
404 NotFound "Patient not found"

Diseases POST

Version Added: 21.3

Attaches a diseaseDef (Problem) to a patient.

PatNum: Required.
DiseaseDefNum: Rarely used. Just use **diseaseDefName** instead, which handles insertion of DiseaseDef automatically.
diseaseDefName: Required unless you choose to use **DiseaseDefNum**.
DateStart: Optional. String in "yyyy-MM-dd" format. Default "0001-01-01".
DateStop: Optional. String in "yyyy-MM-dd" format. Default "0001-01-01".
ProbStatus: Optional. Either "Active", "Resolved" or "Inactive". Default "Active".
Note: Optional.

Example requests:
POST /diseases

```
{
  "PatNum": 74,
  "diseaseDefName": "Diabetes"
}
```

```
{
  "PatNum": 74,
  "DiseaseDefName": "Severe Back Pain",
  "ProbStatus": "Resolved",
  "PatNote": "Patient underwent corrective surgery 04/11/2019",
  "DateStart": "2016-01-01",
  "DateStop": "2019-04-30"
}
```

Example responses:
201 Created
400 BadRequest (Missing or Invalid fields)
404 NotFound "Patient not found" or "DiseaseDef not found"

DiseaseDefs

DiseaseDefs GET

Version Added: 21.3

Gets a list of all diseases (Problems) that can be assigned to patients.

Rarely used. Usually just use Diseases GET and POST.

Example requests:
GET /diseaseDefs
GET /diseaseDefs?Offset=200

Example response:

```
[
{
  "DiseaseDefNum": 58,
  "DiseaseName": "Severe Back Pain",
  "IsHidden": "True",
  "DateTStamp": "2021-02-07 12:27:28",
  "ICD9Code": "",
  "ICD10Code": "",
  "SnomedCode": ""
},
  {
  "DiseaseDefNum": 59,
  "DiseaseName": "Hypertension",
  "IsHidden": "False",
  "DateTStamp": "2021-09-07 14:00:10",
  "ICD9Code": "402",
  "ICD10Code": "",
  "SnomedCode": ""
},
etc…
]
```

DiseaseDefs POST

Version Added: 21.3

Inserts a single DiseaseDef using a unique **DiseaseName**.

Rarely used. Usually just use Diseases GET and POST.

Example request:
POST /diseaseDefs

```
{
  "DiseaseName": "Shingles"
}
```

Example responses:
201 Created
400 BadRequest "DiseaseName is required" or "A DiseaseDef with that name already exists".

Documents

This includes images, PDFs, etc. There are four ways a document can be inserted into Open Dental:

1. Not using any API – Add a file to the patient folder. When the user later load the Imaging Module within Open Dental, any new files found are always recognized and entries are made in the database for them. By default, the new file will be dropped into the first category of images. If you want the new file to go to a specific category, then you can name it with a prefix of "_##_". Example: "_135_". The prefix number should be the DefNum of the category where the document belongs. To determine the DefNum, you will need to look

in the database where definition.Category=18 and pick from that list somehow. Files with prefixes are processed to go into the specified category, and the prefix is removed.

2. Documents POST SetByUrl.

3. Documents POST Upload.

4. Documents POST UploadSftp.

Documents GET

Version Added 21.2

PatNum is required. Gets all documents and mounts for the patient, ordered by DateCreated. If the file storage method is InDatabase, the returned filePath field will blank. Mounts do not have a filePath or a DateTStamp. DateTStamp and serverDateTime are not really useful at the moment for this table, compared to Appointments and Patients. The result does not include the actual document files. Those can be obtained using Documents POST DownloadSftp.

Example request:
GET /documents?PatNum=101

Example response:

```
[
  {
  "DocNum":34,
  "MountNum":0,
  "filePath": "\\server\OpenDentImages\S\SmithJohn15\exampleimage.jpg",
  "Description":"Scanned image",
  "Note": "",
  "DateCreated": "2021-04-08 08:12:21",
  "DateTStamp": "2021-04-18 11:45:01",
  "serverDateTime": "2021-05-20 13:30:54"
},
  {
  "DocNum":0,
  "MountNum":15,
  "filePath": "",
  "Description":" 5/11/2021: 4BW",
  "Note": "Bitewings",
  "DateCreated": "2021-05-11 13:26:39",
  "DateTStamp": "",
  "serverDateTime": "2021-05-20 13:30:54"
}
]
```

200 OK
400 BadRequest "PatNum is required"
404 NotFound "Patient not found"

Documents POST DownloadSftp

Version Added: 21.2

This will place a file on an SFTP site that you specify. After running this method, download the resulting file from your SFTP site. The user with the SFTP credentials must have write permission in this directory. Directory

will be created if it does not exist, and files already existing with the specified name will be overwritten. If the SftpAddress does not contain a file name, the document.FileName will be used. All file storage options (LocalAtoZ, InDatabase, and Cloud) are supported.

Either **DocNum** or **MountNum** are required.

Example Requests:
POST /documents/DownloadSftp

```
{
  "DocNum": 389,
  "SftpAddress": "MySftpSite/myUsername/Documents/SmithJohn389.png",
  "SftpUsername": "myUsername",
  "SftpPassword": "myPassword"
}
```

```
{
  "MountNum": 20,
  "SftpAddress": "MySftpSite/myUsername/Documents/FMX.jpg",
  "SftpUsername": "myUsername",
  "SftpPassword": "myPassword"
}
```

Example responses:
201 Created, "location": The full filepath of the saved file.
400 BadRequest (Invalid fields, invalid file extension, and Sftp connection errors).
404 NotFound "Document not found" and "Mount not found".

Documents POST SetByUrl

Version Added: 21.1

URL goes into the database in document.Note as "_download_:" followed by the URL. The customer never sees this text, but when they later click on the document, the URL is used to perform a download and the prefix is removed.

DocCategory is optional and defaults to the first DocCategory. It is a FK to definition.ItemName, where definition.Category=18.

Example request:
POST /documents/SetByUrl

```
{
  "PatNum":101,
  "url": "https://www.somesite.com/myimage.jpg",
  "DocCategory":47,
  "Description":"Extraoral",
  "ImgType":"Photo"
}
```

Example response:
201 Created
(no "location" Header or object because we don't support GET single documents)

Documents POST Upload

Version Added: 21.1

Upload the actual file as rawBase64. Specify the extension of the file. Examples could include .pdf, .jpg, .dcm, .doc, etc. rawBase64goes into the database in document.Note as "_rawBase64_:" followed by the extension and rawBase64. Example: "_rawBase64_:.jpg_errGEreRi3fWWtB+gWWEgg..." The customer never sees this text, but when they later click on the document, the file is created and the note removed.

DocCategory is optional and defaults to the first DocCategory. It is a FK to definition.ItemName, where definition.Category=18.

Example request:

POST /documents/Upload

```
{
  "PatNum":101,
  "rawBase64": "errGEreRi3fWWtB+gWWEgg...",
  "extension":".jpg",
  "DocCategory":47,
  "Description":"Extraoral",
  "ImgType":"Photo"
}
```

Example response:

201 Created

(no "location" Header or object because we don't yet support GET single documents)

Documents POST UploadSftp

Version Added: 21.2

Prior to running this method, upload a file to your own SFTP site. This method will then pull the uploaded file into the customer's AtoZ folder, database, or cloud storage. Specify the full path of the file as the SftpAddress in the JSON. The user with the SFTP credentials must have read permission in this directory. The document.FileName of the new document will match the file name of the original file.

The ImgType parameter is optional, defaulting to Document. DocCategory is optional, defaulting to the first definition in the category where definition.Category=18. All other fields are required. The filePath of the response object will either be the full filepath of the saved file (AtoZ or cloud) or blank (database).

Example Requests:

POST /documents/UploadSftp

```
{
  "PatNum": 15,
  "DocCategory":239,
  "ImgType":"Photo",
  "SftpAddress": "MySftpSite/myUsername/Documents/SmithJRadiograph.png",
  "SftpUsername": "myUsername",
  "SftpPassword": "myPassword"
}
```

Example responses:

```
{
  "DocNum": 411,
  "filePath": "\\server\OpenDentImages\S\SmithJohn15\SmithJRadiograph.png",
```

```
  "Description": "SmithJRadiograph.png",
  "Note": "",
  "DateCreated": "2021-05-26 08:16:46",
  "DateTStamp": "2021-05-26 08:16:46",
  "serverDateTime": "2021-05-26 08:16:46"
}
```

201 Created
400 BadRequest (missing fields and Sftp connection errors)
404 NotFound "Patient not found"

Documents POST Thumbnails

Version Added: 21.2

Prior to running this method, first call a GET /documents for the patient to see their list of documents. Only standard image files from this list can be made into thumbnails. Mounts and pdfs will be skipped. This method gets the thumbnails for all images for the patient, creating any that do not already exist as thumbnails. Created thumbnails will be 100 x 100 and will use the same filename as the original image.

This will place files on an SFTP site that you specify. After running this method, download the resulting files from your SFTP site. The user with the SFTP credentials must have write permission in this directory. Directory will be created if it does not exist, and files already existing with the specified name will be overwritten. All file storage options (LocalAtoZ, InDatabase, and Cloud) are supported.

Example Requests:
POST /documents/Thumbnails

```
{
  "PatNum": 15,
  "SftpAddress": "MySftpSite/myUsername/Thumbnails/",
  "SftpUsername": "myUsername",
  "SftpPassword": "myPassword"
}
```

Example responses:

```
[
 {
   "DocNum": 440,
   "FileName": "SmithJohn440.gif"
 },
 {
   "DocNum": 441,
   "FileName": "SmithJohn441.gif"
 }
]
```

200 OK "No documents could be made into thumbnails."
201 Created
400 BadRequest (missing fields and Sftp connection errors)
404 NotFound "Patient not found" and "No documents exist for this patient"

Documents POST DownloadMount

Version Added: 21.2

You probably don't want to use this. Gets all the individual images for one mount, but there's no information

about which position in the mount, flipping, rotation, etc. It is much more common to use Documents POST DownloadSftp to get a composite image for a mount instead of the individual images.

Prior to running this method, first call a GET /documents for the patient to see their list of documents and mounts. Use the MountNum to run this method. All images in the mount will be downloaded to the specified folder. Created files will use the same filename as the original image.

This will place files on an SFTP site that you specify. After running this method, download the resulting files from your SFTP site. The user with the SFTP credentials must have write permission in this directory. Directory will be created if it does not exist, and files already existing with the same name will be overwritten. All file storage options (LocalAtoZ, InDatabase, and Cloud) are supported.

Example Requests:
POST /documents/DownloadMount

```
{
  "MountNum": 15,
  "SftpAddress": "MySftpSite/myUsername/Mounts/",
  "SftpUsername": "myUsername",
  "SftpPassword": "myPassword"
}
```

Example responses:

```
[
 {
   "DocNum": 440,
   "FileName": "SmithJohn440.gif"
 },
 {
   "DocNum": 441,
   "FileName": "SmithJohn441.gif"
 }
]
```

200 OK "No mount images could be downloaded."
201 Created
400 BadRequest (missing fields and Sftp connection errors)
404 NotFound "Mount not found."

Etrans

Etrans POST

Version Added: Plan to add to 21.2

Creates one Etrans of the type ERA 835.

Example Requests:
POST /etrans

```
{
  "DateTimeTrans": "2021-05-26 08:16:46",
  "HqClearinghouseNum": 3,
  "MessageText": "ISA*00*       *00*       *30*330989922…",
  "UserNum": 4
}
```

Example responses:
201 Created
(no "location" Header or object because we don't support GET single inssubs)

InsSubs

Links an InsPlan to a Subscriber (patient.PatNum). Also, see PatPlans. BenefitNotes is specifically designed to store automated notes.

InsSubs POST (create)

Version Added: 21.1

This does not create a new insurance plan, change benefits, etc.

Example request:
POST /inssubs

```
{
  "PlanNum": 15,
  "Subscriber": 1,
  "DateEffective":"2019-01-01",
  "DateTerm": "2019-12-31",
  "SubscriberID":"8645332",
  "BenefitNotes":""
}
```

Example response:
201 Created
(no "location" Header or object because we don't support GET single inssubs)

InsSubs PUT (update)

Version Added: 21.1

This can be used to reassign a different plan to a subscriber while leaving the PatPlans (coverage) for family members untouched. Fields other than PlanNum and BenefitNotes will be ignored. There's no way to obtain the InsSubNum from a method in the API; you would instead use a query.

Example request:
PUT /inssubs/82

```
{
  "PlanNum": 15,
  "BenefitNotes":"Add these notes."
}
```

Example response:
200 OK

InsSubs DELETE

Version Added: 21.1

Will fail if any PatPlans exist. There's no way to obtain the InsSubNum from the API; you would instead use a query.

Example request:
DELETE /inssubs/82

Example response:
200 OK
400 Bad Request, "Can't delete InsSub because PatPlans are still attached."

InsVerifies

InsVerifies PUT

Version Added: 21.1 (backported)

This shows at the right of the Insurance Plan window in the Benefits Last Verified box. As with all insverify entries in our database, historical entries are always retained in the insverifyhist table. VerifyType can be "PatientEnrollment" or "InsuranceBenefit". If PatientEnrollment, then FKey must be valid patplan.PatPlanNum. If InsuranceBenefit, then FKey must be valid insplan.InsPlanNum. DefNum is optional or must be a valid where definition.Category=38 (InsuranceVerificationStatus).

Example request:
PUT /insverifies

```
{
  "DateLastVerified": "2021-03-16",
  "VerifyType": "PatientEnrollment",
  "FKey": 325,
  "DefNum": 721,
}
```

Example response:
200 OK

Medications

Medications GET

Version Added: 21.3

Gets the list of medications that can be assigned to patients.

Rarely used. Usually just use MedicationPats GET and POST.

Example Request:

GET /medications

Example responses:

```
[
{
  "MedicationNum": 12,
  "MedName": "Glucophage",
  "GenericNum": 124,
  "genericName": "Metformin",
  "Notes": "Antidiabetic agent",
```

```
  "DateTStamp": "2016-12-01"
},
etc…
]
```

Example responses:
201 Created
400 BadRequest (Missing or Invalid fields)

Medications POST

Version Added: 21.3

Creates a new medication.

Rarely used. Usually just use MedicationPats GET and POST.

MedName: Required.
genericName: Optional. If not provided this will be the same as MedName.
Notes: Optional.

Example Request:

POST /medications

```
{
  "MedName": "Metformin"
}
```

```
{
  "MedName": "Glucophage",
  "genericName": "Metformin",
  "Notes": "Antidiabetic agent"
}
```

Example responses:
201 Created
400 BadRequest (Missing or Invalid fields)
404 Not Found "No Medication with that genericName was found"

MedicationPats

MedicationPats GET

Version Added: 21.3

Gets a list of all medications for a given patient.

Example request:
GET /medicationpats?PatNum=234

Example responses:

```
[
{
  "MedicationPatNum": 45,
```

```
  "PatNum": "234",
  "medName": "Metformin",
  "MedicationNum": 12,
  "PatNote": "500mg, taken twice a day.",
  "DateStart": "2000-06-20",
  "DateStop": "0001-01-01",
  "ProvNum": 1
},
etc…
]
```

MedicationPats POST

Version Added: 21.3

Attaches a medication to a patient.

PatNum: Required.
medName: Required. Tries to match to an existing medication. If a new medication must be created, it will be assumed to be generic rather than brand. For more control, use medication POST.
MedicationNum: Rarely used. Just use **medName** instead, which handles insertion of a Medication automatically. If MedicationNum is used, then medName is not required.
PatNote. Optional.
DateStart: Optional. String in "yyyy-MM-dd" format. Default "0001-01-01".
DateStop: Optional. String in "yyyy-MM-dd" format. Default "0001-01-01".
ProvNum: Optional. Default is 0.

Example requests:
POST /medicationpats

```
{
  "PatNum": 234,
  "medName": "Metformin"
}
```

```
[
{
  "PatNum": 234,
  "medName": "Metformin",
  "PatNote": "500mg, taken twice a day.",
  "DateStart": "2006-02-01",
  "DateStop": "0001-01-01",
  "ProvNum": 1
},
etc…
]
```

Example responses:
201 Created
400 BadRequest (Missing or Invalid fields)
404 NotFound "Patient not found", "Provider not found" or "Medication not found"

Operatories

Operatories GET

Version Added: 21.1

ClinicNum is optional. Gets a list of all operatories.

Example requests:
GET /operatories
GET /operatories?ClinicNum=2

Example responses:


```
[
{
"OperatoryNum": 1,
"OpName": "Dr. Brian Albert",
"Abbrev": "OP-1",
"ItemOrder": 0,
"IsHidden": "false",
"ProvDentist": 1,
"ProvHygienist": 0,
"IsHygiene": "false",
"ClinicNum": 0,
"SetProspective": "false",
"IsWebSched": "false"
},
  {
"OperatoryNum": 2,
"OpName": "Dr. Sarah Lexington",
"Abbrev": "OP-2",
"ItemOrder": 1,
"IsHidden": "false",
"ProvDentist": 3,
"ProvHygienist": 0,
"IsHygiene": "false",
"ClinicNum": 0,
"SetProspective": "false",
"IsWebSched": "false"
}
]
```


PatFields

PatFields GET

PatNum and FieldName are required parameters. If a PatField exists for the patient, it gets the value. If a PatField does not exist, it returns an empty string.

Example request:
GET /patfields?PatNum=101&FieldName=Ins%20Verified

Example responses:


```
{
```

```
  "FieldName": "Ins Verified",
  "PatNum":101,
  "FieldValue":"Yes"
}
```

PatFields PUT

If a PatField already exists for the patient, it gets set to the new value, overwriting the old value. If a PatField does not yet exist for the patient, then an PatField gets inserted into the database. To delete a PatField, set the FieldValue to an empty string.

Example request:
PUT /patfields

```
{
  "FieldName": "Ins Verified",
  "PatNum":101,
  "FieldValue":"Yes"
}
```

Example responses:
200 OK
400 Bad Request (with explanation)

PatientNotes

PatientNotes GET

Version Added: 21.2

PatientNotes have a 1:1 relationship to Patients and contain additional information about the patient. The PatNum must be included in the URL.

Example request:
GET /patientNotes/15

Example response:

```
{
  "Medical": "Diabetes.",
  "ICEName": "Carolyn Wright",
  "ICEPhone": "(820) 202-1134"
}
```

200 OK

PatientNotes PUT

Version Added: 21.2

Include the PatNum in the URL. All JSON fields are optional. If you want to append a note instead of replace, then use PatientNotes GET, and do your own concatenation before calling PUT.

Medical: Replace existing medical history note.
ICEName: Replaces existing ICEName.
ICEPhone: String. Replaces existing ICEPhone.

Example requests:
PUT /patientNotes/15

```
{
  "Medical": "Unknown family history.",
}
```

```
{
  "ICEName": "Herbert Grayson",
  "ICEPhone": "(134) 721-1321"
}
```

Example response:

```
{
  "Medical": "Unknown family history.",
  "ICEName": "Carolyn Wright",
  "ICEPhone": "(820) 202-1134"
}
```

200 OK
400 BadRequest (Invalid fields)
404 NotFound "Patient not found".

Patients

Patients GET (single)

Version Added: 21.1

A unique patient is identified by a PatNum obtained from a previous search of some sort. Birthdate value can be 0001-01-01, which is equivalent to "none".

URL Search Parameters:
PatNum: In URL only

Example request:
GET /patients/15

Example response:

```
{
"PatNum": 15,
"LName": "Smith",
"FName": "John",
"MiddleI": "",
"Preferred": "",
"PatStatus": "Patient",
"Gender": "Male",
"Birthdate": "1979-06-23",
"SSN": "352664588",
"Address": "125 Satin Heights",
"City": "San Jose",
"State": "CA",
"Zip": "05698",
```

```
"HmPhone": "(536)624-5871",
"WkPhone": "(536)265-8587",
"WirelessPhone": "(536)987-5621",
"Guarantor": 15,
"Email": "johns@hotmail.com",
"priProvAbbr": "DOC1",
"secProvAbbr": "",
"BillingType": "Standard Account",
"ImageFolder": "SmithJohn15",
"ChartNumber": "",
"ClinicNum": 0,
"clinicAbbr": "",
"siteDesc": ""
}
```


Patients GET (multiple)

Version Added: 21.1

You can get a list of patients who meet a set of search criteria. Most string parameters support partial match and are not case sensitive. The parameters and results are very similar to the Patient Select window in Open Dental because it uses the same code. The results do not contain every possible patient field, just the ones that you might see in the Patient Select window. For example, the following fields will be blank, 0, or null: Gender, Zip, Guarantor, and ImageFolder. You can GET individual patients if you want that information.

If you do not yet have a PatNum and are looking for a specific patient by name, you should generally use a combination of LName, FName, and Birthdate to help ensure uniqueness.

URL Search Parameters:
LName, FName, Phone, Address, hideInactive, City, State, SSN, ChartNumber, BillingType, guarOnly, showArchived, Birthdate, SiteNum, SubscriberId, Email, Country, clinicNums (comma separated list), clinicAbbr, invoiceNumber.

Example requests:
GET /patients?Offset=700
GET /patients?LName=smi
GET /patients?LName=smi&FName=j&Birthdate=1976-05-24&hideInactive=true

Example response:


```
[
{
"PatNum": 16,
"LName": "Smith",
"FName": "Jane",
"MiddleI": "",
"Preferred": "",
"PatStatus": "Patient",
"Gender": null,
"Birthdate": "1976-05-24",
"SSN": "632458956",
"Address": "125 Satin Heights",
"City": "San Jose",
```

```
"State": "CA",
"Zip": null,
"HmPhone": "(536)624-5871",
"WkPhone": "(536)987-4822",
"WirelessPhone": "",
"Guarantor": 0,
"Email": "smithfam@yahoo.com",
"priProvAbbr": "DOC1",
"secProvAbbr": "",
"BillingType": "Standard Account",
"ImageFolder": null,
"ChartNumber": "",
"ClinicNum": 0,
"clinicAbbr": "",
"siteDesc": ""
},
  {
"PatNum": 15,
"LName": "Smith",
"FName": "John",
"MiddleI": "",
"Preferred": "",
"PatStatus": "Patient",
"Gender": null,
"Birthdate": "1979-06-23",
"SSN": "352664588",
"Address": "125 Satin Heights",
"City": "San Jose",
"State": "CA",
"Zip": null,
"HmPhone": "(536)624-5871",
"WkPhone": "(536)265-8587",
"WirelessPhone": "(536)987-5621",
"Guarantor": 0,
"Email": "johns@hotmail.com",
"priProvAbbr": "DOC1",
"secProvAbbr": "",
"BillingType": "Standard Account",
"ImageFolder": null,
"ChartNumber": "",
"ClinicNum": 0,
"clinicAbbr": "",
"siteDesc": ""
},
etc...
]
```


Example response (no results):


```
[]
```


Patients GET Simple

Version Added: 21.2

This is used instead of GET (multiple) if you need to pass in DateTStamp to get recent changes, or if you need

additional fields that are not included in GET (multiple).

LName: Supports partial string matching and is case-insensitive.
FName: Supports partial string matching and is case-insensitive.
Birthdate: In "yyyy-mm-dd" format.
ClinicNum: A single ClinicNum. Leave blank if not using clinics or want results for all clinics.
PatStatus: Either "Patient", "NonPatient", "Inactive", "Archived", "Deleted", "Deceased", or "Prospective".
DateTStamp: In "yyyy-mm-dd HH:mm:ss" format.
PriProv (added in version 21.3): A single ProvNum. Leave blank if you want results for all primary providers.

All parameters are optional. If you do not yet have a PatNum and are looking for a specific patient by name, you should generally use a combination of LName, FName, and Birthdate to help ensure uniqueness. Use the DateTStamp to filter results for patients updated in the database since that date. The serverDateTime is included in the response to use for subsequent requests.

Example requests:
GET /patients/Simple?Offset=700
GET /patients/Simple?DateTStamp=2021-07-01%2005%3A30%3A00
GET /patients/Simple?LName=smi&FName=eter&Birthdate=1976-05-24
GET /patients/Simple?PatStatus=Inactive

Example response:


```
[
{
"PatNum": 1,
"LName": "Smith",
"FName": "John",
"MiddleI": "",
"Preferred": "",
"PatStatus": "Patient",
"Gender": "Male",
"Birthdate": "1976-05-24",
"SSN": "",
"Address": "123 Elm St",
"City": "Salem",
"State": "OR",
"Zip": "97301",
"HmPhone": "",
"WkPhone": "(123)456-7890",
"WirelessPhone": "",
"Guarantor": 1,
"Email": "name@web.com",
"PriProv": 1,
"priProvAbbr": "DOC1",
"secProvAbbr": "",
"BillingType": "Standard Account",
"ImageFolder": "SmithJohn1",
"ChartNumber": "",
"ClinicNum": 1,
"clinicAbbr": "Southside",
"siteDesc": "",
"DateTStamp": "2021-07-26 14:22:55",
"serverDateTime": "2021-08-05 09:05:42"
},
```

```
  {
"PatNum": 2,
"LName": "Johnson",
"FName": "Brody",
"MiddleI": "",
"Preferred": "",
"PatStatus": "Patient",
"Gender": "Male",
"Birthdate": "0001-01-01",
"SSN": "",
"Address": "",
"City": "",
"State": "",
"Zip": "",
"HmPhone": "",
"WkPhone": "5035035030",
"WirelessPhone": "",
"Guarantor": 2,
"Email": "",
"PriProv": 1,
"priProvAbbr": "DOC1",
"secProvAbbr": "",
"BillingType": "Standard Account",
"ImageFolder": "JohnsonBrody2",
"ChartNumber": "",
"ClinicNum": 1,
"clinicAbbr": "Southside",
"siteDesc": "",
"DateTStamp": "2021-07-23 14:52:07",
"serverDateTime": "2021-08-05 09:05:42"
},
etc...
]
```


200 OK
400 BadRequest (Invalid Fields)

Patients POST (create)

Version Added: 21.1

The only required fields are LName and FName.

Example request:
POST /patients


```
{
"LName": "Smith",
"FName": "Jane",
"Gender": "Female",
"Birthdate": "1976-05-24",
"Address": "125 Satin Heights",
"City": "San Jose",
"State": "CA",
"Zip": "97301",
"HmPhone": "(536)624-5871",
"WkPhone": "(536)987-4822",
```

```
"WirelessPhone": "",
"Email": "smithfam@yahoo.com"
}
```

Example response:
201 Created
Header "location": https://api.opendental.com/api/v1/patients/16

```
{
"PatNum": 16,
"LName": "Smith",
"FName": "Jane",
"MiddleI": "",
"Preferred": "",
"PatStatus": "Patient",
"Gender": "Female",
"Birthdate": "1976-05-24",
"SSN": "",
"Address": "125 Satin Heights",
"City": "San Jose",
"State": "CA",
"Zip": "97301",
"HmPhone": "(536)624-5871",
"WkPhone": "(536)987-4822",
"WirelessPhone": "",
"Guarantor": 16,
"Email": "smithfam@yahoo.com",
"priProvAbbr": "DOC1",
"secProvAbbr": "",
"BillingType": "Standard Account",
"ImageFolder": "SmithJane16",
"ChartNumber": "",
"ClinicNum": 0,
"clinicAbbr": "",
"siteDesc": ""
}
```

Patients PUT (update)

Version Added: 21.2

The following fields cannot be set as part of a PUT: PatNum, priProvAbbr, secProvAbbr, ImageFolder, ChartNumber, clinicAbbr and siteDesc. Attempts to set them will be silently ignored. All remaining fields are optional, and it is common to only set one or two fields. If an empty string is sent for a string fields, then it will clear that field from the database, except LName and FName. ClinicNum 0 is allowed.

Example request:
PUT /patients/16

```
{
"LName": "Smith",
"FName": "Jane",
"MiddleI": "",
"Preferred": "",
"PatStatus": "Patient",
"Gender": "Female",
"Birthdate": "1976-05-24",
```

```
"SSN": "",
"Address": "125 Satin Heights",
"City": "San Jose",
"State": "CA",
"Zip": "97301",
"HmPhone": "(536)624-5871",
"WkPhone": "(536)987-4822",
"WirelessPhone": "",
"Guarantor": 16,
"Email": "smithfam@yahoo.com"
"BillingType": "Standard Account",
"ClinicNum": 0
}
```


Example responses:
200 OK
400 Bad Request (with explanation)
404 Not Found (with explanation)

PatPlans

A PatPlan row in the database indicates coverage aka eligibility. If there is no patplan row, then the patient does not have coverage. So eligibility can be set by adding or removing PatPlan rows.

PatPlans POST (create)

Version Added: 21.1

This adds a PatPlan row to the database. Required: PatNum and InsSubNum. This requires that a valid InsSub is already in place. There's no way to obtain the InsSubNum from the API; you would instead use a query. Ordinal is optional, with a default of 1. If Ordinal is 1, and there is already primary insurance, the other insurance will get bumped to Ordinal 2. Default relationship is Self. If this plan is already linked to this InsSub, then response will be BadRequest.

Example request:
POST /patplans


```
{
  "PatNum": "15",
  "Ordinal":1,
  "Relationship":"Self",
  "InsSubNum":101
}
```


Example response:
201 Created
(no "location" Header or object because we don't support GET single patplans)

PatPlans DELETE

Version Added: 21.1

This removes a PatPlan row from the database, indicating no coverage. There's no way to obtain the PatPlanNum from the API; you would instead use a query.

Example request:
DELETE /patplans/251

Example response:
200 OK

Payments

Payments POST (create)

Version Added: 21.2

Creates a payment for a patient. Does not support income transfers or insurance payments. Includes appropriate splits according to the payment allocation preferences from Setup > Allocations within Open Dental.

PayAmt: Required.
PatNum: Required.
PayDate: Optional. Defaults to today's date. Follows the office's preference to allow future-dated payments.
CheckNum: Optional.
PayNote: Optional.
BankBranch: Optional.
ClinicNum: Optional. Defaults to patient.ClinicNum

Payments will be applied to the patient's payment plan, if one is eligible. The oldest plan will be chosen if there is more than one eligible plan. If needed, the user can later detach the payment from the plan within the Edit Payment Plan window by editing the split and unchecking "Attach to Payment Plan". They can also attach to a different plan in the same manner.

isPatientPreferred: When entering a payment through Open Dental directly, there is a checkbox for this option. This API field allows the same functionality. It causes the splits to go to the patient instead of being split among family members on a FIFO basis.

Example requests:
POST /payments

```
{
  "PayAmt": "129.99",
  "PatNum": 15
}
```

```
{
  "PatNum": 13,
  "PayAmt": 20,
  "PayDate": "2021-07-05",
  "CheckNum": "2058",
  "PayNote": "Check payment through website",
  "BankBranch": "My Credit Union",
  "ClinicNum": "3",
  "isPatientPreferred" : "true"
}
```

Example responses:
201 Created
400 BadRequest (Missing or invalid fields)
404 NotFound (Patient not found)

PayPlans

PayPlans POST (create)

Version Added: 21.3

Creates a payment plan for the patient with the specified terms. This is a patient payment plan, not an insurance or dynamic payment plan. Procedures and adjustments are not associated with the plan. The terms of the payment plan are stored in the PayPlan.Note field and returned in the response. Uses the patient's information to set the provider and clinic (if enabled) associated with the payment plan.

PatNum: Required.
Guarantor: Optional. The person responsible for payments. Default this plan's patient.
PayPlanDate: Optional. The date of the plan agreement. Default today.
principalAmount: Required. The principal amount for the plan.
PayAmt: This or **NumberOfPayments** is required. The amount due per payment plan charge.
NumberOfPayments: This or **PayAmt** is requried. The total number of payments in the payment plan.
APR: Optional. Default 0.
DownPayment: Optional. Default 0.00.
Note: Optional. The terms of the payment plan are stored in this field by default. Additional note text is appended to these terms.
ChargeFrequency: Optional. Either "Weekly", "EveryOtherWeek", "Monthly", "Quarterly", or "OrdinalWeekday" (ie, second Tueday of each month, based on the DatePayPlanStart). Default "Monthly".
DatePayPlanStart: Optional. Due date of first payment. Default one month after the PayPlanDate.
DateInterestStart: Optional. Date the payment plan can start posting interest charges. Default minval.

Example requests:
POST /payplans

```
{
 "PatNum" : 61,
 "principalAmount": 500,
 "PayAmt": 100
}
```

```
{
 "PatNum" : 19,
 "principalAmount": 256.12,
 "NumberOfPayments": 12,
 "APR": 5
}
```

```
{
 "PatNum" : 11,
 "Guarantor": 21,
 "PayPlanDate": "2021-09-06",
 "principalAmount": 500,
 "PayAmt": 75.50,
 "APR": 18,
 "DownPayment": 125,
 "ChargeFrequency": "OrdinalWeekday",
 "Note": "No payments until 12/06/2021. No interest until 01/03/2022. Payments
     due first Monday of each month.",
```

```
 "DatePayPlanStart": "2021-12-06",
 "DateInterestStart": "2022-01-03"
}
```

Example responses:

```
"9/20/2021 - Date of Agreement: 9/20/2021, Total Amount: 500.00, APR: 0, Total Cost of Loan: 500.00 "
```

```
"9/20/2021 - Date of Agreement: 9/20/2021, Total Amount: 256.12, APR: 5, Total Cost of Loan: 263.11 "
```

```
"9/20/2021 - Date of Agreement: 9/6/2021, Total Amount: 500.00, APR: 18, Total Cost of Loan: 511.66 No payments until 12/06/2021. No interest until 01/03/2022. Payments due first Monday of each month."
```

201 Created
400 BadRequest (Missing or Invalid fields)
404 NotFound "Patient not found"

Popups

Popups POST (create)

Version Added: 21.1

Required fields are PatNum and Description. Be careful. Popups tend to be annoying to users and are usually reserved for more important notes.

Example Request:
POST /popups

```
{
  "PatNum": "15",
  "Description":"Needs to fill out paperwork",
  "PopupLevel":"Patient"
}
```

Example response:
201 Created
(no "location" Header or object because we don't support GET single popups)

Preferences

Preferences GET

Version Added: 21.1
PrefName is optional. Otherwise you end up with all ~1000 preferences, paginated.

Example requests:
GET /preferences?PrefName=RecallDaysPast
GET /preferences?Offset=200

Example response:

```
[
```

```
  {
"PrefNum": 14,
"PrefName": "PracticeDefaultBillType",
"ValueString": "40"
},
  {
"PrefNum": 15,
"PrefName": "RecallDaysPast",
"ValueString": "365"
},
  {
"PrefNum": 16,
"PrefName": "RecallDaysFuture",
"ValueString": "7"
},
etc...
]
```


Procedurelogs

Procedurelogs GET

Version Added: 21.1
PatNum is required.

Parameters:

DateTStamp: Only include procedurelogs with a DateTStamp altered after the specified date and time. This provides a good way for you to keep a synchronized copy of procedurelogs. Store serverDateTime (added in v21.2) that gets returned and use it to run the next GET.

Example Requests:
GET /procedurelogs?PatNum=261
GET /procedurelogs?PatNum=261&DateTStamp=2020-07-30

Example response:


```
[
{
"ProcNum": 1,
"PatNum": 1,
"AptNum": 0,
"ProcDate": "2021-04-01",
"ProcFee": "255.00",
"Surf": "MODL",
"ToothNum": "4",
"ToothRange": "",
"Priority": 0,
"priority": "",
"ProcStatus": "C",
"ProvNum": 1,
"provAbbr": "DOC1",
"Dx": "",
"dxName": "",
```

```
"PlannedAptNum": 0,
"ClinicNum": 0,
"CodeNum": 2,
"procCode": "D0120",
"descript": "periodic oral evaluation - established patient",
"DateTStamp":"2021-11-03 05:30:06",
"serverDateTime":"2021-11-03 09:32:45"
}
]
```

Procedurelogs POST (create)

Version Added: Planned for 21.3
Creates a new procedure for a given patient.

PatNum: Required.
ProcDate: Required. String in "yyyy-MM-dd" format.
ProcStatus: Required. Either Treatment Planned (TP) or Complete (C).
procCode: Required. This should be a valid D code, example: D0120.

AptNum: Optional.
ProcFee: Optional.
Surf: Optional.
ToothNum: Optional.
ToothRange: Optional.
Priority: Optional. Definition.DefNum where definition.Category=20. Default is the first definition in that Category. If Priority is used, then priority will be set automatically.
priority: Optional. String version of Priority. If priority is used, then Priority will be set automatically.
ProvNum: Optional. Defaults to the PriProv of the appointment if given, otherwise it will check the patient's default provider. Failing either of the previous options, it will be set to the dental office's default provider. If ProvNum is used, then provAbbr will be set automatically.
provAbbr: Optional. If provAbbr is used, then ProvNum will be set automatically.
Dx: Optional. Definition.DefNum where definition.Category=16. Default is the first definition in that Category. If Dx is used, then dxName will be set automatically.
dxName: Optional. String version of Dx. If dxName is used, then Dx will be set automatically.
PlannedAptNum: Optional. Only set if this procedure is on a planned appointment, otherwise it will be 0.
ClinicNum: Optional. Defaults to the appointment if given, otherwise it will check the provider. Failing either of the previous options, it will be set to the dental office's default clinic.

Example Requests:
POST /procedurelogs

```
{
"PatNum": 1,
"ProcDate": "2021-04-01",
"ProcStatus": "C",
"procCode": "D0120"
}
```

```
{
"PatNum": 1,
"AptNum": 0,
```

```
"ProcDate": "2021-04-01",
"ProcFee": "255.00",
"Surf": "MODL",
"ToothNum": "4",
"ToothRange": "",
"Priority": 119,
"priority": "Low",
"ProcStatus": "C",
"ProvNum": 1,
"provAbbr": "DOC1",
"Dx": 87,
"dxName": "Caries",
"PlannedAptNum": 0,
"ClinicNum": 0,
"procCode": "D0120"
}
```


Example response:
201 Created
400 BadRequest (Missing or invalid fields)
404 NotFound "Patient not found", "Appointment not found", "procCode not found", "Priority not found", "Dx not found"

Providers

Providers GET

Version Added: 21.1

Gets a list of all providers. Can be optionally filtered by either **ClinicNum** or **DateTStamp**. Two different queries are used. ClinicNum is not a field in the Provider table, so it does not get returned in any results.

ClinicNum: Optional. Specifying a ClinicNum returns list of non-hidden providers that are either not restricted to a clinic, or are restricted to the ClinicNum provided.

DateTStamp (added in version 21.3)**:** Optional. String in "yyyy-MM-dd HH:mm:ss" format. Get providers altered after the specified date and time. This is a good way for you to keep a synchronized copy of providers. Store serverDateTime that gets returned, and use it to run the next GET.

Example requests:
GET /providers
GET /providers?DateTStamp=2021-08-01%2005%3A30%3A00

Example responses:


```
[
 {
  "ProvNum": 1,
  "Abbr": "DOC1",
  "LName": "Albert",
  "FName": "Brian",
  "Suffix": "",
  "IsSecondary": "False",
  "DateTStamp": "2021-08-04 08:33:01",
  "serverDateTime": "2021-08-31 12:05:31"
```

```
},
 {
  "ProvNum": 2,
  "Abbr": "HYG1",
  "LName": "Jones",
  "FName": "Tina",
  "Suffix": "",
  "IsSecondary": "True",
  "DateTStamp": "2021-08-15 13:18:51",
  "serverDateTime": "2021-08-31 12:05:31"
}
]
```

Example request:
GET /providers?ClinicNum=4

Example response:

```
{
  "ProvNum": 6,
  "Abbr": "HYG2",
  "LName": "Thomason",
  "FName": "Shirley",
  "Suffix": "",
  "IsSecondary": "False",
  "DateTStamp": "2021-07-25 10:20:21",
  "serverDateTime": ""
}
```

200 OK
400 BadRequest "DateTStamp format must be yyyy-MM-dd HH:mm:ss"
404 NotFound "Clinic not found"

Queries

Queries POST

Version Added: 21.1

Runs a custom query against the database. Queries are screened to be read-only. Temporary tables are allowed. Any command that would change the database will not be run. The Audit Trail for these requests includes the command that was run.

Rarely, an office will insist on higher security for queries. There is an option to add a <UserLow> and <PasswordLow> to the OpenDentalWebConfig.xml file found in the eConnector installation folder. Do not leave a blank <UserLow> tag in the config file, because you would then need to add MySQL permission for the blank user @localhost. This would allow the office to use a different MySQL user for these queries, and that MySQL user could have more restrictive permissions than the MySQL user used for all the other API methods.

The results of the query are written to a file and saved to the SFTP site specified in the JSON. SftpPort is optional with port 22 used by default. The SftpAddress string must contain the full filePath. The user with the SFTP credentials must have write permission in this directory. Directory will be created if it does not exist, and files already exististing with the specified name will be overwritten. Query results are written in comma-delimited CSV format. If there are no results, then the file will only contain "OK".

Example Requests:
POST /queries

```
{
  "SqlCommand": "SELECT PatPlanNum, PatNum, InsSubNum FROM patplan",
  "SftpAddress": "MySftpSite/myUsername/Patient Plans/PatPlans-35.csv",
  "SftpPort": 25,
  "SftpUsername": "myUsername",
  "SftpPassword": "myPassword"
}
```

```
{
  "SqlCommand": " SELECT * FROM patient WHERE Birthdate LIKE '%-06-22' ",
  "SftpAddress": "MySftpSite/myUsername/Birthdays/Jun22.csv",
  "SftpUsername": "myUsername",
  "SftpPassword": "myPassword"
}
```

```
{
  "SqlCommand": "SELECT * FROM InsSub WHERE DateTerm < '2021-01-01'",
  "SftpAddress": "MySftpSite/myUsername/Insurance Subscriptions/Expired2021.csv",
  "SftpUsername": "myUsername",
  "SftpPassword": "myPassword"
}
```

Example responses:
201 Created
400 BadRequest (SQL syntax and Sftp connection errors)
401 Unauthorized (Query is not read-only or is not executing on temporary tables)

Queries PUT ShortQuery

Version Added: 21.2

Runs a custom query against the database, returning at most 100 rows. Queries are screened to be read-only. Temporary tables are allowed. Any command that would change the database will not be run. See Queries Post, above, for an explanation of the completely optional restricted MySQL UserLow. The Audit Trail for these requests includes the command that was run.

The results of the query are returned as a datatable in the JSON. While pagination is supported for results that return over 100 rows, it is recommended to instead use Queries POST for longer results.

Example Requests:
PUT /queries/ShortQuery
PUT /queries/ShortQuery?Offset=200

```
{
  "SqlCommand": "SELECT clinic.Description, COUNT(*) AS NumberOfPatients
     FROM patient,clinic
     WHERE patient.ClinicNum=clinic.ClinicNum
     GROUP BY clinic.ClinicNum"
}
```

```
{
```

```
  "SqlCommand": "SELECT * FROM payperiod"
}
```

Example Responses:

```
[
  {
     "Description":"Southern Office",
     "NumberOfPatients":62
  },
  {
     "Description":"Northern Office",
     "NumberOfPatients":107
  },
  {
     "Description":"Downtown Office",
     "NumberOfPatients":93
  }
]
```

```
[
{
  "PayPeriodNum": 200,
  "DateStart": "2021-05-01T00:00:00",
  "DateStop": "2021-05-31T00:00:00",
  "DatePaycheck": "2021-06-04T00:00:00"
},
{
  "PayPeriodNum": 201,
  "DateStart": "2021-06-01T00:00:00",
  "DateStop": "2021-06-30T00:00:00",
  "DatePaycheck": "2021-07-05T00:00:00"
}
]
```

200 OK
400 BadRequest (SQL syntax errors)
401 Unauthorized (Query is not read-only or is not executing on temporary tables)

Recalls

Recalls PUT Status

Version Added: 21.2

Updates the RecallStatus on a patient's recall. This status describes the recall reminder itself, and not the status of the resulting appointment. To instead change an appointment's status, use Appointments PUT Confirm. A commlog for the patient is also created with CommType.Recall, Mode_.None, and note with "Recall reminder" and the new RecallStatus.

PatNum: Required.
recallType: Required. Either "Prophy", "ChildProphy", or "Perio".
RecallStatus: Any definition.DefNum where Definition.Category=13. Leave blank to set status to "None".
commlogMode: Optional. Either "None", "Email", "Mail", "Phone", "In Person", "Text", or "Email and Text".

commlogNote: Optional. This text will be used instead of the default commlog.Note.

Example requests:
PUT /recalls/Status

```
{
  "PatNum": 71,
  "recallType": "Prophy",
  "RecallStatus": 312,
  "commlogMode": "Phone",
  "commlogNote": "Patient out of town until 08/30/2021."
}
```

```
{
  "PatNum": 46,
  "recallType": "Perio"
}
```

Example responses:
200 OK
400 BadRequest (Invalid Fields)
404 NotFound "Patient not found", "Patient does not have any recalls", or "Patient does not have a recall of the specified RecallType".

RefAttaches

RefAttaches POST

Version Added: 21.2

Attaches a patient to a referral source. The referral source must be specified by either **ReferralNum** or **referralName**. Before calling this method, use Referrals GET to find the ReferralNum of an existing referral source. Alternatively, specify a referralName to search the LName of existing referrals for an exact match. If a match is not found, a new referral with that name is created and used.

PatNum: Required.
ReferralNum: This or **referralName** is required.
referralName: This or **ReferralNum** is required.
RefDate: Optional.
ReferralType: Optional. Either "RefTo", "RefFrom", or "RefCustom". Default "RefFrom".
RefToStatus: Optional. Typically only used with outgoing referrals. Either "None", "Declined", "Scheduled", "Consulted", "InTreatment", or "Complete". Default "None".
Note: Optional.

Example requests:
POST /refattaches

```
{
  "PatNum": 50,
  "ReferralNum": 2
}
```

```
{
  "PatNum": 50,
```

```
  "ReferralNum": 2,
  "ReferralType": "RefTo",
  "RefToStatus": "InTreatment",
  "Note": "08/12 - Dr. Bokish called to confirm this is still in progress.",
  "RefDate": "2021-07-28"
}
```

```
{
  "PatNum": 50,
  "referralName": "Google"
}
```

Example responses:
201 Created
400 BadRequest "Referral is already attached to this patient" or Invalid Fields
404 NotFound "Patient not found" or "Referral not found".

Referrals

Referrals GET

Version Added: 21.2

Gets a list of all referral sources. These can be either a provider, patient, or non-patient. The description of non-patient sources is stored in the LName field.

Example requests:
GET /referrals
GET /referrals?Offset=200

Example response:

```
[
{
  "ReferralNum": 1,
  "LName": "Davidson",
  "FName": "Norm",
  "MName": "",
  "Title": "DMD",
  "specialty": "Endodontics",
  "isPatient": "false",
  "Note": ""
},
{
  "ReferralNum": 2,
  "LName": "Facebook Ad",
  "FName": "",
  "MName": "",
  "Title": "",
  "specialty": "",
  "isPatient": "false",
  "Note": "Ad active from 06/01/2021 to 09/01/2021"
},
{
  "ReferralNum": 3,
```

```
  "LName": "Beringer",
  "FName": "Debbie",
  "MName": "L",
  "Title": "",
  "specialty": "",
  "isPatient": "true",
  "Note": ""
},
etc...
]
```

200 OK

Signalods

Signalods GET

Version Added: 21.2

SigDateTime is a required parameter. If any Signalods exist after the given dateTime, they will be returned. If a matching Signalod does not exist, it returns an empty list. Store serverDateTime that gets returned and use it to run the next GET.

Example request:
GET /signalods?SigDateTime=2021-02-02%2005%3A05%3A00

Example responses:

```
[
{
  "SignalNum": 71,
  "DateViewing": "0001-01-01",
  "SigDateTime": "2021-04-15 08:01:46",
  "FKey": 0,
  "FKeyType": "Undefined",
  "IType": "Security",
  "RemoteRole": "ClientDirect",
  "MsgValue": "",
  "serverDateTime": "2021-05-25 08:01:46"
}
]
```

201 Created
400 BadRequest (SigDateTime Required)
400 Bad Request (SigDateTime format must be yyyy-MM-dd HH:mm:ss)

Userods

Userods GET

Version Added: 21.3

Gets a list of users (called userod since "user" is a reserved word in mysql). Users are separate entities from Providers and Employees, although they can be linked. A user can be a provider, an employee, both, or neither. CEMT users are not included.

ProvNum: Contains the provNum if the user is a provider. Otherwise, 0.
providerName: The full name of the provider, including suffix. Blank if user is not a provider.
EmployeeNum: Contains the employeeNum if the user is a employee. Otherwise, 0.
employeeName: The first and last name of the employee. Blank if user is not a employee.

Example request:
GET /userods

Example responses:


```
[
 {
  "UserNum": 1,
  "UserName": "Lynda",
  "EmployeeNum": 0,
  "employeeName": "",
  "ClinicNum": 1,
  "ProviderNum": 9,
  "providerName": "Lynda Larson, DMD",
  "emailAddress": "LyndaLarson@email.com",
  "IsHidden": "False"
},
 {
  "UserNum": 2,
  "UserName": "Charlie",
  "EmployeeNum": 3,
  "employeeName": "Charles Sorenson",
  "ProviderNum": 0,
  "providerName": "",
  "ClinicNum": 1,
  "emailAddress": "Chuck@hotmail.com",
  "IsHidden": "False"
},
 {
  "UserNum": 3,
  "UserName": "Becca",
  "EmployeeNum": 4,
  "employeeName": "Becca Alexandria",
  "ProviderNum": 6,
  "providerName": "Becca Alexandria",
  "ClinicNum": 2,
  "emailAddress": "BAlexandria@gmail.com",
  "IsHidden": "False"
},
etc...
]
```


200 OK